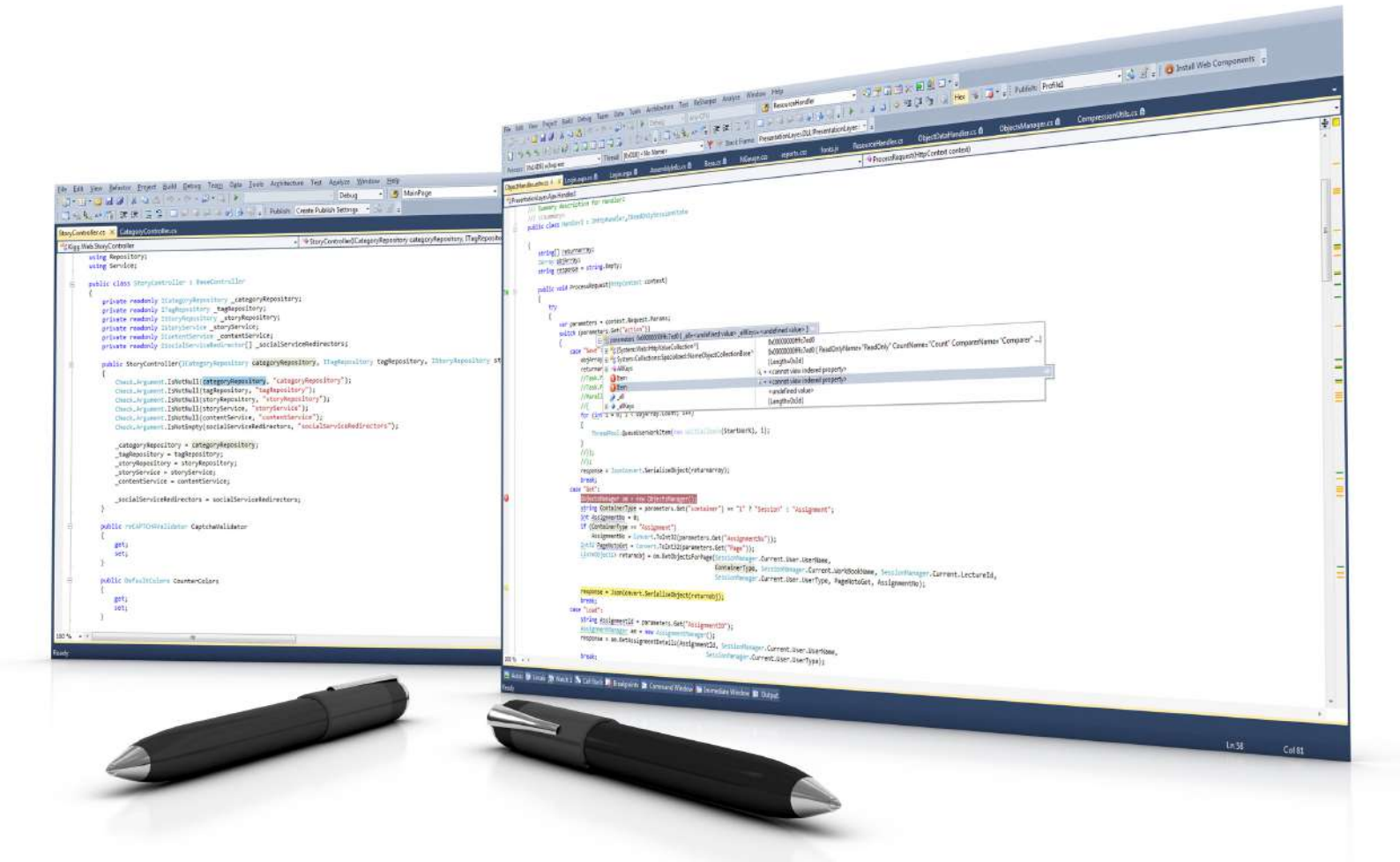
Co-funded by the
Erasmus+ Programme
of the European Union

Toshkent axborot texnologiyalari universiteti

KOMPYUTERDA DASTURLASH 2



609564-EPP-1-2019-1-EL-EPPKA2-CBHE-JP
www.mechauz.uz

KOMPYUTERDA DASTURLASH 2

==========

COMPUTER PROGRAMMING 2

==========

КОМПЬЮТЕРНОЕ ПРОГРАММИРОВАНИЕ 2

«Modernization of Mechatronics and Robotics for Bachelor degree in Uzbekistan through Innovative Ideas and Digital Technology (MechaUz)»

TOSHKENT - 2023

The publication is made on the basis of materials of the international project «Modernization of Mechatronics and Robotics for Bachelor degree in Uzbekistan through Innovative Ideas and Digital Technology (MechaUz)» and in the framework of the Erasmus+ program.

The purpose of teaching science is to learn the basic principles of information collection, storage and processing, transmission through the C++ programming language. Also, to create modern information systems and study the methods and technologies of their creation.

The essence of computer programming 2 subject, its main principles and tasks is to develop students' algorithmic knowledge. To achieve this, modern programming language (C++) capabilities are used.

The task of science is to its students:

- Working with unknown types and unknown namespaces;
- Containers (Collections) and their capabilities;
- Associative and unordered associative containers and their use;
- Using standard algorithms and iterators;
- Working with numerical classes;
- Syntactic analysis;
- Creation of complex sorting algorithms;
- programming in Visual C++ environment;
- Working with components and dialogue windows;
- graphic capabilities of Visual C++;
- the capabilities of OLE, MFC technologies;
- Working with multi-window environments;
- Creating small projects;
- It consists of teaching user interface design.

This project has been funded with support from the European Commission.

This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

© Tashkent University of Information Technologies

Authors: Z.Sh. Abdullayeva, X.E. Xujamatov

Tashkent 2023

Mundarija mazmuni

Ma’ruza mavzulari:

№	Ma`ruzalar	Ajratilgan soat
1	**Noma`lum tiplar va noma`lum nomlar fazosi.** Tiplarni dinamik tarzda aniqlash. Tiplarni almashtirish operatorlari. Yangi nomlar fazosi yaratish. Buferlashtirilgan kiritish va chiqarish.	30
2	**Konteynerlar (Kolleksiyalar).** STL kutubxonalari. Konteyner sinflar.	
3	Chiziqli konteynerlar (array, vector, deque, list, forward_list).	
4	**Assosiativ va tartiblanmagan assosiativ konteynerlar.** Assosiativ konteynerlar (set, map, multiset, multimap).	
5	Tartiblanmagan assosiativ konteynerlar (unordered_set, unordered_map, unordered_multiset, unordered_multimap).	
6	**Konteynerlar adapterlari.** Stack, queue, priority_queue. Konteynerlar bilan ishlash algoritmlari. Funktorlarning qo`llanilishi.	
7	**Standart algoritmlar va iteratorlar.** Standart algoritmlar.	
8	Iteratorlar va ularning qo`llanilishi. Xotirani taqsimlovchilar va ularga qo`yilgan talablar. Standart bo`yicha hotira taqsimlovchi.	
9	**Sonli sinflar bilan ishlash.** Sonli sinflar va ular bilan ishlash (complex, vallaray, slice, gslice va h.k).	
10	**Murakkab saralash algoritmlari.** Amaliy dasturlash. Murakkab saralash algoritmlari. Juda ham katta sonlar bilan ishlash. Large sinfining qo`llanilishi.	
11	**Visual C++ muhitida dasturlash.** Visual C++ muhitida dasturlash. Visual C++ muhitida menyular va uskunalar paneli.	
12	**Komponentalar bilan ishlash.** Komponenta tushunchasi va xususiyatlari. Standart, Additional, System komponentalari.	
13	**Muloqot oynalari bilan ishlash.** Visual C++ muhitida muloqot oynalari va ularni sozlash, boshqarish elementlari.	
14	**Visual C++ning grafik imkoniyatlari.** Visual C++ning grafik imkoniyatlari (grafik holat, tasvirlarni va funksiya grafiklarini qurish). Chart komponentlari.	
15	**Kichik loyihalarni yaratish.** Visual C++ muhitida turli sohaga oid masalalarni yechish uchun (mini) kichik loyihalarni loyihalash usullari. Kichik loyihalar algorimlarini yozish usullari. Visual C++ muhitida hisobot shaklini tayyorlash va chop etish elementlari. Foydalanuvchi interfeysini loyihalash va testlash usullari.	

Amaliy mashg'ulot mavzulari:

№	Amaliy mashg'ulot mavzulari	Ajratilgan soat
1	**Noma`lum tiplar va noma`lum nomlar fazosi.** Tiplarni dinamik tarzda aniqlash. Tiplarni almashtirish operatorlari. Yangi nomlar fazosi yaratish. Buferlashtirilgan kiritish va chiqarish	44
2	**Konteynerlar (Kolleksiyalar).** STL kutubxonalari.	
3	**Konteyner sinflar**. Chiziqli konteynerlar (array, vector, deque).	
4	**Konteynerlar (Kolleksiyalar).** Chiziqli konteynerlar (list, forward_list)	
5	**Assosiativ va tartiblanmagan assosiativ konteynerlar.** Assosiativ konteynerlar (set, multiset) Tartiblanmagan assosiativ konteynerlar (unordered_set, unordered_multiset).	
6	**Assosiativ va tartiblanmagan assosiativ konteynerlar.** Assosiativ konteynerlar (map, multimap).	
7	**Tartiblanmagan assosiativ konteynerlar** (unordered_map, unordered_multimap).	
8	**Konteynerlar adapterlari.** Stack, queue, priority_queue	
9	**Konteynerlar adapterlari.** Konteynerlar bilan ishlash algoritmlari. Funktorlarning qo`llanilishi	
10	**Standart algoritmlar va iteratorlar.** Standart algoritmlar. Iteratorlar va ularning qo`llanilishi	
11	**Standart algoritmlar va iteratorlar.** Xotirani taqsimlovchilar va ularga qo`yilgan talablar. Standart bo`yicha taqsimlovchi	
12	**Sonli sinflar bilan ishlash.** Complex sonlar.	
13	**Sonli sinflar** va ular bilan ishlash (vallaray, slice, gslice va h.k)	
14	**Murakkab saralash algoritmlari. Amaliy dasturlash.** Murakkab saralash algoritmlari.	
15	**Large sinfining qo`llanilishi.** Juda ham katta sonlar bilan ishlash.	
16	**Visual C++ muhitida dasturlash.** Visual C++ muhitida menyular va uskunalar paneli.	
17	**Komponentalar bilan ishlash.** Komponenta tushunchasi va xususiyatlari.	
18	**Standart, Additional, System komponentalari.**	
19	**Muloqot oynalari bilan ishlash.** Visual C++ muhitida muloqot oynalari va ularni sozlash, boshqarish elementlari.	
20	**Visual C++ning grafik imkoniyatlari.** Visual C++ning grafik imkoniyatlari (grafik holat, tasvirlarni va funksiya grafiklarini qurish). Chart komponentlari.	

21	**Kichik loyihalarni yaratish.** Visual C++ muhitida turli sohaga oid masalalarni yechish uchun (mini) kichik loyihalarni loyihalash usullari. Kichik loyihalar algorimlarini yozish usullari Visual C++ muhitida hisobot shaklini tayyorlash va chop etish elementlari.	
22	**Foydalanuvchi interfeysini loyihalash.** Foydalanuvchi interfeysini loyihalash va testlash usullari.	

Mustaqil ish mavzulari:

№	Mustaqil ta’lim mavzulari	Ajratilgan soat
1	Turlarning dinamik tarifi.	106
2	Turlarni almashtirish operatorlari.	
3	Yangi nomlar fazosi uchun joy yaratish.	
4	Buferlashtitilgan KIRISH va chiqish.	
5	STL kutubxonalar.	
6	Konteyner sinflar.	
7	Chiziqli konteynerlar (array, vector, deque, list, forward_list).	
8	Assotsiyativ konteynerlar (set, map, multiset, multimap).	
9	Tartiblanmagan assotsiyativ konteynerlar (unordered_set, unordered_map, unordered_multiset, unordered_multimap).	
10	Stack, queue, priority_queue.	
11	Konteynerlar bilan ishlaydigan algoritmlar.	
12	Funktorlarni qo'llash.	
13	Standart algoritmlar.	
14	Iteratorlar va ularni qo'llash.	
15	Xotirani taqsimlash va unga qo'yiladigan talablar.	
16	Standart taqsimlash.	
17	Raqamli sinflar va ular bilan ishlash (complex, vallaray, slice, gslice va b.sh.).	
18	Murakkab saralash algoritmlari.	
19	Juda katta raqamlar bilan ishlash.	
20	Large sinfini qo’llash.	
21	Visual C++ muhitida dasturlash.	
22	Visual C++ muhitida menyular va uskunalar paneli.	
23	Komponenta tushunchasi va xususiyatlari.	
24	Standart, Additional, System komponentalari.	
25	Visual C++ muhitida muloqot oynalari va ularni sozlash, boshqarish elementlari.	
26	Visual C++ning grafik imkoniyatlari (grafik holat, tasvirlarni va funksiya grafiklarini qurish	
27	Chart komponentalari.	
28	Visual C++ muhitida ko`p oynali muhitni yaratish	

29	Visual C++ muhitida turli sohaga oid masalalarni yechish uchun (mini) kichik loyihalarni loyihalash usullari.	
30	Kichik loyihalar algorimlarini yozish usullari.	

Yuklama

Faoliyat	**Soatlari**
Ma’ruza	30
Amaliy mashg’ulot	44
Mustaqil ish	106
Jami:	**180**

Ta’lim strategiyasi

Dasturlash kursini o‘qitish ta’limning kredit tizimi asosida ma’ruza, amaliy mashg‘ulotlari, videoma’ruzalar, taqdimotlar, hamda mavzu bo‘yicha vazifalar va mustaqil topshiriqlarni o‘z ichiga oladi. Ma’ruza, amaliy ishlariga oid o‘quv materiallarida ko‘rsatilgan mavzular bo‘yicha nazariy va amaliy ma’lumotlar beriladi, amaliy ishlarini bajarish va natijalarni hisoblash tartibi tushuntiriladi. Kurs bo‘yicha qo‘yilgan o‘quv materiallari talabalar tomonidan mustaqil o‘rganiladi, testlar, amaliy ishlari talabalar tomonidan individual tarzda bajariladi.

Mustaqil o’qish

Dasturlash fanini o’rganuvchi talabalar auditoriyada olgan nazariy bilimlarini mustahkamlash va iqtisodiyot sohasidagi amaliy masalalarni yechishda ko’nikma hosil qilish uchun mustaqil ta’lim tizimiga asoslanib, kafedra o’qituvchilari rahbarligida mustaqil ish bajaradilar.

Auditoriya ishlari

Talabaning dars davomida o‘rganilayotgan tushunchalaridagi muammolarni hal qilishda bir qator qiyinchilaklar paydo bo‘ladi. Har bir bo‘lim uchun savolar bloki mavjud.

O‘quv materiallari

Talabalar ushbu hujjatdan quyidagi o‘quv materiallarini olishlari kerak:

- o‘quv qo‘llanmada talabalar uchun o‘quv shartlari va bilimlarini baholashga oid ma’lumotlar mavjud;
- kursning har bir mavzusiga oid slaydlar taqdim etiladi;
- har bir mavzu uchun muammoli vaziyatlar.

Amaliyot ishlari quyidagi ko‘rinishda:

- Ishdan maqsad;
- o‘quv materiallari va bajarish uskunalari;
- vazifalar.

Talabalar bilimini baholash

Talabalar bilimini baholash semestr va yakuniy nazorat davomida o‘qitish materiallarini o‘zlashtirish ko‘rsatkichi (test, topshiriq va yozma ish natijasi)ga asoslangan.

Joriy oraliq va yakuniy nazorat ballari quyidagicha taqsimlanadi:

<table>
<tr><td>Baholash usullari</td><td colspan="3">Onlayn testlar, yozma ishlar, og‘zaki so‘rov, amaliy topshiriqlar prezentatsiyalar va h.k.</td></tr>
<tr><td rowspan="4">Baholash mezonlari</td><td colspan="3">90-100 ball «a’lo»
Fan bo‘yicha xulosa va qaror qabul qilish. Fan yuzasidan ijodiy fikrlay va mustaqil mushohada yurita olish.
Fan bo‘yicha olgan bilimlarini amalda qo‘llay olishi va mohiyatini tushuntirib bera olish. Bilish, aytib berish. Tasavvurga ega bo‘lish.</td></tr>
<tr><td colspan="3">70-89 ball «yaxshi»
Fanini mustaqil mushohada qilish.
Fan bo‘yicha olgan bilimlarini amalda qo‘llay olish. Mohiyatini tushuntirish.
Bilish, aytib berish. Tasavvurga ega bo‘lish.</td></tr>
<tr><td colspan="3">60-69 ball «qoniqarli»
Fan mohiyatini tushuntirish.
Bilish, aytib berish.
Fan haqida tasavvurga ega bo‘lish.</td></tr>
<tr><td colspan="3">0-60 ball «qoniqarsiz»
Fan haqida aniq tasavvurga ega bo‘lmaslik. Bilmaslik.</td></tr>
<tr><td rowspan="7">Reyting baholash turlari</td><td>Joriy nazorat</td><td>Maksimal ball (30)</td><td>O‘tkazish vaqti</td></tr>
<tr><td>6 ta amaliy ishlarini bajarish va hisobot qilish</td><td>5</td><td>O‘quv jarayoni grafigi bo‘yicha</td></tr>
<tr><td>Oraliq nazorat</td><td>Maksimal ball (20)</td><td>O‘tkazish vaqti</td></tr>
<tr><td>6 ta amaliy ishlarini bajarish va hisobot qilish</td><td>5</td><td>O‘quv jarayoni grafigi bo‘yicha</td></tr>
<tr><td>Yakuniy nazorat</td><td>50</td><td>Semestr oxirida</td></tr>
<tr><td>Fan bo’yicha jami</td><td>100</td><td></td></tr>
</table>

Baholash (assessment):

O‘qishni baholash talabalarning butun kurs davomidagi ishtiroki, shuningdek, nazariya va laboratoriya bo‘yicha yakuniy test natijasiga ko‘ra amalga oshiriladi. Baholashning foiz nisbatida quyidagi ko‘rinishda taqsimlanadi:

- Ma’ruza mashg‘ulotlar: 20%
- Yakuniy test: 30%
- Laboratoriya: 30%
- Mustaqil ta’lim 20%

Nazariya bo‘yicha baholash:

Talabalarning bilimlarini baholash uchun ikkita dars mavjud. Nazariy imtihon shaxsan o‘qituvchi tomonidan belgilangan sana, vaqt va joyda o‘tkaziladi va unda talabalarning egallagan bilimlari, hamda shu egallangan bilim, ko‘nikma, tajriba asosida talabalarning muammolarni echish qobiliyatlari baholanadi. Imtihon bahosi kursdagi yakuniy bahoning 30% ini tashkil qiladi.

Laboratoriya bo‘yicha baholash:

Baho har bir laboratoriya ishi uchun va laboratoriya yuzasidan o‘tkaziladigan yakuniy imtihon bo‘yicha qo‘yiladi. Bunday mashg‘ulot laboratoriya mashg‘uloti kabi tashkil etilib, yakuniy mashg‘ulot ham laboratoriya mashg‘uloti tarzida o‘tkaziladi. Laboratoriya ishini baholash (30% tayyorgarlik, 70% ishlab chiqish) laboratoriya uchun bahoning 50%ini tashkil qiladi.

CHora ko‘rish:

Har bir uy ishi uchun ma’lum bir sana belgilanadi. Belgilangan vaqtda topshirmagan talabalarning uy ishi uchun belgilangan bahosi pasaytiriladi.

II SEMESTR UCHUN
1-MA'RUZA
NOMA'LUM TIPLAR VA NOMA'LUM NOMLAR FAZOSI (2 SOAT)

Reja:

1. Tiplarni dinamik tarzda aniqlash.
2. Tiplarni almashtirish operatorlari.
3. Yangi nomlar fazosi yaratish.
4. Buferlashtirilgan kiritish va chiqarish.

Annotatsiya*: Noma'lum tiplar va noma'lum fazolar uchun dinamik tipni aniqlash va tpini almashtirish operatorlari zamonaviy ob'ektga yo'naltirilgan dasturlashni qo'llab-quvvatlovchi C++ dasturlash tilining ikkita vositalari: dinamik tipni aniqlash va qo'shimcha tip quyish operatorlari bilan bog'liq. Tiplarni dinamik tarzda aniqlashning ba'zi usullari, typeid operatori, typeinfo kutubxonasi, type_info sinfi a'zolaridan foydalanish, dastur bajarilish davomida ko'rsatkichli va oddiy o'zgaruvchilarning tiplarini almashtrish operatorlari dynamic_cast, const_cast, static_cast, reinterpret_cast operatorlaridan foydalanish, noma'lum va foydalanuvchining nomlar fazolarini yaratish va dasturda ulardan foydalanish usullari ko'rsatilgan.*

Kalit so'zlar: *Tipni aniqlash, tipni almashtrish, nomlar fazosi, buferlashtirilgan KIRISH va chiqish, typeid, typeinfo, type_info sinfi, dynamic_cast, const_cast, static_cast, reinterpret_cast, namespace, istrstream, oqim, formatlash identifikatori, monupulyatorli formatlash.*

KIRISH

Noma'lum tiplar va noma'lum fazolar uchun dinamik tipni aniqlash va tipini almashtirish operatorlari zamonaviy ob'yektga yo'naltirilgan dasturlashni qo'llab- quvvatlovchi C++ dasturlash tilining ikkita vositalari: dinamik tipni aniqlash (runtime type identification - RTTI) va qo'shimcha tip quyish operatorlari bilan bog'liq. Bu vositalarning biri C++ spesifikatsiyasi qismi emasdi, lekin har ikki runtime polimorfizm uchun qo'llab-quvvatlash oshirish uchun qo'shilgan. RTTI dastur bajarish davomida ob'yekt turini aniqlash imkoniyati muvjudligini bildiradi. Bunda muhokama qilinayotgan tip turi tashlash operatorlari - bu amalni bajarish uchun dasturchiga xavfsiz yo'llarini taklif qiladi. Ulardan biri, dynamic_cast, bevosita RTTI aniqlash bilan bog'liq.

Tiplarni dinamik tarzda aniqlash. Ushbu vosita polimorf bo'lmagan tillarda (masalan, C) mavjud emasligi sababli dinamik turdagi identifikatsiyalash bilan tanish bo'lmasligigiz mumkin, chunki har bir ob'yektning tipi kompilyatsiya vaqtida (ya'ni, dasturni bajarishda) ma'lum bo'lgani uchun dastur bajarilishida ma'lumot olishning hojati yo'q. Biroq, C++ kabi polimorf tillarda ob'yekt tipi kompilyatsiya vaqtida noma'lum bo'lgan holatlar bo'lishi mumkin, chunki dastur bajarila boshlagunga qadar bu ob'yektning aniq tipi (tiplari) aniqlanmaydi. Ma'lumki, C++ sinf ierarxiyasi, virtual funksiyalar va ko'rsatkichlarni (pointer) asos sinflarga ishlatib polimorfizmni amalga oshiradi. Tayanch sinfga ko'rsatkich bu tayanch sinf a'zolariga yoki undan olingan har qanday ob'yekt a'zolariga
murojaat qilish uchun foydalanish mumkin. Shuning uchun asos sinfga ko'rsatgichning istalgan vaqtda qanday turdagi ob'yektga murojaat qilishi har doim ham oldindan ma'lum emas. Bu faqat dastur ishlayotganida — dinamik turdagi identifikatsiya vositalaridan biri yordamida aniqlanadi.

Dastur bajarilishi davomida ob'yekt turini olish uchun *typeid* operatoridan foydalaniladi.

Dastur bajarilayotgan vaqtda ob‘yekt tipini olish uchun <typeinfo> kutubxonasini kiritish va *typeid* operatoridan foydalanish mumkin. Typeid operatoridan foydalanishning eng keng tarqalgan yozilish sintaktisi quyidagicha:

```
typeid(object)
```

Bunda, object element tipini olish uchun ob‘yekt anglatadi. Shuningdek, nafaqat ichki tipini, balki dasturchi tomonidan yaratilgan sinf tipini ham aniqlash mumkin. Buning uchun typeid operatori ob‘yekt tipini tasvirlaydigan type_info ob‘yektiga murojaat qiladi.

type_info sinfda public modifikatori bilan quyidagi statik operatorlar aniqlangan bo‘ldi.

```
bool operator == (const type_info &ob);
 bool operator !=(const type_info &ob);
bool before(const type_info &ob);
const har *name();
```

Bunda qayta aniqlangan (overloaded) operatorlari [==] va [!=] tiplarini taqqoslash uchun ishlatiladi. Chaqiruvchi ob‘yekt parametr sifatida ishlatiladigan ob‘yekt (element ob) oldida ketma-ketlik tartibida bo‘lsa before() funksiyasi true qaytaradi. Bu funksiya asosan ichki foydalanish uchun mo‘ljallangan. Uning natija qiymati merosxo‘r yoki sinf iyerarxik bilan hech qanday aloqasi yo‘q. name() funksiyasi tip nomiga ko‘rsatkich qaytadi.

typeid operatoridan foydalanishga bir misol keltiramiz.

1.1-dastur. Tiplarni taqqoslash[1]

```
#include "stdafx.h"
#include <iostream>
#include <typeinfo>
using namespace std;
class myclass {
// . . .
};
int main(){ int
i, j; float f;
myclass ob;
cout << "i o‘zgaruvchi tipi: " << typeid(i).name()<<endl; cout << "f
o‘zgaruvchi tipi: " << typeid(f).name()<<endl;
cout << "ob o‘zgaruvchi tipi: " << typeid(ob).name()<<endl<<endl; if(typeid(i) == typeid(j))
        cout << " i va j tiplar bir xil.\n"; if(typeid(i) !=
typeid(f))
        cout << " i va f tiplar bir xil emas.\n"; system("PAUSE");
return 0;
}
```

1.1 – dastur natijasi. Output
i o‘zgaruvchi tipi: int f o‘zgaruvchi tipi: float ob o‘zgaruvchi tipi: class myclass i va j tiplar bir xil. i va f tiplar bir xil emas

1.1-dastur tahlili. Dasturda int tipiga oid i va j, float tipida f, myclass tipida ob o‘zgaruvchilari aniqlangan. typeinfo kutubxonasining typeid().name() metodi bilan ularning tiplari aniqlangan. Taqqsolash operatorlari bilan o‘zraguvchilarning tiplari tekshirilgan va natijalar ekranga chiqarilgan. Bu esa tiplar bo‘yicha dastlabki ishlov berish uchun lozim.[1]

1.1-jadval. Type_info sinf a‘zolari va vazifalari

№	Funksiya nomi	vazifasi
1	(constructor)	Sinfda standart va nusxa konstruktori mavjud emas.
2	(destructor) [virtual]	xavfsiz tayanch sinf uchun pointer orqali ob‘yektni o‘chirish mumkin (virtual funksiyasi)
3	operator=	O‘zlashtirish amalidan foydalanib ko‘chirib bo‘lmaydi
4	operator== operator!=	Ob‘yektlar bir xil tipgaligini taqqoslashydi (public funksiya)
5	before	Joriy tipdagi boshqa type_index ob‘yekt tomonidan nusxasi tipini bor yoki yo‘qligini tekshiradi. Ob‘yektlar, aniqrog‘i tiplari, amalga oshirilishi belgilangan tartibda tashkil etiladi. (public funksiya)
6	hash_code	Joriy tipga mos keladigan tiplarni qaytaradi (public funksiya)
7	name	Joriy tipning aniqlangan nomini qaytaradi. (public funksiya)

typeid operatori polimorf tayanch sinf uchun ko‘rsatkich sifatida qo‘llaniladigan bo‘lsa (eslabtib o‘tamiz, polimorf sinf kamida bitta virtual funksiyani o‘z ichiga olgan sinfga aytiladi), u avtomatik ravishda real ob‘yekt tipini qaytaradi va u asossidan olingan sinf tayanch sinf ob‘yekt yoki ob‘yekt emasligini ham ko‘rsatadi.

Shuning uchun typeid operatoridan asos sinfga ko‘rsatgich bilan murojaat qilingan ob‘yekt tipini dinamik ravishda aniqlash uchun foydalanish mumkin. Bu xususiyat quyidagi 1.2. dasturda ko‘rsatilgan.

1.2-dastur tahlili. Dasturda dinamik ko‘rsatkichli Basetipli p o‘zgaruvchisi yaratilgan va 3 ta, base tipiga baseob, myClass_one tipiga ob1, myClass_two tipiga ob2 mos o‘zgaruvchilar yaratilgan. Dastur davomida dinamik o‘zgaruvchiga

[1] Дастур Visual C++ муҳитида тузилган.

oddatiy o‘zgaruvchilar qiymat qilib berilgan. Natija shu ma‘lum bo‘ldiki dinamik o‘zgaruvchining tipi danamik ko‘rinishda o‘zgardi.

typeid operatori polimorf turdagi tayanch sinfga ko‘rsatgich sifatida qo‘llanilsa, haqiqiy manzillangan ob‘yektning tipini dastur bajarilayotgan vaqtda aniqlaydi.

typeid operatori polimorf bo‘lmagan iyerarxik sinflar uchun ko‘rsatkich qo‘llaniladigan barcha hollarda, tayanch tipi uchun ko‘rsatikich bo‘ladi. Boshqa so‘zlar bilan aytganda, bu ko‘rsatkich aslida tipni aniqlash mumkin emas. Tajriba sifatida 1.2-dasturdagi asos sinfdagi f() virtual funksiyani izohga oling va natijani tekshirib ko‘ring. Dasturda bu o‘zgarishni amalga oshirgandan so‘ng har bir turli tipli ob‘yektning tipi base deb aniqlanishini ko‘rasiz, chunki bu p ko‘rsatgichga ega bo‘lgan tipdir.

typeid operatori odatda kengqirrali ko‘rsatkich (dereferenced pointer ya‘ni, [*] operatoriga ega bo‘lgan ko‘rsatkich) uchun qo‘llanilganligi sababli, bu bu ko‘rsatkich null bo‘lib chiqqanda vaziyatni boshqarish uchun maxsus istisno yaratiladi. Bu holda, typeid operatori bad_typeid istisnoga o‘tadi.

1.2-dastur. typeid operatorini polimorf sinflar ierarxiyasiga qo‘llanilishi.

```
#include "stdafx.h"
#include <iostream>
#include <typeinfo>
using namespace std;
class Base {
virtual void f() {}; // Asos sinf uchun polimorf
// . . .
};
class myClass_one: public Base {
// . . .
};
class myClass_two: public Base {
// ...
};
int main(){
Base  *p, baseob;
myClass_one b1;
myClass_two b2;
p = &baseob;
 cout << " O'zgaruvchi p tipdagi obyektga ishora: ";
 cout << typeid(*p).name() << endl;
 p = &ob1;
 cout << " O'zgaruvchi p tipdagi obyektga ishora: ";
 cout << typeid(*p).name() << endl;
 p = &ob2;
```

```
cout << " O'zgaruvchi p tipdagi obyektga ishora: ";
cout << typeid(*p).name() << endl;
system("pause");
return 0;}
```

1.2 – dastur natijasi. Output

```
O'zgaruvchi p tipdagi obyektga ishora: class Base O'zgaruvchi p tipdagi
obyektga ishora: class myClass_one
O'zgaruvchi p tipdagi obyektga ishora: class myClass_two
```

Polimorf sinflar ierarxiyasidagi ob'yektlarga havolalar ko'rsatgichlar kabi ishlaydi. Typeid operatori polimorf sinfning mos yozuvlar uchun qo'llaniladigan bo'lsa, u aslida ob'yekt tipini qaytaradi va u asosiy tipi o'rniga olingan tipdagi ob'yekt bo'lishi mumkin. Bu usul o'zgaruvchilarni funksiyalarga murojaat qilib o'tishda eng ko'p qo'llaniladi. Masalan, quyidagi (1.3-dastur) dasturda WhatType() funksiyasi asosiy tipdagi ob'yektlarga mos yozuvlar parametrini e'lon qiladi. Demak, WhatType() funksiyasi bazaviy turdagi ob'yektlarga murojaatlarni yoki asos sinfdan olingan har qanday sinf ob'yektlariga murojaatlarni keltirib chiqaradi. Bunday parametrlarga qo'llaniladigan typeid operatori funksiyaga o'tgan ob'yektning haqiqiy turini qaytaradi.

1.3-dastur. typeid operatorini havolali parametriga qo'llash.

```
#include "stdafx.h" #include <iostream> #include <typeinfo>
using namespace std;
class Base {
virtual void f() {}; // Asos sinf uchun polimorf
// . . .
};
class myClass_one: public Base {
// . . .
};
class myClass_two: public Base {
// ...
};
void WhatType(Base &ob){
cout << " ob parametri ob'ekt tipiga havolasi: "; cout << typeid(ob).name() << endl;
}
int main(){ Base baseob;
myClass_one ob1; myClass_two ob2; WhatType(baseob); WhatType(ob1); WhatType(ob2);
system("pause"); return 0;
}
```

1.3 – dastur natijasi. Output

```
ob parametri ob'ekt tipiga havolasi: class Base
ob parametri ob'ekt tipiga havolasi: class myClass_one ob parametri
ob'ekt tipiga havolasi: class myClass_two
```

1.3-dastur tahlili. Dasturda WhatType funksiyasi yaratilib, unga asos sinf tipidpgi havolali parametr belgilandi. Asosiy funksiyada 3 ta, base tipiga baseob, myClass_one tipiga ob1, myClass_two tipiga ob2 mos o‘zgaruvchilar yaratilgan. Dastur davomida WhatType funksiyasining havolali parametrga oddatiy o‘zgaruvchilar qiymat qilib berilgan. Natija shu ma‘lum bo‘ldiki, dinamik o‘zgaruvchining tipi danamik ko‘rinishda o‘zgardi.

Dasturlash jarayonlarida o‘zgaruvchilarni tiplarini real tip bilan taqqoslashga to‘g‘ri keladi. Buning uchun typeid operatori agrgument sifatida tipi nomini olish yordamida amalga oshiriladi. Uning yozilishi quyidagicha:

```
tyreid(tip  nomi)
```

Masalan, quyidagi listing amal qiladi.

```
cout<<typeid(int).name();
```

typeid operatorini ushbu variantda ishlatilishining maqsadi tipni taqqoslash uchun foydalanishdir, (belgilangan ma‘lumotlar tipini ta‘riflaydi) ammo, type_info ob‘yektini olish uchun emas.

Dinamik tipni aniqlashni (RTTI) qo‘llash. Dasturda dinamik tipni aniqlash (identifikatsiya) vositasi (RTTI) imkoniyati va foydali bo‘lishi mumkinlingini ko‘rsatib o‘tamiz. Masalan, geometrik shakllar uchun iyerarxik sinf berilgan bo‘lsin, unda doira, uchburchak va to‘rtburchak maydoni hisoblash usullari mavjud. Ushbu dasturda doira, uchburchak yoki to‘rtburchakni yaratish uchun mo‘ljallangan funksiyasi factory() bo‘lsin. Ushbu funksiya yaratilgan ob‘yektga bir nusxani qaytadi. Ob‘yektlarni hosil qiluvchi funksiya ba‘zan ob‘yekt fabrikasi deb ataladi. Yaratilayotgan ob‘yektning o‘ziga xos tipi C++ tasodifiy sonlar generatorining Rand() funksiyasiga KIRISH orqali aniqlanadi. Demak, qanday turdagi ob‘yekt hosil bo‘lishini oldindan bilolmaymiz. 1.4-dasturda o‘nta ob‘yekt yaratadi va har bir turdagi yaratilgan ob‘yektlar sonini hisoblaydi. factory() funksiyasini chaqirishda har qanday shakl turini hosil qilish mumkinligi sababli, dastur aslida yaratilgan ob‘yekt tipini aniqlash uchun typeid operatoridan foydalanadi.

1.4-dastur. Dinamik tipdagi identifikatsiya vositasidan foydalanish.

```
#include "stdafx.h" #include <iostream> #include <typeinfo> #include <cstdlib> using
namespace std; class figure {
protected:
        double x, y; public:
        figure(double i, double j) { x = i;
                y = j;
        }
virtual double area() = 0;
};
class triangle : public figure { public:
        triangle(double i, double j) : figure(i, j) {} double area() {
                return x * 0.5 * y;
```

```
        }
};
class rectangle : public figure { public:
        rectangle(double i, double j) : figure (i, j) {} double area() { return x * y;}
};
class circle : public figure { public:
        circle(double i, double j=0) : figure(i, j) {} double area() {return 3.14 * x * x;}
};
// figure sinf ob‘yektlarini yaratish. Figure
```

```
*factory(){
        switch(rand() % 3 ) {
        case 0: return new circle (rand()%15);
        case 1: return new triangle (rand()%15,rand()%15); case 2: return new rectangle
        (rand()%15, rand()%15);
        }
return 0;
};
int main(){
        figure *p; // asos sinfga ko‘rsatkich int i;
        int t=0, r=0, c=0;
                // 5 ta ob‘yektlarni hosil qilsh va sanash for(i=0; i<15; i++) {
                p = factory(); // ob‘yektni hosil qilish
                cout << "Ob‘yektning tipi: " << typeid(*p).name(); cout << ". ";
                // ob‘yektni tekshirish orqali sanasj if(typeid(*p) == typeid(triangle)) t++;
                if(typeid(*p) == typeid(rectangle)) r++; if(typeid(*p) == typeid(circle)) c++;
                // Yuzasini hisoblash
                cout << " S= " << p->area() << endl;
                }
cout << endl;
cout << "Quyidagi ob‘yektlar hosil qilindi:\n"; cout << " Uchburchaklar: " << t << endl;
cout << " To‘rtburchaklar: " << r << endl; cout << " Doiralar: " << c << endl; system("pause");
return 0;
}
```

1.4 – dastur natijasi. Output

```
Ob‘yektning tipi: class rectangle.  S= 8 Ob‘yektning
tipi: class triangle.  S= 28 Ob‘yektning tipi: class circle.
S= 28.26 Ob‘yektning tipi: class triangle.  S= 35
Ob‘yektning tipi: class rectangle.  S= 12 Ob‘yektning
tipi: class triangle.  S= 55 Ob‘yektning tipi: class
rectangle.  S= 72 Ob‘yektning tipi: class circle.  S=
254.34 Ob‘yektning tipi: class rectangle.  S= 21
Ob‘yektning tipi: class triangle.  S= 33 Ob‘yektning
tipi: class rectangle.  S= 10 Ob‘yektning tipi: class
circle.  S= 379.94 Ob‘yektning tipi: class circle.  S=
254.34 Ob‘yektning tipi: class triangle.  S= 4
Ob‘yektning tipi: class triangle.  S= 36

Quyidagi ob‘yektlar hosil qilindi:
 Uchburchaklar: 6
 To‘rtburchaklar: 5
 Doiralar: 4
```

tipeid operatorini shablon sinflariga qo‘llash. typeid operatorini shablon sinflariga ham qo‘llash mumkin albatta. Shablon sinfi asosida hosil bo‘lgan ob‘yekt tipi ob‘yektni amalga oshirishda uning umumlashgan ma‘lumotlari uchun qanday ma‘lumotlardan foydalanilganligi asosida qisman aniqlanadi. Shuning uchun, turli ma‘lumotlar yordamida yaratilgan bir xil shablon sinfga tegishli bo‘lgan ikki ob‘yektlar tip xil bo‘ladi.

1.5-dastur. Shablon sinflari bilan typeid operatoridan foydalanish.

```
#include "stdafx.h" #include <iostream> #include <typeinfo> using namespace std; template
<class T> class myclass {
        T a; public:
                myclass(T i) { a = i; }
        // . . .
};
int main(){
        myclass<int> o1(28), o2(1); myclass<double> o3(19.80); cout << " o1 ob‘yekt tipi: ";
        cout << typeid(o1).name() << endl; cout << " o2 ob‘yekt tipi: ";
        cout << typeid(o2).name() << endl; cout << " o3 ob‘yekt tipi: ";
        cout << typeid(o3).name() << endl; cout << endl;
        if(typeid(o1) == typeid(o2))
                cout << " o1 va o2 ob‘yektlar bir xil tip.\n"; if(typeid(o1) == typeid(o3)) cout <<
        "Xatolik\n";
        else cout << "o1 va o3 ob‘yektlar bir xil tip emas.\n"; system("pause");
return 0;
}
```

1.5 – dastur natijasi. Output

```
o1 ob‘yekt tipi: class myclass<int>
o2 ob‘yekt tipi: class myclass<int>

o3 ob‘yekt tipi: class myclass<double>
o1 va o2 ob‘yektlar bir xil tip.
o1 va o3 ob‘yektlar bir xil tip emas
```

1.5-dasturdan ko‘rinib turibdiki, ikki ob‘yektlar bir xil shablon sinf ob‘yektlarida bo‘lsa-da, ularning parametrli ma‘lumotlar mos bo‘lmasa, ular turi teng emas. Bu dasturda o1 ob‘yekti myclass <int > tipida, o3 ob‘yekti esa myclass <double>tipida bo‘ladi. Shunday qilib, bu ob‘yektlar turli xilligini ko‘rsatadi.

tipeid operatorini shablon sinflariga qo‘llashning yana bir misolini, ya‘ni 1.4-dastur geometrik shaklni aniqlash dasturining o‘zgartirilgan variantini ko‘rib chiqamiz. Bu safar figure sinfini shablon sinfiga aylantiriladi.

1.6-dastur. Figure ierarxiyasining shablon varianti.

```
#include "stdafx.h"
#include <iostream>
 #include <typeinfo>
#include <cstdlib>
using namespace std;
 template <class T>
class figure{
protected:
        T x, y; public:
(T i, T j) { x = i;
 y = j;
 }
 virtual T area() = 0;
 };
 template <class T>
riangle : public figure<T>{ public:
 triangle(T i, T j) : figure<T>(i, j) {} T area() {
 return x * 0.5 * y;
 }
 };
 template <class T>
ectangle : public figure<T>{ public:
 rectangle(T i, T j) : figure<T>(i, j) {} T area() {
 return x * y;
 }
 };
 template <class T>
ircle : public figure<T>{ public:
 circle(T i, T j=0) : figure<T>(i, j) {} T area() {
 return 3.14 * x * x;
 }
 };
```

```
<double> *generator(){ switch(rand() % 3 ) {
 case 0: return new circle<double> (rand() % 28);
 case 1: return new triangle<double>(rand() % 28, rand() % 28); case 2: return new
 rectangle<double> (rand() % 28, rand() % 28);
 }
 return 0;
 }
 int main(){
 figure<double> *p; int i;
 int t=0, c=0, r=0; for(i=0; i<15; i++) {
 p = generator();
 cout << "Ob‘yekt tipi: " << typeid(*p).name(); cout << ". ";
 if(typeid(*p) == typeid(triangle<double>)) t++; if(typeid(*p) == typeid(rectangle<double>)) r++;
 if(typeid(*p) == typeid(circle<double>)) c++; cout << " S= " << p->area() << endl;
 }
 cout << endl;
 cout << "Ob‘yektlarni sanash:\n";
 cout << " uchburshaklar: " << t << endl; cout << " o‘rtburchaklar: " << r << endl; cout << "
 doiralar: " << c << endl;
 system("pause");
 }
```

1.6 – dastur natijasi. Output

```
Ob‘yekt tipi: class rectangle<double>. S= 90 Ob‘yekt tipi: class triangle<double>. S= 136 Ob‘yekt
tipi: class circle<double>. S= 615.44 Ob‘yekt tipi: class triangle<double>. S= 210 Ob‘yekt tipi:
class rectangle<double>. S= 351 Ob‘yekt tipi: class triangle<double>. S= 202.5 Ob‘yekt tipi: class
rectangle<double>. S= 44 Ob‘yekt tipi: class circle<double>. S= 803.84 Ob‘yekt tipi: class
rectangle<double>. S= 156 Ob‘yekt tipi: class triangle<double>. S= 30 Ob‘yekt tipi: class
rectangle<double>. S= 225 Ob‘yekt tipi: class circle<double>. S= 706.5 Ob‘yekt tipi: class
circle<double>. S= 1384.74 Ob‘yekt tipi: class triangle<double>. S= 66.5 Ob‘yekt tipi: class
triangle<double>. S= 15


Ob‘yektlarni sanash: uchburshaklar: 6

o‘rtburchaklar: 5

doiralar: 4
```

Dinamik tipni aniqlash har bir dasturda ishlatilmaydi. Ammo, dastur bajarilish vaqtida, polimorf turlari bilan ishlashda, undagi ob‘yektlarning har qanday tipini aniqlashga imkonini beradi.

Tiplarni almashtirish operatorlari. C++ tilida beshta tipni almatirish operatorlari mavjud.

Ulardan birinchisi bu odatdagi (an‘anaviy) uslubda ishlatiladigan [()](cast operatori) azaldan C++ga qurilgan. U quyidagicha ishlatiladi:

```
float f = (float) 5;

bool b = (bool) 5;
```

Tipni almatirish to‘rtta operatori (dynamic_cast, const_cast, reinterpret_cast va static_cast) 10 yil oldinC++ tilga qo‘shildi. Bu operatorlar tiplar bilan amallarni bajarganda tur xil imkoniyatlarni yaratib beradi Ularning har birini alohida ko‘rib chiqaylik.

Dynamic_cast operatori. Dynamic_cast operatori dastur bajarilishi davomida polimorf turdagi quyish amalini bajaradi. Yangi operatorlardan eng muhimi dynamic_cast tipidagi tip almashtirish (kasting, casting) operatoridir. Dasturni bajarish davomida taklif qilinayotgan amalning bajarilishini tekshiradi. Agar belgilangan amal chaqirilganda yaroqsiz bo‘lsa, hech qanday turdagi kasting amalga oshirilmaydi. Dynamic_cast operatorini qo‘llashning umumiy formati quyidagicha:

```
dynamic_cast<type> (expr)
```

Bunda type elementi bu amalning maqsadi bo‘lgan yangi tipni, expr elementi esa bu yangi tipga almashadigan ifodani bildiradi. Tip turi ko‘rsatkich yoki mos yozuvlar bilan almashishi kerak va expr ifoda ko‘rsatkich yoki mos yozuvlar uchun tashlanishi kerak. Shu tarzda dynamic_cast operatoridan bir turdagi ko‘rsatgichni boshqa turdagi ko‘rsatgichga yoki bir turdagi murojaatni boshqa turdagi murojaatga aylantirish uchun foydalanish mumkin.

Bu operator asosan polimorf tiplar orasida dinamik tipni quyish amallari uchun ishlatiladi. masalan, agar polimorf sinflar B va D bo‘lsin. sinf D va sinf V olingach, dynamic_cast operatori yordamida, har doim ko‘rsatkichni Dga aylantirish mumkin. Chunki *in pointer* tayanch sinfga ko‘rsatkich har doim bazasidan olingan sinf ob‘yektini ko‘rsatish uchun foydalanish mumkin. Biroq, dynamic_cast operatori faqat ko‘rsatkich aylantirishi mumkin. D ob‘yekt ko‘rsatkich uchun murojaat qilinganda, albatta, sinf D ob‘yekti bo‘ladi. Umuman olganda, dynamic_cast operatori faqat polimorf tipdagi kastingga ruxsat berilsa, muvaffaqiyatli bajariladi, ya‘ni yangi tipdagi ko‘rsatikch ushbu yangi tipdagi ob‘yektga yoki undan olingan ob‘yektga havola qilishi mumkin. Aks holda, agar belgilangan tipdagi almashish amalini bajarib bo‘lmasa, bu amalda ko‘rsatgichlar ishtirok esa dinamic_cast operatorining natijasi null qiymatga aylanadi. Agar ushbu amalni bajarilishi muvaffaqitsiz bo‘lsa, uzilishlar ishtirok etgan bo‘lsa, bad_cast istisnosi avtomatik bajariladi.

Oddiy misolni ko‘rib chiqamiz. Base asos sinfi polimorf sinf bo‘lsin va myClass sinf esa Base sinfidan olingan deb faraz qilaylik.

```
Base *bp, b_ob;
myClass *dp, d_ob;
bp = &d_ob; // myClass sinf ob‘yektga Base sinf uchun ko‘rsatkich.
dp = dynamic_cast< myClass *> (bp); // myClass sinf ko‘rsatgichga almashtirish
if(dp) cout << " Tipni almashtirish muvaffaqiyatli bo‘ldi!";
```

Bu yerda bp ko‘rsatkichini (asos sinf) dp ko‘rsatkichiga (myClass sinf) almashtirish muvaffaqiyatli bo‘ladi, chunki bp aslida sinf ob‘yektiga ishora qiladi.

myClass shuning uchun ushbu misolni bajarishda tipni almashtirish muvaffaqiyatli chiqdi. bp aslida sinf bazasini ob‘yektga ishora qilgandi. Quyidagi misolda tipni almashtirishda myClass sinf ob‘yekt bo‘lsa, ko‘rsatgichga tayanch sinfga ob‘yektni tashlash uchun muvaffaqiyatsiz va noto‘g‘ri olingan hisoblanadi.

```
Base *bp, b_ob;
myClass *dp, d_ob;
bp = &b_ob; // Base sinfga ko‘rsatkich sifatida Base sinf ob‘yekt berilgan. dp =
dynamic_cast< myClass *> (bp); // xatolik
if(!dp) cout << " Tipni almashtirish muvaffaqiyatsiz ";
```

Ushbu dastur fragmentida tipni almashtirish amali muvaffaqiyatsiz bo‘ldi va tipni almashtirish uchun muvaffaqiyatsiz bo‘lganligini ekranga chiqaradi.

dynamic_cast operatoridan foydalanishda quyidagi 1.7-dasturda tipni almashtirish holatlari aniqlaymiz.

1.7-dastur. dynamic_cast operatoridan foydalanish.

```
#include   "stdafx.h"
#include <iostream>
#include <typeinfo>
using namespace std;

class Base {
        public:
                virtual void f() { cout << " Base sinfi orqali.\n"; }
// . . .
};
class myClass : public Base {
        public:
                void f() { cout << "myClass sinfi orqali.\n"; }
};
int main(){
        Base *bp, b_ob;
        myClass *dp, d_ob;
        dp = dynamic_cast<myClass *> (&d_ob);


        if(dp) {
                cout << " tipni almashtirish" <<" (myClass* => myClass *) bajarildi. "; dp-
                        >f();
        } else cout <<"Xatolik\n";
        cout << endl;
        bp = dynamic_cast<Base *> (&d_ob);
        if(bp) {
                cout << " tipni almashtirish " <<"(myClass* => Base *) bajarildi. "; bp-
                >f();
        } else cout << "Xatolik\n";
        cout << endl;

        bp = dynamic_cast<Base *> (&b_ob);
        if(bp) {
                cout << " tipni almashtirish " <<"(Base* => Base *) bajarildi. "; bp-
                >f();
        } else cout << "Xatolik\n";
        cout << endl;
        dp = dynamic_cast<myClass *>
        (&b_ob); if(dp) cout <<"Xatolik\n";
        else
        cout <<" tipni almashtirish "<<"(Base*=> myClass*) bajarilmadi.\n"; cout
        << endl;

        bp = &d_ob; // bp myClass ob'eyktiga ko‘rsatkich dp
        = dynamic_cast<myClass *> (bp);
```

```
    if(dp) {
cout << " bp ni myClass tipiga almashtirish bajarildi. ";
                dp->f();
        } else cout << "Xatolik\n";
        cout << endl;

        bp = &b_ob; // br Base ob'eyktiga ko‘rsatkich dp
        = dynamic_cast<myClass *> (bp);
        if(dp) cout << "Xatolik";
        else cout <<" bp myClass tipiga almashtirish bajarilmadi.\n"; cout
    << endl;
        dp = &d_ob; // dp myClass ob'eyktiga ko‘rsatkich bp
        = dynamic_cast<Base *> (dp);

        if(bp) {
                cout <<" dp ni myClass tipiga almashtirish bajarildi. "; bp-
                >f();
        }
        else cout <<"Xatolik\n";
        system("pause");
return 0;
}
```

1.7 – dastur natijasi. Output

```
tipni almashtirish (myClass* => myClass *) bajarildi. myClass sinfi orqali. tipni almashtirish
(myClass* => Base *) bajarildi. myClass sinfi orqali. tipni almashtirish (Base* => Base *)
bajarildi.  Base sinfi orqali.
tipni almashtirish (Base*=> myClass*) bajarilmadi.
bp ni myClass tipiga almashtirish bajarildi. myClass sinfi orqali. bp myClass tipiga
almashtirish bajarilmadi.
dp ni myClass tipiga almashtirish bajarildi. myClass sinfi orqali.
```

Dynamic_cast operatori ba‘zan typeid operatori o‘rniga ishlatilishi mumkin. Masalan, Base sinf polimorf va myClass sinf uchun asos sinf, deb faraz qilaylik. Quyidagi dastur fragmentini bajarishda, ob‘yekt aslida myClass sinf ob‘yekti bo‘lsada dp ko‘rsatkich bp ko‘rsatkich tomonidan ob‘yekt manzili beriladi.

```
Base *bp;
Derived *dp;
// . . .
if(typeid(*bp) == typeid(Derived)) dp = (Derived *) bp;
```

Bu holda, odatiy tipni almashtirish ishlatildi. Bu juda xavfsiz, chunki if operatori typeid operatori yordamida typeid amalining haqiqiyligini bajarilishidan oldin tekshiradi. Typeid operatori va if operatori bilan almashtirish orqali yanada samaraliroq qilish mumkin.

dynamic_cast:

dp = dynamic_cast<myClass *> (bp);

Dynamic_cast operatori faqat ob‘yektning belgilangan tipi yoki belgilangan tipi almashtirilgan tipi bilan bir ob‘yekt bo‘lsa muvaffaqiyatli natija beradi, bu almashtirish tugagandan so‘ng, dp ko‘rsatkich olingan ob‘yektga null qiymatini yoki ko‘rsatkichni ham o‘z ichiga oladi. Belgilangan tipni tashlash amali to‘g‘ri bajarilgan bo‘lsa dynamic_cast operatori faqat muvaffaqiyatli bajariladi, uning almashtirish mantig‘ini muayyan vaziyatlarda soddalashtirilgan mumkin. Quyidagi 1.8-dastur typeid operatorini dynamic_cast operatori bilan almashtirish mumkinligini ko‘rsatilgan. Bu yerda bir xil amallar to‘plami ikki marta bajariladi, masalan, avval typeid operatori, so‘ngra dynamic_cast operatori yordamida.

1.8-dastur. Typeid operatori o‘rniga dynamic_cast operatoridan foydalanish.

```
#include "stdafx.h"
#include <iostream>
#include <typeinfo>
using namespace std;
class Base {
        public:
                virtual void f() {}
};
class MyClass : public Base {
        public:
                void derivedOnly() {
                        cout << "Bu MyClass ob‘yekti.\n";
                }
};
int main(){
        Base *bp, b_ob;
        MyClass *dp, d_ob;
                //  typeid operatoridan foydalanish
bp = &b_ob;
        if(typeid(*bp) == typeid(MyClass)) { dp
                = (MyClass *) bp;
                dp->derivedOnly();
        } else
        cout <<" Base tipidagi ob‘yekt MyClass almashtirilmadi. \n"; bp
        = &d_ob;
        if(typeid(*bp) == typeid(MyClass)) { dp
                = (MyClass *) bp;
                dp->derivedOnly();
        } else
        cout <<"Xatolik, tip almashtirish yozilmagan! \n";
        // dynamic_cast operatoridan foydalanish bp
        = &b_ob;
        dp = dynamic_cast<MyClass *> (bp);
        if(dp) dp->derivedOnly();
```

```
                else cout << " Base tipidagi ob‘yekt MyClass almashtirilmadi. \n"; bp =
                &d_ob;
                dp = dynamic_cast<MyClass *> (bp);
                if(dp) dp->derivedOnly();
                        else cout << "Xatolik, tip almashtirish yozilmagan! \n"; system("pause");
 return 0;
}
```

1.8 – dastur natijasi. Output

```
 Base tipidagi ob‘yekt MyClass almashtirilmadi. Bu
MyClass ob‘yekti.
 Base tipidagi ob‘yekt MyClass almashtirilmadi.
Bu MyClass ob‘yekti.
```

1.8-dasturdan ko‘rinadiki, asos sinfning ko‘rasatkichini MyClass ko‘rsatkichiga almashtirishda dynamic_cast operatorini qo‘llash dastur mantig‘ini qulayroq qiladi. Yuqoridagi dastur natijasiga qarang. Shuningdek, dynamic_cast operatorini shablon sinflarga ham qo‘llash mumkin.

const_cast operatori. Const_cast operatori const va/yoki volatile o‘zgartirgichlarni qayta aniqlash uchun xizmat qiladi.

Const_cast operatori const va / yoki volatile o‘zgartirgichlarni ochiq qayta aniqlash/joriy qilish uchun ishlatiladi. const yoki volatile ning atrubutlariga istesno tariqasida, yangi tipi joriy tipi bilan mos bo‘lishi kerak. Ko‘pincha const_cast

operatori const atributini olib tashlash uchun ishlatiladi. Uning umumiy formati quyidagicha:

```
const_cast<type> (expr)
```

Bu yerda type elementi yangi tipga almatirish tipini o‘rnatadi, expr elementi esa yangi tipga tipi almatirilishi kerak bo‘lgan o‘zgaruvchi(ifoda)ni bildiradi.

1.9-dastur. Const_cast operatoridan foydalanish.

```
#include "stdafx.h"
#include <iostream>
#include <typeinfo>
using namespace std;
void f (const int *p){
        int *v;
```

```
    v = const_cast<int *> (p); //  const-o‘zgaruvchini qayta almashtirish.
        *v = 1980; // yangi qiymat berish, modifikatsiyalash
}
int main(){
        int x = 2004;
        cout <<" f() funksiya chaqirilmasdan oldin x ni qiymati: " << x<< endl; f
        (&x);
        cout <<" f() funksiya chaqirilgandan so‘ng x ni qiymati: " << x<< endl; system("pause");
return 0;
}
```

1.9 – dastur natijasi. Output

```
f() funksiya chaqirilmasdan oldin x ni qiymati: 2004
f() funksiya chaqirilgandan so‘ng x ni qiymati: 1980
```

Ko‘rib turganingizdek, x o‘zgaruvchi qabul qilgan parametr const ko‘rsatgichi sifatida o‘rnatildi va f() funksiyasi tomonidan o‘zgartirilgan.

const atributini o‘chirish uchun const_cast operatoridan foydalanish xavfli darajadagi vosita ekanligini alohida ta‘kidlash lozim. Shuning uchun, juda diqqat bilan foydalanishni tavsiya qilamiz.

Faqat const_cast operatori const atributini o‘chirishi mumkinligini bilib oldingiz. Boshqa so‘zlar bilan aytganda, na dynamic_cast, static_cast, yoki reinterpret_cast ob‘yekt const atributini o‘zgartirish uchun foydalanish mumkin emas.

static_cast operatori. Static_cast operatori polimorf bo‘lmagan tipdagi ko‘rsatkichlarni almashtirish uchun foydalaniladi. Har qanday tipni almashtirish uchun foydalanish mumkin. Biroq, dastur bajarilatgan vaqtda hech qanday tekshirish amalga oshirilmaydi. Static_cast operatori quyidagi umumiy yozuv formatiga ega:

```
static_cast<type> (expr)
```

Bu yerda type elementi yangi tipdagi almashtirish tipini belgilaydi va expr elementi bu yangi tipga tipi almashtirilishi kerak bo‘lgan ifodani bildiradi.

Static_cast operatori tipni almashtirishning orginal operatoridir. U faqat polimorf bo‘lmagan tiplar uchun amalga oshiradi. Masalan, quyidagi 1.10-dasturni ishlaganda float tipidagi o‘zgaruvchi int tipiga aylantiradi.

1.10-dastur. Static_cast operatoridan foydalanish.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
int main(){
        int i; float f;
        f = 199.22F;
        i = static_cast<int> (f); cout <<
        i;
        return 0;
   system("pause");
}
```

reinterpret_cast operatori. Reinterpret_cast operatori fundamental tipni almashtirishni amalga oshiradi.

einterpret_cast operatori tipni tubdan farq qiladigan tipga aylantiradi. Masalan, bir ko‘rsatkichni int tipiga hamda bir ko‘rsatkichni bir int qiymati uchun ko‘rsatkichga aylantirish uchun foydalanishingiz mumkin. Bundan tashqari, tabiatan mos kelmaydigan ko‘rsatkich tiplarni tashlash uchun foydalanish mumkin. Ushbu operator quyidagi umumiy formatga ega:

```
reinterpret_cast<type> (expr)
```

Bu yerda type elementi yangi tipdagi tipni almashtirishni belgilaydi va expr elementi bu yangi tipga almatiriladigan ifodani bildiradi.

Reinterpret_cast operatoridan foydalanish quyidagi 1.11-dasturni ko‘rib chiqamiz.

1.11-dastur. Reinterpret_cast operatoridan foydalanish.

```
#include "stdafx.h" #include <iostream>
using namespace std;
int main(){
        int i;
        char *p = "Salom Buxoro";
        i = reinterpret_cast<int> (p); // ko‘rsatkichni int tipiga almashtirish. cout << i<<endl;
        system("pause");
        return 0;
}
```

Bu yerda reinterpret_cast operatori p ko‘rsatkichni butun son int qiymatiga aylantiradi. Bu o‘zgartirish asosiy tipni o‘zgarish ifodalaydi. Bu tip almashtirishni oddiy kasting bilan ham amalga oshirish mumkin.

Yangi to‘rtta cast operatorlari bilan an‘anaviy tipni almashtirish amali (kasting) qanday farq mavjud. Ba‘zi dasturchilar to‘rt cast operatorlari yuqorida tasvirlangan butunlay an‘anaviy tipni almashtirish amalining o‘rniga deb o‘ylashlari mumkin. Ularning o‘rniga har doim odatiy tipni almashtirishning yangi vositalaridan foydalanish kerakmi? – degan savol qiziqtirishi mumkin. Aslida barcha dasturchilar uchun umumiy qoida yo‘q. Yangi operatorlar bir ma‘lumottipini boshqasiga almashtirishni ancha xavfli amallar xavfsizligini yaxshilash uchun yaratilganligi sababli, ko‘pgina C++ dasturchilari faqat shu maqsadda ishlatilishi kerakligiga ishonch hosil qilgan. Bunga e‘tiroz qilib bo‘lmaydi. Boshqa dasturchilar an‘anaviy tipi almashtirish amali ko‘p yillar davomida ularga "sadoqat" bilan xizmat qilib bergan, undan shunday voz kechish oson emas. Masalan, oddiy va nisbatan xavfsiz tipdagi kasting amallarini bajarish uchun read() va write() funksiyalarini chaqirganda talab qilinadi.

Ammo shunday dasturlashda jarayonlar borki, ularda cast operatorlaridan foydalanish maqsadga muvofiqdir, masalan, polimorf tiplar uchun cast amallarini bajarishda albatta dynamic_cast operatoridan foydalanishga to‘g‘ri keladi.

Yangi nomlar fazosi yaratish. Nomlar fazosi haqida qisqacha ma‘lumotga ega bo‘lsangiz kerak. Endi bu tushunchani batafsilroq ko‘rib chiqamiz va foydalanuvchining nomlar fazosi qanday yaratilishini ko‘rib chiqamiz. Namespaces tushunchasi C++ dasturlash tilida nisbatan yaqinda paydo bo‘lgan tushunchalarda biridir. Ular nomlarning o‘zaro bir xilligini oldini olish (kolliziya saqlanish) uchun identifikatorlarning nomlarini lokalizatsiya qilish uchun mo‘ljallangan. C++ dasturlash muhitida, o‘zgaruvchan nomlari aql bovar qilmaydigan bir qator operatorlari, funksiyalari, metodlari va sinflar mavjud. Nomlar fazosi tushunchasi kiritilishidan oldin, bu nomlarning barchasi bir xil global nomlar bo‘lib, ko‘plab nizolarni (xatoliklarni, o‘zaro bir xilliklarni) keltirib chiqargan. Agar dasturda dasturchi o‘zining toupper() funksiyasini yaratgan bo‘lsa, masalan, agar, u (uning

parametrlari ro‘yxatiga qarab) standart kutubxona toupper(), funksiyasini almashtirish mumkin. Ammo, ikkala fnuksiyalarning nomlari bir xil global namespaceda saqlangan bo‘ladi. Bir xil dasturlarni turli ishlab chiqaruvchilarning funksiyalari va sinflar, kutubxonalar foydalanganda nomlar bir xil bo‘lgan funksiyalar dastur bajarilishida ziddiyatlarga duch keladi. Bunday holda, bir kutubxonada aniqlangan nomlar boshqa kutubxonada aniqlangan nomlar bilan to‘qnashishi mumkin.

Barcha muammolar nomlar fazosi tushunchasi va namespace kalit so‘z tomonidan hal qilindi. Bu kalit so‘z yangi nomlar fazosi e‘lon qilish orqali barcha funksiya, sinf va kutubxonalarni mahalliylashtirish imkonini beradi. Bu nomlar fazosi turli kontekstlarda bir xil nomdagi funksiya va sinflar ziddiyatga olib kelmasdan ishlatish imkonini beradi. std standart kutubxona (std -Standart Library) juda ko‘p foydalanilgan bo‘lishi mumkin. Tilning dastlabki variantlarida C++ kutubxonasi global nomlar fazosida belgilangan. Sezilarli nomlar ziddiyatlari ehtimolini kamaytirish uchun std namespace nomlar fazosi yaratilgan. Bundan tashqari, ziddiyatlarni oldini olish uchun nomlar fazrsi ko‘lamini mahalliylashtirish uchun dasturda o‘z nom fazongizni yaratishingiz mumkin. Bu shaxsiy sinf yoki funksiya kutubxonalar yaratishda, ayniqsa, muhim ahamiyatga ega.

Namespace kalit so‘zi nomlar fazosida sohalarni qo‘shib global namespacedan ajratish imkonini beradi. Aslida, bir nomlar fazosi doirasi belgilaydi. Namespace kalit so‘zidan foydalanishning asosiy shakli quyida ko‘rsatilgan:

```
namespace Space{
        //....
}
```

Namespace kontentida aniqlangan ixtiyoriy har bir element, bu namespace doirasi bajariladi.

1.12-dastur. myNameSpace nomli namespace e‘lon namuna.

```
namespace
        MyNameSpace {
        int i,k;
        void myfunc (int j) { cout <<j; } class
        myclass {
                public:
                        void set(int x) { i = x; } int
                        get() { return i; }
        };
};
```

Mynamespace nomli namespaceda i va k o‘zgaruvchilarning nomlari, myfunc() funksiyasi va myclass sinfi yaratilgan.

Bu namespace ichida o‘zgaruvchilarga, funksiyava sinflarga to‘g‘ridan- to‘g‘ri murojaat qilish va yangisini KIRISH mumkin. Biroq, namespace kalit so‘z ma‘lum bir dastur sohasini belgilaydi, chunki, namespace tashqaridan namespace ichida e‘lon ob‘yektlarni foydalanish imkoniyatini faqat ziddiyatlarni oldini olgan holda foydalanishga ruxsat beradi. Masalan, Mynamespace nomli namespace tarkibiga kirmagan dasturning bir qismida i o‘zgaruvchiga 28 qiymat berish uchun quyidagicha ifoda yozish lozim:

```
        MyNameSpace::i = 28;
```

Mynamespace nomli namespace sohasiga kiradigan Myclass tipidagi ob‘yektni odatdagidek e‘lon qilishimiz mumkin, ammo kirmagan Myclass tipidagi ob‘yektni quyidagicha e‘lon qilish lozim:

```
        MyNameSpace::myclass ob;
```

Shunday qilib, bu soha tashqarida nomlar fazosining nomi bilan kengaytirish operatori (::) bilan murojaat qilish mumkin.

Agar dasturda namespace a‘zolariga tez-tez murojaat bo‘ladigan bo‘lsa, har doim uning nomi va kengaytirish operatori yozish dasturning ko‘rinishiga va dasturchining ko‘p yozishiga sabab bo‘lishi mumkin. Ushbu muammoni hal qilish uchun foydalanuvchining nomlar fazosidan foydalanish fragmenti ishlab chiqilgan. Ushbu fragment ikkita asosiy shaklga ega:

```
using namespace MyNameSpace;

using MyNameSpace::myclass;
```

Fragmentning birinchi qatorida keltirilgan shaklda a‘zolar kirmoqchi bo‘lgan nomlar fazosining nomini bildiradi. Foydalanish fragmentida bu shakl yordamida yozilgan nomlar fazosidagi barcha a‘zolarga joriy nom mavjud deb olinadi va to‘g‘ridan-to‘g‘ri murojaat qilish mumkin, fazo nomi va kengaytirish operatori har safar yozish kerak bo‘lmaydi.

Fragmentning ikkinchi qatorida using fragmentining ikkinchi shaklidan foydalanilganda faqat fragmentda ko‘rsatilgan nomdosh a‘zosi ko‘rinadi. Masalan, yuqorida ta‘riflangan mynamespace nomli namespace bo‘lsa, unda quyidagi fragmentlar va operatorlaridan foydalanish to‘g‘ri:

```
using MyNameSpace::k;
int main(){
 k = 10;
 return 0;
}
```

Fragment asosida faqat k o‘zgaruvchi ko‘rinadi. Demak, ko‘rsatma to‘g‘ri.

```
using namespace MyNameSpace;

int main(){
        k = 10;
        i = 10;
        myclass ob; myfunc(25);
return 0;
}
```

Bu fragmentda MyNameSpace fazoning barcha a‘zolari ko‘rinadi va Demak, ko‘rsatma to‘g‘ri.

Shuningdek, bir xil nom bilan bir nechta fazoni e‘lon qilish imkoniyati mavjud. Bu bir necha fayllar ichiga fazolarni ajratish imkonini beradi, yoki hatto bitta fayl ichida fazolarni ajratish imkonini. Buni dasturchining dasturga ishlatayotgan funksiya va sinflarini ajratish uchun qo‘llash mumkin. Quyidagi misolni ko‘rib chiqamiz:

```
namespace NS { int a;}
namespace NS { int b;}
int main(){
        NS::a = 10;
        NS::b = 10;
return 0;
}
```

Bu yerda NS namespace ikki qismga bo‘lingan. Shunga qaramay, har bir qismi a‘zolariga ham bir xil nom bilan murojaat qilingan, ya‘ni NS fazo.

Bir dastur faylida bir nomli fazolarni e‘lon qilish maqsadga muvofiq. Ammo, dasturda 2 va undan ortiq dastur fayllari ishlatilsa va fazolarni birlashtirish kerak bo‘lsa, shunda fazoni ikkiga ajratish usulidan foydalanish mumkin. Boshqa so‘zlar bilan aytganda, bir namespace faqat boshqa namespace bo‘lishi mumkin emas. Bu bir nomli fazo e‘lon qilinishi mumkin emas degan ma‘noni anglatadi, masalan, bir funksiya ichida.Nomsiz fazo - fazoning maxsus turidir. Muayyan fayl ichida noyob identifikatorlar, a‘zolarni yaratish imkonini beradi. Nomsiz fazoning asosiy shakli quyidagicha bo‘ladi:

```
namespace {
        const float f = 3.1415;
        int myIf(int a, int b){return a>b?1:0;}
}
```

Fazolarni nomsiz nomlashlar faqat bitta fayl doirasida ma‘lum bo‘lgan noyob identifikatorlar va funksiya, sinflarni o‘rnatish imkonini beradi. Shu tarzda, noma‘lum nomli fazoni o‘z ichiga olgan fayl ichida, ushbu fazoning a‘zolariga to‘g‘ridan-to‘g‘ri, hech qanday tushuntirishga muhtoj bo‘lmasdan foydalanish mumkin. Ammo bu fayldan tashqarida bunday identifikatorlar noma‘lum bo‘lib qoladi.

Siz kichik va o‘rta dasturlar uchun o‘z fazolaringizni yaratishingiz shart emas. Ammo, agar qayta foydalanish uchun mo‘ljallangan fnuksiyalar, ssinflar va/yoki sinflar kutubxonalar yaratish uchun ba‘zi namespace ichida sizning kodlaringiz joylashtirish yozgan dasturingizni tushunarliroq qiladi.

1.13-dasturda nomlar fazosini yaratish va foydalanishga misol keltirilgan. 1.13-dastur. Nomlar fazosini yaratish va foydalanish.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
// birinchi nomlar fazosi
namespace NS1{
        class demo{
                int i; public:
                        demo(int x) { i=x; } void set(int x) { i=x; } int get() { return i; }
        };
        char str[] = "Nomlar fazolarini o‘rnatish\n"; int counter;
}
// ikkinchi nomlar fazosi
namespace NS2 { int x, u; }
```

```
int main(){
        NS1::demo ob(10);
        cout <<" ob = " << ob.get() << endl;
        ob.set(99);
        cout << "Endi ob = "<< ob.get() << endl;
        using NS1::str; cout << str;
        using namespace NS1;
        for(counter = 1; counter<=10; counter++)
                cout << counter << ends;
        cout << endl;
        NS2::x = 10; NS2::u = 20;
        cout << "o‘zgaruvchilar x, y = "<< NS2::x<<", " << NS2::u << endl;
    using  namespace  NS2;
        demo x_ob(x), y_ob(u);
            cout << "x_ob, y_ob ob‘yektlarning qiymati: "<< x_ob.get(); cout<< " " <<
y_ob.get() << endl;
system("pause");
return 0;
}
```

1.13 – dastur natijasi. Output

```
ob = 10
Endi ob = 99
Nomlar fazolarini o‘rnatish 1 2 3 4 5 6 7 8 9 10
o‘zgaruvchilar x, y = 10, 20
x_ob, y_ob ob‘yektlarning qiymati: 10 20
```

Dasturda bir muhim nuqta ko‘rsatilgan: bir nechta nomlarni birgalikda ishlatganda, bir nomni boshqasi almashtirmaydi. Hozirgi fragmentda ba‘zi nomlarni kiritganingizda, uning nomlari o‘sha paytda boshqa nomdagi nomlarning o‘zida bo‘lishidan qat‘i nazar, bu fragmentda qo‘shiladi. Shunday qilib, dastur bajarilayotgan vaqtda, std, NS1 va NS2 namespaces global namespace bo‘lib qo‘shildi.

Yuqorida aytib o‘tilganidek, fayllar orasida yoki bitta fayl ichida nomlar fazosini ajratish mumkin, keyin esa bu nomlar fazosining a‘zolarini mazmunan birlashtiriladi. Birgalikda nomlar fazosini birlashtirish misolini ko‘rib chiqamiz.

1.14-dastur. Namespace larni birlashtirish.

```
#include "stdafx.h"
#include <iostream>
using namespace std;
namespace Demo {
        int a; // Demo fazoda a o‘zgaruvchi
}
int x; // Global fazoda x o‘zgaruvchi namespace
```

```
Demo {
        int b; // Demo fazoda b o‘zgaruvchi
}
using namespace Demo;
int main(){
        a = b = x = 101;
cout << a << " " << a << " " << x << endl;
system("pause");
return 0;
}
```

1.14 – dastur natijasi. Output

```
101 101 101
```

Bu misolda a va b o‘zgaruvchilari ham bir xil fazoda - Demo fazoda va x o‘zgaruvchisi global fazoda e‘lon qilingan. Ammo, using namespace Demo; fragmenti bilan, a va b o‘zgaruvchilarini va global o‘zgaruvchini ishlatish mumkin.

Yuqorida aytib o‘tilganidek, C++ standart o‘z std fazosiga ega va butun kutubxonalarini qamrab oladi. Shuning uchun ushbu kitobdagi barcha dasturlar quyidagi ko‘rsatmaga ega:

```
using namespace std;
```

Bu fragment std namespace joriy qiladi va bevosita C++ standart kutubxonasida belgilangan funksiyalari va sinflar nomlarini KIRISH imkonini beradi. std namespace kengaytirish operatori yordamida har doim ishlatish maqsadga muvofiq emas.

Biroq, agar xohlasangiz, har bir identifikatordan oldin std namespace nomini va kengaytirish operatorini qo‘yishingiz mumkin, xato bo‘lmaydi. Masalan, quyidagi 1.15-dasturda standart std kutubxonasi global doirada kiritilmagan.

1.15-dastur. Ochiq foydalanish uchun nom ko‘rsatilgan.

```
#include   "stdafx.h"
#include <iostream>
int main(){
        double val;
        std::cout << "Son kiriting:"; std::cin
        >> val;
        std::cout  <<  "Son:"; std::cout
        << val << std::endl;
system("pause");
return 0;
}
```

Ushbu dasturda ko‘rsatilganidek, standart KIRISH va chiqish oqimlari cin va cout operatorlaridan foydalanish uchun nomlaridan oldin ularning qaysi fazodanligini belgilash kerak.

Agar dasturingiz C++ standart kutubxonasidan keng foydalanish zarur bo‘lmasa, std namespaceni global doirada kiritish shart emas. Ammo, agar sizning dasturingiz standart kutubxona a‘zolari uchun minglab murojaatlarni o‘z ichiga olsa, har a‘zo uchun std fazosini chaqirish juda ham oson kechmaydi. Shuning uchun uni global kiritish nisbatan dasturchi oson va std barcha standart identifikatorlarni o‘z ichiga oladi.

Agar dasturingizda standart kutubxonadan faqat bir necha a‘zodan foydalansangiz, bu bir necha a‘zolarni alohida belgilash uchun using fragmentidan foydalanish qulayroq bo‘lishi mumkin. Ushbu yondashuvning afzalligi shundaki, siz dasturda ko‘rsatilgan a‘zolarni fazosini ko‘rsatmasdan kiritishingiz mumkin va ayni paytda butun standart kutubxonani global nomda kiritishingiz shart emas. Masalan:

```
#include "stdafx.h"
#include <iostream>
using std::cin;
using std::cout;
int main(){
        double val;
        cout << "Son kiriting:"; cin
        >> val;
        cout << "Son:";
        cout << val << std::endl;
system("pause");
return 0;
}
```

Bu yerda standart KIRISH va chiqish oqimlari cin va cout operatorlari to‘g‘ridan-to‘g‘ri ishlatilishi mumkin, ammo ayni paytda std fazoning boshqa a‘zolar joriy soxadan tashqarida qoldiriladi. Maslan, endl;

Avvalroq, C - tilining kutubxonasi global nomda belgilanganligi haqida aytib o‘tilgan edi. Agar eski C dasturini yangilashingiz kerak bo‘lsa, siz ham unda foydalanish uchun namespace std foydalanish kerak bo‘ladi, yoki kengaytirish operatori va std namespace nomini kiritish lozim. Agar yangi uslub sarlavhalari bilan eski header fayllarni o‘rniga, ayniqsa, muhim ahamiyatga ega (*.h). Eski header fayllar global namespace o‘z mazmunini joylashtiradi va yangi uslub sarlavhalari uchun std namespace o‘z mazmunini joylashtiradi.

Agar, nomlar fazosini global sifatida bitta faylga e‘lon qilib, cheklamoqchi bo‘lsangiz, S tilida statik deb e‘lon qilish kerak edi, ya‘ni, statik identifikator bilan. Masalan, quyidagi ikki fayl bir xil dasturning bir qismi deb faraz qilaylik:

Birinchi fayl	Ikkinchi fayl

static int counter; void
f1(){
counter == 99; // to‘g‘ri
}
 	
	extern int counter; void
	f2() {
	counter = 10; // Xato
	}

counter o‘zgaruvchisi birinchi faylda aniqlanganligi uchun ham uni birinchi faylda ishlatishingiz mumkin. Ikkinchi faylda counter o‘zgaruvchisi bilan fragmentdagi extern identifikatorini ko‘rsatishiga qaramay, bu o‘zgaruvchini ishlatishga urinish xatolikka olib keladi. Birinchi fayldagi counter o‘zgaruvchisini statik deb e‘lon qilib, uning ko‘lamini shu faylga cheklangan.

Statik identifikator bilan global o‘zgaruvchilar deklaratsiyalari C++ hali ham mavjud bo‘lsa-da, bu qochish maqsadda eng yaxshi yo‘l Nomlar fazosidan foydalanish hisoblanadi, quyidagi misolda ko‘rsatilgandek.

Birinchi fayl	Ikkinchi fayl

```
namespace {
int counter;
}
void f1(){
        counter == 99; // to‘g‘ri
}
```

```
extern int counter; void
f2() {
        counter = 10; // to‘g‘ri
}
```

Buferlashtirilgan kiritish va chiqarish. C++ tilida ekran va faylga IO dan tashqari kiritish va chiqarish vazifalarini lokalizatsiyalash sifatida massivlardan foydalanadigan qator funksiyalarni qo‘llab-quvvatlaydi. Massivlarga asoslangan kiritish / chiqarish konseptual (array-based I/O) echimga o‘xshasada, C tilining IO funksiyalari bilan anologdir (ayniqsa, sscanf() va sprintf() funksiyalari). Massivlar yordamida kiritish/chiqarish C++ tilida juda moslashuvchan va foydalidir, chunki unda foydalanuvchi tomonidan yaratilgan o‘zgaruvchilarni kiritish va chiqarishda ham foydalaniladi. Massivlar bilan kiritish/chiqarishning barcha jihatlarini qamrab olish mumkin emas, ammo, eng muhim va tez-tez ishlatiladigan xususiyatlari amalga oshirish mumkin.

Massivlar yordamida IOni amalga oshirish uchun ham oqimlarni talab qilishini tushunish muhimdir. Oldingi ko‘nikmalaringizdan C++ da IO amalarini bilishingiz mumkin. Biroq, kiritish/chiqishda paramertlar sifatida ob‘yektlardan foydalanishning barcha afzalliklari haqida dasturchilarning ko‘nikmalari muhimdir. Shuningdek, ma‘lumot olish uchun, siz bir necha yangi xususiyatlarni boshqaradigan funksiyalar bilan tanishishingiz kerak. Bu funksiyalar istalgan oqimni ma‘lum xotira maydoniga bog‘lash uchun mo‘ljallangan. Bu amallar tugagandan so‘ng, barcha IO xusisiyatlari allaqachon dasturchi biladigan IO vazifalari yordamida amalga oshiriladi degan ko‘nikmaga olib keladi.

Kiritish/chiqarish ob‘yektlari sifatida massivdan foydalanishni boshlashdan oldin, <strstream> kutubxonasini dasturingizga kiritilgan bo‘lishi kerak. Bukutubxonadagi istrstream, ostrstream, va strstream sinflardan foydalanish mumkin. Bu sinflar, o‘z navbatida, kiritish, chiqish va KIRISH/chiqish uchun oqimlardan foydalanish asoslangan. Ios sinfi bu sinflar uchun tayanch sinfdir, shuning uchun istream, ostream va iostream sinflarining barcha funksiyalari va manipulyatorlari istrstream, ostrstream va strstream sinflarida ham mavjud. Satr massivini chiqish uchun ostrstream sinf konstruktordan quyidagi asosiy shaklda foydalaniladi:

```
ostrstream chiqish_oqimi(char *bufer, streamsize size, openmode mode = ios::out);
```

Bu yerda, chiqish_oqimi massivli satr bilan bog‘laydigan oqim bo‘lib, buffer pointer (buferlangan ko‘rsatkich) orqali beriladi. size parametri massivning o‘lchamini belgilaydi. Mode - bu standart rejim parametri, odatda normal chiqish holatiga o‘rnatiladi, lekin siz ios sinfda belgilangan har qanday chiqish rejimi indentifikatori o‘rnatishingiz mumkin.

Chiqish uchun massiv ochilganda, simvolar massivga sig‘guncha joylashtiriladi.To‘lgan massiv bo‘lishi mumkin emas. Ixtiyoriy to‘lgan massiv kiritish/chiqarishni xatolikka olib kelishi mumkin. Masssivdag belgilarni sonnini aniqlash uchun pcount() funksiya-a‘zosi ishlatiladi.

```
int pcount()
```

Funksiya faqat bir oqim bilan aloqada bo‘lib, shu munosabati bilan chaqiriladi va u massivga yozilgan belgilar sonini qaytaradi. Shu jumladan null tugatishni bildiradi.

Undan massivga kiritish uchun istrstream sinf konstruktorining quyidagi shaklidan foydalaniladi.

```
istrstream KIRISH_oqimi (const char * bufer) ;
```

Bu yerda, bufer belgilar kiritiladigan massivga ko‘rsatgich hisoblanadi va KIRISH_oqimi KIRISH oqimi bilan ko‘rsatiladi. Biror massivdan KIRISH ma‘lumotlarini o‘qishda eof() funksiyasidan foydalaniladi va massivning oxiriga etganida true qaytaradi.

KIRISH/chiqish uchun massiv ochish uchun strstream sinf konstruktor quyidagicha shakldan foydalaniladi:

```
strstream kch_oqimi(char* bufer, streamsize size, openmoda mode = ios : : in | ios: : out) ;
```

Bu yerda, kch_oqimi – bu kirio‘/chiqish oqimi, belgilar o‘lchamida belgilangan uzunligidagi massivga bufer ko‘rsatkich orqali KIRISH/chiqish ob‘yekti sifatida ishlatiladi.

Yuqorida keltirilgan barcha IO vazifalari, shuningdek, oqim bilan ishlash, ikkilik, shu jumladan tasodifiy KIRISH uchun muhimligini tushinish lozim.

Massivlar uchun oqim sinflaridan foydalanish C++ standarti tomonidan kam ko‘llab quvvatlanadi. Kelajakda C++ versiyalarida qo‘llab-quvvatlanishi bo‘lmasligi ham mumkin. Belgilarni massivla oqimlari uchun mo‘ljallangan bir xil vazifalar C++da sinf-konteyner asosida takomillashtirilgan.

1. Massivga ma‘lumotlarni yozish va chiqish uchun oddiy misol keltiramiz.

1.16-dastur. Massivga chiqarish.

```
#include "stdafx.h"
#include <iostream>
#include <strstream>
using namespace std;
int main(){
char buf[255]; // chiqarish uchun bufer
ostrstream ostr(buf, sizeof buf); // chiqarish uchun massiv ostr
<< "KIRISH/chiqish massiv oqimi bilan ishlaydi.\n"; ostr <<
"oddiy KIRISH/chiqish kabi: \n " << 100;
ostr <<' ' << 123.23 <<'\n';
// manipulyatordan ham foydalanish mumkin
ostr << hex << 100 << ' ';
// identifikatordam ham
ostr.setf(ios::scientific); ostr <<
123.45 << ' ';
ostr << ends;
// oxirgi natijani chiqarish
cout << buf;
system("pause");
return 0;
}
```

1.16 – dastur natijasi. Output

```
KIRISH/chiqish massiv oqimi bilan ishlaydi.
oddiy KIRISH/chiqish kabi:
100 123.23 64 1.234500e+002
```

Dasturda IO operatorlari qayta aniqlangan, hamda KIRISH/chiqish uchun ajralmas manipulatorlar, a‘zo funksiyalari va massivdvn foydalanishda to‘liq format indentifikatorlari oqimlar

uchun foydalanilgan. Bu barcha manipulyatorlar uchun amal qiladi va lokal sinflar uchun IO operatorlari yaratish imkonini beradi.

Yuqoridagi dasturda ends manipulyatori massivga tugatish null belgisini qo‘shish uchun ishlatiladi. Null belgi avtomatik ravishda massivga kiritilgan yoki yo‘qligini amalga oshirish dasturchining o‘ziga bog‘liq. Shuning uchun, bu dasturchi uchun muhim, chunki massiv qatorining oxiriga null belgi yozish yaxshidir.

1.17-dastur. Massivga ma‘lumotlarni kiritish.

```
#include "stdafx.h"
#include
<iostream>
#include
<strstream>
using namespace
std;

int main(){
char buf[] = "SalomBuxoro 1980 28.01 B";
istrstream istr(buf);
char str[80]; float f; char c; int i; istr
>> str >> i >> f >> c;
cout << str << ' ' << i << ' ' << f << ' ' << c << '\n'; system("pause");
return 0;
}
```

1.17 – dastur natijasi. Output
SalomBuxoro 1980 28.01 B

Bu dastur bufer ko‘rsatkichida massivda mavjud ma‘lumotlarni o‘qiydi va qayta ishlab ekranga chiqaradi.

Agar massiv oqim bilan bog‘langan bo‘lsa, fayl kabi ishlatiladi. Masalan, massiv ma‘lumotlarni bufer ko‘rsatkichi bilan o‘qish va dasturda eof() va get() foydalanishni ifodalaydi.

1.18-dastur. Massivlar asosida eof() va get() funksiyalarning IO da foydalanish.

```
#include "stdafx.h"
#include <iostream>
#include <strstream>
using namespace std;
int main(){
        char buf[] = "SalomBuxoro 1980 28.01 B";
        istrstream istr(buf);
        char c;
        while(!istr.eof()) {
                istr.get(c);
                if (!istr.eof()) cout << c;
        }
```

```
        cout<<endl;
        system("pause
        ");
  return 0;
}
```

1.18 – dastur natijasi. Output

```
SalomBuxoro 1980 28.01 B
```

Quyidagi dastur massiv ma‘lumotlarni kiritish va chiqarishga oid.

1.19-dastur. Massivlar yordamida KIRISH/chiqish.

```
#include "stdafx.h"
#include <iostream>
#include <strstream>
using namespace std;
int main(){
        char iobuf[255];
        strstream iostr (iobuf, sizeof iobuf) ; iostr <<
        "tekshrish \n";
        iostr << 100 << hex << ' ' << 100 << ends; char
        str[80]; int i ;
        iostr.getline(str, 79); iostr
        >> dec >> i;
        cout << str << ' ' << i << ' '; iostr >>
        hex >> i;
        cout << hex << i << endl;
        system("pause");
 return 0;
}
```

1.19 – dastur natijasi. Output

```
tekshrish  100 64
```

Buferlashtirilgan kiritish va chiqarishda C++ da IO ni boshqarish vazifalari uchun a‘zo funksiyalarini, identifikatorlarni va manipulyatorlarni formatlash orqali ta‘minlanadi. Indentifikatorlar, funksiyalar va manipulyatorlar bir xil vazifani bajaradi - ular axborotni kiritish/chiqarish uchun maxsus formatni o‘rnatadi. C++ da ekrandan/ga kiritish/chiqarish mos ravishda cin va cout operatorlari yordamida amalga oshiriladi, ya‘ni formatlash manipulyatorlari ushbu IO operatorlari bilan birgalikda ishlatiladi. Indentifikator vazifalari va formatlash manipulyatorlar o‘rtasidagi farq ularga qo‘llaniladigan usuldir. Endi formatlash obektlaridan foydalanish yo‘llarini ko‘rib chiqamiz.

Asosiy formatlash a‘zo funksiyalari:

cout.fill('symbol'); to‘ldiruvchi belgini o‘rnatadi. Bunda symbol to‘ldiruvchi belgi va tirnoq ichida beriladi.

cout.width(width_field); maydon uzunligini berish. width_field – pozitsiyalar soni, har bir pozitsiyada bitta simvol sig‘adi.

cout.precision(number); haqiqiy sonlarda nuqtadan keyingi belgilar soni berish. number - haqiqiy sonlarda nuqtadan keyingi belgilar soni.

Funksiyalarga nuqta amali orqali KIRISHiladi va qavslar ichida kerakli argument yoziladi. Fill() funksiyasiga argumentni bitta qo‘shtirnoq bilan belgi sifatida yoki son(belgi kodi) sifatida yozish mumkin. KIRISH/chiqish oqimlarni formatlash uchun bitta funksiya yetarli emas. C++da indentifikatorlar bilan formatlash usullarini ko‘ramiz.

Indentifikatorlar orqali formatlash KIRISH/chiqish parametrlaridan birini yoqish yoki o‘chirish imkonini beradi. IO nindentifikatorni o‘rnatish uchun setf() funksiyasini ishlatamiz va agar chiqish indentifikatorni o‘chirmoqchi bo‘lsangiz, unsetf() funksiyasidan foydalaniladi. Quyidagi indentifikatorni o‘rnatish va o‘chirishga doir instruksiya ko‘rsatilgan.

Chiqish indentifikatorni o‘rnatish

```
cout.setf( ios::name_flag );
```

name_flag - indentifikator nomi. Chiqarish operatorlariga nuqta orqali ruxsat beriladi. setf() funksiyasi bitta argument qabul qiladi va indentifikator nomidir. Indentifikatorlar ios sinfiga tegishli bo‘lganligi uchun oldin shu sinf nomi yoziladi va kengaytirish amali orqali kerakli indentifikator tanlanadi.

Chiqish indentifikatorni o‘chirish

```
cout.unsetf( ios::name_flag);
```

name_flag - indentifikator nomi.

Agar KIRISH/chiqishda bir nechta identifikatorlarni ishlatish kerak bo‘lsa, Yoki [|] mantiqiy amalidan foydalanish mumkin.

Bir nechta identifikatorlarni o‘rnatish

```
cout.setf( ios::name_flag1 | ios:: name_flag2 | ios:: name_flag_n );
```

Bir nechta identifikatorlarni o‘chirish

```
cout.unsetf( ios::name_flag1 | ios:: name_flag2 | ios:: name_flag_n );
```

1.1-jadvalda asosiyformatlash identifikatorlarini keltirilgan va ulardan foydalanishga doir misollar ko‘rsatilgan.

1.1-jadval. C++ da formatlash identifikatorlari.

Identifi-kator	Vazifasi	misol	Misol natijasi
boolalpha	Mant ko‘rinishida mantiqiy kattalikni kiritish (true, false)	cout.setf(ios::boolalpha); bool log_false = 0, log_true = 1; cout << log_false << endl << log_true << endl;	false true
oct	Sakkizlik sanoq sistemasida kattaliklarni kiritish (avval dec identifikatorni o‘chiramiz va oct identifikator o‘rnatamiz)	cout.unsetf(ios::dec); cout.setf(ios::oct); int value; cin >> value; cout << value << endl;	KIRIS H: 9910 chiqish: 1438

dec	O‘nlik sanoq sistemasida kattaliklarni kiritish/chiqarish (identifikator joriy o‘rnatilgan)	cout.setf(ios::dec); int value = 148; cout << value << endl;	148

hex	O‘n oltilik sanoq sistemasida kattaliklarni kiritish (avval dec identifikatorni o‘chiramiz va oct identifikator o‘rnatamiz)	cout.unsetf(ios::dec);
		cout.setf(ios::hex); int value;
		cin >> value;
		cout << value << endl;
			KIRISH: 9910
			chiqish: 6316

showbase	Sanoq sistemaga asoslangan indikatorni chiqarish	cout.unsetf(ios::dec);
		cout.setf(ios::oct | ios::showbase);
		int value;
		cin >> value;
		cout << value << endl;
			KIRISH: 9910
			chiqish: 01438

uppercase	16 lik SSda katta harflarni chiqarish (joriy holatda kichik harflar o‘rnatilgan)	cout.unsetf(ios::dec);
		cout.setf(ios::hex | ios::uppercase);
		int value;
		cin >> value;
		cout << value << endl;
			KIRISH: 25510
			chiqish: FF16

showpos	Musbat sonlar uchun + ishorasini chiqarish	cout.setf(ios::showpos); int value = 15; cout << value << endl;	+15

scientific	Eksponensial sonlarni chiqarish	cout.setf(ios::scientific);
		double value = 1024.165;
		cout << value << endl;
			1.024165 e+003

fixed	Fiksirlangan holda haqiqiy sonlarni chiqarish (joriy holatda)	double value = 1024.165; cout << value << endl;	1024.165
right	O‘ng tomondan tekslish	cout.width(40); cout << «cppstudio.com» << endl;	__cppstudio.com

left	Chap tomondan tekislash	cout.setf(ios::left);
		cout.width(40);
		cout << «cppstudio.com» << endl;
			cppstudio.com____

Shuningdek, monipulyatorlar bilan formatlashning usullari ham bor. manipulyator – maxsus tip bo‘lib, oqimlarga o‘tgan ma‘lumotlarni formatlash uchun kiritish/chiqish oqimlarni nazorat qiladi. Lekin, ko‘pchilik manipulatorlarining funksiyalari identifikatorlar bilan bir xil formatlash amallarini bajaradi. Ba‘zida, identifikatorli yoki formatlash funksiyalaridan foydalanish osonroq bo‘lsada, ba‘zan formatlashda manipulyatorlaridan foydalanish qulayroq. Shuning uchun C++da KIRISH / chiqishni formatlashning bir necha vositalari bilan to‘ldirilgan. 1.2-jadvalda formatlashning monipulyatorlari va misollari keltirilgan.

1.2-jadval. Formatlashning monipulyatorlari.

Manipulyator	vazifasi	misol	natijasi
endl	Chiqarishda yangi qatorga o‘tish	cout <<«Salom:» << endl <<«mbbGroup»;	website: cppstudio.com
boolalpha	Mantiqiy kattalikni matn ko‘rinishida chiqarish (true, false)	bool log_true = 1; cout << boolalpha << log_true <<endl;	true
noboolalpha	Mantiqiy kattalikni son ko‘rinishida chiqarish (true, false)	bool log_true = true; cout << noboolalpha << log_true <<endl;	1
oct	8 SSda sonlarni chiqarish	int value = 64; cout << oct <<value << endl;	1008
dec	10 SSda sonlarni chiqarish	int value = 64; cout << dec << value << endl;	6410
hex	16 SSda sonlarni chiqarish	int value = 64; cout << hex <<value << endl;	408
showbase	SS uchun asosiy indikatorni chiqarish	int value = 64; cout << showbase << hex << value <<endl;	0x40
noshowbase	SS uchun asosiy indikatorni chiqarmaslik (joriy holat).	int value = 64; cout << noshowbase << hex << value <<endl;	40
uppercase	16 SS da katta harflarni	int value = 255;	FF16
	chiqarish (joriy holatda kichik harflar)	cout << uppercase << hex << value <<endl;	
nouppercase	16 SS da kichik harflarni chiqarish (joriy holat)	int value = 255; cout << nouppercase << hex << value <<endl;	ff16
showpos	Sonlarni + belgisi bilan chiqarish	int value = 255; cout << showpos<< value << endl;	+255
noshowpos	Sonlarni + belgisiz chiqarish (joriy holat)	int value = 255; cout <<noshowpos<< value << endl;	255
scientific	Sonlarni eksponensial ko‘rinishda chiqarish	double value = 1024.165; cout << scientific << value << endl;	1.024165e+003

fixed	Fiksirlash holda sonlarni chiqarish (joriy holat).	double value = 1024.165; cout << fixed << value << endl;	1024.165
setw(int number)	Maydon uzunligini o‘rnatish, number — simvollarning pozitsiyalari soni, parameterli monupulyator	cout << setw(40) << “mbbGroup” << endl;	___mbbGroup
right	O‘ngdan tekslash	cout << setw(40) << right << «mbbGroup » <<endl;	___mbbGroup
left	Chapdan tekislash	cout << setw(40) << left<< «mbbGroup» << endl;	mbbGroup ____
setprecision(int count)	Verguldan keyi kalit belgi berish, count — verguldan keyingi belgilar soni	cout << fixed <<setprecision(3) << (13.5 / 2) <<endl;	6.750
setfill(int symbol)	Qo‘shimcha belgilarni o‘rnatish	cout << setfill(‘0′) << setw(4) << 15 << ends <<endl;	0015

C++ da buferlashtirilgan formatlangan KIRISH/chiqish dasturlashdagi eng oddiy mavzulardan biri hisoblanadi. Ma‘lum formatlash vositalaridan qanday foydalanish yuqoridagi jadvallarda ko‘rsatilgan, shuning uchun bu mavzuda hech qanday qiyinchilik bo‘lmasligi kerak.

Nazorat savollari

1. Polimorf sinf qanday xususiyatga ega bo‘lishi lozim?
2. Dastur bajarilishi davomida ob‘yektturini olish uchun qanday operatordan foydalaniladi?
3. type_info sinfda public modifikatori bilan qaysi statik operatorlar aniqlangan?
4. Type_info sinf a‘zolaridagi qanday operatordan foydalanish taqiqlangan?
5. Typeid operatorini havolali parametriga qo‘llash mumkinmi?
6. C++ tilida necha tipni almatirish operatorlari mavjud va ularni nomlari, ishlatish uslublarini ayting.
7. Dynamic_cast operatori dastur bajarilishi davomida qaysi turdagi quyish amalini bajaradi?
8. Dynamic_cast va const_cast tip almashtirish operatorlarining vazifalari, kamchiliklari va yutuqlari eslang.
9. O‘zgarmas e‘lon qilingan identifikatorning tipini almashtirish mumkinmi. Mumkin bo‘lsa qanday amalga oshiriladi. Mumikn bo‘lmasa, nima sababdan izohlayu bering.
10. Polimorf bo‘lmagan va bo‘lgan sinf ob‘yektining tipini almashtirish uchun qaysi operatorlardan foydalaniladi, har birini izog‘layu bering.
11. To‘rtta cast operatorlari dynamic_cast, const_cast, static_cast, reinterpret_cast bilan an‘anaviy tipni almashtirish amali (kasting) qanday farq mavjud.
12. Yangi nomlar fazosi yaratish uchun kerak, misollar bilan eslab tushuntirib bering.
13. Jamoaviy dasturlash vaqtida dasturdaga nomlarning ziddiyatlarni oldini olish uchun nima

vazifani amalga oshrish kerak va ziddiyatlarni bekor qilishning eng oddiy va professonal usulini tushuntiring.

14. Nomlar fazosini 2 ga ajratish mumkinmi, mumkin bo‘lsa, qanday va nima uchun nomlar fazosini 2 ga ajratish kerak bo‘ladi.

15. Using kalit so‘zni ishlatmasdan turib, nomlar fazosiga murojaat qilish mumkinmi? Mumkin bo‘lsa, misol bilan asoslab bering.

16. C++ tilida ekran va faylga IO dan tashqari kiritish va chiqarish vazifalarini lokalizatsiyalash sifatida qanday vositadan foydalanadigan qator funksiyalarni qo‘llab-quvvatlanadi.

17. Kiritish/chiqarish ob‘yektlari sifatida massivdan foydalanish uchun qanday header fayldan foydalanish lozim.

18. Massivga ma‘lumotlarni yozish va chiqish uchun qanday tipdan foydalaniladi.

19. Massivlar asosida eof() va get() funksiyalarning IO da foydalanish nima uchun zarur.

20. KIRISH/Chiqish indentifikatorni o‘rnatish va o‘chirish usullari aytib bering va misollar orqali asoslang.

21. Formatlash identifikatorlari nima uchun kerak va u qanday ishlatiladi.

22. Sanoq sistemaga asoslangan indikatorni chiqarish uchun qaysi formatlash identifikatoridan foydalanamiz.

23. Chap tomondan tekislab formatlashda qaysi formatlash identifikatoridan foydalanamiz.

24. Monipulyatorlar bilan formatlashning usullari nima uchun kerak va u qanday ishlatiladi.

25. Sanoq sistemaga asoslangan indikatorni chiqarish uchun qaysi monipulyatorlar bilan formatlashning usuldan foydalanamiz.

2-MA’RUZA
KONTEYNERLAR (KOLLEKTSIYALAR) (2 SOAT)

Reja:

1. STL kutubxonalari.
2. Konteyner sinflar.

Annotatsiya*: STL (Standard Template Library) kutubxonalari bilan tanishib, uning to‘plamlari va har to‘plamning vazifasi, ishlatish maqsadi, maxsus usullari va algoritmlarini, iterator va dinamik struktura xususiyatlarini, iteratorlarning turlarini, iteratorlarning usullaridan foydalanishni, barcha to‘plamlarda mavjud bo‘lgan asosiy usullarni, STL kutubxona algoritmlarini, barcha to‘plam elementlari tanlash va ularga ishlov berish usullari, saralash usullari, to‘plam a’zolari ustida ma’lum arifmetik amallarni bajarish usullari, pridekatlar, oqim xavfsizligi, konteyner sinflar, ketma-ket konteynerlar, assotsiativ konteynerlar, konteynerlarning xususiyatlari, konteyner sinflarning umumiy usullari, chiziqli konreynerlar (array, vector, deque, list, forward_list), konteynerlar birlashishlash funksiyalari ko‘rsatilgan.*

Kalit so‘zlar: *STL kutubxonasi, to‘plam, shablon, vector, list, map, set, multimap, multiset, string, wstring, strstream, satrli oqim, iterator, dinamik ma‘lumotlar tuzilmasi, pridekat, konteyner sinflar, ketma-ket konteynerlar, chiziqli konreynerlar (array, vector, deque, list, forward_list), bir aloqali ro‘yxat* <forward_list>*, ikki aloqali ro‘yxat (ikkilangan ro‘yxat) <list>, ikki tomonlama navbat <deque>, dinamik massiv (vektor) <vector>, statik massiv < array>, allocator.*

Dasturlash texnologiyalarining rivjlanishi dasturlash tillarini ishlab chiqaruvchilar oldiga juda jiddiy masallarni paydo qilmoqda. Shulardan biri bu dasturlash tillari uchun turli xil to‘plamlar bilan

ishlashdir. Masalan, xayotdan olib qaraydigan bo‘lsak, barcha narsalar qandaydir to‘plam, ammo qonuniyati har xil. Bu to‘plamlarni qanday dasturlash kerak degan muammo paydo bo‘ladi. Shuning uchun barcha ob‘yektga yo‘naltirilgan dasturlash tillarida STL, ya‘ni standart shablonlar kutubxonasi tushunchasi kiritilgan. STL (Standard Template Library) kutubxonalari, to‘plam turlari, xususiyatlari, usullari va funksiyalar, konteyner va iteratorlar bo‘yicha nazariy materiallarni keltiramiz.

STL (Standard Template Library) kutubxonalari. Shablon mexanizmlari C++ kompilyatoriga moslab qurilgan bo‘lib, dasturchilarga umumiy dasturlash yordamida dastur fragmentlarini qisqartirishga imkon beradi. Tabiiyki, bunday mexanizmlarni amalga oshiruvchi standart kutubxonalar ham mavjud. Bugungi kunda C++ dasturlash tilida eng samarali STL kutubxonasi hisoblanadi.

STL kutubxonasining ko‘plab tatbiqlari mavjud bo‘lib, ularning har biri aniq standart doirasida yaratilgan bo‘lsa - da, o‘z kengaytmalariga ega. Ammo bunday yondashuvning bir kamchiligi bor: dastur fragmentini har doim turli kompilyatorlar bilan bir xil tarzda ishlamaydi. Shuning uchun, dasturchi qanchalik mohirlik bilan kutubxonani yaratsa va foydalansa, o‘ziga xos bajarilishini tushunsa ham, imkon qadar an‘anaviy usullardan foydalanishni tavsiya qilamiz.

C++ dasturlash tilining kutubxonalaridan eng mashhur to‘plamlarni ko‘rib chiqaylik. Ularning har biri muhim vazifalarni hal qilinishi mumkin doirasi uchun o‘z shablon parametrlariga ega.

To‘plamlarni dastur fragmentida ishlatish uchun quyidagi fragmentdan foydalaniladi.

```
#include <T>
```

Bunda T – to‘plamning nomi.

Odatda quyidagi to‘plamlar ko‘p ishlatiladi.

vector – elementlar to‘plami, o‘lchamini o‘zgartish kerak bo‘lgan massivda saqlanadigan elementlar to‘plami (odatda ortib boradigan); Dasturga ulanish uslubi:

```
#include <vector>
```

list – elementlar to‘plami, elementlarni ikki tomonlama bog‘langan ro‘yxat sifatida saqlaydigan to‘plam; Dasturga ulanish uslubi:

```
#include <list>
```

map – elementlar to‘plami, har bir elementi shakli <const kalit, T> juftlikda saqlanadigan to‘plam, bu oddiy bir juftlik <kalit, qiymati> juftligi (har bir kalit bitta qiymati mos keladi). Kalit – taqqoslash amali uchun qiymatini tavsiflovchi ma‘lum bir qiymati. Juftlikda kalit tez qidirishni amalga oshirish imkonini beradi, kalit asosida tartiblashtirilgan shaklida saqlanadi. Tartiblashni amalga oshirish uchun oldindan tashkil qilish qonuniyatini aniqlab olish kerak. Dasturga ulanish uslubi:

```
#include <map>
```

set – elementlar to‘plami, faqat kalitlarning qiymati bo‘yicha tartiblangan to‘plamidir, ya‘ni taqqoslash amali qo‘llaniladigan, ammo takrorlanmaydigan qiymatlar — har bir kalit to‘plamda (inglizcha set-to‘plam degan ma‘noni beradi) faqat bir marta foydalaniladi; Dasturga ulanish uslubi:

```
#include <set>
```

multimap – map, juftlikda kalitlar unikal emas, takrorlanadigan to‘plam. Agar kalit bo‘yicha qidirsangiz, siz bitta qiymatni emas, balki bir xil kalit qiymatiga ega bo‘lgan elementlar to‘plamini olasiz. Dasturga ulanish uslubi:

```
#include <map>
```

multiset – set, juftlikda kalitlar unikal emas, takrorlanadigan to‘plam. Agar kalit bo‘yicha qidirsangiz, siz bitta qiymatni emas, balki bir xil kalit qiymatiga ega bo‘lgan elementlar to‘plamini olasiz. Dasturga ulanish uslubi:

```
#include<set>
```

Belgilar to‘plami bo‘lgan satr (qator)ni ham to‘plam sifatida qarash mumkin. Shuning uchun ixtiyoriy kutubxona satrlar bilan ishlash va ifodalash uchun o‘zining sinflarga ega. STLda satrlar ASCII va Unicode formatlarida ifodalanadi.

string — to‘plam, ASCII formatidagi bir baytli belgilar to‘plami; **wstring** — to‘plam, Unicode formatidagi ikki baytli belgilar to‘plami; Dasturga ulanish uslubi:

```
#include <string>
#include <wstring>
```

Shuningdek, belgilar to‘plami bo‘lgan, satrli oqimlarni ham to‘plam sifatida qarash mumkin. **strstream**-oddiy ma‘lumotlar tiplarini STL-string ko‘rinishada saqlashni tashkil qilish uchun ishlatiladi. Dasturga ulanish uslubi:

```
#include <strstream>
```

Dasturlarni tahlil qilishni shu sinfdan boshlaymiz.

2.1-dastur. Satrli oqim to‘plam sifatida ishlatish

```
#include "stdafx.h"
#include <iostream>
#include <strstream>
#include <string>
using namespace std;
        int main(){
    strstream xstr;
    for (int i = 0; i < 10; i++)
    {
        xstr << "Demo " << i << endl;
    }
    cout << xstr.str() << endl;
    string str;
```

```
        str.assign(xstr.str (), xstr.pcount()); cout <<
    str.c_str() << endl;
        system("pause");
    return 0;
}
```

2.1-dastur natijasi. Output

```
Demo 0
Demo 1
Demo 2
Demo 3
Demo 4
Demo 5
Demo 6
Demo 7
Demo 8
Demo 9
═══════════════════════════════════════¤¤¤¤lllllllyu■yu■
Demo 0
Demo 1
Demo 2
Demo 3
Demo 4
Demo 5
Demo 6
Demo 7
Demo 8
Demo 9
```

Satrli oqim - oqim oxirida bir null terminator bilan tugaydigan buferdir. Shuning uchun uning bo‘sh qolgan elemetlari ixtiyoriy belgi bilan to‘ldiriladi. Satrli oqimning haqiqiy qismini olish uchun pcount() elementlarni sanagichidan foydalanish mumkin. Keyin esa, satrli oqimning "haqiqiy qismi" olinadi va chop qilinadi.

Iteratorlar ham dinamik strukturaga ega to‘plamdir. Iterator dinamik ma‘lumotlar tuzilmalarini amalga oshirishda juda muhim tushunchadir. Tushunishimiz uchun, iteratorni ma‘lum bir cheklovlar bilan bir ko‘rsatkich sifatida abstrakt ko‘rinishda aniqlash mumkin. Sirasini aytganda, iterator umumiy tushuncha bo‘lib va bir ko‘rsatkich uchun ob‘yekt to‘plami bo‘ladi, ammo ko‘rsatkich bu iterator emas. Iterator sinfini quyidagicha qurib olish mumkin:

2.2-dastur. Iteratorning sinfni qurish.

```
class    Iterator{
     T*  pointer;
     public:
          T* GetPointer (){
               return this -> pointer;
          }
          void SetPointer (T* pointer){ this -
               > pointer = pointer;
          }
};
```

Iteratorning ba’zi bir formallashtirilgan ta’riflari:

Iteratorlar to‘plam elementlariga KIRISHni ta‘minlaydi. Har bir aniq STL sinfi uchun iteratorlar to‘plamda sinf ichida alohida aniqlanadi.

Iteratorlarning uch turi mavjud:

- (forward) iterator - to‘plamni kichik indeksdan kattaroq indeksga o‘tkazish uchun;

- reverse iterator - to‘plamni katta indeksdan kichikroq indeksli o‘tkazish uchun;
- random access iterator - to‘plamdan tasofidiy, har qanday yo‘nalishda tanlash uchun.

To‘plamning yarim elementlarini o‘chirishga doir misol keltiramiz.

2.3-dastur. To‘plamning yarim elementlarini o‘chirish.

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
void printInt (int number); int main (){
      vector<int> myVec; vector<int>::iterator first, last; for (long i=0; i<10; i++){
            myVec.push_back(i);
      }
      first = myVec.begin();
          last = myVec.begin() + 5; if (last >= myVec.end()){
            return - 1;
      }
```

```
      myVec.erase(first, last);
      for_each (myVec.begin(), myVec.end(), printInt); system("pause");
      return 0;
}
void printInt (int number)
{
 cout << number << endl;
}
```

2.3-dastur natijasi. Output

```
5
6
7
8
9
```

2.3-dasturdagi ba‘zi funksiyalarni keyinroq tushuntirib o‘tamiz. Shuni ham tushunish kerakki, to‘plamning qandaydir elementi uchun iterator olishda to‘plamni ketma-ket o‘zgartirish iteratorni yaroqsiz holga keltirish mumkin.

Iteratorning iteratsiyasida oldinga va shunga o‘xshash orqaga o‘tish quyidagicha fragmenti asosida sodir bo‘ladi:

```
for (iterator element = begin(); element < end(); element++)
{ t = (*element); }
for (iterator element = end(); element < begin(); element--)
{ t = (*element); }
```

Tasodifiy tanlash iteratoridan foydalanganda, masalan, quyidagicha fragmentni yozish mumkin:

```
for (iterator element = begin(); element < end(); element+=2)
  { t = (*element); }
```

Barcha to‘plamlarda mavjud bo‘lgan asosiy usullar quyidagi 2.1-jadvalga keltiramiz.

2.1-jadval. To‘plam usullari.

№	nomi	vazifasi
1	empty	To‘plamni bo‘shligini tekshiradi
2	size	To‘plamning o‘lchamini qaytaradi
3	begin	To‘g‘ri iteratorning birinchi elementini ko‘rsatadi
4	end	To‘g‘ri iteratorning oxirgi elementini ko‘rsatadi. Elementi yo‘q to‘plamga oxirgisidan keyinga o‘tadi
5	rbegin	Teskari iteratorning birinchi elementini ko‘rsatadi
6	rend	Teskari iteratorning oxirgi elementini ko‘rsatadi.
7	clear	To‘plamni tozalash, barcha elementlarini o‘chiradi.
8	erase	Ajratilgan elementlarni to‘plamdan o‘chiradi.
9	capacity	To‘plamning sig‘imini qaytaradi, ya‘ni bu to‘plam uchun mumkin bo‘lgan elementlar soni (aslida to‘plam uchun qancha xotira ajratilganini qaytaradi)

To‘plamning sig‘imi (hajmi), boshida aytib o‘tilganidek, kerak bo‘lganda o‘zgaradi, ya‘ni, agar to‘plam uchun ajratilgan barcha xotiralar to‘lgan bo‘lsa, unda yangi element qo‘shilganda, to‘plamning sig‘imi oshiriladi va o‘sishdan oldin undagi barcha qiymatlar yangi xotira maydoniga ko‘chiriladi - bu dasturchilar uchun juda qimmat amal bo‘lib hisoblanadi. Dasturlashda to‘plamning hajmi va quvvati ishonch hosil qilish muhim dasturlash element hisoblanadi. To‘plamning sig‘imi turlicha ekanligiga ishonch hosil qilish uchun quyidagi dasturni keltiramiz.

2.4-dastur. To‘plamning sig‘imini tekshirish.

```
#include   "stdafx.h"
#include <iostream>
#include <vector>

using namespace std;
int main(){
vector<int> vec;
cout << "Real size of array in vector: " << vec.capacity() << endl; for
(int j = 0; j < 25; j++){
      vec.push_back (10);
}
cout << "Real size of array in vector: " << vec.capacity() << endl;
system("pause");
return 0;
}
```

2.3-dastur natijasi. Output

```
Real size of array in vector: 0
Real size of array in vector: 28
```

vector – eng ko‘p ishlatiladigan to‘plam vektor hisoblanadi. Bu to‘plamning operator[] operatoriga ega ekanligi juda qulay. Odatiy massiv kabi ishlatiladi. Xuddi shuningdek, bu operator map, deque, string i wstring to‘plamlariga ham mavjud.

Vektorning quvvati dinamik ravishda o‘zgarishini tushunish muhimdir. Odatda, multiplikativ yondashuv hajmini oshirish uchun ishlatiladi: zarur bo‘lsa, vektor uchun ajratilgan xotiraning marta soni ortadi, ya‘ni, yangi element qo‘shib to‘plam sig‘imining oshirishga sabab bo‘lsa, operatsion tizimi dasturi uchun yangi xotira maydoni ajratadi. Masalan, ikki barobar katta bo‘lish uchun, eski xotira maydoni barcha elementlari nusxasi yangi qiymat qilib qo‘shish.

STL kutubxona algoritmlari. STL kutubxonasini ishlab chiquvchilar shablonli ma‘lumotlar tuzilmalari majmuasiga ega kutubxona yaratishda o‘zlariga ancha jiddiy vazifa qo‘yishgan. STL kutubxonasi to‘plamlari bilan ishlash imkonini beradigan, mashhur algoritmlarni optimal tatbiqlari va katta majmuini o‘z ichiga oladi. Barcha amalga oshirilgan funksiyalarni uch guruhga bo‘lish mumkin:

1. Barcha to‘plam elementlari tanlash va ularga ishlov berish usullari:

```
sount, count_if, find, find_if, adjacent_find, for_each, mis match, equal,
search,copy, copy_backward, swap, iter_swap, swap_ranges, fill, fill_n,
generate, generate_n, replace, replace_if, transform, remove, remove_if,
remove_copy, remove_copy_if, unique, unique_copy, reverse, reverse_copy,
rotate, rotate_copy, random_shuffle, partition, stable_partition
```

Bu usularni vazifalarini ixtiyoriy manbadan, splusplus.com veb sahifasidan foydalanib bilish mumkin.

2. Saralash usullari:

```
sort, stable_sort, partial_sort, partial_sort_copy, nth_element,
binary_search, lower_bound, upper_bound, equal_range, merge, inplace_merge,
includes, set_union, set_intersection, set_difference,
set_symmetric_difference, make_heap, push_heap, pop_heap, ‘s ort_heap, min,
max, min_element, max_element, lexographical_compare, next_permutation,
prev_permutation
```

Bu usularni vazifalarini ixtiyoriy manbadan, splusplus.com veb sahifasidan foydalanib bilish mumkin.

3. To‘plam a‘zolari ustida ma‘lum arifmetik amallarni bajarish usullari:

```
Accumulate, inner_product, partial_sum,
adjacent_difference
```

Bu usularni vazifalarini ixtiyoriy manbadan, splusplus.com veb sahifasidan foydalanib bilish mumkin. Ushbu usularni sanab o‘tishdan maqsad STL kutubxonasi tomonidan taqdim etilgan boy vositalar to‘plami bilan tanishtirishdir. Qo‘shimcha ma‘lumot olish uchun, C++ dasturlash tiliga oid sinflarning tegishli hujjatlarida tanishib chiqish mumkin.

Pridekatlar. Ko‘pgina STL kutubxona algoritmlari uchun algoritm to‘plamning muayyan a‘zosi bilan nima qilish kerakligini aniqlaydigan shartni o‘rnatishingiz lozim bo‘ladi. Predikat - bu funksiya, bir necha parametrlarni oladi va Boolean qiymatini qaytaradi (true/false). Standart predikatlar to‘plami ham mavjud.

Oqim xavfsizligi. Bu STL butunlay xavfsiz kutubxona emasligini tushunish muhim ahamiyatga ega. Lekin bu muammoni hal qilish juda oddiy: ikki oqimlar bir xil to‘plamdan foydalanayotgan bo‘lsa, Mutex seksiyasini amalga oshirish zarur.

STL cross-platform kutubxona hisoblanadi. Albatta, ushbu kutubxona kompilyatorning har qanday versiyasi uchun mavjudligiga mutlaq kafolat yo‘q. Masalan, u kamdan-kam hollarda mobil qurilmalarda amalga oshiriladi, chunki amalga oshirilgan ma‘lumotlar tuzilmalarining aksariyati xotirani tejamasdan, tezlik foydasini tanlaydi hamda xotira mobil platformalarda eng qimmatli texnik resursdir, kompyuterda esa u hozir juda ko‘p. Shuning uchun tez-tez o‘z STL kutubxonangizni lokalizatsiyasini yaratish kerak bo‘ladi, masalan, ilovasini mobil platformaga ko‘chirish uchun.

Konteyner sinflar. Konteyner sinflar muayyan tarzda tashkil qilingan ma‘lumotlarni saqlash uchun mo‘ljallangan sinflar. Turli xil tipdpgi ma‘lumotlarni saqlash uchun bir xil turdagi konteynerdan foydalanishingiz mumkin. Bu xususiyat sinf shablonlari yordamida amalga oshiriladi, shuning uchun C++ kutubxonasining konteyner sinflarini, shuningdek algoritmlarni va iteratorlarni o‘z ichiga olgan qismi standart shablonlar kutubxonasi (STL) deb ataladi.

Ma’lumotlar konteynerlarda saqlanadi va ular bilan turli amallar konteyner usullari va moslanuvchan algoritmlar bilan aniqlanadi va bajariladi. Iteratorlar bu ikki elementni bir-biriga bog‘lagan holda ishlaydi. Ular tufayli har qanday algoritm har qanday konteyner bilan ishlashi mumkin.

Professonal dasturlashni kutubxona sinflarisiz foydalanishni tasavvur qilish mumkin emas, shuningdek alohida konteynerlarsiz ham. Ulardan foydalanish dasturlarning ishonchliligi, joriy qilish samaradorligi, moslashuvchanligi va ko‘p qirraliligini oshirish hamda dastur tuzish vaqtini kamaytirishga imkonini beradi. Kutubxonani yaratish ko‘p ish va mashaqqat talab qiladi, amao, dastur yaratish vaqtida o‘zini oqlaydi.

STL kutubxonasi dasturlarni yozishda ishlatiladigan asosiy ma‘lumotlar tuzilmalarini amalga oshiruvchi konteynerlarni o‘z ichiga oladi: vektorlar, navbatlar, ro‘yxatlar, lug‘atlar va to‘plamlar. Konteynerlarni ikki turga bo‘lish mumkin: ketma-ket va assotsiativ konteynerlar .

Ketma-ket konteynerlar. Ular uzluksiz ketma-ketlikda o‘xshash miqdorlarning chekli sonini saqlashni ta‘minlaydi. Konteynerlar sifatida vektor (vector), ikki tomonlama navbat (deque), ro‘yxat (list) va bir aloqali ro‘yxati (forward_list), shuningdek konteyner variantlar asosida adapterlar, stek (stack), navbat (queue) va ustuvorlik bilan navbat (priority_queue) sinflarini o‘z ichiga oladi. Massiv ham amallar bilan cheklangan holda konteynerning yana bir turidir.

Konteynerning har bir turi ma‘lumotlar ustida o‘z amallar to‘plamini ta‘minlaydi. Siz tanlagan konteyner turi dasturdagi ma‘lumotlar bilan nima qilishni xohlashingizga bog‘liq. Masalan, agar ketma-ketlik o‘rtasida ma‘lumotlar tez - tez joylashtirish va o‘chirish kerak bo‘lsa, ro‘yxatlardan foydalanish kerak, ma‘lumotlarni oxirida yoki boshida, birinchi navbatda joylatrish kerak bo‘lsa, ikki tomonlama navbatdan foydalanish maqsadga muvofiq.

Assotsiativ konteynerlar. Assotsiativ konteynerlar asosiy ma‘lumotlarga kalitlar asosida tezkor murojaat qilishni ta‘minlaydi. Bu konteynerlar muvozanatli daraxtlarga asoslangan. Assotsiativ konteynerlarning besh turi mavjud: lug‘atlar (map), ko‘p lug‘atlar (multi) (multimap), to‘plamlar (set), multi to‘plamlar (multiset) va bitli to‘plam (bitset).

Dasturchi standart kutubxonada mavjud bo‘lgan sinflarga asoslanib o‘z konteyner sinflarini yaratishi mumkin. Bunga KIRISHishdan oldin muhim tushunchani bilishinggiz shart, ya‘ni STL kutubxonasining fundamental tushunchasi bu shablondir

Barcha konteyner sinflari standartlashtirilgan interfeysi bilan ta‘minlangan. Turli konteynerlar uchun bir xil amallarning ma‘nosi bir xil bo‘lishi tabiiydir. Asosiy amallar konteynerlarning barcha turlari uchun qo‘llaniladi. Standart esa faqat konteyner interfeysini belgilaydi, shuning uchun turli xil dasturlar samaradorlikda katta farq qilishi mumkin. Barcha konteynerlar o‘z xotirasini o‘zi boshqaradi, shuning uchun dasturchi bu haqida o‘ylashning hojati yo‘q.

Deyarli har qanday konteynerlarning quyidagi xususiyatlari bor:

2.2-jadval. Konteyner xususiyatlari.

№	Xususiyat nomi	mazmuni
1	value_type	Konteyner elementining tipi
2	size_type	Elementlar indeksining tipi
3	iterator	Iterator
4	const_iterator	O‘zgarmas iterator
5	reverse_iterator	Teskari iterator
6	const_reverse_iterator	O‘zgarmas teskari iterator
7	reference	Elementga havola
8	const_reference	O‘zgarmas elementga havola
9	key_type	Kalit tipi (assotsiativ konteynerlar uchun)
10	key_compare	Taqqoslash mezoni (assotsiativ konteynerlar uchun)

Iterator bir element uchun ko‘rsatkichga ekvivalentidir. Ular konteynerlarni to‘g‘ri yoki teskari yo‘nalishda ko‘rish uchun ishlatiladi. Iteratordan talab qilinadigan barcha amallar konteyner elementiga murojaat qilish va uning keyingi elementiga o‘tish amalini amalga oshirishdir. Konteynerlarning elementlarining qiymatlari o‘zgarmaganda o‘zgarmas iteratorlardan foydalaniladi.

Iteratorlar yordamida haqiqiy ma‘lumotlar tiplari haqida o‘ylamasdan, konteyner elementlarga murojaat qilish uchun foydalanish mumkin. Buning uchun, har bir konteyner quyidagi 2.3-jadvalda keltirilgan bir necha usullardan foydalanishni tavsiya qilinadi. Har bir konteyner uchun bu tiplar va usullarni, ularni realizatsiyasini amalga oshirishga bog‘liq tarzda belgilanadi. Shuningdek, ixtiyoriy konteynerlarda ularning hajmi haqida ma‘lumot olish uchun usullari mavjud:

size()– elementlar soni;

max_size() – konreynerning maksimal o‘lchami (1 miliard ta element uchun);

empty()– mantiqiy usuli, konteyner bo‘shligini tekshiradi;

Zaruriyat bo‘lganda dasturchi konteynerning maydonlari va usullari ketma- ket o‘zlashtirib boraveradi. Biz ham zaruriyat tug‘ilganda foydalanamiz.

2.3-jadval. Konteyner sinflarning umumiy usullari.

№	Usullar	Izoh
1	iterator begin() const_iterator begin () const	Birinchi elementni ko‘rsatadi
2	iterator end() const_iterator end () const	Oxirgisidan keyingi elementni ko‘rsatadi
3	reverse_iterator rbegin() const_reverse_iterator rbegin () const	Teskari ketma-ketlikda birinchi elementni ko‘rsatadi
4	reverse_iterator rend() const_reverse_iterator rend () const	Teskari ketma-ketlikda oxirgisidan keyingi elementni ko‘rsatadi

3-MA’RUZA
CHIZIQLI KONTEYNERLAR (2 SOAT)

Reja:

1. Chiziqli konteynerlar.
2. Array, vector, deque.
3. List, forward_list.

Annotatsiya*: STL (Standard Template Library) kutubxonasiga tegishli bo’lgan chiziqli konteynerlar bilan tanishib, uning to‘plamlari va har to‘plamning vazifasi, ishlatish maqsadi, maxsus usullari va algoritmlarini, iterator va dinamik struktura xususiyatlarini, iteratorlarning turlarini, iteratorlarning usullaridan foydalanishni, barcha to‘plamlarda mavjud bo‘lgan asosiy usullarni, STL kutubxona algoritmlarini, barcha to‘plam elementlari tanlash va ularga ishlov berish usullari, saralash usullari, to‘plam a’zolari ustida ma’lum arifmetik amallarni bajarish usullari, pridekatlar, oqim xavfsizligi, konteyner sinflar, ketma-ket konteynerlar, assotsiativ konteynerlar, konteynerlarning xususiyatlari, konteyner sinflarning umumiy usullari, chiziqli konreynerlar (array, vector, deque, list, forward_list), konteynerlar birlashishlash funksiyalari ko‘rsatilgan.*

Kalit so‘zlar: *STL kutubxonasi, to‘plam, shablon, vector, list, string, wstring, strstream, satrli oqim, iterator, dinamik ma‘lumotlar tuzilmasi, pridekat, konteyner sinflar, ketma-ket konteynerlar,*

chiziqli konreynerlar (array, vector, deque, list, forward_list), bir aloqali ro'yxat <forward_list>*, ikki aloqali ro'yxat (ikkilangan ro'yxat) <list>, ikki tomonlama navbat <deque>, dinamik massiv (vektor) <vector>, statik massiv < array>, allocator.*

KIRISH

Ketma-ket konteynerlar. Ular uzluksiz ketma-ketlikda o'xshash miqdorlarning chekli sonini saqlashni ta'minlaydi. Konteynerlar sifatida vektor (vector), ikki tomonlama navbat (deque), ro'yxat (list) va bir aloqali ro'yxati (forward_list), shuningdek konteyner variantlar asosida adapterlar, stek (stack), navbat (queue) va ustuvorlik bilan navbat (priority_queue) sinflarini o'z ichiga oladi. Massiv ham amallar bilan cheklangan holda konteynerning yana bir turidir.

Konteynerning har bir turi ma'lumotlar ustida o'z amallar to'plamini ta'minlaydi. Siz tanlagan konteyner turi dasturdagi ma'lumotlar bilan nima qilishni xohlashingizga bog'liq. Masalan, agar ketma-ketlik o'rtasida ma'lumotlar tez - tez joylashtirish va o'chirish kerak bo'lsa, ro'yxatlardan foydalanish kerak, ma'lumotlarni oxirida yoki boshida, birinchi navbatda joylatrish kerak bo'lsa, ikki tomonlama navbatdan foydalanish maqsadga muvofiq.

Chiziqli konreynerlar (array, vector, deque, list, forward_list). Standart konteynerlarni ikkita katta guruhga chiziqli va assotsiativ konteynerlarga bo'lishini bilamiz. O'z navbatida chiziqli konteynerlarni bog'langan ro'yxat (forward_list va list) va tasodifiy KIRISH konteynerlariga (ekrin konteyner deb ham yuritiladi) (deque, vector va array) bo'linadi.

Assotsiativ konteynerlar quyidagi variantlarning kombinatsiyasi bo'lgan sakkizta konteyner bilan ifodalanadi (tegishli standart sinflar nomlarining qavslar ichida beriladi): to'plam (*set) yoki lug'at (*map), elementlarni takrorlashga imkon beruvchi (*multi*) yoki ruxsat bermaydigan, tartibga solingan solinmagan (unordered*).

Barcha konteynerlar iterator va const_iterator tiplari o'z ichiga olish uchun o'z navbatida o'qish-yozish va faqat o'qish iteratorlarini aniqlaydi. Konteyner tarkibida olgan iteratorlar oralig'ini begin() va end() (o'qish-yozish uchun iterator), shuningdek cbegin() va cend() (o'qish uchun const_iterator) funksiyalari yordamida olish mumkin. Konteynerlar ikki tomonlama iteratorlar bilan ham teskari o'tish uchun rbegin(), rend(), crbegin() va crend() funksiyalari mavjud.

Barcha konteynerlarni empty funksiyasi tomonidan bo'sh uchun tekshirish mumkin. Qachonki cont.begin() == cont.end() bo'lsa, o'zgarmas konteyner bo'sh hisoblanadi. end() konteynerning oxirgi elementi keyingi shartli elementga havola qiluvchi iteratorni qaytaradi. Elementlar sonini size() funksiyasi yordamida olish mumkin (forward_list bundan mustasno). Konteyner tarkibidagi funksiyalarni tozalash uchun clear() funksiyasidan foylalaniladi (array bundan mustasno).

Iteratorda begin(), end(), cbegin(), cend(), rbegin(), rend(), crbegin() va crend() erkin funksiyalarining shablonlarini aniqlash mumkin. Bular bir xil nomli funksiyalarga havola qilinadi, shuningdek, bunday aniqlangan funksiyalar statik massiv va std::valarray uchun aniqlandi. for(:) takrorlanish operatori xuddi shanday shakliga tayanadi. Masalan, cr bir konteyner bo'lsin (shuningdek, statik massiv ham bo'lishi mumkin), shunda dasturda for() fragmenti quyidagicha ishlatiladi.

```
for (T x : cr) work(x);
```

Semantik jihatdan quyidagi fragmentga teng

```
{
    using std::begin; using
    std::end;
    const auto e = end(cr);
    for (auto p = begin(cr); p != e; ++p)
    { T x = *p; work(x); }
}
```

Bu yerda T tipi konteyner elementlarining tipiga mos kelishi shart emas va auto asosida konstruksiya yoki havolali tip ham bo‘lishi mumkin (eng ko‘p ishlatiladigan variantlar auto& va auto&&).

Chiziqli konteynerlarni (array dan boshqa) belgilangan qiymatlardan assign() funksiyasini chaqirib to‘ldirish mumkin (eski qiymalari o‘chiriladi) va resize() funksiyasi bilan o‘lchamlarini (hajmi kamayganda elementlar oxiridan o‘chiriladi va hajmi ortganda, yangi elementlar oxiridan qo‘shiladi) aniqlash mumkin.

Konteynerning birinchi elementiga to‘g‘ridan-to‘g‘ri murojaat (tartiblanmagan assotsiativ konteynerlardan tashqari) front() funksiya yordamida amalga oshiriladi. Iteratorlari ikki tomonlama bo‘lgan barcha konteynerlarning oxirgi elementga murojaat uchun back() funksiyasi, shuningdek, teskari yo‘naltirilgan iteratorlar reverse_iterator va const_reverse_iterator funksiyalaridan foydalaniladi. Tegishli intervallarni rbegin(), rend() i crbegin(), crend() funksiyalari yordamida olish mumkin.

Barcha konteynerlarni tenglik va tengsizlik uchun taqqoslash mumkin va ularning mazmunini swap() funksiyasi yordamida almashtirish mumkin. Chegaralanmagan assotsiativlardan tashqari barcha konteynerlarni <, <= , > va >= operatorlari leksikografik jihatdan taqqoslash mumkin.

Standart konteynerlarda dinamik xotirani boshqarish uchun Allocator ajratuvchi maxsus sinflardan foydalanadi. Allocator xotirani boshqarishning minimal birligini belgilaydigan va bir qator yordamchi ta‘riflarni taqdim etadigan element tipiga bog‘liqdir. Bu vazifa to‘rt asosiy funksiyalari yordamida amalga oshiriladi: elementlari berilgan qator uchun xotira ajratishda allocate, xotirani tozalash uchun deallocate, konstruktor qurish uchun construct, va destruktor uchun destroy. Allokator boshqa tipdagi elementlar uchun analog allokator olishda rebind metafunksiyasini amalga oshirishi kerak.

Alloc – elementlarning aniq tiplari uchun allokator berilgan bo‘lsin. U tipli elementlar uchun uning allokatorini olish varianti quyidagi dastur fragmentida keltirilgan.

```
using AllocForU = typename Alloc::template rebind<U>::other;
```

C++da standart kutubxona (<memory> sarlavha fayl bilan) new/new[] va delete/delete[] operatorlarini qo‘llash orqali allocator<T> ni ta‘minlaydi. Bundan model va o‘zingizning allokatorlaringgizni yozishda foydalanish mumkin.

Bir aloqali ro‘yxat <forward_list>. forward_list<T, a = allocator<T>> bir tomonlama iterator orqali T tipdagi elementlarga KIRISHni ta‘minlaydi. Ro‘yxatning tugunlarini yaratish uchun A::rebind orqali yaratilgan allokatorday foydalaniladi.

Bir aloqali ro‘yxatning o‘ziga xosligi shundaki, faqat belgilangan joriy tugundan keyin elementlarni qo‘shish va o‘chirish mumkin:

insert_after - element qo‘shish.

emplace_after – yangi element yaratish. Bunda belgilangan parametrlar uchun konstruktor chaqiriladi.

erase_after - element o‘chirish.

Birinchi elementdan oldin degan ko‘shimcha aniqlangan pozitsiya mavjud. Bu before_begin va cbefore_begin (const_iterator qaytaruvchilar uchun variant) funksiyalari yordamida bajariladi. Shuningdek, fl.push_front(item) funksiyasidan foydalanib, elementlarni oldindan joylashtirish mumkin (yuqoridagiga ekvivalent), masalan, quyidagicha dastur fragmenti:

```
fl.insert_after(fl.before_begin(), item)
```

Konstruktorga har qanday parametrlarni joylashtirish uchun fl.emplace_front(...) funksiyasi ishlatiladi. Bunga ekvivalent sifatida quyidagi dastur fragmentini yozish mumkin:

```
fl.emplace_after(fl.before_begin(), ...)
```

pop_front funksiyasi ro‘yxatdan birinchi elementni o‘chiradi.

C++ standart kutubxonasida ro‘yxat konteynerlar xususiyati uchun samarali bo‘lgan amallarni bajarishga faqat iteratorlarda foydalanishga ishonchsizligi yuqori darajali amallarni qo‘llab-quvvatlash hisoblanadi:

merge - ikkita tartiblangan ro‘yxatni bir-biriga birlashtiradi, elementlar ko‘chirilmaydi lekin o‘ngdan chap ro‘yxatga o‘tiladi

splice_after - berilgan ro‘yxatni ko‘rsatilgan elementdan keyin joylashtiradi.

remove – berilgan elementga teng bo‘lgan barcha elementlarni o‘chiradi. remove – berilgan pridikat asosida barcha elementlarni o‘chiradi. reverse – elementlarni tartibiga murojaat qiladi.

unique - barcha ketma-ket dublikatlarni o‘chiradi. sort - ro‘yxatni joyida tartiblaydi.

Umuman olganda, bu funksiyalar qandaydir standart algoritmlarga o‘xshaydi, lekin juda tez va qulay ishlaydi. Ularga murojaat qilish uchun umumiy ruxsat olish kerak, masalan, sort() funksiyasi uchun std::sort(from, to). Ammo, tasodifiy KIRISH iteratorlari kabi talab qilingan ro‘yxatlar uchun amal qilmaydi.

Ikki aloqali ro‘yxat (ikkilangan ro‘yxat) <list>. list<T, a = allocator<T>> ikki tomonlama iterator orqali T tipdagi elementlarga KIRISHni ta‘minlaydi. Bir aloqali ro‘yxatdan farqli o‘laroq, elementlarni belgilangan pozitsiyadan oldin joylashtirildi (insert, emplace, splice funksiyalari), oxiridan qo‘shish va o‘chirish (push_back, emplace_back, pop_back), va iterator ko‘rsatayotgan elementni o‘chirish (erase) funksiyalari bilan amalga oshiriladi. *_after funksiyasi mavjud emas.

Ikki tomonlama navbat <deque>. deque<T, A = allocator<T>> , maxsus KIRISH iteratori orqali elementlar uchun KIRISH imkonini beradi. Faqat ro‘yxati kabi, u samarali qo‘shish va har ikki tomonidan ma‘lumotlar olib tashlash imkonini beradi. Indeksga KIRISH uchun ikkita funksiya ishlatiladi: [] operatori va at(indeks). Birinchisidan farqli o‘laroq, ikkinchisi indeksni tekshiradi va qiymati haqiqiy bo‘lmasa, out_of_range istisnoga murojaat qiladi.

Ikki tomonlama navbat konteyneri uchun ro‘yxatni bir xil tarzda har qanday holatda elementlarni kiritish va o‘chirish imkonini beradi. Lekin ikki tomonlama navbatda bu amallar ular konteyner hajmi, vaqti chiziqli bo‘lishini talab qilishi mumkin. Bundan tashqari, oldindan iteratorga saqlangan elementlarni tartibini kiritish va o‘chirishni buzishi mumkin shuning uchun xotirada saqlangan elementlarning ko‘rsatkichlarini yodda saqlash maqsadga muvofiq (Agar iteratorlar ro‘yxat kabi saqlanayotgan bo‘lsa, elementlarga bo‘lgan ko‘rsatkichlar o‘chirilmaydi).

Dinamik massiv (vektor) <vector>. Vektor<T, a = allocator <T>> elementlarga maxsus KIRISH iteratori orqali KIRISHni ta‘minlaydi. Deque dan farqli o‘laroq, u kiritish va boshida elementlarni o‘chirish uchun ruxsat bermaydi. Dinamik massiv xotiraning uzluksiz bo‘limida ketma-ket saqlangan elementlarning joylashishini kafolatlaydi, uning manzili data funksiyasi bilan qaytariladi.

Elementlarni qo‘shish va oldindan ajratilgan saqlashni tugatishda elementlar ko‘chiriladigan joyga katta hajmdagi yangi dinamik qator ajratish mumkin. Eski saqlab o‘chiriladi, va barcha saqlangan iteratorlar yo‘qoladi, o‘chirilgan ob‘yektlar uchun ko‘rsatkich bo‘lib qoladi.

Dinamik massiv reserve funksiyasi yordamida yetarli hajmda saqlash uchun oldindan massivni tayyorlash imkonini beradi (u ko‘chib o‘tishda elementlarni olib kelishi mumkin). Saqlash hajmini capacity funksiyasi yordamida bilib olishingiz mumkin. shrink_to_fit funksiyasi haqiqiy ishlatiladigan hajmini ajratadi va elementlarni foydalanilmagan xotiradan olib tashlaydi.

Statik massiv < array>. array<T, N> iteratorning tasodifiy KIRISHi orqali elementlarga KIRISH imkonini beradi va statik array T[N] to‘plami hisoblanadi, kerakli manzili data funksiyasi tomonidan olinishi mumkin. T[N] massivdan farqli ravishda array<T, N> - qiymatlari ko‘chirilishi (qiymat bo‘yicha funksiyalarga o‘tishi) mumkin bo‘lgan va qiymatlari avtomatik ravishda ko‘rsatgichlarga aylantirilmaydigan to‘la huquqli tipdir. array < T, N> dan tayanch sinf sifatida foydalanish mumkin (lekin ehtiyotkorlik bilan, faqat har qanday konteyner kabi, standart konteynerlar bu maqsad uchun mo‘ljallangan emas, chunki, xususan, virtual funksiyalarni o‘z ichiga olmaydi). deque va vector kabi bir xil tarzda indeksi tomonidan elementlarga murojaat kilish mumkin. Elementlar sonini o‘zgartirish mumkin emas, shuning uchun array elementlarni kiritish yoki o‘chirish uchun hech qanday funksiyalar aniqlanmaydi. fill funksiyasi massivni qiymat nusxalari bilan to‘ldiradi.

1.1-jadval. Ketma-ket konteynerlar uchun asosiy funksiyalar

Headers		**<array>**	**<vector>**	**<deque>**	**<forward_list>**	**<list>**
Members		**array**	**vector**	**deque**	**forward_list**	**list**
	constructor	*implicit*	vector	deque	forward_list	list
	destructor	*implicit*	~vector	~deque	~forward_list	~list
	operator=	*implicit*	operator=	operator=	operator=	operator=
iterators	begin	begin	begin	begin	begin before_begin	begin
	end	end	end	end	end	end
	rbegin	rbegin	rbegin	rbegin		rbegin
	rend	rend	rend	rend		rend
const iterators	cbegin	cbegin	cbegin	cbegin	cbegin cbefore_begin	cbegin
	cend	cend	cend	cend	cend	cend
	crbegin	crbegin	crbegin	crbegin		crbegin
	crend	crend	crend	crend		crend

capacity	size	size	size	size		size
	max_size	max_size	max_size	max_size	max_size	max_size
	empty	empty	empty	empty	empty	empty
	resize		resize	resize	resize	resize
	shrink_to_fit		shrink_to_fit	shrink_to_fit		
	capacity		capacity			
	reserve		reserve			
element access	front	front	front	front	front	front
	back	back	back	back		back
	operator[]	operator[]	operator[]	operator[]		
	at	at	at	at		
modifiers	assign		assign	assign	assign	assign
	emplace		emplace	emplace	emplace_after	emplace
	insert		insert	insert	insert_after	insert
	erase		erase	erase	erase_after	erase
	emplace_back		emplace_back	emplace_back		emplace_back
	push_back		push_back	push_back		push_back
	pop_back		pop_back	pop_back		pop_back
	emplace_front			emplace_front	emplace_front	emplace_front
	push_front			push_front	push_front	push_front
	pop_front			pop_front	pop_front	pop_front
	clear		clear	clear	clear	clear
	swap	swap	swap	swap	swap	swap
list operations	splice				splice_after	splice
	remove				remove	remove
	remove_if				remove_if	remove_if
	unique				unique	unique
	merge				merge	merge
	sort				sort	sort
	reverse				reverse	reverse
observers	get_allocator		get_allocator	get_allocator	get_allocator	get_allocator
	data	data	data			

Nazorat savollari

1. STL sinfi uchun iteratorlar to‘plamda sinfda kanday aniqlanadi va turlari nechta? Har bir turini tushuntirib bering?
2. Iteratorda begin(), end(), cbegin(), cend(), rbegin(), rend(), crbegin() va crend() erkin funksiyalari nimalarini aniqlashi mumkin?
3. Chiziqli konteynerlarni belgilangan qiymatlardan qaysi funksiyasini chaqirib to‘ldirish mumkin?
4. Allocator xotirani boshqarishning minimal birligini belgilaydigan va bir qator yordamchi ta‘riflarni taqdim etadigan element tipiga bog‘liqdir. Bu vazifa nechta va qanday asosiy funksiyalari yordamida amalga oshiriladi?
5. Bir aloqali ro‘yxatning nimalarini yaratish uchun A::rebind orqali yaratilgan allokatorday foydalaniladi?
6. Qaysi funksiyasidan foydalanib, elementlarni oldindan joylashtirish mumkin.
7. Ikki aloqali ro‘yxat va ikki tomonlama navbatning farqlarini sanab bering?
8. Dinamik va statik massivlarning o‘xshash tomonlarini misollar yordamida tushuntirib bering?
9. Staiik massiv elementlarni kiritish yoki o‘chirish uchun qanday funksiyalar aniqlangan.

4- MA’RUZA

ASSOSIATIV VA TARTIBLANMAGAN ASSOSIATIV KONTEYNERLAR (2 SOAT)

Reja:

1. Assosiativ konteynerlar
2. Set, map, multiset, multimap.
3. Tartiblanmagan assosiativ konteynerlar (unordered_set, unordered_map, unordered_multiset, unordered_multimap).

***Annotatsiya**: Assosiativ konteynerlarning asosiy foydalanish ko‘nikmalari, assosiativ konteynerlar (set, map, multiset, multimap) va tartiblanmagan assosiativ konteynerlar (unordered_set, unordered_map, unordered_multiset, unordered_multimap) bilan yaqindan tanishib, ularning xususiyatlari, funksiyalari, sinflari, yaratish va foydalanish uslublari, amallari, talablari, vazifalari va usullari keltirilgan.*

Kalit so‘zlar. *to‘plam, shablon, map, set, multimap, multiset, unordered_set, unordered_map, unordered_multiset, unordered_multimap, satrlioqim, iterator, dinamik ma’lumotlar tuzilmasi, pridekat, konteynersinflar.*

KIRISH

Konteynerlar va ketma-ket konteynerlar bo‘yicha nazariy va amaliy imkoniyatlarini bilamiz. Shuning uchun konteynerlarning ikkinchi turi bo‘lgan assosiativ konteynerlarni boshqa konteynerlar farqli bo‘lishi kerak. Assosiativ konteynerlari kalit asoisda ma‘lumotlarni to‘plamdan tez izlab topish uchun ishlatiladi. Tartiblangan konteynerlar muvozanatlashgan binar daraxt ustiga quriladi va qat‘iy tartibga tayanadi ("kichik" amali [<] operatori asosida hisoblangan). Ikkita element bir biridan kichik bo‘lmasa, ekvivalent hisoblanadi.

Buning kutubxonasi 4 ta asosiy *set* (to‘plam), *multiset* (multi to‘plam, ko‘p to‘plam), *map* (lug‘at) va *multimap* (mulilug‘at, ko‘p lug‘at) assosiativ konteynerlar bilan ishlashga qaratilgan. Bu konteynerlar Key kalitini o‘zlariga asosiy paramert sifatida oladi va Compare munosabati bo‘yicha tartiblaydi. Tartiblash *Key* paramert bilan amalsha oshiriladi. Shuningdek, *map* va *multimap* erkin *T* tipida Key bilan assosiativlanadigandir (sherikdir). *Compare* ob‘yektining tipi taqqoslanaligan konteyner ob‘yekti (*comparison object)* deb aytiladi.

Shuning uchun bu konteynerlarning umumiy xususityati va amallaridan boshlaymiz. Barcha assotsiativ konteynerlar quyidagi amallarni qo‘llab- quvvatlaydi:

count – elementlar sonini qaytaradi, belgilangan kalit bo‘yicha elementlar sonini qaytaradi;

find –elementga ko‘rsatkichga mos bo‘lgan iteratorni qaytaradi, agar bunday bo‘lmasa end() funksiyasini vazifasini bajaradi.

equal_range - berilgan intervaldigi barcha elementlar uchun iteratorlar juftligini qaytaradi.

Katitlarning tengli xaqida *munosabatlarning ekvivalentligi* Kalitlar uchun shartli taqqoslash va (*not*) *operator== tahlil qilamiz.*

Key1 va key2 kalitlar o‘zaro teng hisoblanadi, agar taqqoslanadigan ob‘yektlar uchun comp chin qiymat qaytarsa, ya‘ni:

comp(k1, k2) == *false* && comp(k2, k1) == *false*

Assosiativ konteynerlar har qanday kalit qiymat uchun bitta eng katta elementning qiymatini saqlasa, unikal (takrorlanmaydigan) kalitlarni (unique keys) qo‘llab quvvatlaydi. Aks hoda ular teng qiymatli kalitlarni (*equal keys*) ham qo‘llab quvvatlaydi. *set* va *map* konteynerlar unikal kalitlarni, *multiset* va *multimap* konteynerlar teng qiymatli kalitlar bilan ishlashga mo‘ljallangan.

set va *multiset* konteynerlar uchun aniqlanadigan tip va kalit tipi bir xil bo‘ladi.

map va *multimap* konteynerlar uchun *pair<const Key, T>* jufligi bo‘ladi.

Assosiativ konteynerlarning iteratorlari ikki tomonlama iteratorlar sinfiga kiradi. *Insert* konteyner havolalari va iteratorlarning inkor qilmaydi. Elementlarni o‘chirishda *erase* faqat iterator va havolalarni inkor qiladi.

1.1-jadvalda assosiativ konteynerlarning talablari keltirilgan (konteynerlarga qo‘shimacha). Unda x bu assosiativ sinf, a – X ning qiymati, Agar X unikal kalitni ko‘llab quvvatlasa, a_uniq – X ning qiymati, Agar X bir nechta kalitni ko‘llab quvvatlasa, a_eq – X ning qiymati, i va j – iteratorning KIRISH talablarini qanoatlantiruvchi va value_type elementni ko‘rsatadi (beradi), [i, j) - interval, p – a uchun ruxsat berilgan iterator, q – a uchun o‘zgartiriladigan iterator, [q1, q2) - a da ruxsat berilgan interval, t - X::value_type ning qiymati, k - X::key_type ning qiymati.

1.1-jadvalda assosiativ konteynerlarning talablari.

Ifoda		Qaytarila-digan tipi	tasdiq/ oldingi yoki keyingi holatiga izoh	murakkabligi
X::key_type		Key		Kompilyatsiya vaqti
X::key_compa re		Compare	less<key_type> joriy holat	Kompilyatsiya vaqti
X::value_comp are		binar predikat turi	xuddi, key_compare uchun set va m ultiset; munosabati tartiblangan juftlik, map va multimap uchun chaqirilgan birinchi elementdek (Key)	Kompilyatsiya vaqti

X(c) X a(c);	.		Bo‘sh konteyner yaratish; taqqoslash ob‘yekti sifatida foydalanish uchun.	Doimiy
X() X a;	.	Bo‘sh konteyner yaratish; taqqoslash ob‘yekti sifatida Compare() dan foydalanish uchun		Doimiy
X(i,j,c) X a(i,j,c);	.	Bo‘sh konteyner yaratish [i, j) intervaldan qiymat qo‘shish; taqqoslash ob‘yekti sifatida foydalanish uchun..		NlogN (N - i dan j gacha); chiziqli , agar[i,j) value_comp() bilan saralangan bo‘lsa
X(i,j) X a(i,j);	.	Bo‘sh konteyner yaratish [i, j) intervaldan qiymat qo‘shish; taqqoslash ob‘yekti sifatida Compare() dan foydalanish uchun		NlogN (N - i dan j gacha); chiziqli , agar [i,j) value_comp() bilan saralangan bo‘lsa
a.key_comp()	X::key_co mpar e	Taqqoslash ob‘yektini qaytaradi		doimiy
a.value_comp()	X::value_c omp are	Taqqoslash ob‘yekti uchun yaratilgan value_compare ob‘yektni qaytaradi		doimiy
a_uniq.insert(t)	pair<iterat or, bool>	Agar konteynerda t kalitli element bo‘lmasa, t ni qo‘shish, juftlikda komponentning bool qiymati qo‘shish		logarifmik

		bajarilganligini, iterator t kalitli elementni ko‘rsatadi	

a_eq.insert(t)	iterator	t ni qo‘shadi va iteratorni qaytaradi, qo‘shishilgan elementni ko‘rsatadi.	logarifmik
a.insert(p, t)	iterator	t ni qo‘shish, agar faqat va faqat unikal kalitli konteynerda t kalitga teng kalitli element bo‘lmasa; har doim nusxalari bilan konteynerlarga t ni qo‘shish. har doim t kalitli elementni ko‘rsatuvchi iteratorni qaytaradi. p iterator – qo‘shish qaerdan boshlanishini ko‘rsatuvchi izoh	umuman logarifmik, lekin domiyga intiladi, agar t to‘g‘ri p ni yoniga qo‘yilgan bo‘lsa.
a.insert(i, j)	Natija ishlatilmaydi	[i, j) intervaldan konteynerga elementlarni qo‘shadi;	Umuman, Nlog(size()+N) (N – i dan j gacha); chiziqli, agar [i,j) value_comp() bilan kelishgan holda saralangan bo‘lsa
a.erase(k)	size_type	konteynerda k kalitga teng bo‘lgan barcha e lementlarni o‘chiradi. o‘chirilgan elementlarning sonini qaytaradi.	log(size()) + count(k)
a.erase(q)	Natijasi ishlatilmaydi	kkooko‘rsatilgan elementni o‘chiradi	domiyga intiladi
a.erase(ql, q2)	Natijasi ishlatilmaydi	((ql, q2) intervaldagi barcha elementlarni cho‘chiradi.	log(size())+ , N - ql dan q2 gacha.
a.find(k)	iterator; const_iterator – a uchun o‘zgarmas	k kalitli elementni ko‘rsatuvchi iterator yoki agar bunday element topilmasa a.end() qaytaradi	logarifmik
a.count(k)	size_type	kalitli elementlar sonini qaytaradi.	Log(size()) + count(k)
a.lower_bound (k)	iterator; const_iterator a uchun o‘zgarmas	k kalitli elementdan kichik bo‘lmagan birinchi elementni ko‘rsatuvchi iteratorni qaytaradi	logarifmik
a.upper_bound (k)	iterator; const_iterator a uchun o‘zgarmas	k kalitli elementdan katta bo‘lgan birinchi elementni ko‘rsatuvchi iteratorni qaytaradi	logarifmik
a.equal_range(	pair<iterator, iterator>;	make_pair(lower_boud(k), bound (k)) ga ekvivalent	logarifmik

	pair<const_iterator, const_iterator> a uchun o‘zgarmas		

Assotsiativ konteyner iteratorlarining asosiy xususiyati shundaki, ular konteynerlar orqali o‘sish tartibidagi kalitlar tartibida iteratsiyalanadi, bu yerda o‘sish tartibi ularni yaratish uchun ishlatilgan taqqoslash bilan belgilanadi.

Ikkiia ixtiyoriy o‘zguruvchan I va j iteratorlar uchun I dan j gacha masofa musbat bo‘ladi, ya‘ni *value_comp (*j, *i) == false. Unikal kalitli* assotsiativ konteynerlar uchun kuchliroq *value_comp (*i, *j) == true* holat saqlanib turibdi.

Barcha tartiblangan konteynerlar ikki tomonlama iteratorlar yordamida elementlarga murojaat qilish imkonini beradi. Kalit asosida qidirishda berilgan kichik bo‘lmagan birinchi elementni lower_bound() funksiyasi va birinchi katta elementni upper_bound() funksiyasi yordamida amalga oshiriladi (1.1-jadvalga qarang). Agar mymap konteyner berilgan bo‘lsa, unda mymap.equal_range(key) funksiyasi bilan ekvivalnt bo‘ladi (1.1-dasturga qarang).

1.1-dastur. Tartiblangan konteynerlar funksiyalari.

```
// created by Mbbahodir
#include "stdafx.h"
#include <iostream>
#include <map>
using namespace std;
int main ()
{
  map<char,int> mymap;
  map<char,int>::iterator itlow,itup;
  pair<map<char,int>::iterator,map<char,int>::iterator>  ret,  make_ret;

  mymap ['a']=101;    mymap ['c']=303;    mymap ['f']=404;    mymap ['b']=202;

  cout << mymap.count('c') << endl;
  cout << mymap.find('c')->second << endl;

  ret = mymap.equal_range('b');

  char key = 'b';

  make_ret = make_pair(mymap.lower_bound(key),mymap.upper_bound(key));

  cout << "Kichik bo‘lmagam element: ";

  cout << ret.first->first << " => " << ret.first->second << '\t';
  cout << make_ret.first->first << " => " << make_ret.first->second << '\t'; itlow =

  mymap.lower_bound ('b');

  cout << itlow->first << " => " << itlow->second << endl;
  cout << "katta bo‘lgan birinchi element: ";
```

```
cout << ret.second->first << " => " << ret.second->second << '\t';

cout << make_ret.second->first << " => " << make_ret.second->second << '\t';

itup = mymap.upper_bound ('b');

cout << itup->first << " => " << itup->second << endl;

system("pause");

return 0;

}
```

1.1-dastur. Output

```
1
303
Kichik bo‘lmagam element: b => 202     b => 202     b =>
202 katta bo‘lgan birinchi element: c => 303          c =>
303                                    c => 303
```

1.1-dastur tahlili. Dasturda 4 ta elementdan iborat map konteyner berilgan. ‘s‘ kalitli element soni count() funksiya bilan aniqlagan. Xuddi shu kalit bilan uning qiymati find() funksiya bilan topilgan. equal_range('b'), mymap.lower_bound('b'); mymap.upper_bound('b');
make_pair(mymap.lower_bound(key), mymap.upper_bound(key));
funksiyalarining ekvivalent ishlari ko‘rsatilgan.

Takrorlanmaydigan (unikal) tartiblangan elementlar to‘plami **<set>.** set<K, C = less<K>, A = allocator<K>> - Bu konteyner joyida qiymatlarnio‘zgartirishga yo‘l o‘zgartirishga yo‘l qo‘ymaydi. <set> - bu assosiativ konteyner bo‘lib, unikal kalitni qo‘llab quvvtlaydi (bir xil qaymatli kalitlarni qo‘llab quvvatlamaydi) va kalit yordamida tez qidirish imkonini beradi. set<K>::iterator va set<K>::const_iterator iteratorlar funksional ekvivalet hisoblanadi va o‘zgarganda const K& qiymat qaytaradi. Set to‘plamda qiymatni o‘zgartirish uchun avval o‘chirib tashlab so‘ng yangi qiymatni yoki o‘zgartirilgan varianti qo‘shiladi**.**
<set> ning shabloni:

```
template <class Key, class Compare = less<Key>,
        template <class U> class Allocator = allocator>
```

<set> sinfning ochiq xususiyatlari, operatorlari va funksiyalari:

1. typedef operatorlari:

```
typedef Key key_type;
typedef Key value_type;
typedef Allocator<Key>::pointer pointer;
typedef Allocator<Key>::reference reference;
typedef Allocator<Key>::const_reference const_reference;
typedef Compare key_compare;
typedef Compare value_compare;
typedef iterator;
typedef iterator const_iterator;
typedef size_type;
typedef difference_type;
typedef reverse_iterator; typedef
const_reverse_iterator;
```

2. Xotirani ajratish va bo‘shatish operatorlari (allocation/deallocation):

```
set(const Compare& comp = Compare());
template <class InputIterator>
set(InputIterator first, InputIterator last,
      const Compare& comp = Compare());
set(const set<Key, Compare, Allocator>& x);
~set();
set<Key, Compare, Allocator>& operator=(const set<Key, Compare,
      Allocator>& x);
void swap(set<Key, Compare, Allocator>& x);
```

3. Ruxsat berish vositalarining operatorlari (accessors):

```
key_compare key_comp() const; value_compare
value_comp() const; iterator begin() const;
iterator end() const;
reverse_iterator rbegin() const;
reverse_iterator rend() const;
bool empty() const;
size_type size() const; size_type
max_size() const;
```

4. Qo‘shish va o‘chirish operatorlari (insert/erase):

```
pair<iterator, bool> insert(const value_type& x); iterator
insert(iterator position, const value_type& x); template <class
InputIterator>
void insert(InputIterator first, InputIterator last); void
erase(iterator position);
size_type erase(const key_type& x); void
erase(iterator first, iterator last);
```

5. To‘plam amallari (set operations):
6. Taqqoslash operatorlari:

<set> to‘plamda:

iterator (iterator) bu – const value_type ko‘rsatuvchi domiy ikki tomnlama iterator. Aniq tipni belgilash realizatsiya qilishga bog‘liq va Allocatorda aniqlanadi.

sonst_iterator - ham iterator (iterator) kabidir.

size_type - ishorasiz butun tip, Aniq tipni belgilash realizatsiya qilishga bog‘liq va Allocatorda aniqlanadi.

```
iterator find(const key_type& x) const;
size_type count(const key_type& x) const;
iterator lower_bound(const key_type& x) const;
iterator upper_bound(const key_type& x) const;
pair<iterator, iterator> equal_range(const key_type& x) const;
```

difference_type - ishorali butun tip, Aniq tipni belgilash realizatsiya qilishga bog‘liq va

```
template <class Key, class Compare, class Allocator>
bool operator==(const set<Key, Compare, Allocator>& x,
       const set<Key, Compare, Allocator>& y);

template <class Key, class Compare, class Allocator>
bool operator<(const set<Key, Compare, Allocator>& x,
       const set<Key, Compare, Allocator>& y);
```

Allocatorda aniqlanadi.

Elementlarni qo‘shishi insert() funksiyasi tomonidan amalga oshiriladi va bu funksiyaning bir nechta variantlari mavjud:

- juft iteratorlarni kitirishda intervaldan qiymat qo‘shish;

```
template <class InputIterator>
 void insert (InputIterator first, InputIterator last);
```

- qo‘yish mumkin bo‘lgan joyga ko‘rsatish orqali bitta qiymat qo‘shish (agar yangi element bevosita oldin pozitsiyani egallasa, o‘rnatish o‘zgarmas vaqt mobaynida amalga oshiriladi);

```
pair<iterator,bool> insert (const value_type& val);
```

- berilgan qiymatni qo‘shish (asosiy usul);

```
iterator insert (iterator position, const value_type& val);
```

Oxirgi variant insert funksiyasi bir juftli (iterator, Boolean) qaytaradi. Juftlikdagi birinchi element qo‘shish joyni yoki topilgan qiymatini, ikkinchi joylashtirilgan qiymatini qo‘shish bajarilganligi (true) yoki qo‘shish vaqtida to‘plamda borligini (flase) bildiradi.

Elementlarni o‘chirish erase funksiyasi orqali amalga oshiriladi. Bu funksiya qiymat yoki berilgan elementlar to‘plami yoki iterator yoki iteratorlar intervallari qabul qiladi.

```
(1)     size_type erase (const value_type& val);
(2)     void erase (iterator position);
(3)     void erase (iterator first, iterator last);
```

Birinchi parametr elementni o‘chiradi, qolgan ikka parametrlar elementni ko‘rsatadigan iterator qaytaradi. Birinchi variant o‘chirilgan elementlarning sonini beradi (set bo‘lsa, 1 yoki 0 qiymat beradi).

1.2-dastur. Set to‘plamga elementlarni qo‘shish.

```
// created by MBBahodir
#include "stdafx.h"
#include <iostream>
#include <set>
using namespace std;

void erase_subs(set<int> &s, const int &subs)
{
     set<int>::iterator p, pe, pd; p =
     s.begin(), pe = s.end();

   while (p != pe)
   {if (p == s.find(subs)) p = s.erase(p);
      else
          ++p;
   }
}


int main ()
{
   set<int> myset; set<int>::iterator it;
    pair<set<int>::iterator,bool>   ret;

   for (int i=1; i<=5; ++i)
    myset.insert(i*10);
   ret =myset.insert(20);
   if (ret.second == false)
    it=ret.first;
   myset.insert(it,25);
   myset.insert(it,24);
   myset.insert (it,26);
int myints[] = {5,10,15,60,28};
   myset.insert(myints,myints+5);
   cout << "myset contains after insert:";
   for (it=myset.begin(); it!=myset.end(); ++it) cout << ' ' << *it;
     cout << '\n';
     // set to‘plam tltmentlarini o‘chirish it =
     myset.begin();
     ++it;
     myset.erase (it);
   myset.erase (40);
   it = myset.find (60);
   myset.erase (it, myset.end());
   cout << "myset contains after erase :";
   for (it=myset.begin(); it!=myset.end(); ++it) cout << ' ' << *it;
```

```
    cout << '\n';
    erase_subs(myset,24);
    cout << "myset contains after erase_subs :";
    for (it=myset.begin(); it!=myset.end(); ++it)
        cout << ' ' << *it;
    cout << '\n';
    system("pause"); return
    0;
}
```

1.2-dastur. Output

```
    myset contains after insert: 5 10 15 20 24 25 26 28 30 40 50 60
     myset contains after erase : 5 15 20 24 25 26 28 30 50
myset contains after erase_subs : 5 15 20 25 26 28 30 50
```

Tartiblangan multito‘plam <set> - <multiset>. <set> to‘plamdan farqli o‘laroq, insert funksiyasi qiymat qo‘shish uchun qo‘shiladigan qiymatni ko‘rsatuvchi iterator qaytaradi. <multiset> - assosiativ konteynet bo‘lib, teng qiymatli kalitlarni saqlaydi (mumkin qadar bir kalit qiymatli elmentlar to‘plamini saqlaydi) va kalit orqali tez qidirish imkonini beradi.

<multiset> ning shabloni (<set> ni shabloni bilan bir xil):

```
template <class Key, class Compare = less<Key>,
template <class U> class Allocator = allocator>
```

<multiset> sinfning ochiq xususiyatlari, operatorlari va funksiyalari:

1. typedef operatorlari - <set> niki bir xil.
2. Xotirani ajratish va bo‘shatish operatorlari (allocation/deallocation):

```
const Compare& comp = Compare()); template <class InputIterator> multiset(InputIterator first,
InputIterator last,
        const Compare& comp == Compare()); multiset(const multiset<Key, Compare,
Allocator>& x);
~multiset();
ıpare, Allocator>& operator=(const multiset<Key, Compare, Allocator>& x);
void swap(multiset<Key, Compare, Allocator>& x);
```

3. Ruxsat berish vositalarining operatorlari (accessors) - <set> niki bilan bir xil.
4. Qo‘shish va o‘chirish operatorlari (insert/erase):

```
iterator insert(const value_type& x);
iterator insert(iterator position, const value_type& x);
template <class InputIterator>
void insert(InputIterator first, InputIterator last);
void erase(iterator position);
size_type erase(const key_type& x);
void erase(iterator first, iterator last);
```

5. To‘plam amallari (set operations) - <set> niki bir xil.

6. Taqqoslash operatorlari:

```
template <class Key, class Compare, class Allocator>
bool operator==(const multiset<Key, Compare, Allocator>& x, const multiset<Key,
                Compare, Allocator>& y);
template <class Key, class Compare, class Allocator>
bool operator<(const multiset<Key, Compare, Allocator>& x, const multiset<Key,
                Compare, Allocator>& y);
```

<multiset> to‘plamda:

iterator (iterator) bu – const value_type ko‘rsatuvchi domiy ikki tomnlama iterator. Aniq tipni belgilash realizatsiya qilishga bog‘liq va Allocatorda aniqlanadi.

sonst_iterator - ham iterator (iterator) kabidir.

size_type - ishorasiz butun tip, Aniq tipni belgilash realizatsiya qilishga bog‘liq va Allocatorda aniqlanadi.

difference_type - ishorali butun tip, Aniq tipni belgilash realizatsiya qilishga bog‘liq va Allocatorda aniqlanadi.

<*set*> va <*multiset*> to‘plamlarning xususiyatlari, operatorlari va funksiyalari umuman olganda bir xil. Shuning uchun 2 ta to‘plamni birga o‘rganish maqsadga muvofiq.

set va *multiset* uchun foydalaniladigan iteratorlar o‘zgarmas ikki tomonlama iterator bo‘lganligi uchun *algorithm* kutubxonasida mazkur tipni qo‘llamaydigan funksiyalari uzatish mumkin emas. Shuning uchun joriy sinfda aniqlangan massiv elementlari bilan ishlaydigan usullari, funksiyalardan (umumlashgan algoritmmlari o‘rnida) foydalanish kerak.

Yuqorida ta‘kidlab o‘tilgandek, *multiset* to‘plamning *set* to‘plamdan farqi faqat kalit bir nechta bir xil qiymatli kalitlarni saqlashidadir. Shuning uchun ixtiyoriy sohada ishlatishda *multiset* sinfining ishlatilishi *set* sinfinikidan farq qilmaydi.

set va *multiset* sinflarining ob‘yektlarining tipi kalit bilan yonma-yon bitta shablonli parametr olishi mumikn. Bu shablon taqqoslash (*comp*) funksiyasidir.

Agar shundan funksiya mavjud bo‘lsa, oshkormas less<> funksiyasi bilan beriladi (< amali).

set (*multiset)* sinflarining ob‘yektlarini quyidagi konstruktolar bilan yaratish mumkin:

Bo‘sh to‘plam konstruktori - set<type, comp> ar; yoki set<type, comp> ar(Comp);

Nusxalash konstruktori - set<type, comp> ar(other);

Iteratorlar yordamida qo‘shish konstruktori - set<type, comp> ar(first, last); yoki set<type, comp> ar(first, last, Somp);

Ro‘yxat asosida initsializatsiya qilish konstruktori - set<type, comp> ar {init}; yoki set<type, comp> ar(init); yoki set<type, comp> ar(init, Comp);

Bu konstruktorlarda Comp – konteynerlarning kalitlarini taqqoslash funksyasi (ixtiyoriy). Agar dasturchi o‘zining allokator funksiyasini yaratsa, qo‘shish uchun taqqoslash funksiyasining yonida konstruktorga majburiy bo‘lmagan Allocator() funksiyasi bor.

set (*multiset)* sinflarining ob‘yektlarini o‘chirish uchun destruktor - ar.~set();

Hajmlar va o‘lcham bilan ishlash usullari: empty() – true qaytaradi, agar to‘plam bo‘sh bo‘lsa false, size() - to‘plamdagi elementlar sonini qaytaradi, max_size() – mumkin bo‘lgan elementlar sonini qaytaradi.

Modifikatorlari: clear() – konteynerni tozalaydi, insert() – element qo‘shish, erase() – elemeni o‘chirish, swap() – o‘rin almashtirish, emplace() – joydi elementni yaratish.

Qidirish usullari: set to‘plam bilan ishlaganda qidirish muhim xisoblanadigan aspekt bo‘lib hisoblanadi. Qidirishni samarali tashkil qilish uchun bir nechta o‘zning usullariga ega: count(key), find(key), equal_range(key), lower_bound(key), upper_bound(key)

Taqqoslash va birlashtirish amallari: set to‘plam uchun barcha taqqoslash amallari amal qiladi. ning ketma-ket konteynerlarga amal qilgan barcha amal o‘rinli.

[=] (operator=) operatorni elementlar to‘plamini olish uchun ishlatiladi. Masalan, other to‘plamning nusxasini yangi bir other_one to‘plamga nusxalash uchun ishlatiladi (other_one = other).

Semantik nusxalashni amalga oshirish usuli other to‘plamning nusxasini yangi bir other_one to‘plamga to‘liq nusxalash uchun ishlatiladi va other to‘plami o‘chirilmaydi, ammo uning holatini aniqlab bo‘lmaydi other_one = move(other).

<set> va <multiset> doir masalalr va dasturlarni keltiramiz.

1- masala. [10, 50] intervalda N ta ketma- ket sonlar tasofidiy berilgan. {10, 20, 30, 40, 50} berilan sonlardan joriy ketma-ketlikda nechtadan bor.

1.3(a)-dastur. 1-masalaning dasturi.

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <set>
#include <random>
#include <ctime>
using namespace std;

int main() {
    default_random_engine rnd(time(0));
    uniform_int_distribution<unsigned> g(10, 50);
    vector<int> myVektor;
    int n, k = 0;
    cout << "n = "; cin >> n; for
    (int i = 0; i < n; i++) {
        int d = g(rnd);
            myVektor.push_back(d);
        cout << myVektor[i] << " ";
    }
    set<int> mySet;
      for (int i=1; i<=5; ++i)
      mySet.insert(i*10); cout << endl;
    for (int i = 0; i < n; i++)
            if (mySet.count(myVektor[i]))
    k++; cout << k << "  ta element
    topildi."  << endl;
      system("pause
    "); return 0;
  }
```

1.3(a)-dastur. Output

```
n =  255
47 50 18 29 44 47 31 31 25 19 40 32 43 29 38 22 10 39 47 10 43 46 26 47 18 50 30 24 34 47 31
13 35
41 29 26 25 26 19 46 18 17 23 32 18 47 37 14 13 13 32 28 14 42 41 29 21 19 23 38 12 31 26 48
34 44
20 21 34 13 43 42 30 26 44 10 34 22 15 26 23 26 30 22 44 41 44 12 36 33 29 37 20 17 10 28 33
26 19
12 50 31 49 33 50 41 32 42 38 35 27 23 18 25 28 31 12 36 18 42 36 44 24 37 10 34 49 26 29 11
39 33
31 15 11 42 37 40 49 34 41 20 50 39 19 29 30 24 38 47 14 41 37 21 49 21 42 14 37 45 12 40 17
46 17
10 45 43 37 19 25 39 11 49 33 17 11 12 10 22 19 26 12 47 34 16 21 33 25 44 26 40 28 38 35 15
```

```
31 26
10 32 40 42 45 32 17 20 41 30 36 13 23 24 44 19 23 29 21 38 30 26 20 23 33 31 39 20 22 50 34
18 24
21 30 21 42 13 22 41 19 35 40 14 34 37 16 14 35 17 44 40 13 37 28 45 24
34  ta element topildi.
```

Oldinroq juda tez-tez elementlarni kiritish va o‘chirish algoritmlarni foydalanish kerak emas deb aytgan. Vektor va to‘plam bilan ishlash samaradorligini taqqoslaylik. To‘plam va vektorning massivlarini tasodifiy sonlar bilan to‘ldiradigan va belgilangan qiymatni qidiradigan dastur yarataylik.

1.3(b) – dastur. Vektor va to‘plam bilan ishlash samaradorligini taqqoslash.

```
// Created by MBBahodir
#include "stdafx.h"
#include <iostream>
#include <set>
#include <vector>
#include <algorithm>
#include <random>
#include <ctime>
#include <chrono>
using namespace std;

int main() {
	int n;
	default_random_engine rnd(time(0));
	uniform_int_distribution<unsigned> g(1, 10000);
	 cout << "n = "; cin >> n;
const int k = 100;
 int a = n;
	auto start = chrono::system_clock::now();
	vector<int> ar1;
	while (a--) ar1.push_back(g(rnd));
	    auto result1 = find(ar1.begin(), ar1.end(), k);
	    auto end = chrono::system_clock::now();
	auto elapsed = end - start; cout << "vector => "<< elapsed.count()<< endl;
	a = n;
	start = chrono::system_clock::now();
	set<int> ar2;
	    while (a--) ar2.insert(g(rnd)); auto
	            result2 = ar2.find(k);
	end = chrono::system_clock::now();
	elapsed = end - start;
	cout << "set => " << elapsed.count() << endl;

	system("pause");
	return 0;
}
```

1.3(b)-dastur. Output

```
n = 10000
vector => 289989
set => 1160000
```

To‘plamning ishlashi vektordan sezilarli darajada past bo‘ladi. Shuni yodda tutingki, to‘plamni to‘ldirish amali saralash bilan amalga oshiriladi va bu albatta, qimmatbaho amal. Lekin, faqat qidirish uchun, har ikki konteynerlarni ishlash solishtiradigan bo‘lsa, vaziyat <set> foydasiga o‘zgaradi:

1.3(c)-dastur. Output

```
n = 10000
vector => 70042
set => 20044
```

Bundan qanday xulosa chiqarish mumkin? set yordamida (yoki multiset) takrorlanish jarayonlarini dasturlashda ketma ketlikdan elementlarni olish, yoki tez-tez o‘chirish va elementlar kiritish bu noto‘g‘riyondashuv hisoblanadi. Set to‘plam ob‘yektini yaratish uchun takrorlanish jarayonlari asosida shakllantirilgan ketmk ketlik foydalanish kerak yoki tayyor to‘plam uchun insert funksiyasidan foydalanish maqsadga muvofiq.

Ammo, dastur tez-tez elementlarni qidirish ishlatilsa, set raqobatga chiqa olmaydi.

pair - yordamchi sinfi. Boshqa assotsiativ konteynerlar bilan ishlashni yanada ko‘rib chiqish uchun pair yordamchi shablon sinfi bilan tanishishimiz kerak. Bu sinf birlik sifatida ikkita ob‘yektlarni saqlash imkoniyatini beradi. pair STD ning turli joylarida, xususan minmax() algoritmida, set equal_range() sinfining usulida, shuningdek juft (key, value) bo‘lgan assosiativ konteynerlar elementlari bilan ishlash uchun ishlatiladi.

pair - bir juft ob‘yekt yaratish uchun quyidagi konstruktor sintaktiki ishlatiladi:

```
pair<T1, T2> p1;
pair<T1, T2> p2(val1, val2);
pair<T1, T2> p3(p1);
```

Amalari:

p.first – havola bo‘yicha juftlikning birinchi elementiga murojaat. p.second –
havola bo‘yicha juftlikning ikkinchi elementiga murojaat.
p->first – ko‘rsatkich bo‘yicha juftlikning birinchi elementiga murojaat.
p->second – ko‘rsatkich bo‘yicha juftlikning ikkinchi elementiga murojaat.
p = other_p – qiymat qilib berish (C++ qoidasiga asosan oshkormas tip almatirish yordamida).
p.swap(other_p) yoki swap(p, other_p) –p va other_p lar uchun o‘rin almashtirish.
<, <=, >, >=, ==, != – taqqoslash amallari make_pair(val1,
val2) – juftlikni beradi.

1.4-dastur. Pair konstruktori va amallarini ishlatish.

```
// Created by MBBahodir
#include "stdafx.h"
#include <iostream>
using namespace std;

int main() {
	int a1 = 16, b1 = 11; int
	a2 = 18, b2 = 03;
	pair<int, int> par1(a1, b1);
	pair<int, int> *par2 = new pair<int, int>(a2, b2); cout
	<< par1.first << '.' << par1.second << endl; cout <<
	par2->first << '.' << par2->second << endl;
	    system("pause");
	return 0;}
```

1.4-dastur. Output

```
16.11
18.3
```

Assosiativ konreynerlar uchun map va multimap sinflari. map assotsiativ konteyneri (xarita, lug‘at) o‘z-o‘zini muvozanatlovchi daraxtga (qizil-qora daraxt) asoslangan to‘plamlar bilan bir xil tarzda amalga oshiriladi. To‘plam va lug‘at o‘rtasidagi asosiy farq shundaki, map massivi kalitlardan tashkil topgan elementlar juftlarining tartibli assotsiativ massivi (konteyneri) va ularning mos qiymatlaridan iborat. Lug‘atda kalitlari unikal (takrorlanmaydigan) va multimap bir nechta nusxadagi kalitlari bo‘ladi. Lug‘atlarda kalit va qiymat turlari fundamental tip yoki abstrakt tip bo‘lishi mumkin. Agar kalit tipi abstrakt tip bo‘lsa, uning uchun saralash yo‘nalishini belgilovchi komparator funksiyasi (taqqoslash funksiyasi) aniqlanishi kerak. Kalit doimiy qiymatga ega va kalit qiymatini o‘zgartirish mumkin emas. Juftlikning ikkinchi element qiymatini (asosiy qiymatini) o‘zgartirish mumkin.

Map (yoki multimap) sinflari bilan ishlashni boshlash uchun dastur sarlavhasiga (ikkala sinf bilan birgalikda) quyidagi dastur fragmenti yozildi (bu oldin ham barcha assosiativ konteynerlar uchun aytilgan edi):

```
#include <map>
```

Konstruktorlar. Lug‘at ob‘yektlari shablon parametrlarini: kalit tipi va kalitning qiymati tipi (key va T), taqqoslash (comp)funksiyasini o‘z ichiga oladi. Agar bunday funksiya bo‘lmasa, u oshkormas less <> funksiya (operation <) tomonidan o‘rnatiladi.

Quyidagi konstruktorlar yordamida map sinf ob‘yektlarini yaratiladi:

Oddiy map konreyner konstruktorlari: map<Key, T, comp> ar; yoki map<Key, T, comp> ar(Comp);

Nusxalash uchun konstruktor: map<Key, T, comp> ar(other);

Iteratorlar yordamida qo‘shish uchun konstruktorlar: map<Key, T, comp> ar(first, last); yoki map<Key, T, comp> ar(first, last, Somp);

Ro‘yxat asosida initsializatsiya qilish konstruktorlari: map<Key, T, comp> ar {init}; yoki map<Key, T, comp> ar(init); yoki map<Key, T, comp> ar(init, Comp); *Eslatma*. Ro‘yxatni initsializatsiya qilishda har bir juftlik alohida figurali qavs ichiga olinishi kerak!

```
map<string, int> ar {{"a1", 10}, {"www", 17}, {"j8", 100}};
```

Bu yerda Comp konteyner kalitlari solishtirish uchun funksiyasi (ixtiyoriy). Bundan tashqari, taqqoslash funksiyaning yonida konstruktor ixtiyoriy Allocator() vazifasini o‘z ichiga oladi (soddaligi uchun yozilmaydi), dasturchi uning allocator vazifasini ham belgilaydi.

Map sinfining destruktori: ar.~map();

Lug‘at usullari. Lug‘at usullari set to‘plam usullari bilan bir xil bo‘lganligi uchun ularni takrorlab keltirmaymiz. Bu usullar qanday ishlashini misol yordamida ko‘rsatamiz. Aytaylik, quyidagicha lug‘at yaratishimiz kerak: kalitga tasodifiy (string), qiymatga tasodifiy (integer) o‘rnatiladi. So‘ngra key1 va key2 orasidagi barcha key = > qiymat juftliklarining chiqishi talab qilinsin. Dastur oxirida key 3 kalit uchun maksimal qiymatni chiqarish talab qilinadi (dasturning o‘zidagi izohlarga qarang).

1.5-dastur. Lug‘at usullaridan foydalanish.

```
// Created by MBBahodir
#include "stdafx.h"

#include <iostream>
#include <random>
#include <ctime>
#include <string>
#include <map>
#include <vector>
using namespace std;

int main() {
    map<string, int> rat;
    vector<string> lit;
        lit.push_back("a1");    lit.push_back("a2");    lit.push_back("a3");
        lit.push_back("b1");    lit.push_back("b2");    lit.push_back("b3");
        lit.push_back("c1"); lit.push_back("c2"); lit.push_back("c3");
    default_random_engine          rnd(time(0));
    uniform_int_distribution<int> d(-10, 20);
    uniform_int_distribution<int> c(0, 8);
    //============================1============================//
    // lug‘atni to‘ldirish: tasofidiy kalit tasodifiy olinadi                    //
    //=======================================================//
    int n;
    cout << "Lug‘at elementlari sonini kirit: "; cin
    >> n;
    while (n--) {
        string S = lit[c(rnd)]; int
        D = d(rnd);
        pair<string, int> tpr = make_pair(S, D);
        rat.insert(tpr);
        // emplace uchun juftlik yaratish shartmas:
        //rat.emplace(S, D);
```

```
    }
    //============================2============================//
    // 2 ta har xil amal bilan kalit va qiymat qoshish                    //
    //=============================================================//
    rat.emplace("d1", 10);
        rat.insert(pair<const string, int>("d3", -15)); rat.insert(pair<const
    string, int>("d3", 6));
    //============================3============================//
    // birinchi va ikkinchi kalit orasidagi elementlarni chiqarish//
    //=============================================================//
    string el_l, el_r;
    cout << "kalitlarni kirit:\n"; cout <<
    "1-kalit: "; cin >> el_l; cout << "2-
    kalit: "; cin >> el_r; auto fst =
    rat.lower_bound(el_l); auto lst =
    rat.upper_bound(el_r); while (fst !=
    lst) {
        cout << fst -> first
             << " => "
             << fst -> second
             << endl;
        fst++;
    }
    //============================4============================//
    // kalitga mos eng katta elementni aniqlash                    //
    //=============================================================//
    string tmp;
    cout << "Kalit kirit: "; cin
    >> tmp;
    auto first = rat.equal_range(tmp).first; auto
    last = rat.equal_range(tmp).second; if (first
    == last)
        cout << 0 << endl;
    else
        while (first != last) {
            auto r = first -> second; if (r
            >= 10)
                cout << r << " ";
            first++;
        }
          system("pause");
    return 0;
}
```

1.5-dastur. Birinchi sinov Output

```
Lug‘at elementlari sonini kirit: 10
a2 => 14
a3 => 13
b2 => -8
b3 => 3
c2 => -4
c3 => -7
d1 => 10
d3 => -15
kalitlarni kirit: 1-
kalit: a1
2-kalit: c2 a2 =>
14
a3 => 13
b2 => -8
b3 => 3
c2 => -4
Kalit kirit: c4 0
```

1.5-dastur. Ikkinchi sinov Output

```
Lug‘at elementlari sonini kirit: 9 a1 => 11
a2 => 10
b1 => 2
b2 => -10
c1 => 3
c2 => 9
d1 => 10
d3 => -15
kalitlarni kirit: 1-
kalit: a1
2-kalit:
 c2 a1 => 11
a2 => 10
b1 => 2
b2 => -10
c1 => 3
c2 => 9
Kalit kirit: d3
```

Elementlarga murojaat. Lug‘atlarda elementlarga KIRISHning yana ikkita usuli mavjud (iteratorlar bilan ishlashdan tashqari). Birinchi usul at(key) usulini qo‘llashdir (bilsangiz kerak). Biroq, indeksni argument sifatida qabul qilmaydi, lekin kalitning qiymatini qabul qiladi. Bu usul kalitga ekvivalent bo‘lgan kalit bilan mos elemeni qiymatiga mos yozuvlar qaytaradi. Agar bu element mavjud bo‘lmasa, out_of_range istisno funksiyasining qiymati qaytariladi. Boshqacha aytganda, massivda bunday kalit bor-yo‘qligini tekshiradi. Ikkinchi usul overloaded[] funksiyadan foydalanishga asoslangan. Lekin quyidagi sabablarga ko‘ra ehtiyotkorlik bilan foydalanish zarur. ar(key) mavjud bo‘lmasa, standart konstruktor yordamida kalit bilan yangi element qator qo‘shiladi.

Aks holda elementga ko‘rsatkich qaytariladi. Eslattb o‘tamizki, indekslash funksiyasini assotsiativ konteyner sinflarning ikki turga (map va unordered_map) foydalanish mumkin. Ushbu usullarni ishlab chiquvchilar tomonidan qo‘shilishining sabablari aniq: dastur juda qisqa va chiroyli bo‘ladi.

at() funksiyasi istisno qaytargani uchun try - catch funksiyasi bilan quyidagicha boshqarish mumkin:

```
map<string, int> rat; try
{
    rat.at("U");
}
// istesnoni ushlab olish catch
(std::out_of_range) {
        // uni qayta ishlash
        rat["U"] = 15;
}
```

Dastur fragmentida "U" kalitli juftlik topilmasa, uni yaratib, yangi qiymat beradi.

rat["U"] = 15; (hiyla, ruscha fishka) indekslash amali u faqat asosiy qiymatini olish emas, balki bu qiymatini o‘zgartirish mumkin degan ma‘noni anglatadi, bir qiymatini qaytaradi. Boshqacha qilib aytganda, ushbu dastur fragmenidagi amallarini bajarish mumkin:

```
rat["b1"]=1980;
cout << rat["b1"] << endl;
rat["b1"] +=1;
cout << rat["b1"] << endl;
--rat["b1"];
cout << rat["b1"] << endl;
```

map<K, T, C = less<K>, A = allocator<pair<const K, T>>> to‘plam pair<const K, T> tipidagi qiymatni saqlaydi. Bunda K kalit, tanlash imkonini beradi va T saqlanadigan qiymat. Shuning uchun iteratordan foydalangannimizda first kalit qiymatni va second saqlangan qiymatni qaytaradi, o‘zgartirish imkonini beradi. Yuqorida aytib o‘tilgandek operator[] asosiy amallardan hisoblanadi (vazifasini bilsangiz kerak). Agar map<K, T> m bo‘lsa va m[k] = t ma‘no jihatidan quyidagi fragmentga teng:

```
m.insert(make_pair(k, T())).first->second = t
```

Bu kabi elementlarni qo‘shish samarali hisoblanib, T ob‘yektni yaratishga imkon bermaydi (yuqoridagi fragmentda aniq T() konstruktor chaqirilgan).

Agar operator[] noqulay va samarasiz deb hisoblasangiz, yana unga ikkita alternativ variantlar bor. Birinchisi find funksiyasidan foydalanib,kalit bo‘yicha (key, value) juftligiga mos iteratorni olish yoki kalit lug‘atda bo‘lmasa, end() funksiyadan foydalanish mumkin. Ikkinchisi at() funksiyasidan, ya‘ni kalit asosida qiymatni qaytaradi. Agar kalit bo‘lmasa, istisno holatga o‘tadi.

Map sinfining umumiy shabloni quyidagicha:

```
template <class Key, class T, class Compare = less<Key>,
template <class U> class Allocator = allocator>
```

Map sinfining operatorlari, xususiyatlari, funksiyalari va usullarini qo‘rib chiqamiz.

typedef operatorlari :

```
typedef Key key_type;
```

```
typedef pair<const Key, T> value_type;
typedef Compare key_compare;
class value_compare
        : public binary_function<value_type, value_type, bool> {
        friend class map;
protected:
        Compare comp; value_compare(Compare
        c) : comp(c) {}
public:
        bool operator()(const value_type& x, const value_type& y) {
         return comp(x.first, y.first);
        }
};
typedef iterator;
typedef const_iterator;
typedef Allocator<value_type>::pointer pointer;
typedef Allocator<value_type>::reference reference;
typedef Allocator<value_type>::const_reference const_reference;
typedef size_type;
typedef difference_type;
typedef reverse_iterator;
typedef const_reverse_iterator;
```

Xotirani ajratish va bo‘shatish (xolli qilish) operatorlari (allocation/deallocation):

```
map(const Compare& comp = Compare());
template <class InputIterator>
map(InputIterator first, InputIterator last, const Compare&comp = Compare());
map(const map<Key, T, Compare,  Allocator>& x);
~map();
map<Key, T, Compare, Allocator>&
operator=(const map<Key, T, Compare, Allocator>& x); void swap(map<Key, T, Compare,
Allocator>& x);
```

Ruxsat etish vositalari (accessors):

```
key_compare key_comp() const;
value_compare value_comp() const;
rator begin() const_iterator begin() const; iterator end();
        const_iterator end() const;
        reverse_iterator rbegin();
        const_reverse_iterator rbegin();
        reverse_iterator rend();
        const_reverse_iterator rend();
         bool empty() const;
        size_type size() const;
         size_type max_size() const;
Allocator<T>::reference operator[](const key_type& x);
```

Qo‘shish va o‘chirish funksiyalar (insert/erase):

```
pair<iterator, bool> insert(const value_type& x);
 iterator insert(iterator position, const value_type& x);
 template <class InputIterator>
 void insert(InputIterator first, InputIterator last);
 void erase(iterator position);
size_type erase(const key_type& x);
void erase(iterator first, iterator last);
```

Map sinf usulari:

```
iterator find(const key_type& x); const_iterator find(const key_type& x)
const; size_type count(const key_type& x) const; iterator
lower_bound(const key_type& x);
const_iterator lower_bound(const key_type& x) const; iterator
upper_bound(const key_type& x);  const_iterator
upper_bound(const key_type& x) const; pair<iterator, iterator>
equal_range(const key_type& x);
pair<const_iterator, const_iterator> equal_range(const key_type& x)const;
```

Mantiqiy operatorlari:

```
template <class Key, class T, class Compare, class Allocator> bool
    operator==(const map<Key, T, Compare, Allocator>& x, const
                                map<Key, T, Compare, Allocator>& y);

template <class Key, class T, class Compare, class Allocator> bool
        operator<(const mapr<Key, T, Compare, Allocator>& x,
                        const map<Key, T, Compare, Allocator>& y);
```

iterator- ikki tomonlama iterator bo‘lib value_type ishora beradi. Aniq tipi amalga oshirish bilan bog‘liq va Allocator asosila belgilanadi.

const_iterator- doimiy ikki tomonlama iterator bo‘lib const value_type ishora beradi. Aniq tipi amalga oshirish bilan bog‘liq va Allocator asisida belgilanadi. Bunda iterator va const_iterator uchun konstruktor bor, bu kafolatlanadi.

size_type – butun ishorasiz tip. Aniq tipi amalga oshirish bilan bog‘liq va Allocator asisida belgilanadi.

difference_type - butun ishorali tip. Aniq tipi amalga oshirish bilan bog‘liq va Allocator asisida belgilanadi.

Assotsiativ konteynerlar uchun standart usullar to‘plamidan tashqari, lug‘at *Allocator::reference operator[](const key_type&)* amalini ta‘minlaydi. Masalan, m lug‘atda k kalit yuilan m[k] kuyidagi fragment bilan ekvmvalent:

```
(*((m.insert(make_pair(k, T())))).first)).second
```

Assotsiativ konteynerlar asosiy xususiyatlari

Headers		<set>		<map>	
Members		set	multiset	map	multimap
	constructor	set	multiset	map	multimap
	destructor	~set	~multiset	~map	~multimap
	assignment	operator=	operator=	operator=	operator=
iterators	begin	begin	begin	begin	begin
	end	end	end	end	end
	rbegin	rbegin	rbegin	rbegin	rbegin
	rend	rend	rend	rend	rend
const iterators	cbegin	cbegin	cbegin	cbegin	cbegin
	cend	cend	cend	cend	cend
	crbegin	crbegin	crbegin	crbegin	crbegin
	crend	crend	crend	crend	crend
capacity	size	size	size	size	size
	max_size	max_size	max_size	max_size	max_size
	empty	empty	empty	empty	

Tartiblangan unikal kalitga ega bo‘lmagan map – multimap sinfi hisoblanadi. Bu sinf operator[] standart operatorni qo‘llab quvvatlamaydi, va multiset juftligini eslatadi. Bunda qidirish faqat birinchi maydon, ya‘ni kalit maydon bilan amalga oshiriladi. multimap sinfini o‘rganib chiqish uchun, misol sifatida matnli faylga yozilgan lug‘atga murojaat amallarini ko‘rsatamiz. Faraz qilaylik lug‘at so‘zlar juftligidan tashkil topgan va birinchi so‘z kalit bo‘lsin va takrorlanishi mumkin.

“so‘z - tarjimasi” tipini aniqlash. Bunda so‘z juftligini olamiz va lug‘al element deb qaraymiz.

```
using Entry = pair<string, string>;
```

Lug‘at tipini aniqlash

```
using Dictionary = multimap<string, string>;
```

Standart KIRISH va chiqish operatorlari quyidagicha aniqlaymiz.

```
namespace std
{
  istream& operator>>(istream &is, Entry &en)
  {
    return is >> en.first >> en.second;
  }

  ostream& operator<<(ostream &os, const Entry &en)
  {
    return os << en.first << ' ' << en.second;
  }
  istream& operator>>(istream &is, Dictionary &d)
```

```
    {
      istream_iterator<Entry> begin(is), end; d
      = Dictionary(begin, end);
      return is;
    }

    ostream& operator<<(ostream &os, const Dictionary &d)
    {
      ostream_iterator<Entry> out(os, "\n");
      copy(d.begin(), d.end(), out);
      return os;
    }
}
```

Lug‘atga KIRISH uchun amal. Belgilangan lug‘at uchun yangi lug‘at chiqaradi.

```
Dictionary reverse(const Dictionary &d)
{
  Dictionary result;
  for (const auto &el : d)
  result.emplace(el.second, el.first);
  return result;

  }
```

Lug‘at elementiga murojaat va o‘qish.

```
void dictionary_reverse(istream &from, ostream &to)
{
  Dictionary read;
  from >> read;
  to << reverse(read);
}
```

Std sinfida aniqlangan istream_iterator va ostream_iterator tiplaridan foydalanganlik uchun [<<] va [>>] operatorlari shu fazoaga moslab yaratilgan. Bu foydalanuvchiga KIRISH va chiqish operatorlari va standart tiplardan foydalanish ruhiyatiga ta‘sir qilmaydi.

Multimar sinfning konstruktori map sinfiniki bilan bir xil.

Multimar sinfning map sinfinikidan farf qiladigan operatorlari:

```
friend class multimap;
        protected:
                Compare comp; value_compare(Compare c) :
                comp(c) {}
        public:
                bool operator()(const value_type& x, const value_type& y) {
                return comp(x.first, y.first);
                }
        };
```

Mantiqiy operatorlari:

```
template <class Key, class T, class Compare, class Allocator>
bool operator==(const multimap<Key, T, Compare, Allocator>& x, const multimap<Key, T, Compare, Allocator>& y);

template <class Key, class T, class Compare, class Allocator> bool
operator<(const multimap<Key, T, Compare, Allocator>& x,
        const multimap<Key, T, Compare, Allocator>& y);
```

iterator- ikki tomonlama iterator bo‘lib value_type ishora beradi. Aniq tipi amalga oshirish bilan bog‘liq va Allocator asosila belgilanadi.

const_iterator- doimiy ikki tomonlama iterator bo‘lib const value_type ishora beradi. Aniq tipi amalga oshirish bilan bog‘liq va Allocator asisida belgilanadi. Bunda iterator va const_iterator uchun konstruktor bor, bu kafolatlanadi.

size_type – butun ishorasiz tip. Aniq tipi amalga oshirish bilan bog‘liq va Allocator asisida belgilanadi.

difference_type - butun ishorali tip. Aniq tipi amalga oshirish bilan bog‘liq va Allocator asisida belgilanadi.

Nazorat savollari

1. Qanday konteynerlar uchun aniqlanadigan tip va kalit tipi bir xil bo‘ladi?
2. Assosiativ konteynerlarning iteratorlari necha tomonlama iteratorlar sinfiga kiradi?
3. Kalit asosida qidirishda berilgan kichik bo‘lmagan birinchi elementni qanday funksiya va birinchi katta elementni qanday funksiya yordamida amalga oshiriladi?
4. Set to‘plamda qiymatni o‘zgartirish uchun avval nima funksiyani bajarib, so‘ng yangi qiymatni yoki o‘zgartirilgan varianti qo‘shiladi?
5. Xotirani ajratish va bo‘shatish operatorlari set sinfi uchun sanab bering?
6. Ishorali butun tip, aniq tipni belgilash realizatsiya qilishga bog‘liq va Allocatorda aniqlanadi. Gap qaysi funksiya haqida.
7. To‘plam - assosiativ konteynet bo‘lib, teng qiymatli kalitlarni saqlaydi (mumkin qadar bir kalit qiymatli elmentlar to‘plamini saqlaydi) va kalit orqali tez qidirish imkonini beradi. Gap qaysi to‘plam haqida bormoqda.
8. set va multiset sinflarining ob‘yektlarining tipi kalit bilan yonma-yon bitta shablonli parametr olishi mumikn. Bu shablon qaysi funksiya?
9. [=] (operator=) operatorni nima uchun ishlatiladi?
10. To‘plam va vektorning massivlarini tasodifiy sonlar bilan to‘ldiradigan va belgilangan qiymatni qidiradigan dastur tuzing.

11. Pair qanday sinf va nima uchun kerak?

12. p.first – havola bo‘yicha juftlikning nechanchi elementiga murojaat?

13.Lug‘atda kalitlari unikal (takrorlanmaydigan) va qaysi sinfda bir nechta nusxadagi kalitlari bo‘ladi.

14. Oddiy map konreyner konstruktorlari: map<Key, T, comp> ar; va map<Key, T, comp> ar(Comp); ni ishlash tartibini tushuntirib bering.

5- MA’RUZA
TARTIBLANMAGAN ASSOSIATIV KONTEYNERLAR (2 SOAT)

Reja:

1. Tartiblanmagan assosiativ konteynerlar.
2. unordered_set, unordered_map, unordered_multiset, unordered_multimap.

Annotatsiya*: Tartiblanmagan assosiativ konteynerlarning asosiy foydalanish ko‘nikmalari va unordered_set, unordered_map, unordered_multiset, unordered_multimap bilan yaqindan tanishib, ularning xususiyatlari, funksiyalari, sinflari, yaratish va foydalanish uslublari, amallari, talablari, vazifalari va usullari keltirilgan.*

Kalit so‘zlar. *to‘plam, shablon, map, set, multimap, multiset, unordered_set, unordered_map, unordered_multiset, unordered_multimap, satrlioqim, iterator, dinamik ma’lumotlar tuzilmasi, pridekat, konteynersinflar.*

KIRISH

Tartiblanmagan assosiativ konteynerlar <unordered_set> va <unordered_map>. Tartiblanmagan konteynerlar <unordered_set>, <unordered_map> xesh – jadval asosida qurilgan (odatda bu xesh jadvallar ekvivalent elementlar ro‘yxati - bloklar) va xesh funksiyalarga (joriy holatida standart tiplar uchun <functional> sinfidan std::hash<T> bilan aniqlanadi) va tenglik amaliga (joriy holatda [==] operatori) tayanadi.

Ikkita element uchun xeshlari teng bo‘lsa va tenglik amali chin (true) qiymat qaytarsa, ular ekvivalent hisoblanadi. Tartiblanmagan konteynerlari bir tomolama iteratorlar bilan elementlariga murojaat qilish kerak.

Tartiblanmagan standart assosiativ konteynerlarga quyidagi shablon sinflar kiradi*:*

unordered_set<K, H = hash<K>, E = equal_to<K>, A = allocator<K>>,
unordered_map<K, T, H = hash<K>, E = equal_to<K>, A = allocator<pair<const K, T>>>,
unordered_multiset,
unordered_multimap.

Bularning funksionalligi tartiblangan assosiativ konteynerlarga o‘xshaydi.

Tartiblanmagan konteynerlarda xesh jadvalni shakllantirishning o‘ziga xosligi lower_bound va upper_bound funksiyalariga egamasligidir. Ammo, berilgan kalitga mos bir intervaldagi elementlarni olish uchun equal_range funksiyasi ishlatiladi.

O‘rtacha tariblanmagan konteynerlarda elemetnlarni qo‘shish, qidirish va o‘chirish o‘rtacha doimiy vaqt talab va tartiblanganlarga nisbatan sezilarli darajada tezroq bo‘lishi mumkin. Biroq, eng yomon holatda ham vaqtga chiziqli erishish mumkin. Samaradorligi hesh funksiyasini amalga oshirish sifatiga bog‘liq, lekin butunlay bunday vaqt ajratishni oldini olish mumkin emas. Shuning uchun,

interaktiv dasturlarda (masalan, o‘yinlar), tartibga solinmagan konteynerlar davriy kechikishlar tufayli tartiblanganlardan ko‘ra yomonroq bo‘lishi mumkin.

Katta sondagi elementlar qo‘shish uchun, kerakli vaqtda bir qator kiritishda "rehash"ga oshirish mumkin (ko‘p vaqt oladigan ajratilgan saqlash va qayta tarqatish elementlar hajmini oshirish) rehash funksiyasini chaqirib to‘g‘ri hisoblanadi (vektor uchun reserve funksiyasiga o‘xshaydi).

Ayrim vazifalarda saralash uchun vaqt ko‘p ajratilsa avtomatik saralashni muhim kamchilik deb hisoblash mumkin. Bu holda, tartiblangan to‘plam va lug‘atlar sinflarini tartiblanmagan assotsiativ konteynerlar sinflar (unordered_set, unordered_multiset, unordered_map va unordered_multimap) tomonidan o‘zgartirilishi mumkin. Tartiblanmagan konteynerlarda qidirish, qo‘shish va o‘chirish amallari o‘rtacha doimiy vaqt murakkabligiga ega va u O (1) ga tengdir.

Tartiblanmagan konteynerlar xesh jadvallar sifatida amalga oshiriladi. Xesh- jadvalda amalni bajarish xesh-funksiyada (xeshlash) kalitdan boshlanadi. Olingan xesh qiymati (shuningdek, xesh yoki xesh kodi) massivda indeks rolini bajaradi. Xesh jadvali oddiy lug‘atga o‘xshaydi, unda tom ma‘noda xesh qiymati deb hisoblash mumkin.

Tartiblanmagan konteynerlarning xesh jadvallarida yacheykalari bor va ular cheksiz ko‘p elementlarni o‘z ichiga olishi mumkin. Bu yacheeykani segment deb ataymiz. Aslida, segment (xesh bilan bog‘liq) ulangan ro‘yxatdir.

To‘plam hajmiga nisbatan saqlanadigan elementlar soni mos kelsa, (xesh funksiyasi uchun mumkin bo‘lgan elementlar soni) xesh jadvalni to‘ldirish (load factor) koeffitsienti deb yuritiladi va o‘rtacha amal ijro vaqt belgilaydigan muhim parametr hisoblanadi.

Unordered_set va unordered_multiset sinflari bilan ishlashni boshlash uchun unordered_set sarlavhasini (ikkala sinf bilan birgalikda) quyidagicha qo‘shish kerak:

```
#include <unordered_set>
```

unordered_map va unordered_multimap sinflari ham xuddi yuqoridagidek qo‘shiladi:

```
#include <unordered_map>
```

Tartiblanmagan konteynerlarning tiplari quyidagi shablon parametrlarini o‘z ichiga oladi:

- const Key – kalit tipi (unordered_set, unordered_multiset, unordered_map, unordered_multimap).
- T – qiymatning tipi (unordered_map, unordered_multimap).
- alloc –allokator funksiya (majburiy emas, erkin).
- size_type bucket_count – initsializatsiya qilishda foydalanish uchun minimal yacheykalar soni (agar ko‘rsatilmagan bo‘lsa, joriy holat bo‘yicha olinadi).
- hash – xesh-funksiya (erkin).
- equal – taqqoslash fueksiyasi, kalitlarni solishtirish uchun ishlatiladi (erkin).

Quyidagi konstruktorlar asosida unordered_set va unordered_map sniflarining ob‘yektini quradi:

```
unordered_set<Key> ar; unordered_set<Key, hash,
equal> ar; unordered_map<Key, T> ar;
unordered_map<Key, T, hash, equal> ar;
```

Agar hash xesh funksiya yozilmagan bo‘lsa, joriy holat bo‘yicha hash<key_type> funksiyasi ishlatiladi.

Agar equal taqqoslash funksiyasi yozilmagan bo‘lsa, joriy holat bo‘yicha equal_to<> (operator==) funksiyasi ishlatiladi.

Soddaligi uchun Allocator () va bucket_count yozilmaydi.

Nusxalash konstruktrlari:

```
unordered_set<Key> ar(other);
unordered_map<Key, T> ar(other);
```

Iteratorlar asosida qo‘shish:

```
unordered_set<Key> ar(first, last); unordered_set<Key,
hash, equal> ar(first, last); unordered_map<Key, T>
ar(first, last);
unordered_map<Key, T, hash, equal> ar(first, last);
```

Ro‘yxat bo‘yicha **initsializatsiya qilish:**

```
unordered_set<Key> ar {init}; unordered_set<Key>
ar(init); unordered_set<Key, hash, equal> ar(init);
unordered_map<Key, T> ar {init};
unordered_map<Key, T> ar(init);
unordered_map<Key, T, hash, equal> ar(init);
```

unordered_set va unordered_map sinf ob‘yektlarini o‘chirish uchun destruktorlar:

```
ar.~unordered_set();
ar.~unordered_map();
```

Tartiblanmagan konteynerlarda ishlatilmaydigan usullar:

- [>], [<,] [>=], [<=] taqqoslash amallari ([==]va [!=]amallarni qo‘llab quvvatlaydi)
- lower_bound va upper_bound
- Ikki tomonlama iteratorlar: rbegin, crbegin, rend, crend Tartiblanmagan konteynerlarning usullari:

Xeshlash uslubi (Bu usullar tartibsiz konteynerlarning ishlashini boshqarish imkonini beradi):

- load_factor() – to‘ldirish koeffitsentini qaytaradi (yuqoriga qarang)
- max_load_factor() – Agar argumentsiz foydalanilsa, maksimal to‘ldirish qiymati uchun float tipini qaytaradi, agar argument sifatida float tipi foydalanilsa, maksimal to‘ldirish qiymati uchun o‘rnatiladi . Agar terminator omili ushbu chegaradan oshsa, konteyner avtomatik ravishda segmentlar sonini oshirishi mumkin.
- rehash(size_type count) – konteyner uchun qayta xeshlash o‘tkazadi, ya‘ni yacheykalarni sonini (backet_count) backet_count >= count va backet_count > size/max_load_factor bo‘lishi uchun.
- reserve(size_type count) – songa yacheykalar sonni o‘rnatadi, konteynerni qayta xeshlash o‘kazmaslik va maksimal to‘ldirish koeffitsienti uchun, oxirgi yacheykaning count hisoblanadi.

Maksimal samaradorlikka erishish uchun har doim maksimal yacheykalar sonini aniq belgilash kerak. Tavsiya etilgan qiymatlar 0.7 dan 0.8 gacha oraliqda bo‘ladi. Element qidirish uchun standart maksimal yacheykalar soni o‘rnatiladi (1.0ga teng) va dastuochilar tomonidan 0.7 o‘rnatiladi.

Xesh funksiyasining o‘zi konteyner faoliyatini yaxshilashi mumkin, ammo bu mavzu doirasidan chiqib ketishga olib keladi.

Interfeys segmentlari. Segment elementlari bilan bevosita ishlash uchun (bog‘langan ro‘yxat sifatida) quyidagi usullardan foydalaniladi:

- begin(), end(), cbegin(), cend() – iteratorla. E‘tibor bering, bular segmentlar uchun, massiv uchun emas! Konteynerlar uchun shularning anologlari yaratilgan.
- size_type bucket_count() – massivdagi segmentlar sonini qaytaradi.
- size_type max_bucket_count() – muuumkin bo‘lgan maksimal elementlar soni (tizimga va joriy qilishga bog‘liq).
- size_type bucket_size(size_type n) –n indeksli segmentdagi elementlar soni.
- size_type bucket(Key) – kalit asosida segment sonini qaytaradi.

Bu usullar asosida xesh jadvalni vizual ko‘rinishini tasavvur qilish mumkin.

1.1-dastur. Tartiblanmagan konteynerlardan foydalanish.

```
#include "stdafx.h"

#include <iostream>
#include <iomanip>
#include
<unordered_set>
#include <random>
#include <ctime>
#include <chrono>
using namespace std;

int main() {
    default_random_engine rnd(time(0));
    uniform_int_distribution<unsigned> g(100, 999);
    unordered_multiset<int> unm;
    size_t n = 60;
    unm.reserve(n);
    while (n--) unm.emplace(g(rnd));
    cout << "Konteynerdagi segmentlar soni: "
        << unm.bucket_count()
        << "\n"
        << "Maksimal to‘ldirishlar ko'ffitsenti: "
```

```
            << unm.max_load_factor()
            << endl;
    auto start = chrono::system_clock::now(); auto
    res = unm.find(200);
    auto end = chrono::system_clock::now();
    auto elapsed = end - start;
    cout << "Natija => "<< elapsed.count()<< endl;
    // O‘zimiz tomondam Maksimal to‘ldirishlar ko'ffitsenti o‘rnatamiz
    unm.max_load_factor(0.7);
    unm.rehash(200);
    cout << "Konteynerdagi segmentlar soni: "<< unm.bucket_count()<< "\n"
         << "Maksimal to‘ldirishlar ko'ffitsenti: "<< unm.max_load_factor()<< endl;
    start = chrono::system_clock::now();
    res = unm.find(200);
    end = chrono::system_clock::now();
    elapsed = end - start;
    cout << "Natija => "
         << elapsed.count()
         << endl;
    cout << " Hash tuzilmasi " << endl;
    for (size_t i = 0; i < unm.bucket_count(); i++) { cout
        << "Segment N:" << setw(3) << i << " => ";
        // Iteratorga lokal segmentlarni olish auto
        first = unm.cbegin(i);
        auto last = unm.cend(i);
        while (first != last) {
            cout << *first++ << " ";
        }
        cout << endl;
    }
      system("pause");
    return 0;
}
```

1.1-dastur. Output

```
Konteynerdagi segmentlar soni: 32 Maksimal
to‘ldirishlar ko'ffitsenti: 1 Natija => 30023
Konteynerdagi segmentlar soni: 256 Maksimal
to‘ldirishlar ko'ffitsenti: 0.7 Natija => 10004
Segment N:  0 =>        Segment N:  1 =>        Segment N:  2 =>        Segment N:  3
=> Segment N:  4 =>
Segment N:  5 =>        Segment N:  6 =>        Segment N:  7 =>        Segment N:  8
=> Segment N:  9 => 646
Segment N: 10 =>        Segment N: 11 =>        Segment N: 12 =>        Segment N: 13
=> Segment N: 14 =>
```

Segment N: 15 => Segment N: 16 => Segment N: 17 => Segment N: 18 => Segment N: 19 =>
Segment N: 20 => Segment N: 21 => Segment N: 22 => Segment N: 23 => Segment N: 24 => 940
Segment N: 25 => Segment N: 26 => Segment N: 27 => Segment N: 28 => Segment N: 29 =>

Segment N: 30 => 717 Segment N: 31 => Segment N: 32 => Segment N: 33 => Segment N: 34 =>
Segment N: 35 =>
Segment N: 36 =>
Segment N: 37 =>
Segment N: 38 =>
Segment N: 39 =>

Segment N: 40 => Segment N: 41 => Segment N: 42 => Segment N: 43 => Segment N: 44 =>
Segment N: 45 => 841 738

Segment N: 46 => Segment N: 47 => Segment N: 48 => Segment N: 49 => Segment N: 50 => Segment N: 51 => Segment N: 52 => 768 Segment N: 53 => Segment N: 54 => Segment N: 55 => Segment N: 56 => Segment N: 57 => Segment N: 58 => Segment N: 59 => Segment N: 60 => 631 Segment N: 61 => Segment N: 62 => 316 Segment N: 63 => Segment N: 64 => Segment N: 65 => Segment N: 66 => Segment N: 67 => Segment N: 68 => Segment N: 69 => Segment N: 70 => Segment N: 71 => Segment N: 72 => Segment N: 73 => Segment N: 74 => Segment N: 75 => Segment N: 76 => Segment N: 77 => Segment N: 78 => Segment N: 79 => Segment N: 80 => Segment N: 81 => Segment N: 82 => Segment N: 83 => Segment N: 84 => Segment N: 85 => Segment N: 86 => Segment N: 87 => Segment N: 88 => Segment N: 89 => Segment N: 90 => Segment N: 91 => Segment N: 92 => 498 Segment N: 93 => Segment N: 94 => 187 476
Segment N: 95 => Segment N: 96 => Segment N: 97 => Segment N: 98 => 280 Segment N: 99 => Segment N:100 => 225 Segment N:101 => Segment N:102 => Segment N:103 => Segment N:104 => Segment N:105 => 973 Segment N:106 => Segment N:107 => Segment N:108 => Segment N:109 => Segment N:110 => Segment N:111 => Segment N:112 => Segment N:113 => Segment N:114 => Segment N:115 => Segment N:116 => 960 Segment N:117 => Segment N:118 => Segment N:119 => Segment N:120 => Segment N:121 => 630 Segment N:122 => Segment N:123 => Segment N:124 => 567 Segment N:125 => Segment N:126 => Segment N:127 => Segment N:128 => Segment N:129 => Segment N:130 => Segment N:131 => Segment N:132 => Segment N:133 => 351 Segment N:134 => 227 Segment N:135 => Segment N:136 => Segment N:137 => Segment N:138 => Segment N:139 => Segment N:140 => Segment N:141 => Segment N:142 => Segment N:143 => Segment N:144 => Segment N:145 => Segment N:146 => 761 Segment N:147 => Segment N:148 => 145 Segment N:149 => Segment N:150 => Segment N:151 => Segment N:152 => Segment N:153 => Segment N:154 => Segment N:155 => Segment N:156 => Segment N:157 => Segment N:158 => Segment N:159 => Segment N:160 => Segment N:161 => Segment N:162 => Segment N:163 => Segment N:164 => Segment

N:165 => Segment N:166 => Segment N:167 => 242 Segment N:168 => Segment N:169 => Segment N:170 => Segment N:171 => Segment N:172 => Segment N:173 => Segment N:174 => Segment N:175 => Segment N:176 => Segment N:177 => Segment N:178 => Segment N:179 => Segment N:180 => 458 Segment N:181 => Segment N:182 => Segment N:183 => Segment N:184 => Segment N:185 => Segment N:186 => Segment N:187 => Segment N:188 => Segment N:189 => Segment N:190 => Segment N:191 => Segment N:192 => 852 Segment N:193 => Segment N:194 => Segment N:195 => Segment N:196 => Segment N:197 => 287 Segment N:198 => Segment N:199 => Segment N:200 => Segment N:201 => Segment N:202 => Segment N:203 => Segment N:204 => Segment N:205 => Segment N:206 => Segment N:207 => Segment N:208 => Segment N:209 => 483 Segment N:210 => 697

Segment N:211 => Segment N:212 => Segment N:213 => Segment N:214 => Segment N:215 => 967 Segment N:216 => Segment N:217 => Segment N:218 => Segment N:219 => Segment N:220 => Segment N:221 => Segment N:222 => Segment N:223 => Segment N:224 => Segment N:225 => Segment N:226 => Segment N:227 => Segment N:228 => Segment N:229 => 666 Segment N:230 => 484 Segment N:231 => Segment N:232 => 603 Segment N:233 => Segment N:234 => Segment N:235 => 254 Segment N:236 => Segment N:237 => Segment N:238 => Segment N:239 => Segment N:240 => Segment N:241 => Segment N:242 => Segment N:243 => Segment N:244 => Segment N:245 => Segment N:246 => Segment N:247 => Segment N:248 => Segment N:249 => Segment N:250 => Segment N:251 => Segment N:252 => Segment N:253 => Segment N:254 => Segment N:255 =>

Tartiblanmagan konteynerlarning sinflari murakkab tuzilishga ega, ammo standart vositalar bilan ham tartiblangan konteynerlardan kam bo‘lmagan mukammal ishlashni ko‘rsatadi. Tartiblangan konteynerlar yoki tartiblanmagan konteynerlar orasidagi tanlov vazifaga bog‘liq bo‘ladi. Assotsiativ konteynerlarning asosiy usullari keng tarqalganligi sababli, dasturning bir bajarilishidan boshqasiga o‘tish oson va testlarni o‘tkazganingizdan so‘ng tanlovingizni bir yoki bir nechta turli konteynerda amalga oshirib, tanlov qilishingiz mumkn.

Tartiblanmagan konteynerlarning sinflari xususiyatlari.

Headers		**<unordered_set>**		**<unordered_map>**	
Members		**unordered_ set**	**unordered_ multiset**	**unordered_ map**	**unordered_ multimap**
	constructor	unordered_ set	unordered_ multiset	unordered_ map	unordered_ multimap
	destructor	~unordered_ set	~unordered_ multiset	~unordered_ map	~unordered_ multimap
	assignment	operator=	operator=	operator=	operator=
	begin	begin	begin	begin	begin

iterators	end	end	end	end	end
	rbegin				
	rend				
const iterators	cbegin	cbegin	cbegin	cbegin	cbegin
	cend	cend	cend	cend	cend
	crbegin				
	crend				
capacity	size	size	size	size	size
	max_size	max_size	max_size	max_size	max_size
	empty				

Nazorat savollari

1. Ro‘yxatni initsializatsiya qilishda har bir juftlik alohida qanday qavs ichiga olinishi kerak.
2. Lug‘atlarda elementlarga KIRISHning yana ikkita usuli mavjud (iteratorlar bilan ishlashdan tashqari). Birinchi va ikkinchi usularni tushuntirib bering.
3. Indekslash funksiyasini assotsiativ konteyner sinflarning qaysi sinflariga foydalanish mumkin.
4. at() funksiyasi istisno qaytargani uchun try - catch funksiyasi bilan boshqarish mumkinmi?
5. operator[] noqulay va samarasiz deb hisoblasangiz, yana unga ikkita alternativ variantlar bor. Bu variantlarni ayting.
6. Qaysi to‘plamda qidirish faqat birinchi maydon, ya‘ni kalit maydon bilan amalga oshiriladi.
7. Qanday konteynerlar xesh – jadval asosida qurilgan va xesh funksiyalarga va tenglik amaliga tayanadi.
8. Ayrim vazifalarda saralash uchun vaqt ko‘p ajratilsa avtomatik saralashni muhim kamchilik deb hisoblash mumkin. Bunda dasturchi qanday yo‘l tutadi.
9. Tartiblanmagan konteynerlarning xesh jadvallarida nima bor va ular cheksiz ko‘p elementlarni o‘z ichiga olishi mumkin.
10. Tartiblanmagan konteynerlarning tiplari uchun shablon parametrlarini sanab bering.
11. Tartiblanmagan konteynerlarda agar equal taqqoslash funksiyasi yozilmagan bo‘lsa, joriy holat bo‘yicha qaysi funksiya ishlatiladi.
12. Xeshlash uslubi nima uchun kerak.
13. rehash(size_type count) – konteyner uchun qayta xeshlash o‘tkazadi, ya‘ni yacheykalarni sonini qaysi shartda bo‘lishi uchun.
14. Segment elementlari bilan bevosita ishlash uchun qanday usullardan foydalaniladi.
15. Tartiblangan konteynerlar yoki tartiblanmagan konteynerlar orasidagi tanlov nimaga bog‘liq bo‘ladi.

6-MA’RUZA
KONTEYNERLAR ADAPTERLARI. (2 SOAT)

Reja:

1. Stack, queue, priority_queue, deque.
2. Konteynerlar bilan ishlash algoritmlari.
3. Funktorlarning qo‘llanilishi.

Annotatsiya*: Adapterdarning muhim tushunchalari asosida stek, navbat, ustivor navbat, ikki tomonlama navbat sin fob’ekttlari va ularning xususiyatlari, funksiyalari, dasturlashda o‘zlarini tutishlari, konteynerlar bilan ishlashga mo‘ljallangan asosiy algoritmlar va ularning dasturlashdagi ahamiyati, funktor funksiyalarni yaratish va foydalanish uslublari, amallari, talablari, vazifalari va usullari keltirilgan bo‘lib, dasturchining adapterlar, algoritmlar va funktor funksiyalar bilan ishlashdagi tanlash imkoniyati bo‘yicha tajriba-xulosalari ko‘rsatilgan.*

Kalit so‘zlar: . *Konteyner, adapter, shablon, sinf, moslashtirish, stack,(stek), queue (navbat), priority_queue (ustuvor bilan navbat), deque (ikki tomonlama navbat), ma‘lumotlar tuzilmasi, push_back, pop_back, pop_front, LIFO, FILO, LIFO, for_each(), find(), funksional ob‘yektlar, Operator(), Standart ML va funktor, Haskell va funktor.*

KIRISH

Adapterlar standarti shablon kutubxonasining alohida toifasidir (vakilidir). Adapterlar yangi tushunchalar yoki ilovalar emas, balki mavjud kutubxona tushunchalarining o‘ziga xos, tez-tez ishlatiladigan maqsadlar uchun moslashtirishlari. Bu moslashtirish (adaptatsiyalash) ko‘pincha adapterning so‘rovlarnnig asosiy tushunchasiga asoslanib, funksiyalarini cheklash orqali amalga oshiriladi. Kutubxona konteynerlarning adapterlari, iteratorlar va funksiyalarni o‘z ichiga oladi.

Konteynerlarning adapterlari. Konteynerlarning adapterlari asosida adapterlarni hosil qilishning eng oson yo‘li misol orqali tushuntirishdir. Bular stack (stek), queue (navbat), priority_queue (ustuvor bilan navbat) misol bo‘ladi. Bu misollardan tushunarli bo‘ldiki, adapterlar:

- juda keng va tez-tez ishlatiladigan ma‘lumotlar tuzilmalari;
- har qanday realizatsiyani alohida bajarilishi kerak emas. Ularning funksiyalarini ta‘minlash uchun STL standart konteyneri har qanday moslashtirishda tayanch sifatida push_back, pop_back, yoki pop_front amallardan (adapter turiga qarab) foydalanish mumkin;
- adapter uchun tayanch konteynerning to‘plamidan qo‘shimcha amallar chiqarib tashlanishi kerak (vahimalar yaratmaslik uchun, masalan, indekslash amali uchun, konteyner vektor bo‘lsa, stek moslashtirish ishlatiladi);

Konteynerlarning adapterlari uchun sintaktik ta‘riflar misollar orqali qarab chiqish yaxshi natija beradi, shuning uchun misollar bilan tushntirib o‘tamiz. Birinchi navbvtda konteynerlarning adapterlari uchun *<stack>, <queue> kabi sinflarni dasturga qo‘shish lozim.*

Stek (stack, ma‘lumotlar yig‘indisi): stekda uning elementlariga faqat bir uchidan murojaat qilish mumkinligi bilan xarakterlanadi va stekning yuqori qismi deb ataladi. Bu LIFO (Last In — First Out , birinchi kirgan oxiri chiqadi) tamoyili bo‘yicha faoliyat ko‘rsatuvchi ma‘lumotlar to‘plamidir. Stekka oid deyarli barcha funksiyalarini ko‘rsatadigan dastur keltirilgan (1.1-dastur).

1.1-dastur. Stekka oid funksiyalardan foydalanish.

```
// Created by MBBahodir
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <stack>
#include <string>
#include <array>
using namespace std;
int main() {
  stack<int> st, st_one, st_two;
  stack<string,vector<string>> vst;
  vst.emplace("str1");
  vst.emplace("str2");
  vst.push("eostr");
    for (int i=0; i<5; ++i) st.push(i);
  for (int i = 0; i < 15; i++)
  st_one.emplace(i*10);
    st_two = st;
    st.swap(st_one);
    cout << st.size() << " " << st_one.size() << " " << st_two.size() << " " << vst.size()
<< endl;
    while (!vst.empty()) {
          cout << vst.top() << " : stack dagi o‘rni " << vst.size() << endl; vst.pop();
          }

  int size = st.size(); while
  (size--) {
           cout << st.top() << " : stack dagi o‘rni " << st.size() << endl; st.pop();
  }

  while (st.size() != 0){
           cout << st_two.top() << " : stack dagi o‘rni " << st_two.size() << endl; st_two.pop();
  }
  while (!st.empty()) {
           cout << st.top() << " : stack dagi o‘rni " << st.size() << endl; st.pop();
  }
  cout << st.size() << " " << st_one.size() << " " << st_two.size() << " " << vst.size()
<< endl;
```

```
cout << " stack st " << (st.empty() ? "" : "not ") << " empty" << endl; cout << "
stack vst" << (vst.empty() ? "" : "not ") << " empty" << endl;
cout << " stack st_one " << (st_one.empty() ? "" : "not ") << " empty" << endl;
system("pause");
return 0;
}
```

1.1-dastur. Output.

```
15 5 5 3
eostr : stack dagi o‘rni 3 str2 :
stack dagi o‘rni 2 str1 : stack
dagi o‘rni 1 140 : stack dagi
o‘rni 15
130 : stack dagi o‘rni 14
120 : stack dagi o‘rni 13
110 : stack dagi o‘rni 12
100 : stack dagi o‘rni 11
90 : stack dagi o‘rni 10
```

```
80 : stack dagi o‘rni 9
70 : stack dagi o‘rni 8
60 : stack dagi o‘rni 7
50 : stack dagi o‘rni 6
40 : stack dagi o‘rni 5
30 : stack dagi o‘rni 4
20 : stack dagi o‘rni 3
10 : stack dagi o‘rni 2
0 : stack dagi o‘rni 1
0 5 5 0
 stack st  empty stack
 vst empty
 stack st_one not  empty
```

1.1 - dasturda keltirilgan stek adapterining funksiyalari bo‘yicha dasturni tahlil qilish orqali ba‘zi xulosalar keltiramiz:

- Stack<string> elementlari satr bo‘lgan o‘zgaruvchini e‘lon qiladi. String ob‘yektlar o‘zlari STL konteynerlar ekanligiga e‘tibor bering. Shunday qilib, stek konteynerlarning har qanday elementlarini o‘z ichiga olgan konteynerlarni olishi mumkin (bu boshqa STL konteynerlari uchun xosdir).

- stack<string> e‘lon qilishni misollarning ko‘pchiligida stekka tavsif sifatida ko‘rish mumkin. Ko‘pchilik dasturchi buni boshqa bo‘lishi bilmasligi va tassafur qilmasligi mumkin. Lekin bir xil ta‘rif mavjud stack<string, vector<string>> satrlari bir steki, bunda tayanch sinf vector<string> tanlab olingan. Masalan, *vector*, *list* va *deque* yoki hatto dasturchining konteyner sinfi tayanch sinflar sifatida foydalanish. Ba‘zan so‘rashdi, nima uchun stack< vektor<string>> yozish mumkin emas (birinchi stringni o‘chirgan holda)? Chunki stack konstruktorini amalga oshirishda bu yo‘l belgilangan va bu juda mumkin. Ammo, butunlay boshqa turdagi tavsif bo‘ladi: string vektorlari to‘plami stack< vektor<string>> umuman boshqa stek bo‘ladi (konteynerlarning tarkibiy joylashtirish haqidagi eslatma).

- Juda ko‘p holarda stekni vektor ko‘rinishida initsializatsiya qilinda, ammo bu faqat C++11 variantida mavjud. 1.1-dasturda emplace() funksiyasidan stekka element qo‘shish uchun foydalanilgan.

- Keyingi fragmentlarda, deyarli barcha amallarni (usullarini) ko‘rish mumkin: stack:push() – stekka bitta element qo‘shish, top() – stekning eng yuqorida turgan elementiga ko‘rsatkich olish, pop() – yuqori elementini chiqarib tashlash, size() – stekning joriy hajmi, empty() – stekning bo‘shligini tekshirish, swap() – ikki stekning almashtirish, operatop= stekni qiymat qilib boshqa bir tekka berish (o‘zlashtirish).

- Dasturdan ko‘rish mumkinki, stek adapteri tayanch konteynerga xos usullarni yo‘qotdi (at(), operator[] va hokazo.), lekin push(), pop() funksiyalarni qayta aniqlash orqali orttirilgan hisoblanadi.

Stekni tushinib olib, o‘xshash funksiyalarni navbatga joriy qilish qiyin masala emas. Stekdan farqli o‘laroq, navbat FIFO (First In — First Out birinchi kirgan birinchi chiqadi) tamoyili bo‘yicha faoliyat yuritadigan ma‘lumotlar to‘plamidir. (Bu usulni bir uchiga oqib KIRISH, keyin boshqa uchidan oqib chiqadigan quvurga o‘xshatish mumkin). 1.1- dasturda til semantikasi talabi bo‘yicha tahrirlarni qilib navbat uchun deyarli o‘zgarishsiz qoldiramiz va 1.2 dasturni olamiz.

1.2-dastur. Navbat sinfining funksiyalari.

```
// Created by MBBahodir
#include "stdafx.h"
#include <iostream>
 #include <list>
#include <queue>
```

```
 #include <string>
#include <array>
using namespace std;
int main() {
        queue<int> st, st_one, st_two;
        queue<string,list<string>> vst;

 vst.emplace("str1");
 vst.emplace("str2");
 vst.push("eostr");

  for (int i=0; i<5; ++i) st.push(i);

  for (int i = 0; i < 15; i++) st_one.emplace(i*10);

  st_two = st;
  st.swap(st_one);
```

```
  cout << st.size() << " " << st_one.size() << " " << st_two.size() << " " << vst.size()
<< endl;

  while (!vst.empty()) {
        cout << vst.front() << " : stack dagi o‘rni " << vst.size() << endl; vst.pop();
      }

int size = st.size(); while
(size--) {
        cout << st.front() << " : stack dagi o‘rni " << st.size() << endl; st.pop();
}
while (st.size() != 0){
        cout << st_two.front() << " : stack dagi o‘rni " << st_two.size() << endl; st_two.pop();
}

while (!st.empty()) {
        cout << st.front() << " : stack dagi o‘rni " << st.size() << endl; st.pop();
}
```

```
 cout << st.size() << " " << st_one.size() << " " << st_two.size() << " " << vst.size()
<< endl;


 cout << " stack st " << (st.empty() ? "" : "not ") << " empty" << endl; cout << "
 stack vst" << (vst.empty() ? "" : "not ") << " empty" << endl;
 cout << " stack st_one " << (st_one.empty() ? "" : "not ") << " empty" << endl;
 system("pause");
 return 0;
 }

 1.2-dastur.Output
100 : stack dagi o‘rni 5
110 : stack dagi o‘rni 4
120 : stack dagi o‘rni 3
130 : stack dagi o‘rni 2
140 : stack dagi o‘rni 1
0 5 5 0
 stack st  empty stack
 vst empty
 stack st_one not  empty
```

1.1-dasturda o‘zgartirish talab qilinganlarini o‘zgartirib, quyidagi xulosalarga olib keladi:

- navbatda pop_front() usuli vektor konteyner ustiga qurib bo‘lmaydi, , lekin uni ro‘yxat konteyner uchun qurish mumkin.

- Deque sinfi uchun *front(), push_back(), pop_front*()), usullari har qanday konteynerlarda ishlatiladi (faqat pop_front() usuli borlarida).

- shuning uchun navbat uchun yasaladigan ob‘yektda, ya‘ni deque sinfida top() usuli o‘rniga front() usuli ishlatiladi.

Ma‘lum bir tartibda qayta ishlanishi kerak ma‘lumotlarni uchun umumiy ma‘lumotlar tuzilishiga asoslangan to‘plami bu – stek va navbatdir (stack, deque). Masalan, biror funksiya o‘z navbatida uchinchi funksiyani chaqiradigan boshqa funksiyani chaqirsa, uchinchi funksiya birinchi funksiyani emas, ikkinchi funksiyaga qaytishi muhimdir.

Bu ma‘lumotlarni qayta ishlash tartibini amalga oshirish uchun dasturchi o‘zining navbat funksiyasi tashkil qilish lozim. Stekka qo‘shilgan oxirgi qiymat birinchi bajariladi va aksincha, stekka qo‘shilgan birinchi qiymat oxirgi bajariladi. Shunday qilib, ma‘lumotlar tuzilmasining o‘zi chaqirishlarning to‘g‘ri belgilangan tartibda berilishini ta‘minlaydi.

Konseptual, ma‘lumotlar tuzilishi - stek juda oddiy: u ma‘lumotlarni kiritish yoki chiqarish uchun ma‘lum bir tartibni belgilaydi Har safar bir element qo‘shiladi, bu stekka yuqori element bo‘lib tugaydi. Stekdan olib tashlanadigan yagona element stekning yuqori qismida joylashgan elementdir. Shunday qilib, stekka “birinchi KIRISH, oxirgi chiqish — FILO” yoki “oxiri KIRISH, birinchi chiqish

— LIFO” deb ham aytiladi. Demak, Stekka qo‘shilgan birinchi element undan oxirgi chiqariladi.

Xo‘sh, bu nima? Nima uchun stek kerak? Yuqorida aytib o‘tganimizdek, funksiyalarni (qiymatlarni) chaqirishni tashkil qilish uchun qulay yo‘lidir. Aslida, stek chaqirish (call stack) tez-tez ishlaydigan yoki boshqa funksiyalarning qaytish qiymatini kutayotgan funksiyalar ro‘yxatiga murojaat qilish uchun ishlatiladigan atamadir.

Shuningdek, stek informatika faning fundametntal tilining bir qismidir. Agar birinchi kelib, oxirish ketish tamoyili asosida fikrlasangiz, unda stek terminini ishlatish lozim.

Bundan tashqari, bunday (stek kabi) navbatlar ko‘p jarayonlarda ishtirok etadi, nazariy informatikadan tortib, bunday push-down funksiyalari va juda ham ko‘p bag usullarida.

Stek assosiativ usullarga ega:

Push – stekka element qo‘shish.. Pop – stekdan element o‘chirish Top – elementni ko‘rish.

LIFO - stek xarakati.

FILO - stek xarakati (ekvivalet LIFO ga).

Stack (stek) ma‘lumotlar tuzilmasini amalga kutubxona yaratish masalani qaraymiz. Umumlashgan dasturlash usuliga mos keladigan umumiy stack, ya‘ni Stack sinfi uchun shablon yaratish degani. Agar shablonlar bilan tanish bo‘lmasangiz, funksiya shablonlariga oid nazariy ma‘lumotlarga qarang, shablonlar bilan ishlash mexanizmi batafsil bilish lozim. Agar shablonlar bilan tanish, lekin sinf shablonlari bilan ishlashni unutgan bo‘lsangiz, sinf shablonlari haqida nazariy materiallarni o‘qib o‘rganing.

Bu stack shablonlari bilan amalga oshirildigan kutubxonani deyarli har qanday ma‘lumotlar tipi uchun foydalanish mumkin. Bundan tashqari, stack hajmi dastur ijrosi davomida jadal belgilanadi. Stackga qo‘shimcha funksiya ham qo‘shildi peek(), bu stackning yuqori qismidan n- elementini qaytaradi.

1.3-dastur. Stack shablon sinfi

```
#ifndef STACK_H
#define STACK_H
// Created by MBBahodir
#include <cassert>
#include <iostream>

#include <iomanip>

template <typename T>
class Stack{
private:
    T *stackPtr;                        // stackka ko‘satkich
    const int size;                     // stackdagi maksimal elementlar soni
    int top;                            // stackning joriy elementi
public:
    Stack(int = 10);                    // stackning joriy holatda elementlar soni 10 ta
    Stack(const Stack<T> &);            // nusxalsh konstruktori
    ~Stack();                           // destruktor

    inline void push(const T & );       // elementni stackning yuqori qismiga joylashtirish
    inline T pop();                     // stack ichidan eng yuqoridagi elementni o‘chirish
    inline void printStack();           // stackni ekranga chiqarish
    inline const T &Peek(int ) const; // stackning yuqorisidagidam keyingi n-element inline int
    getStackSize() const;  // stack o‘lchamini olish
    inline T *getPtr() const;           // stackka ko‘rsatkich olish
    inline int getTop() const;          // joriy elementni olish
};

// Stack sinfi shablon funksiyalarini realizatsiyz qilish

// konstruktor Steka
template <typename T>
Stack<T>::Stack(int maxSize): size(maxSize) // o‘zgarmaslarni initsalizatsiyalash
{
    stackPtr = new T[size]; // xotira ajratish top = 0;
    // joriy elementno 0 bilan toldirish;
}
```

```
// nusxalsh konstruktori template
<typename T>
Stack<T>::Stack(const Stack<T> & otherStack) : size(otherStack.getStackSize())
    // o‘zgarmaslarni initsalizatsiyalash
{
    stackPtr = new T[size]; // yangi xotira ajratish top =
    otherStack.getTop();

    for(int ix = 0; ix < top; ix++) stackPtr[ix] =
        otherStack.getPtr()[ix];
}
// destruktor template
<typename T>
Stack<T>::~Stack()
{
    delete [] stackPtr; // ochirish
}
// element qo‘shish
template <typename T>
inline void Stack<T>::push(const T &value)
{
    // o‘lchamni o‘zgartirish
    assert(top < size); // joriy element soni o‘lchamdan kichik bo‘lishi kerak

    stackPtr[top++] = value; // elementni joylashtirish
}
// elementni o‘chirish
template <typename T>
inline T Stack<T>::pop()
{
    // o‘lchamni tekshirish
    assert(top > 0); // joriy element 0 katta bo‘lishi kerak

    stackPtr[--top]; //   elementni o‘chirish
return 0;
}
// yuqori elementdan n-elementni qaytaradi
template <class T>

inline const T &Stack<T>::Peek(int nom) const
```

```
{
   //
   assert(nom <= top);

   return stackPtr[top - nom]; // n-elementni qaytarish
}

// ekranga chiqarish
template <typename T>
inline void Stack<T>::printStack()
{
      for (int ix = top - 1; ix >= 0; ix--)
            cout << "|" << setw(4) << stackPtr[ix] << endl;
}

// o‘lchamni qaytarish
template <typename T>
inline int Stack<T>::getStackSize() const
{
      return size;
}
// ko‘rsatkich qaytarish (nusxalash konstruktori uchun)
template <typename T>
inline T *Stack<T>::getPtr() const
{
      return stackPtr;
}
```

```
// o‘lchamni qaytarish
template <typename T>
inline int Stack<T>::getTop() const
{
    return top;
}
#endif // STACK_H
```

Stack shablon sinfi *.h (header) faylda yaratildi. Shablon sinf interfeysi va amalga oshirish ham bir xil faylda bo‘lishi kerak, bo‘lmasa, shunga o‘xshash mazmun bilan xato berishi mumkin. error undefined reference to "method of a class template»

Ya‘ni, xatolik, shablon sinf usullari havolalarda aniqsizlik.

Shablon sinf interfeysining fragmentlarida 9 -dan 28gacha yangilandi. Sinfning barcha usullari izohlarni o‘z ichiga oladi va mening fikrimcha, ularning funkiyalarini alohida ta‘riflash mantiqiy to‘g‘ri emas. Stack shablon sinfning barcha usullari inline funksiyalari sifatida e‘lon qilinadi. Bu sinf ishini tezlashtirish maqsadida amalga oshiriladi. Sinfning ajralmas funksiyalari tashqi funsiyalarga qaraganda tezroq ishlaydi.

Shablon interfeysidan so‘ng, stack sinf usullari amalga oshiriladi, kutubxona fragmentlarida 32-117gacha yangilandi. Agar stack sinf, shablonlar va sinflar usullarini amalga oshirish haqida bilsangiz shablon sinfni yaratishda murakkab hech narsa yo‘q. Sinfda ikkita konstrkutor bor, birinchisi fragmentning 32-33 qatorlarida - bu standart konstruktor hisoblanadi. Fragmentning 41-50 qatordagi nusxa konstruktor hisoblanadi. Bir ob‘yektni boshqasiga nusxalash uchun kerak. Peek usuli (funksiyasi), fragmentning 80-88 fatorlarida stack elementlarni ko‘rish imkonini beradi. Faqat element raqamini kiritishingiz kerak, hisoblash stack yuqorisidagi elementdan boshlanadi va keraklisini qaytaradi. Boshqa funksiyalari xizmatchi funksiyalar hisoblanadi, ular tashqi sinfdna foydalanish uchun mo‘ljallangan, ya‘ni prinstack() funksiyasidan tashqari, bu funksiya stack elementlarini ekranga chiqaradi.

Endi stack shablon sinf uchun dastur tuzaylik va har doimgidek, stack sinf shablonni sinov bo‘ladi hamda asosiy funksiyalaridan foydalanishni ko‘ramiz.

1.4-dastur. Shablon sinfdan foydalanish.

```
// Created by MBBahodir
#include "stdafx.h"
#include <iostream>
using namespace std;
#include "stack.h"
int main()
{
```

```
    Stack<char> stackSymbol(5);
    int ct = 0;
    char ch;

    while (ct++ < 5){ cin >> ch;
        stackSymbol.push(ch); // element qo‘shish
    }
```

```
    cout << endl;

    stackSymbol.printStack(); // ekranga chiqarish

    cout << "\n\n Element o‘chirish\n";
    stackSymbol.pop();
    stackSymbol.printStack(); // ekranga chiqarish
    Stack<char> newStack(stackSymbol);
    cout << "\n\nNusxalash konstruktori!\n";
    newStack.printStack();
    cout << "2 tartibdagi element: "<< newStack.Peek(2) << endl;
        system("pause");
    return 0;
}
```

1.4-dastur. Output.

```
b
a
h
o
q

|     q
|     o
|     h
|     a
|     b

 Element o‘chirish
|     o
|     h
|     a
```

```
|     b
Nusxalash konstruktori!
|     o
|     h
|     a
|     b
 2 tartibdagi element: h
```

1.4-dasturda Bir stack ob‘yektini yaratilgan, stack hajmi 5ga teng va stack ko‘pi o‘z ichiga olishi mumkin bo‘lgan elementlar soni 5 ga teng. stackni while takrorlanish operatori bilan to‘ldiriladi. Ekranda stack elementlarini chiqariladi, keyin stackdan bir elementi o‘chiriladi va yana ekranda stack elementlarini chiqariladi. Natija asosida u o‘zgardi, aniq bir element o‘chirildi.

Shuningdek, dasturda nusxa konstruktor ishlatiladi. Peek() funksiyasi stackning ikkinchi elementini qaytardi.

Stack shablon sinf yaxshi yaratildi va to‘g‘ri ishlamoqda. uni, masalan, int ma‘lumotlar turida sinab ko‘rcangiz bo‘ladi. Ishonchim komilki, hamma narsa to‘g‘ri ishlaydi.

Navbat (queue). Navbat ma‘lumotlar tuzilmasi bo‘lib (yuqorida aytib o‘tilganidek) bu LILO tamoyilini qo‘llab quvvatlaydi (last in-last out: oxirgi kirgan oxirgi chiqadi). C++ allaqachon tayyor STL konteyner-navbat (queue) mavjud.

Navbatda, birinchi elment kiritilgan bo‘lsa, u ham birinchi chiqariladi. 4 ta elementni qo‘shsangiz, birinchi qo‘shilgan element birinchi, ikkinchi qo‘shilgan element ikkinchi chiqadi.

Navbat qanday ishlashini tushunish uchun do‘kon navbatini tasavvur qilishingiz mumkin. Siz esa uning o‘rtasida turibsiz, shunda kassaning oldida turibsiz, avval oldingizdagi barcha odamlarga xizmat qilinishi. Lekin oxirgi navbatda o‘tgan kishi uchun, o‘zi tashqari barcha odamlarga xizmat qilish kerak emas.

1	9	8	0	1	2	8

Bu yuqorida keltirilgan ro‘yxatga qarang. Bulardan birortasini chiqarish uchun shu ketma-ketlik asosida chiqarib, keraklisiga to‘xtaladi. Masalan, 0 sonnini chiqarish, uchun avvl 3 ta son chiqariladi so‘ng kerakli son.

Stack shablon sinfida bir peek() (bu indeks bo‘yicha elementni qaytaradi), bu navbat shablonida muayyan elementni ko‘rish mumkin emas.

Agar barcha navbat elementlarga KIRISH kerak bo‘lsa, lekin, navbatda buni amalga oshirish mumkin. Keyinroq bu qanday amalga oshirishni ko‘rsatib o‘tamiz.

C++da navbat yaratish. Agar C++ da navbat shablonini ishlatmoqchi bo‘lsangiz, avval <queue>kutubxonasini ulashingiz kerak.

Keyin navbatni e‘lon qilish uchun quyidagi sintaktikni ishlatishingiz kerak:

```
queue <type> <name>;
```

Birinchi queue so‘zini yozishimiz kerak.

Type ga navbatni to‘ldirish uchun kerakli tipni ko‘rsatishimiz lozim. Navbat ob‘yektining nomini ko‘rsatishamiz kerak.

Masalan,

```
queue <int> q;
```

Navbat usullari va funksiyalari:

Navbatda funksiya bu bir xil funksiyadir, lekin u faqat STL konteynerlari bilan ishlaydi. Bu funksiyalarga push(), pop(), front(), back(), empty() funksiyalar kiradi.

push() – navbatga yangi element qo‘shish.

pop() – navbatdan birinchi elementni o‘chirish, ikkinchi element birinchi element o‘rniga suriladi.

front() – navbatdagi birinchi elemeniga murojaat back() - navbatdagi birinchi elemeniga murojaat

empty() – navbat bo‘sh yoki bo‘shmasligini tekshiradi. Agar bo‘sh bo‘lsa true, aks holda false qiymat qaytaradi.

Bu keltirilgan funksiyalarga doir 1.5-dasturini keltiramiz.

1.5-dastur. Navbat funksiyalaridan foydalanish.

```
// Created by MBBahodir
#include "stdafx.h" #include
<iostream> #include <queue>
using namespace std;
int main() { queue
  <int> q;

  for (int h = 0; h < 7; h++) { q.push(rand()%100);
  }
```

```
    cout << "Navbatdigi birinchi element " << q.front() << endl; cout <<
    "Navbatdigi oxirgi element " << q.back() << endl; q.pop();
    cout << "Navbatdigi birinchi element (o‘chirishdan so‘ng): " << q.front() << endl; cout <<
    "Navbat bo‘sh " << (q.empty()?"":"emas") << endl;
    system("pause"); return 0;
}
```

1.5-dastur.Output.

```
Navbatdigi birinchi element 41
Navbatdigi oxirgi element 78
Navbatdigi birinchi element (o‘chirishdan so‘ng): 67
Navbat bo‘sh emas
```

Massiv yordamida navbatlarni yaratish. Yuqorida aytganimizdek, navbatni massiv orqali amalga oshirish mumkin. Odatda, agar dasturchi bunday navbatni yarasa, massiv navbat deb ataladi. Shuningdek, buni massiv deb ham aytish mumkin, ammo massiv so‘zi C++ da bor. Shuning uchun, navbat shablon deb aytish ham mumkin va bir xil tarzda uni chaqiriladi.

Buning uchun 2 ta ko‘shimcha o‘zgaruvchi yaratish kerak. Birinchisi start bo‘lsin va navbatning birinchi elemenitini bildirsin. Ikkinchisi ends bo‘lsin va navbatning oxirgi elemenitini bildirsin.

queue[ends] - ifodasini ishlatib navbatning oxirgi elementini qiymatini qaytaradi.

queue[start] - ifodasini ishlatib navbatning oxirgi elementini qiymatini qaytaradi.

Navbatdagi birinchi elementni o‘chirish uchun start ni birga kamaytirish kerak xolos, navbatni bo‘shligini tekshirish uchun start=ends mantiqiy shartni tekshirishning o‘zi kifoyadir. Agar mantiqiy shart true qaytarsa navbat bo‘sh, akas holda bo‘sh emas.

1.6-dastur. Massiv yordamida navbatlardan foydalanish.

```
// Created by MBBahodir
#include "stdafx.h"
#include <iostream>
#include <queue>
using namespace std;

int main() {
```

```
   //queue <int> q; int
   q[3];
   int start = 0, ends = 0;
   for (int h = 0; h < 3; h++) { int a;
            cin >> a; q[start++] = a;
   }
   cout << "Navbatdagi birinchi element " << q[start - 1] << endl; start-
   -;  // bir elementni o‘chirish
   cout << " Navbatdagi birinchi element (o‘chirilgandan so‘ng) " << q[start - 1] << endl;
   cout << " Navbatdagi oxirgi element " << q[ends] << endl;
   cout << "Navbat bo‘sh " << ((start == ends)?"":" emas ") << endl;
```

```
   system("pause");
   return 0;
}
```

1.6-dastur.Output

```
9
7
5
Navbatdagi birinchi element 5
  Navbatdagi birinchi element (o‘chirilgandan so‘ng) 7 Navbatdagi oxirgi
  element 9
Navbat bo‘sh   emas
```

Ustuvor navbat (Priority_queue). Ustuvor navbat (Priority_queue) bilan navbat (queue) - oddiy navbatdek, lekin ustuvor navbatga yangi element qo‘shilasa, shunda navbat kamayish tartibida saralanadi. Shu tariqa ustuvor navbatda eng katta element birinchi o‘ringa chiqadi. Ustuvor navbat shablonidan quyida sintaktik orqali foydalanish kerak:

```
priority_queue <type> <name>;
```

Birinchi priority_queue so‘zini yozishimiz kerak.

Type ga navbatni to‘ldirish uchun kerakli tipni ko‘rsatishimiz lozim. Navbat ob‘yektining nomini ko‘rsatishamiz kerak.

Ustivor navbatda yangi element qo‘shish uchun push() funksiyasi, birinchi o‘rindagi elementiga murojaat qilish uchun top() funksiyasi ishlatiladi. Front() – funksiyasi ishlatilmaydi. Shuningdek, back() – oxirgi elementga murojaat funksiyasi ham ishlatilmaydi.

1.7-dastur. Ustuvor navbatni (Priority_queue) yaratish va usullaridan foydalanish.

```
// Created by MBBahodir
#include "stdafx.h"
#include <iostream>
#include <queue>
using namespace std;

int main() {
        priority_queue <int> priority_q, priority_q_one;

   for (int h = 0; h < 7; h++) {
            priority_q.push(rand()%100);
   }

   priority_q_one  = priority_q;

   cout << "priority_q navbatdigi birinchi element " << priority_q.top() << endl; cout << "
    priority_q_one navbatdigi birinchi element " << priority_q_one.top() <<
endl;
   priority_q.pop();

   cout << "priority_q navbatdigi birinchi element (o‘chirishdan so‘ng): " << priority_q.top() <<
endl;
```

```
   cout << "priority_q navbat bo‘sh " << (priority_q.empty()?"":"emas") << endl;

   while (!priority_q_one.empty())
   {
            cout << "priority_q_one navbatdigi element : " << priority_q_one.top() <<
endl;
            priority_q_one.pop();
   }
   cout << " priority_q_one navbat bo‘sh " << (priority_q_one.empty()?"":"emas") << endl;
   system("pause");
   return 0;
}
```

1.7-dastur. Output

```
priority_q navbatdigi birinchi element 78
 priority_q_one avbatdigi birinchi element 78
priority_q navbatdigi birinchi element (o‘chirishdan so‘ng): 69 priority_q
navbat bo‘sh emas
priority_q_one navbatdigi element : 78
priority_q_one navbatdigi element : 69
priority_q_one navbatdigi element : 67
priority_q_one navbatdigi element : 41
priority_q_one navbatdigi element : 34
priority_q_one navbatdigi element : 24
priority_q_one navbatdigi element : 0

priority_q_one navbat bo‘sh
```

Navbatlarni yaratishni ikki usulini kshrib chiqdik. Queue sinf shabloni va massiv asosida. Aslida koʻproq queue sinf shablonidan foydalangan maqul. Chunki bu massivga nisbattan tez ishlaydi.

Ikki tomonlama navbat (deque, double-ended queue). Deque – shunday maʻlumotlar tuzilmasiga aytiladiki, unda maʻlumotlarni qoʻshish va oʻchirish oldindan va orqadan amalga oshiriladi. Deque ham xuddi queue dek xotirada saqlanadi. Dek (Deque) bu shunday dinamik massivki, u ikki tomondan kengayishi va boshqa navbatlar va toʻplamlarga qaraganda tezroq.

Bundan tashqari, u har qanday holatda oʻzi elementlarni boshida yoki oxirida kiritish uchun ruxsat beradi, lekin oʻzboshimchalik holatda elementlarni tez kiritish imkoniyati saqlab qolinmasligi mumkin. Vektor oʻxshash amallari kam boʻlishi mumkin.

Dek konteyner bilan tanishar ekanmiz u nimasi bilan yaxshi, boshqa konteynerlardan fanday farq qiladi degan savollar paydo boʻlishi tabiiy albatta. Dekning umumiy tavsifidan boshlaymiz. STL da bu dek oʻzi nima? Tushuntirish uchun, uzoqqa bormaymiz, jismoniy tarbiya darsini olamiz. Koʻpchiligimiz "boʻy boʻyicha saflanish" nimaligini bilamiz. Shunday qilib, haqiqiy Dek shunday qurilish mexanizmiga ega. Unda shunday bir qator hosil boʻladiki va bu qator uning oxirida va uning boshidan toʻldiriladi, zarur boʻlsa, bunda ketma-ket elementlarni oʻrnini almashtirishi ham mumkin. Qancha tezroq turish kerak boʻlsa, shuncha yaxshi va stek, navbat maʻlumotlar tuzilmasiga qaraganda? Albatta, yo boshida yoki qatorning oxirida va qatorning oʻrtasida taxminan sizning boʻy balandligiga qarash va keyin solishtirish va oʻrinlarni oʻzgartirish kerak. Biroq, bir qator oʻrtasida joylashtirilgan boʻlsa, butun elementlar suriladi. Bu hayotdan dekka qiyosiy misol edi.

Shakllantirilayotgan toʻplamning boshida yoki oxirida elementlarni qoʻshish kerak boʻlganda, baʻzi elementlarni oʻrtaga qoʻyish kerak boʻladi dekni ishlatish mantiqan toʻgʻri keladi (mos ravishda shu tipdagi dek tanlash kerak). Dekda ketma-ketlik oʻrtasida qoʻshish va oʻchirish qoʻllab-quvvatlash boʻlsa-da, bu amallar chiziqli vaqtida amalga oshiriladi. Dasturda bu kabi amallar koʻp bajarish kerak boʻlsa, roʻyxatlar eng yaxshi tanlov boʻladi.

Aslida, Dek-bu boshida yoki oxirida elementlarning tezroq qoʻshilishi imkoni boʻlgan vektordir. Shu bilan birga, Dekning boshqa amallari vektorlarning tegishli amallari bilan bir xil ishlaydigan vaqtga (amallar koʻnaysa sekinroq) ega.

Dek (navbat queue) bilan va vektor (vector) oʻrtasidagi farqlar:

1. Deklarning boshiga va oxiriga ham elementlarni qoʻshish tezkor amaldir,

ammo vektorlar uchun faqat oxiriga qo‘shish tezkor amal hisoblanadi. Bu qo‘shish doimiy vaqt talab etadi.

2. Dekning funksiyalar va elementlarni qo‘shishga nisbattan vektordagi shunga o‘xshash funksiyalar odatda, bir oz sekin.

3. Dekning iteratorlari maxsus tipdagi aqlli ko‘rsatkichlarga ega bo‘lishi kerak, chunki doimiy ko‘rsatkichlar ular turli qismlari o‘rtasida biridan ikkinchisiga o‘tish kerak.

4. Xotira bloklari hajmiga chegaralari bor tizimlarida ikki tomonlama navbat dek xotirani ortiqg‘i bilan ajratadi, chunki vektor olishi mumkin bo‘lgan ma‘lumotlarga nisbattan ko‘p. Shunday qilib, max_size () funksiyasining natijasi vektorlarga nisbatan ikki tomonlama navbatlarga katta bo‘lishi mumkin

5. Ikki tomonlama navbatlar imkoniyatlarni boshqarishni ta‘minlamaydi va xotira qayta ajratilganda unga joy aniqlanmaydi. Xususan, boshida yoki oxirida elementlarni qo‘shish boshqa har qanday qo‘shish yoki o‘chirish kabi ikki tomonlama navbatda elementlar bilan bog‘liq barcha ko‘rsatkichlarni, murojaatlarni iteratorlar bekor qilmaydi. Biroq, xotiraning taqmislanishi vektorlarga qaraganda yaxshiroq ishlashi mumkin, chunki ular odatda ichki tuzilishiga ko‘ra, ikki tomonlama navbatlar xotirani qayta joylashtirishda barcha elementlarining nusxalash shart emas.

6. Deklarda ishlatilmaydigan xotira bloklari band qilinmaydi va real o‘lchamdan so‘ng xotiraning bo‘sh joylari ozod qilinadi. Buesa Dekning xotira hajmi kamaytirishga imkon beradi (ammo, qachon va qanday qilib bu sodir bo‘lshi uni amalga oshirishga bog‘liq).

Dekka ta‘luqli bo‘lgan vektorlarning afzalligi.

1. To‘plamning o‘rtasida elementlarni qo‘shish va o‘chirish sekinroq amal hisoblanadi. Chunki bir o‘zgarish uchun barcha elementlarning o‘rinlarini almashtirish kerak bo‘ladi.

2. Vektorlar va deklarning iteratorlari erkin ruxsat turiga mansub.
3. Dek sinfi vektor sinfining funksiyalari bilan bir xil funksiyalarga ega.
4. Tez-tez qo‘shish yoki o‘chirish dekning boshida yoki oxirida sodir bo‘lishi.
5. Konteyner elementlariga havola qilaolmaysiz.
6. Konteyner ishlatilmay qolganda xotirani xoli qilish amalga oshirilishiga C++ qafolat bermaydi (amalga oshirish tamoyiliga qarab, xoli bo‘lshi va bo‘lmasligi ham mumkin)

Inobatga olishini kerak bo‘lgan holatlar:

1. Dekning at() boo‘qa biror bir funksiyasi iterator va indeksni tekshira olmaydi.

2. Qo‘shish va o‘chirish qo‘shimcha xotira talab qilish mumkin. Shuning uchun qo‘shish va o‘chirishga doir amallar bajarilganda dekning boshqa elementlari uchun ko‘rsatkichlar, havolalar va iteratorlarni aniq ko‘rsata olmaydi. Bunga istisno bo‘lib,

Dekning boshiga yoki oxiriga element qo‘shish olinadi. Boshiga yoki oxiriga qo‘shilganda ko‘rsatkich va havolalar aniq bo‘ladi, ammo iterator yo‘q.

3. Dek va vektorning interfeyslari deyarli bir xil. Farqi shundaki, ba‘zi bir funksiyalash yaratilgan bo‘lishi mumkin. Masalan, dek sinfi capacity, reserve kabi funksiyalarga ega emas. Buning sababi dek vektorlarga o‘xshab ishlashini yaxshilash kerak emas.

4. Deklar bilan ishlaganda, ayniqsa ma‘lumotlarni qo‘shishda ko‘rsatkich va iteratorlarga bohliq dastur fragmenini yozishdan ehtiyot bo‘lish va mustaqil yozish kerakmas.

Umuman olganda, shaxsiy deklar bu vektorlardir, ular bilan ishlash ham vektorlar kabi amalga oshiriladi. Ammo qachon dekni, qachon vektorni tanlashni bilish kerak.

Ma‘lumotlar tuzilmasining boshiga va oxiriga qo‘shish uchun optimizatsiyalangan (yaxshilangan, ixcham qilingan) vektor – bu dekdir. Bunda qo‘shish aniq o‘zgarmas vaqtda bajariladi va nusxalash konstruktorini bir marta chaqiradi.

Agar element dekning o‘rtasiga qo‘shilsa, eng yomoni bu amal vaqt talab qiladi, chunki qo‘shish nuqtadan boshigacha va oxirigacha bo‘lgan masofalar orasida kichik aniqlash lozim.

insert, push_front i push_back funksiyalari dekdagi hamma iteratorlarni bekor qilishi mumkin. Shuningdek dekning o‘rtasiga ma‘lumot qo‘shilganda havolalarni xam bekor qiladi.

Berilgan navbatdagi elementlarni guruhlash masalasini qaraylik. Guruhlashni amalga oshirish uchun guruhlash shartini tanlaymiz va shu asosida navbat ichida guruhlaymiz, shartga tegishli bo‘lganlar navbat boshidan, bo‘lmaganlar oxiridan joylatiriladi.

1.8-dastur. Navbvtdagi elementlarni guruhlash.

```
// Created by MBBahodir
#include "stdafx.h"
#include <iostream>
#include <deque>
#include <algorithm>
using namespace std;

  bool mypred(const int x){
     return x <= 51;    // guruhlash uchun shart
  }
int main(){
    int Arr[]={1,78,89,23,51,49,100,18,50};
    deque<int> d(&Arr[0],&Arr[9]);
    cout << "Joriy deque: " << endl;
    for (deque<int>::iterator it=d.begin();it!=d.end();it++) std::cout<<*it<<" | ";
    cout << endl;

    stable_partition(d.begin(),d.end(),mypred);
```

```
    cout << "Natija: " << endl;
    for (deque<int>::iterator it=d.begin();it!=d.end();it++) std::cout<<*it<<" | ";
    cout << endl;

       system("pause");
       return 0;

 }
```

4.8-dastur.Output

Joriy deque:

1 | 78 | 89 | 23 | 51 | 49 | 100 | 18 | 50 | Natija:

1 | 23 | 51 | 49 | 18 | 50 | 78 | 89 | 100 |

1.7-dasturdagi fragment asosiy guruhlash sharti hisoblanadi.

```
bool mypred(const int x){
      return x <= 51;       // guruhlash uchun shart
  }
```

Bunda agar x <= 51 bo‘lsa true, aks holda flase qaytaradi. Bunga ekvivalent sifatida quyidagicha fragment ham yozish mumkin.

```
bool mypred(const int x){
    if (x <= 51) return true;          // guruhlash uchun shart
    return false;
}
```

Dek uchun keltirilgan materiallar dastlabki o‘rganuvchilar uchun yetarli deb hisoblaymiz. Masalaning shartiga qarab dek yoki vektorni tanlash dasturchining mahoratiga bog‘liq.

Dekning funksiyalariga quyidagilar kiradi:

1.1-jadval. Dekning funksiyalari

push_front	Boshidan yangi element qo‘shish
push_back	Oxiridan yangi element qo‘shish
pop_front	Birinchi elementni olish
pop_back	Oxirgi elementni olish
front	Birinchi element qiymatini ko‘rish
back	Oxirgi element qiymatini ko‘rish
size	Elementlar sonini ko‘rish
clear	Barcha elementlarni o‘chirish

Konteynerlar bilan ishlash algoritmlari. Barcha konteynerlar umumiy yondashuvi tufayli, amalda qiziqtirgan asosiy algoritmlar konteynerlarning har qanday turi uchun qo‘llanilishi mumkin bo‘lgan umumiy shaklda amalga oshirilishi mumkinligi tufayli STL konteynerlar uzoq amaliy foydalanish uchun bir arzigulik ixtiro bo‘ldi.

Algoritmlar kutubxonaning eng katta va ommabop qismidir. Juda ko‘p algoritmlar mavjudki, ularning hammasini batafsil bayon qilish uchun katta kitob kerak. Quyida ularni guruhlarga ajratib, ularning nomlari bilan ataymiz (ularning hammasi emas) va ulardan faqat ayrimlari misol tariqasida keltiramiz.

for_each()algoritmi. for_each() eng ko‘p ishlatiladigan algoritmdir. Konteyner elementlar guruhi (ehtimol, barcha elementlari) uchun ketma-ket harakatni amalga oshiradi. U har qanday STL konteynerda foydalanish mumkin. Quyidagi 1.9-dasturda for_each() algoritm massiv va vektor uchun ishlatilgan.

1.9-dastur. for_each() algoritmidan foydalanish.

```
// Created by MBBahodir
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
inline ostream& operator <<( ostream& out, const vector< unsigned > & obj ) { cout
    << "< ";
    for( auto& p: obj )
        cout << p << " ";
    return out << ">";
}
void pow2( unsigned& i ) { i *= i; } int

main( void ) {
    const int examples = 4;
    for( int i = 0; i < examples; i++ ) {
        unsigned ai[] = { 1, 2, 3, 4 , 5, 6, 7, 8, 9 }, ni =
            sizeof( ai ) / sizeof( ai[ 0 ] );
        vector< unsigned > vi( ai, ai + ni ); cout
        << vi;
        switch( i ) {
            case 0:
                for_each( vi.begin(), vi.end(), pow2 ); cout
                << " => " << vi << endl;
                break;
            case 1:
                for_each( ai, ai + ni, pow2 );
                cout << " => " << vector< unsigned >( ai, ai + ni ) << endl; break;
            case 2:
                for( auto& i : ai ) pow2( i );
                cout << " => " << vector< unsigned >( ai, ai + ni ) << endl; break;
```

```
            case 3:
                for_each( vi.begin() + 2, vi.end() - 2, pow2 ); cout
                << " => " << vi << endl;
                break;
        }
    }
    system("pause");
    return 0;
}
```

1.9-dastur.Output.

```
< 1 2 3 4 5 6 7 8 9 > => < 1 4 9 16 25 36 49 64 81 >
```

```
< 1 2 3 4 5 6 7 8 9 > => < 1 4 9 16 25 36 49 64 81 >

< 1 2 3 4 5 6 7 8 9 > => < 1 4 9 16 25 36 49 64 81 >
< 1 2 3 4 5 6 7 8 9 > => < 1 2 9 16 25 36 49 8 9 >
```

Dasturning birinchi global (auto &x : ...) takrorlanish fragmenti ishlashini ko‘rsatilgan (C++11 standarti tomonidan joriy qilingan). Bu kabi fragmentni massivlar va konteynerlarga ham qo‘llanilishi mumkin (bu vektorni oqimga chiqarish operatori variant funksiyasida ko‘rsatilgan). Bu takrorlanish fragmenti aslida STL kutubxona yoki algoritmlarni qismi emas, lekin u for_each() algoritm bilan bir xil ta‘sir ko‘rsatadi, ya‘ni to‘plam barcha elementlar uchun izchil qo‘llash mumkin.

Bu yuqoridagi dastur barcha algoritmlarni berilgan interval asosida tashkiliy asosiy mantiq qismini ko‘rsatadi (shart emas, barcha konteynerlar). Boshi va oxiri bilan cheklangan iteratorga (ko‘pincha, birinchi berilgan 2 ta parametrlar) funksiya, funktor va predikat (bu funksiyalarga - qaysidir elementlari bo‘yicha saralash imkoni bo‘lsa va mantiqiy qiymat qaytaradigan funksiyalar kiradi) navbat bilan qo‘llaniladi.

Find(). Navbatdagi algoritm – bu find(). Bu nomidan ma‘lumki, to‘plamdagi elementlarni qidirish funksyasidir. E‘tibor bersak, juda ko‘p konteynerlar bunday find() – funksiyasiga ega va u ob‘yekt uchun **obj.find(…) sietaksisida ishlatiladi. Shu vaqtda algoritm quyidagicha funksiyani find(obj:iteator, …)** ishga tushuradi. Aslida, bu bitta algoritm emas, balki ularning butun bir katta guruhi maveud, ular to‘plam elementlarini ba‘zi atribut, shart bilan tanlashlari asosida birlashtirilishi mumkin. Bu guruh predikatlariga **find()**, **find_if()**, **find_if_not()**, **find_first_of()**, **find_end()**, **adjacent_find()** funksiyalari kiradi. Shuningdek bu guruhda kengaytirilgan algoritmlar ham tegishlidir va bularga **count()**, **count_if()**, **search()**, **binary_search()**, **min()**, **max()**, **minmax_element()**, **min_element()**, **max_element()**, **equal() kabi bir qator algoritmlar kiradi.**

To‘plamga ishlov berish (aralashtirish) bir shartli guruh algoritmlar mavjud va ular joylarda elementlarni qayta tartibga soluvchi va qiymatlarni o‘zgartiruvchi algoritmlardir: **fill()**,

replace_copy(), **reverse()**, **rotate()**, **rotate_copy()**, **shuffle()**, **random_shuffle()**, **transform()**, **replace()**, **replace_if() va boshqa funksiyalar.**

Ikki to‘plamlar ustida bajariladigan algoritimlar guruhi ham mavjud. Bular asosan nusxalash, tarkibini ko‘chirib o‘tish (xar tipdagi konteynerlar uchun bo‘lishi mumkin. Masalan, **vector<>** ni **set<> ga**) funksiyalarini bajaradi. Bularga **copy()**, **copy_if()**, **move()**, **swap_ranges()**, **remove_copy()**, **remove_copy_if()**, **merge()**, **set_intersection()**, **set_difference() va boshqa algoritmlar.**

Nihoyat, juda maxsus guruh algoritmlari, to‘plam ichida elementlar turli saralash bilan bog‘liq funksiyalar **sort()**, **stable_sort()**, **is_sorted()**, **is_sorted_until()va boshqa algoritmlar.** Ushbu qiziqarli guruhni keyinroq, alohida batafsil ko‘rib chiqish kerak.

Kutubxonada vaqt o‘tishi bilan ortib borayotgan algoritmlarning ko‘pligi va ularning aksariyati adabiyotda hech qanday ta‘riflanmasligiga qaramasdan, tabiiy savol tug‘iladi: bu xilma-xillikni qanday tushunish kerak? Bu qiyinchiliklar qanday engio‘ mumkin:

- Barcha STL ob‘yektlar (konteynerlar, algoritmlar) shablon sintaktiki bo‘yicha tasvirlangan. Shuning uchun, ularning tavsiflari, ularning header fayllarida dastur fragmentlari sifatida tuzilgan bo‘lishi kerak.
- Kerakli algoritimlar uchun header fayllarni standart </usr/include/c++> katalogiga o‘ting va stl_algo* kabi sarlavha fayllarini toping - uda barcha algoritm funksiyasi prototiplarini (o‘xshashlarini) topasiz. Bundan tashqari, har bir prototipi algoritm maqsadini tushuntirib va parametrlarini tushuntirishning batafsil izoh oldindan yozib ko‘yilgan.
- Bir necha asosiy STL algoritmlarni foydalanishga doir misollar ko‘rib chiqing – kutubxonalar tizimida ularning ko‘pchiligi bor. O‘xshashlik bilan boshqa barcha algoritmlarni yaratish mumkin.

Shablon sinf kutubxonalar shablon jihatidan **template** termin bilan belgilangan, chunki o‘sha, kompilyatsiya sintaktiki xato bo‘yicha xabarlarni chiqaradi. Birinchisi yozma xabarlar fragmentlari bo‘yicha va ikkinchi xatolarni izlash uchun. Bu shablon kabi kuchli mexanizmi tomonidan qayyta ishlanidi (tekshiriladi) va dasturchi bu uchun tayyor bo‘lishi kerak.

Yuqorida aytib o‘tilganidek, misollarni o‘rganish juda ko‘p savollarga javob beradi, shuning uchun dastur keltiramiz. Endi o‘rgangan algoritmlarni diqqat bilan eslang (har birini sharhlab, kutubxona sinflaridan tashqarida bo‘lgan juda ko‘p STL algoritmlari mavjud, ammo ularning barchasi bir-biriga o‘xshab ishlaydi):

1.10-dastur. Standart algoritmlardan foydalanish.

```
#include "stdafx.h"
#include <iostream>
#include <cstring>
#include <vector>
#include <map>
#include <set>
#include <algorithm>
using namespace std;

inline ostream& operator <<( ostream& out, const vector<char>& obj ) { for(
    auto p: obj ) cout << p;
    return out;
}

int main( void ) {
    char s[] = "Muhammad al-Xorazmiy nomidagi Toshkent Axborot Texnologiyalari
Universiteti"; // nazvanie knigi
    vector<string> vs;
    vs.push_back("Muminov");
    vs.push_back("Bahodir");
    // copy & find :
    vector<char> v1( strlen( s ) ); copy( s,
    s + v1.size(), v1.begin() ); int nb = 0;
    for( auto is = find( v1.begin(), v1.end(), ' ' ); is != v1.end(); is = find(
                ++is, v1.end(), ' ' ) ) nb++;
    cout << "bo‘sh joylar soni: " << nb << " (" << nb + 1
         << " so‘z)" << endl;
    // min & max :
    auto mm = minmax_element( v1.begin(), v1.end() );
    cout << "belgilar uzunligi: '" << *mm.first << "' ... '"
         << *mm.second << "'" << endl;
    // fill & reverse & rotate & shuffle :
    vector<char> suv( vs[ 0 ].size() );
    copy( vs[ 0 ].begin(), vs[ 0 ].end(), suv.begin() ); cout <<
    suv << endl;
    random_shuffle( suv.begin(), suv.end() );
    cout << suv << endl;
```

```
    reverse( suv.begin(), suv.end() ); cout
    << suv << endl;
    rotate( suv.begin(), suv.begin() + suv.size() / 2, suv.end() ); cout <<
    suv << endl;
    // set_intersection & set_difference set<
    char > sus, pns;
    for( char s: vector<char>( vs[ 0 ].begin(), vs[ 0 ].end() ) ) sus.insert( s );
    for( char s: vector<char>( vs[ 1 ].begin(), vs[ 1 ].end() ) ) pns.insert( s );
```

```
    vector<char> outi( 100 ), outd( 100 );
    auto ret = set_intersection( sus.begin(), sus.end(), pns.begin(),
                                        pns.end(), outi.begin() ); cout
    << "umumiy belgi " << ( ret - outi.begin() ) << " : "
          << outi << endl;
    ret = set_difference( sus.begin(), sus.end(), pns.begin(),
                                    pns.end(), outd.begin() );
    cout << "unikal belgi " << ( ret - outd.begin() ) << " : "
          << outd << endl;
    system("pause");
    return 0;
}
```

1.10-dastur. Output

```
          bo‘sh  joylar  soni:  6  (7  so‘z)
          belgilar uzunligi: ' ' ... 'z'
          Muminov
          nuvmMoi
          ioMmvun
          mvunioM
          umumiy belgi 2 : io
unikal belgi 5 : Mmnuv
```

Dasturda char uchun konteynerlardan foydalanilgan. Deyarli barcha berilgan konteynerlar uchun turli xil algoritmlar amalga oshirilgan. (ixcham, ammo zerikarli fragmentlar).

Funktorlarning ko‘llanilishi. Funktor so‘zi C++ da “funksional ob‘yektlar” ning qisqartmasidan olingan. C++ da funksional ob‘yekt - operator() bilan aniqlangan sinfning nusxasi hisoblanadi. Agar C++ sinfi operator() aniqlasangiz, u funksiya kabi ishladi, ammo holatni saqlay olmaydi. Masalan, 1.11-dasturga qarang.

1.11-dastur. Operator()ni qayta ishlatish.

```
#include "stdafx.h"
#include <iostream>
#include <string>

using namespace std;
class SimpleFunctor {
    string name_;
public:
    SimpleFunctor(const char *name) : name_(name) {}
    void operator()() {   cout << "Salom, " << name_ << " ishlar zo‘rmi!" << endl; }
};

int main() {
    SimpleFunctor
    sf("Akbar"); sf();
        system("pause")
        ; return 0;
}
```

1.11-dastur. Output

```
Salom, Akbar ishlar zo‘rmi!
```

Dasturda sf ob‘yekt bo‘lishidan qa‘tiy nazar sf() funksiyasini chaqiramiz. Chunki SimpleFunctop sinfida operator() aniqlangan.

Ko‘p hollarda STL kutubxonasining taqqoslash va xatolik funksiyasining algoritmlarida funktorlar C++ da predikatlar sifatida ishlatiladi. Faraz qilaymiz, sonlar ketma ketligi berilgan. Bu ketma ketlikdagi barcha juft sonlar va toq sonlar yig‘indisini hisoblash kerak bo‘lsin. Bu masalani yechish uchun funktor va for_each algoritmidan foydalanish mumkin.

1.12-dastur. Funktor va for_each algoritmidan foydalanish.

```
#include "stdafx.h"
#include <algorithm>
#include <iostream>
#include <list> using
namespace std;
class EvenOddFunctor {
    int even_;
```

```
    int odd_;
public:
    EvenOddFunctor() : even_(0), odd_(0) {}
    void operator()(int x) {
        if (x%2 == 0) even_ += x;
        else odd_ += x;
    }
    int even_sum() const { return even_; } int
    odd_sum() const { return odd_; }
};

int main() {
    EvenOddFunctor evenodd;

    int my_list[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };
    evenodd =  for_each(my_list,
                    my_list+sizeof(my_list)/sizeof(my_list[0]),
                    evenodd);

     cout << " Juftlar yig‘indisi: " << evenodd.even_sum() << endl; cout
     << " Toqtlar yig‘indisi: " << evenodd.odd_sum() <<  endl;
        system("pause");
        return 0;
}
```

1.12-dastur. Output
Juftlar yig‘indisi: 30 Toqtlar yig‘indisi: 25

Dasturda for_each da EvenOddFunctor sinf ob‘yekti berilgan. for_each my_list bo‘yicha ketma ket o‘tganda funktor funksiyani chaqirmoqda. Har takrorlanishda evenodd funktor nusxasini qaytaradi. Evenodd funktor esa juft yoki toq elmentlarni yig‘indisini saqlaydi.

Standart MLda funktorlar. ML dagi funksiya interfeyslarning umumiy tadbiqlari mavjud va OYD jihatidan shakllantirish qiyin masallardan hisoblanadi. ML standart jihatidan funktorlar ML modul tizimining bir qismidir va ular tuzilmalarni yaratishga imkon beradi. Masalan, plagin tizimini yozish masalasini olaylik. Aytaylik barcha plaginlar soddaroq bo‘lishi uchun, faqat perform funksiyasini o‘z ichiga oladigan zarur interfeysini amalga oshirish kerak bo‘lsin. MLda avval plaginlar uchun maxsus tasdiqlovchi (imzolovchini)ni yaratish zarur.

```
Signature Plugin =
sig
   val perform: unit -> unit
end;
```

Plaginlar uchun tasdiqlovchi interfeysni aniqladik, ikkita plaginni, aytaylik, Loudplugin va Silentpluginni tuzilmalar orqali amalga oshirishimiz mumkin.

LoudPlugin ni amalga oshirish:

```
structure LoudPlugin :> Plugin =
struct
   fun perform() = print"KATTA HARFLAR!\n"
end;
```

Silentplugin ni amalga oshirish:

```
structure SilentPlugin :> Plugin =
struct
   fun perform () = print "kichik harlar\n"
end;
```

Endi funktorlarga yaqinlashamiz. ML dagi funktorlar tuzilmalarni argument sifatida qabul qilishadi, shuning uchun **Plugin** argument sifatida berilgan.

```
functor Performer(P : Plugin) =
struct
   fun job () = P.perform ()
end;
```

Bu funktor Plugin ni P argument sifatida qabul qiladi va P plaginni perform funksiyasini chaqiradigan job funksiyasi uchun ishlatadi.

Performer funktorni ishlatamiz (Funktor tuzilma qaytarishini bilamiz).

```
structure LoudPerformer = Performer(LoudPlugin);
structure SilentPerformer = Performer(SilentPlugin);

LoudPerformer.job ();
SilentPerformer.job ();
```

Standard ML uchun bk oddiy misol edi. Ammo chuqurroq o‘rganish uchun S tilini ham bilish talab qilinadi.

Haskellda funktorlar. Haskelldagi funktorlar haqiqiy funktorlar bo‘lishi bilan farqlanadi. Haskelldagi funktorlar kategoriya nazariyasidan matematik funktorlarga juda o‘xshaydi (Kategoriya

nazariyasi-matematikaning ob‘yektlarning ichki tuzilishiga bog‘liq bo‘lmagan matematik ob‘yektlar o‘rtasidagi munosabatlarning xususiyatlarini o‘rganadigan bo‘limi). Kategoriya nazariyasida funktor F kategoriyalar o‘rtasida shunday taqsimlanishi kerakki, kategoriya strukturasi saqlanib qolsin yoki boshqacha qilib aytganda, u ikki kategoriya orasidagi gomomorfizm bo‘lsin.

Bu ta‘rif Haskellda oddiy tipdagi sinf sifatida amalga oshiriladi.

```
class Functor f where
  fmap :: (a -> b) -> f a -> f b
```

ML ga qarab, masalan, Haskell tipi plagin sinfiga o‘xshaydi, joriy Sinfning nusxasi bo‘lish uchun qaysi amallarni qanday qilib yaratishni aniqlaydi. Bu holatda, birgina Functor tiplar bo‘yicha aniqlanmagan, balki f konstruktor asosida aniqlangan. Bundan shuni bilish mumkinki Functor – fmap funksiyasini yaratib beradi. fmap funksiyasi a tipini qabul qiladi va b tipini qaytaradi. fa (f konstruktori asosida qurilgan tip, a ga qo‘llaniladi) qabul qilinadigan tip va fb qaytariladigan tip.

Uni tushunish uchun, ba‘zi konreynerda har bir element uchun amal qiladigan funksiyasi sifatida fmap funksiyasini o‘ylab ko‘ring. Funktorlarning eng oddiy misoli muntazam ro‘yxatlar va map funksiyasi bo‘lib, u ro‘yxatdagi har bir element uchun funksiyani qo‘llaydi.

```
Prelude> map (+1) [1,2,3,4,5]
[2,3,4,5,6]
```

Bu oddiy misolda fmap funksiyasi faqat map va f tipidagi konstruktor [] - list tipidagi konstruktordir. Shuning uchun, ro‘yxatlar uchun, masalan, Functor sifatida belgilangan.

```
instance Functor [] where
  fmap = map
```

Bu map o‘rniga fmap yordamida fragmentni shakllantirib olsak, albatta, to‘g‘ri bo‘ldi.

```
Prelude> fmap (+1) [1,2,3,4,5]
[2,3,4,5,6]
```

Lekin funktor ta‘rifi tuzilishini asrab-avaylash haqida hech narsa demaydi, unutmang! Shuning uchun har qanday normal funktorlar matematik funksiyalar ta‘rifiga kiruvchi funktorlar qonunlarini qanoatlantirishi kerak. Fmapning ikki qoidalari bor:

```
fmap id = id
fmap (g. h) = fmap g. fmap h
```

Birinchi qoidada konteynerda har bir element uchun bir xil funksiyani xaritalash hech qanday ta‘siri yo‘q deb aytilgan. Ikkinchi qoidaga ko‘ra, konteynerdagi har bir element ustida ikkita funksiyaning tarkibi birinchi funksiyani ko‘rsatish bilan, so‘ngra ikkinchisini ko‘rsatish bilan bir xil bo‘ladi.

Ularni yaqqol ko‘rsatish uchun funktorlarga yana bir misol daraxtlar ustida amallar hisoblanadi. Bir konteyner sifatida daraxt, shajara (ierarxiya, ichma ich joylashgan ob‘yektlar) berilgan bo‘lsin. Daraxt tuzilishini saqlab qolish uchun,daraxt xususityalariga fmap funksiyasini qo‘llaymiz. Buning uchun birinchi daraxtni belgilab boshlaymiz.

```
data Tree a = Node (Tree a) (Tree a)
            | Leaf a
              deriving Show
```

Bu fragmentda daraxt turi ikki daraxtning (chap va o‘ng tugunlaruga) tuguni (node) yoki barg (leaf) ekanligi aytiladi. **deriving Show** ifodasi show funksiyasi orqali ko‘rish imkonini beradi. Endi Tree daraxti ustida funktorni aniqlay olamiz.

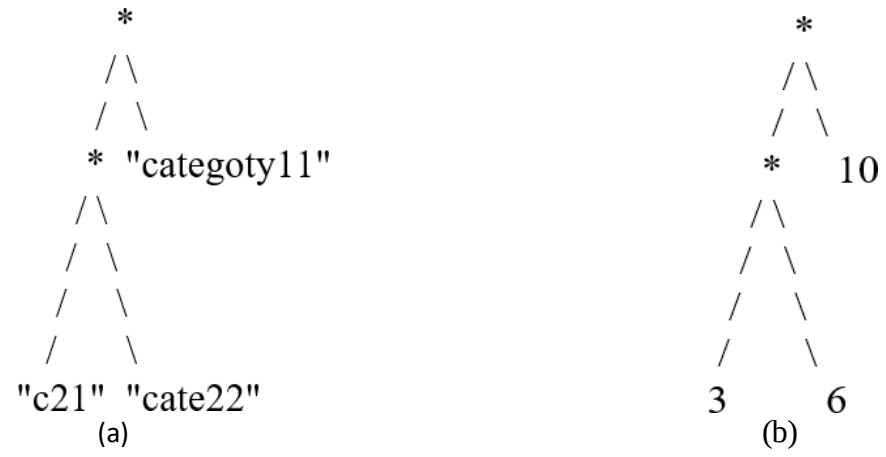
```
instance Functor Tree where fmap g (Leaf v) = Leaf (g v)
  fmap g (Node l r) = Node (fmap g l) (fmap g r)
```

Bu fragmentda fmap tipidagi g funksiyasi argumuent sifatida Leaf tipli v qiymatni qabul qiladi va Leaf tipidagi gni vga qiymat qilib beradi. Ikkinchi qoidada fmap tipiga oid g funksiyasi Node tipidagi l va r qiymatlarni qabul qiladi va natijada Node funksiyasi fmap tipidagi gl va gr argumentlarni qiymat qilib beradi.

Fmap funksiyasini tree ob‘yektlari bilan ishlash uchun tayyorlaymiz. Tree sinfni satrli (string) ro‘yxat bilan quramiz va har bir ro‘yxat uzunligini length()funksiyasi bilan aniqlaymiz.

```
Prelude> let tree = (Node (Node (Leaf “c21”) (Leaf “cate22”) (Leaf “categoty11”))
Prelude> fmap length tree
Node (Node (Leaf 3) (Leaf 6)) (Leaf 10)
```

Quyidagicha tree quriladi:



1.1-rasm. (a) tree ko‘rinishi, (b) length() bo‘yicha ko‘rinishi.

Aslida, Haskell da funktorlar fundamental funktor sinfi hisoblanadi va applicative functors va strelkalar barchasi bular ham funktorlaridir. Haskell funksiyalar murakkab hisoblanadi, agar bu funktorlarni haqiqiy xayot qonuniyatlari bag‘lab o‘rganmoqchi bo‘lsangi, alohida fan va manbalarni o‘rganish lozim.

Prolog funktorlari. Bu hammasidan oson funktorlardir. Ikkita misol qaraymiz. Birinchisi atom - bu tuzilmaning birinchisidir. Ifodani qaraymiz:

```
?- likes(olma, nok)
```

Birinchi atom funktordir – likes

Ikkinchisi- functor deb nomlangan ajralmas predmet. Bu argumentlari va funktor tuzilmasini qaytaradi. masalan,

```
?- functor(likes(olma, nok), Functor, Arity).
Functor = likes
Arity = 2
```

Bu Prolog funktorlarga misol edi.

Funktorlarni yaratish dasturchilarga xos emas albatta. Chunki bugungi kunda tayyor foydalanish uchun ko‘plab funktorlar yaratilgan. Ularni asosan instrument yaratuvchilar, ya‘ni dasturchilar uchun dastur tuzuvchilar yaratadi ap mantiqiy dasturlash funktorlarning o‘rni muhim ahamiyat kasb etadi.

Nazorat savollari

1. Adapterlar nima va qaysi sinflar kiradi?
2. Konteynerlarning adapterlari qaysi sinflar kiradi.
3. Juda keng va tez-tez ishlatiladigan ma‘lumotlar tuzilmalari bu – nima?
4. Konteyner adapterlarining funksiyalarini ta‘minlash uchun STL standart konteyneri har qanday moslashtirishda tayanch sifatida qaysi amallardan (adapter turiga qarab) foydalanish mumkin.
5. Birinchi navbvtda konteynerlarning adapterlari uchun qanday sinflarni dasturga qo‘shish lozim.
6. Stek nima va u nimasi bilan harakterlanadi.
7. Birinchi kirgan oxiri chiqadi tamoyili bo‘yicha faoliyat ko‘rsatuvchi ma‘lumotlar to‘plami bu nima.
8. Navbvt nima va u nimasi bilan harakterlanadi.
9. Birinchi kirgan Birinchi chiqadi tamoyili bo‘yicha faoliyat ko‘rsatuvchi ma‘lumotlar to‘plami bu nima.
10. Navbatning qaysi usuli vektor konteyner ustiga qurib bo‘lmaydi, lekin uni ro‘yxat

konteyner uchun qurish mumkin.

11. Deque sinfi uchun *front(), push_back(), pop_front*()), usullari har qanday konteynerlarda qachon ishlatiladi?

12. Deque sinfida kaysi usuli o‘rniga front() usuli ishlatiladi?

13. Ma‘lum bir tartibda qayta ishlanishi kerak ma‘lumotlarni uchun umumiy ma‘lumotlar tuzilishiga asoslangan to‘plami bu – qanday to‘plamlar?

14. Ma‘lumotlar tuzilishi - stek juda oddiy: u qanday tartibni belgilaydi? 15.Stekning assosiativ usullarini sanab bering?

16. Navbatda, birinchi elment kiritilgan bo‘lsa, u qachon chiqariladi?

17. Agar C++ da navbat shablonini ishlatmoqchi bo‘lsangiz, avval qanday kutubxonasini ulashingiz kerak.

18. empty() – navbat uchun nima vazifani bajaradi.

19. Massiv yordamida navbatlarni yaratish mumkinmi?, mumkin bo‘lsa izohlab bering.

20. Ustuvor navbat (Priority_queue) bilan navbat (queue) ob‘yektlarining farqi nimada.

21. Ustivor navbvtda yangi element qo‘shish uchun push() funksiyasi, birinchi o‘rindagi elementiga murojaat qilish uchun esa qaysi funksiyasi ishlatiladi.

22. Ikki tomonlama navbat va odiiy navbatning farqi nimada? 23.Dek va vektorning o‘xshash tomonlari va farqlari.

24. Ikki tomonlama navbatning avzalliklarini sanab bering.

25. Nima uchun ikki tomonlama navbat dusturchiga kerak. 3 ta asos keltiring. 26.Dekning maummolarini bilasizmi? Dek navbat qachon dasturchiga pand berishi mumkin.

27. Konteyner elementlar guruhi (ehtimol, barcha elementlari) uchun ketma ket harakatni amalga oshiradigan algoritmning aniqlang.

28. To‘plamdagi elementlarni qidirish funksyasi find()ga ta‘rif bering. 29.To‘plamga ishlov berish (aralashtirish)uchun bir shartli guruh algoritmlar mavjud va ular joylarda elementlarni qayta tartibga soluvchi va qiymatlarni o‘zgartiruvchi algoritmlardir. Ularni sanab bering.

30. Ikki to‘plamlar ustida bajariladigan algoritimlar guruhi qaysi algoritmlar misol bo‘la oladi.

31. Funktor so‘zi C++ da qaysi so‘zning qisqartmasidan olingan.

32. C++ da funksional ob‘yekt – qaysi operator bilan aniqlangan sinfning nusxasi hisoblanadi.

33. Ko‘p hollarda STL kutubxonasining taqqoslash va xatolik funksiyasining algoritmlarida funktorlar C++ da nimalar sifatida ishlatiladi.

34. ML standart jihatidan funktorlar ML modul tizimining bir qismidir va ular nimalarini

yaratishga imkon beradi.

35. Haskelldagi funktorlar kategoriya nazariyasidan qaysi funktorlarga juda o‘xshaydi.

7- MA’RUZA
MAVZU. STANDART ALGORITMLAR VA ITERATORLAR. (2 SOAT)

Reja:

1. Standart algoritmlar.
2. Iteratorlar va ularning qo‘llanilishi.
3. Xotirani taqsimlovchilar va ularga qo‘yilgan talablar.
4. Standart bo‘yicha taqsimlovchi.

Annotatsiya:

C++ tilining standart algoritmlar kutubxonasi, iteratorlar, xotirani taqsimlash talablari, dinamik xotira ajratish va foydalanishga asosida qidirish, saralash, taqqoslash funksiyalari, dasturlashda o‘zlarini tutishlari, iteratorlar, o‘zgarmas iteratorlar, teskari iteratorlar bilan ishlashga mo‘ljallangan asosiy funksiyalar va ularning dasturlashdagi ahamiyati, xotirani taqsimlovchi sniflarni yaratish va talablari, turli tipli massivlar uchun dinamik xotirani ajratish va foydalanish operatorlar uslublari, amallari, talablari, vazifalari va usullari keltirilgan bo‘lib, nazariy bilimlarni asoslash uchun dasturlar tuzib ko‘rsatilgan. Bilimlarni mustahkamlash uchun 30 ta nazariy savol va amaliy ko‘nikma va malakalarni rivojlantrish uchun 3 ta assisment topshirig‘i va har assismentda 10(9,7) ta topshiriq, jami 26ta topshiriq berilgan. Bu topshiriqlarni bajarish mavzuni mustahkamlash uchun xizmat qiladi.

Kalit so‘zlar. a*lgoritm, kutubxona, iterator, o‘zgarmas iterator, teskari iterator, qidirish, saralash, o‘rin almashtirish, shablon, sinf, find(), sort(), swap(), begin(), end(), rbegin(), rend(), crbegin(), crend(), xotira, pul, xotira bloki, new, delete, for_each(), funksional ob‘yektlar.*

KIRISH

Bugungi kunda dasturlash juda soddalashib bormoqda, ba‘zi bir mutaxassislar dasturlash uchun matematika kerakmas deb aytsa, ba‘zilari matematika dasturlashning asosi deb aytadi. Shuning negizida nima yotibdi. Albatta algoritmlar, isteratorlar va xotirani taqsimlovchilar va ularning talablari yotadi. Algoritmlar tayyo bo‘lsa, faqat bu algoritmlarini joy joyiga qo‘yish bu juda osondir. Ammo algoritmlarni yaratish va uni dasturlash joriy qilish o‘ta mushkul va mashaqqatli masaladir. Shuning uchun bugungi kunda juda ko‘plab algoritmlarni saqlovchi standart va no standart kutubxonalar mavjud.

Standart algoritmlar. Yangi dasturlashni boshlanuvchilar odatda saralash, qidirish yoki qator elementlarini sanash kabi nisbatan oddiy vazifalarni bajarish uchun maxsus takrorlanishga asoslangan

usullarni yozish uchun juda ko‘p vaqt sarflashadi. Bu usullar ularda xato qilish qanchalik osonligi jihatidan ham, umumiy ishonchlilik va mavjudlik nuqtai nazaridan ham muammoli bo‘lishi mumkin, chunki bu usullar tushunish qiyin bo‘lishi mumkin.

Qidiruv, sanash va saralash dasturlashda juda keng tarqalgan amallar bo‘lganligi sababli, C++ standart kutubxonasi dastlab bu vazifalarni bir necha qator kodda bajaradigan katta funksiyalar to‘plamini o‘z ichiga olgan. Bundan tashqari, bu xususiyatlar oldindan sinovdan o‘tgan, samarali va turli xil konteynerlarni qo‘llab-quvvatlaydi. Bu xususiyatlarni ham paralel qo‘llab – quvvatlash tezroq uni bajarish uchun bir xil masala uchun bir necha MP (markaziy protsessor) oqimlarini ajratish qobiliyati ega.

Algoritm kutubxonasi tomonidan taqdim etiladigan funksiyalar odatda uch toifadan biriga kiradi:

Inspektorlar (nazoratchilar) - konteynerdagi ma‘lumotlarni (masalan, qidirish yoki elementlarni soni hisoblash amallari) ko‘rish uchun ishlatiladi.

Mutatorlar - konteynerdagi ma‘lumotlarni o‘zgartirish uchun ishlatiladi (masalan, saralash yoki elementlarni qayta tartibga solish amallari).

Fasilitatorlar (koordinatorlar) - elementlarining qiymatlari asosida natija hosil qilish uchun ishlatiladi (masalan, qiymatlarni ko‘paytiruvchi ob‘yektlar yoki elementlarning qaysi tartib juftliklarida tartiblanishi kerakligini aniqlovchi ob‘yektlar).

Bu algoritmlar algoritm kutubxonasida (<algorithm >sarlavha faylida) joylashgan. Oldingi biladigan va eng ko‘p tarqalgan algoritmlardan ba‘zilarini imkoniyatlarini chuquroq ko‘rib chiqamiz va bu algoritmlarning barchasi iteratorlardan foydalanadi.

find() algoritmi – berilgan qiymat bo‘yicha elementlarni qidirish algoritmi. Berilgan qiymatni birinchi topgunicha ishlaydi. Argument sifatida 3 ta paramertni oladi.

ketma-ketlikda boshlang‘ich element uchun iterator; ketma-ketlikda oxirgi element uchun iterator; qidirishning qiymati.

Qidirish natijada qidirilayotgan qiymatli elementga (agar u topilsa) yoki iteratorning oxiriga (agar bunday element topilmasa) ishora qiluvchi iterator qaytadi.

1.1-dastur. Find () funksiyasidan foydalanish.

```
// Created by MBBahodir
#include "stdafx.h"
#include <algorithm>
#include <array>
#include <iostream>
using namespace std;
int main(){
	array<int,10> arr = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19 };

	cout << "Qidirish va o‘zgartirish uchun son kirit: ";
	int search;int replace; cin >> search >> replace;
//	array<int,10>::iterator found;
auto found = find(arr.begin(), arr.end(), search); if (found == arr.end())
	{cout << search << " topilmadi. " << endl;}
	else{*found = replace;}
	for (int i : arr){cout << i << ' ';}
cout << endl; system("pause");
 return 0;
}
```

1.1-dastur.Output
Birinchi sinov
Qidirish va o‘zgartirish uchun son kirit: 19 21 1 3 5 7 9 11 13 15 17 21
Ikkinchi sinov
Qidirish va o‘zgartirish uchun son kirit: 10 25

10 topilmadi. 1 3 5 7 9 11 13 15 17 19

find_if() algoritmi - shart orqali elementni izlash. Juda ko‘p hollarda konteynerda mavjud elementlar orasida ma‘lum bir qiymat saqlaydigan element bormi degan masala duch kelamiz. Masalan, konteynerda satrli elementlar berilgan bo‘lsa, biror bir satr qismi bormikan degan masala bo‘lsin. Mana shunday masalalarni yechishda find_if() algoritmi eng mos va qulay hisoblanadi. Bu algoritm find() algoritmi bilan analog sifatida ishlaydi ammo, qidirish uchun qiymat o‘rniga funksiya ko‘rsatgichlariga moslangan element qiymatida o‘xshashlik bor bo‘lgan ob‘yekt chaqiriladi.

find_if() funksiyasi buning uchun har bir elementni chaqiradi va to shunday elementni topilmaguncha yoki konteynerda elementlar tugaguncha.

Ma‘lum bir berilgan satr bo‘yicha, berilgan konteynerda shu satrni o‘zida qism satr qilib saqlovchi elemeni bor ki yo‘qligini aniqlovchi dasturni tuzish kerak bo‘lsin.

5.2-dastur. find_if() funksiyasidan foydalanish.

```
// Created by MBBahodir
#include "stdafx.h"
#include <algorithm>
#include <array>
#include <iostream>
#include <string>

using namespace std;

bool containsNut(string str)
{return (str.find("oro") != string::npos);}

int main()
{array<string, 4> arr = { "Toshkent", "Buxoro", "Samarqand", "Navoiy" };
	auto found = find_if(arr.begin(), arr.end(), containsNut);

	if (found == arr.end())
	{
		cout << " Topilmadi" << endl;
	}
	else
	{
		cout << " Topildi: " << *found << endl;
	}
	system("pause");
	return 0;

}
```

Agar odatdagi standart tarzdagidek muammoni hal qilshganimizda, kamida ikki takrorlanish kerak edi (birinchi takrorlanish elementlarni saralash uchun va ikkinchisi satr qismini solishtirish uchun). Standart C++ kutubxonasining vazifalari bir nechta fragment satrlarida ham shunday qilish imkonini beradi.

count() va count_if() algoritmlari – konteynerdan berilgan elementlarini sanash vazifasini bajaradi. Berilgan mezon (shart) bo‘yicha barcha elementlarni tekshirib, sanaydi. Berilgan konteynerda nechta element berilgan satr qismi borligini sanash dasturini keltiramiz.

1.3-dastur. count() va count_if() algoritmlaridan foydalanish.

```
// Created by MBBahodir
#include "stdafx.h"
#include <algorithm>
#include <array>
 #include <iostream>
#include <string>
#include  <vector>
using namespace std;

bool containsNut(string str)
{
        return (str.find("daryo") != string::npos);
}
bool counter(string str){
        return true;
}
int main()
{
        array<string, 6> arr = { "Sirdaryo", "Amudaryo", "Qashadaryo", "Suxondaryo",
"Zarafshon", "Qashadaryo" };

        auto countWithIf = count_if(arr.begin(), arr.end(), containsNut); int
        countofArr = count(arr.begin(), arr.end(), "Qashadaryo");
        cout <<  "daryo satr qismi borlarining soni: " << countWithIf << endl; cout
      <<  " elementlar soni: " << countofArr << endl;

        system("pause
```

```
        "); return 0;
}
```

1.3-dastur. Output

```
daryo satr qismi borlarining soni: 5
 elementlar soni: 2
```

sort() algoritmi – foydalanuvchining saralash funksiyasini qurish uchun ishlatiladi. Oldinlari sort() funksiyasi bilan massivlarni saralagan bo‘lshingiz mumkin, ammo sort() funksiya bu bilan chegaralanib qolmaydi. Bu funksiya yordamchi bir funksiya oladi va bu argument asosida foydalanuvchi o‘zining saralash uslubini amalga oshirishi mumkin. Funksiya esa taqqoslash uchun paramert ikkita parametr oladi. Agar birinchi element ikkinchisidan oldin bo‘lishi kerak bo‘lsa, ikkinchi paramert true qiymat qaytarishi kerak. Odatda sort() funksiyasi o‘sish tartibida saralaydi. Berilgan massivni kamayish tartibida saralash dasturini tuzaylik.

1.4-dastur. sort() funksiyasidan foydalanish.

```
#include "stdafx.h"
#include <algorithm>
#include <array>
#include <iostream>

using namespace std;
bool greater(int a, int b)
{
   return (a > b);
}
bool d_greater(int a, int b)
{
   return (b > a);
}

int main()
{array<int,5> arr = { 28, 18, 16, 25, 25 }, arr_one;
     arr_one = arr; sort(arr.begin(), arr.end(), greater);
    sort(arr_one.begin(), arr_one.end(),d_greater );

    for (int i : arr) {cout << i << ' ';}

    cout << endl;
```

```
    for (int i : arr_one ) { cout << i << ' ';
    } cout << endl;
  system("pause"); return
  0;

}
```

1.4-dastur. Output

```
28 25 25 18 16
16 18 25 25 28
```

Dasturdan ko‘rinadiki, o‘zimiz saralash algoritmini boshidan yozishimiz shart emas ekan, buning uchun bir qator fragment yetarli bo‘ldi. Ammo C++ tili, juda ko‘p saralashning kamayish tartibda foydalanganligi uchun greater{} tipli funksiyasi bo‘lishi mumkin (<functional> kutubxonasida bor). Biz o‘zimizning greater funksiyasidan foydalanish uchun sort(arr.begin(), arr.end(), greater); dastur fragmentini yozdik, ammo kutubxonanikini chaqirish uchun sort(arr.begin(), arr.end(), std::greater{}); kabi fragment yozishimiz kerak. E‘tibor bering, greater{} funksiyasi figurali qavsga muxtoj, shuning uchun uni funksiya deb atalmaydi, ma‘lumot tipi deb yuritiladi va undan foydalanish uchun nusxasini yaratishimiz kerak. Figurali qavslar anonim ob‘yektlarni yaratish uchun xizmat qiladi.

for_each() algortmi – konteynerning barcha elementlari o‘tish. Bu for_each() funksiyasi kiruvchi ma‘lumot sifatida to‘plamni qabul qiladi va foydalanuvchining funksiyasini joriy to‘plamdagi har bir element uchun qabul qiladi. Buning qulaylik tomoni shundaki, agar bir amalni to‘plamning barcha elementlari bo‘yicha bajarish kerak bo‘lsa, for_each() funksiyasi as qotadi. Berilgan massivning barcha elementlarini chiqarish uchun for_each() funksiyasidan foydalanishni dasturini keltiraylik.

1.5-dastur. for_each() funksiyasidan foydalanish.

```
#include "stdafx.h"

#include <algorithm>
#include <array>
#include <iostream>
using namespace std;
```

```
void doubleNumber(int &i){ i = i;
}
int main(){
        array<int,4> arr = { 1, 2, 3, 4 };
    for_each(arr.begin(), arr.end(), doubleNumber);
   for (int i : arr){ cout << i << ' ';
   }
    cout << endl; system("pause"); return 0;
}
```

Yosh dasturchilar uchun bu takrorlanish jarayonini oddiy for operatoriga aniq parametrlarni qo‘yib ishlatish maquldir. for_each() funksiyasining yutuqlari shandan iboratki, takrorlanish tanasidan qayta foydalanish va paralellashtirish, katta proektlarda katta ma‘lumotlar, turli tiplari har xil ma‘lumotlar tuzilmasi bilan ishlgandan eng yaxshi isntrument bo‘lib xizmat qiladi.

Algoritmlarning bajarilish tartibi . Algoritm kutubxonasida eng ko‘p algoritmlar ma‘lum bir ijro tartibini kafolatlamaydi. Bunday algoritmlar uchun har qanday belgilangan funksiyalar bajarilish tartibini o‘z zimmasiga olmasligiga ishonch hosil qilish kerak, chunki bu funksiyalarni chaqirish tartibi ishlatiladigan kompilyatorga qarab farq qilishi mumkin. Quyidagi algoritmlar aniq bajariladi:

```
for_each()
  copy()
  copy_backward()
  move()
  move_backward()
```

Agar parametr sifatida biror funksiya berilmagan bo‘lsa, aniqlanmagan holat o‘tadi. Buning uchun har bir algoritmga arr.begin() va arr.end() aniq berilishi kerak. Ammo C++20 standartida faqat massiv berilsa o‘zi aniqlanadi.

Algoritm kutubxonasining algoritmlari ro‘yxati:

1	**O‘zgartirilmaydigan ketma-ketlik amallar:**	
1.1	all_of	to‘plamdagi barcha elementlar uchun holat shartini tekshirish
1.2	any_of	to‘plamdagi har qanday element uchun holat shartini bajarishini tekshirish
1.3	none_of	to‘plamdagi hech bir element uchun holat shartga javob bermasligini tekshirish

1.4	for_each	funksiyani to‘plamga qo‘llash
1.5	find	to‘plamda qiymatini topish
1.6	find_if	to‘plamda shartni bajaruvchi elementni topish
1.7	find_if_not	to‘plamda shartni bajarmaydigan elementni topish
1.8	find_end	to‘plamdagi oxirgi elementni topish
1.9	find_first_of	to‘plamdagi elementlardan birinchisini topish
1.10	adjacent_find	to‘plamda teng qo‘shni elementlarni topish
1.11	count	to‘plamdagi elementlar sonini hisoblash
1.12	count_if	shartga javob beradigan to‘plamdagi elementlar sonini qaytaradi
1.13	mismatch	ikki farqli intervallarni birinchi o‘rnini qaytaradi
1.14	equal	to‘plamdagi Ikki elementlar tengligini tekshirish
1.15	is_permutation	to‘plamda almashtirish bor yoki yo‘qligini tekshirish
1.16	search	ketmk-ketlikdan qidirish
1.17	search_n	intervaldan qidirish
2	**Ketma-ketlik amallarini o‘zgartirish:**	
2.1	copy	to‘plam elementlardan nusxa olish
2.2.	copy_n	elementlarni ko‘paytirish
2.3	copy_if	shart asosida diapazon elementlarini nusxalash
2.4	copy_backward	elementlar to‘plamidan Orqaga nusxa ko‘chirish
2.5	move	elementlar qatorini ko‘chirish
2.6	move_backward	elementlar to‘plamni orqaga ko‘chirish
2.7	swap	ikki ob‘yektning qiymatlarini almashtirish
2.8	swap_ranges	ikki intervalning qiymatlarini almashtirish
2.9	iter_swap	ikki iteratorlarning ko‘rsatikich ob‘yektlari qiymatlarini almashish
2.10	transform	to‘plam ob‘yektlarini aylantirish
2.11	replace	to‘plamning qiymat almashtirish
2.12	replace_if	Qiymatlarni shart asosida to‘plamda almashtirish
2.13	replace_copy	almashtirish qiymatlari to‘plamni nusxalash
2.14	replace_copy_if	almashtirish qiymatlari to‘plamni shart asosida nusxalash

2.15	fill	to‘plamni Qiymat bilan to‘ldirish
2.16	fill_n	ketma-ketligini Qiymat bilan to‘ldirish
2.17	generate	funksiya bilan to‘plam uchun qiymatlarni yaratish
2.18	generate_n	funksiya bilan ketma-ketlik uchun qiymatlarni hosil qilish
2.19	remove	to‘plam qiymatini o‘chirish
2.20	remove_if	to‘plam ma‘lumotlar shart asisida o‘uchirish
2.21	remove_copy	qiymatni o‘chirish diapazonini nusxalash
2.22	remove_copy_if	qiymatni o‘chirish diapazonini shart asosida nusxalash
2.23	unique	to‘plamdagi takrorlangan elementlarni o‘chirish
2.24	unique_copy	to‘plamdagi takrorlangan elementlarni nusxalash
2.25	reverse	teskari to‘plam
2.26	reverse_copy	teskari to‘plamni nusxalash
2.27	rotate	to‘plamda elementlarning chapga aylantirish
2.28	rotate_copy	to‘plamda elementlarning chapga aylantirishni nusxalash
2.29	random_shuffle	to‘plamdagi elementlarning tasodifiy qo‘yish
2.30	shuffle	to‘plamdagi elementlarning generator yordamida tasodifiy qo‘yish
3	**Bo‘laklar (qismlar) bilan amallari**	
3.1	is_partitioned	to‘plamning bo‘linganligini tekshirish
3.2	partition	ikkiga bo‘lish orlig‘i
3.3	stable_partition	qonuniyat asosida Ikkiga bo‘lish orlig‘i
3.4	partition_copy	bir to‘plamni ikkiga bo‘lish
3.5	partition_point	bo‘lish nuqtasini olish
4	**Saralash amallari:**	
4.1	sort	to‘plam ma‘lumotlar saralash
4.2	stable_sort	ekvivalentlar tartibini saqlab qolgan holda elementlarni saralash
4.3	partial_sort	to‘plamda elementlarni qisman saralash

4.4	partial_sort_copy	to‘plamda elementlarni qisman saralash va nusxalash
4.5	is_sorted	to‘plamning tartiblanganligini tekshirish
4.6	is_sorted_until	to‘plamdagi ajratilmagan elementni topish
4.7	nth_element	to‘plamdagi elementni saralash
5	**Binar qidiruv (bo‘lingan / tartiblangan intervallar bilan ishlash):**	
5.1	lower_bound	iteratorni quyi chegarasini qaytarish
5.2	upper_bound	iteratorni yuqori chegarasini qaytarishga qaytarish
5.3	equal_range	teng elementlardan to‘plam osti olish
5.4	binary_search	qiymatning tartiblangan ketma-ketlikda mavjudligini tekshiring
6	**Birlashtirish (tartiblangan intervallar bilan ishlash):**	
6.1	merge	tartiblangan to‘plamlarni birlashtirish
6.2	inplace_merge	ketma-ket tartiblangan intervallarni birlashtirish
6.3	includes	tartiblangan to‘plamlarni boshqa tartiblangan to‘plamni o‘z ichiga olishi yoki olmasligini tekshirish
6.4	set_union	ikkita tartiblangan to‘plamni birlashtirish
6.5	set_intersection	ikki tartiblangan intervallarning kesishishi
6.6	set_difference	ikki tartiblangan intervallar orasidagi farq
6.7	set_symmetric_difference	ikki tartiblangan intervallar orasidagi simmetrik farq
7	**Elementlar yig‘indisi amallari:**	
7.1	push_heap	elementlar yig‘indisini qatoriga surish
7.2	pop_heap	elementlar yig‘indisidan elementi o‘chirish
7.3	make_heap	elementlar yig‘indisini yasash
7.4	sort_heap	elementlar yig‘indisini tartiblash
7.5	is_heap	elementlar yig‘indisi ni tekshirish
7.6	is_heap_until	elementlar yig‘indisidan birinchisini topish
8	**Kichik va katta amallari:**	
8.1	min	eng kichik elementni qaytaradi

8.2	max	eng katta elementni qaytaradi
8.3	minmax	eng kichik va eng katta elementni qaytaradi
8.4	min_ element	to‘plamdagi eng kichik elementni qaytaradi
8.5	max_element	to‘plamdagi eng katta elementni qaytaradi
8.6	minmax_element	to‘plamdagi eng kichik va eng katta elementni qaytaradi
9	**Boshqa yordamchi amallar:**	
9.1	lexicographical_compare	leksikografik jihatdan qiyoslash
9.2	next_permutation	to‘plam aylantirish uchun keyingi ideks
9.3	prev_permutation	to‘plam aylantirish uchun oldingi ideks

Bu yerda 85 ta algoritmlar keltirilgan, ammo bularning hammasi shablon funksiya bo‘lganligi uchun C++ dasturlash versiyalarida farqi bo‘lishi mumkin. Shuningdek, eslab ko‘ring, bu algoritmlarning ko‘plaridan oldin mavzularimizda foydalanganmiz.

Nazorat savollari

1. Berilgan qiymat bo‘yicha elementlarni qidirish algoritmi qaysi va nechta parametr qabul qiladi?
2. Konteynerda mavjud elementlar orasida ma‘lum bir qiymat saqlaydigan element qidirish uchun qaysi funksiya ishlatiladi?
3. Konteynerdan berilgan elementlarini sanash vazifasini bajaruvchi algoritmlarni sanab bering?
4. Funksiya taqqoslash uchun paramert sifatida ikkita parametr oladi. Bu qaysi funksiya.
5. Ikki ob‘yektning qiymatlarini almashtirish algoritmni ayting.
6. Tartiblangan to‘plamlarni boshqa tartiblangan to‘plamni o‘z ichiga olishi yoki olmasligini tekshirish algoritmini ayting.
7. Iteratorlar nima uchun foydalaniladi.
8. **begin()** va **end()** funksiyalaridan nima maqsadda foydalaniladi.
9. Agar **begin()** va **end()** funksiyalari o‘zaro teng bo‘lmasa, ularning orasidan kamida nechta element bor.
10. Keyingi elementga murojjat qilish uchun iteratorni harakatlantirish amalni ayting.
11. Agar konteynerda o‘zgarmas qiymatli elementlar bo‘lsa, bu holda konteyner elementlariga murojaat qilish uchun nimadan

foydalanishsh kerak.

12. cbegin() va cend()funksiyalaridan nima saqsadda foydalaniladi.
13. Agar konteynerda o‘zgaruvchi qiymatli elementlar bo‘lsa, bu holda konteyner elementlariga murojaat qilish o‘zgarmas iteratorlardan foydalanishsh mumkinmi.

8- MA’RUZA

MAVZU. ITERATORLAR VA ULARNING QO`LLANILISHI. XOTIRANI TAQSIMLOVCHILAR VA ULARGA QO`YILGAN TALABLAR. (2 SOAT)

Reja:

1. **Iteratorlar va ularning qo‘llanilishi.**
2. **Xotirani taqsimlovchilar va ularga qo‘yilgan talablar.**
3. **Standart bo‘yicha taqsimlovchi.**

Annotatsiya:

C++ tilining iteratorlar, xotirani taqsimlash talablari, dinamik xotira ajratish va foydalanishga asosida qidirish, saralash, taqqoslash funksiyalari, dasturlashda o‘zlarini tutishlari, iteratorlar, o‘zgarmas iteratorlar, teskari iteratorlar bilan ishlashga mo‘ljallangan asosiy funksiyalar va ularning dasturlashdagi ahamiyati, xotirani taqsimlovchi sniflarni yaratish va talablari, turli tipli massivlar uchun dinamik xotirani ajratish va foydalanish operatorlar uslublari, amallari, talablari, vazifalari va usullari keltirilgan bo‘lib, nazariy bilimlarni asoslash uchun dasturlar tuzib ko‘rsatilgan. Bilimlarni mustahkamlash uchun 30 ta nazariy savol va amaliy ko‘nikma va malakalarni rivojlantrish uchun 3 ta assisment topshirig‘i va har assismentda 10(9,7) ta topshiriq, jami 26ta topshiriq berilgan. Bu topshiriqlarni bajarish mavzuni mustahkamlash uchun xizmat qiladi.

Kalit so‘zlar. K*utubxona, iterator, o‘zgarmas iterator, teskari iterator, qidirish, saralash, o‘rin almashtirish, shablon, sinf, find(), sort(), swap(), begin(), end(), rbegin(), rend(), crbegin(), crend(), xotira, pul, xotira bloki, new, delete, for_each(), funksional ob‘yektlar.*

Iteratorlar va ularning qo‘llanilishi. Iteratorlar konteynerlarning elementlariga murojaat qilish uchun foydalaniladi. Iteratorlar bilan elementlar bilan ishlash juda qulay hisoblanadi. Iterator iterator tipi bilan yoziladi. Har qanday konteyner uchun iteratorlarning tiplari farq qiladi. Masalan, list<int> tipidagi konteyner uchun **list<int>::iterator tipi,** vector<int> tipidagi konteyner uchun esa **vector<int>::iterator** tipi ishlatiladi. C++ tiladi konteynerlardan iteratorlarni ajratib olish uchun **begin()** va **end()** funksiyalaridan foydalaniladi. **begin()** funksiyasi konteynerdagi birinchi element ko‘rsatuvchi iteratorni qaytaradi (agar konteynerda elementlar bo‘lsa). **end()** funksiyasi konteynerdagi oxirgi elementdan keyingi pozitsiyani ko‘rsatuvchi iteratorni (ya‘ni konteyner oxirini) qaytaradi. Agar konteyner bo‘sh bo‘lsa, **begin()** va **end()** funksiyalari bir xil qiymat qaytaradi. Agar **begin()** va **end()** funksiyalari o‘zaro teng bo‘lmasa,

larning orasidan kamida bitta element bor. Bu funksiyalar aniq tipli konteyner uchun iterator qaytaradi. Iterator yaratishga misol:

```
std::vector<int> v = { 1,2,3,4 };
std::vector<int>::iterator iter = v.begin();
```

Bu misolda tipiga mansub bo‘lgan vektor tipidpgi konteyner - vektor yaratilgan. Konteyner doimiy qiymatlar bilan to‘ldirilgan. **begin()** funksiyasi (usuli) bilan vektor elementini olish uchun iterator keltirilgan. Bu iterator vektor konteynerning birinchi elementini ko‘rsatadi.

Iterator amallari:

- ***iter** – iterator ko‘rsatadigann elementni olish;
- **++iter** - keyingi elementga murojjat qilish uchun iteratorni harakatlantirish
- **--iter** - oldingi elementga murojjat qilish uchun iteratorni harakatlantirish.

forward_list konteyner iteratorlari dekrement amalini qo‘llab quvvatlamaydi.

- **iter1 == iter2** - ikki iterator teng, agar ular bir xil iteratorni aniqlagan bo‘lsa.
- **iter1 != iter2** ikki iterator teng emas, agar ular bir xil iteratorni aniqlagan bo‘lsa.

```
#include "stdafx.h"
#include<iostream>
#include <vector>
 using  namespace  std;
int            main(){
vector<int> myvector;
       for (int i = 0; i < 15; i++)
       {myvector.push_back(rand() % 100);}
     //  vector<int>::iterator iter = myvector.begin();
       auto iter = myvector.begin();
       while(iter!=myvector.end()) {cout << *iter << " | ";

        ++iter;}
       cout << endl;
       system("pause");

    return 0;
```

1.1-dastur. Iterator amallaridan foydalanish.

Konteynerlar bilan ishlaganda, konteynerdagi elementlarni qo‘shish yoki o‘chirish ushbu konteyner uchun barcha joriy iteratorlarni, shuningdek, uning elementlariga havola va ko‘rsatgichlarni bekor qilishiga olib kelishi mumkin. Iteratorlar nafaqat elementlarni olish, balki ularni o‘zgartirish imkonini beradi.

1.2-dastur. Iterator yordamida konteynerni elementlari qiymatini o‘zgartirish.

```
#include "stdafx.h"
#include <iostream>
#include <vector>
using namespace std;
int main()
{vector<int> myvector;
        for (int i = 0; i < 15; i++)
        {myvector.push_back(rand() % 10);}
         vector<int>::iterator iter = myvector.begin();
        //auto iter = myvektor.begin();
        while(iter!=myvector.end())
    {*iter = (*iter) * (*iter);++iter;}

        for(iter = myvector.begin(); iter!=myvector.end(); ++iter)
    {cout << *iter << " | ";}
        cout  <<  endl;
    system("pause");
    return 0;
}
```

1.2-dastur. Output
1 | 49 | 16 | 0 | 81 | 16 | 64 | 64 | 4 | 16 | 25 | 25 | 1 | 49 | 1 |

Dasturda while takrorlanish operatori konteynerning elementlarini iteratorga olib, o‘zini o‘ziga ko‘paytirib, yana shu iteratorga yozadi. Shuning uchun ekranga konteynerdagi sonlarning kvadratlari chiqadi.

O‘zgarmas iteratorlar. Agar konteynerda o‘zgarmas qiymatli elementlar bo‘lsa, bu holda konteyner elementlariga murojaat qilish uchun o‘zgarmas iteratorlardan foydalanishsh kerak. Buning uchun const_iteratop tipi ishlatiladi. Bu iteratorlar faqat elementlarni sanash imkonini beradi. O‘zgartirish mumkin emas.

```
vector<int>::const_iterator iter;
        for(iter = myvector.begin(); iter!=myvector.end(); ++iter)
    {cout << *iter << " | ";
            //*iter = (*iter) * (*iter);
    }
```

O‘zgarmas iteratorning qiymatlarini olish uchun cbegin() va cend() funksiyalari ham ishlatiladi. Agar iterator o‘zgarmas deb olinmagan bo‘lsa ham, bu funksiyalar uni o‘zgarmas qilib beradi. Shungdek, o‘zgartirish mumkin emas bo‘ladi.

1.3-dastur. O‘zgarmas iteratordan foydalanish.

```
// ConsoleApplication1.cpp : main project file.

#include "stdafx.h"
#include <iostream>
#include <vector>
using namespace std;

int main(){
        int myints[] = {16,2,77,29,28};
        const vector<int> myvector (myints, myints + sizeof(myints) / sizeof(myints[0])
); vector<int> myvector_one; for (int i = 0; i < 15; i++)
                myvector_one.push_back(rand() % 10);
        vector<int>::const_iterator   it;
        for (it = myvector.begin(); it != myvector.end(); ++it) cout << '
                ' << *it;
        cout << endl;
        for (it = myvector_one.cbegin(); it != myvector_one.cend(); ++it) cout <<
                ' ' << *it;
        cout << endl;
        system("pause");
        return 0;
}
```

1.3-dastur.Output

```
        16 2 77 29 28
 1 7 4 0 9 4 8 8 2 4 5 5 1 7 1
```

Teskari iteratorlar. Bu iteratorlar konteynerning elementlarini teskari olishni amalga oshiradi. Buning uchun reverse_iterator tipi ishlatiladi. Bu tipdagi iteratorlarga konteynerning elementlarini olish uchun rbegin() i rend() funksiyalaridan foydalaniladi.

```
#include "stdafx.h"
#include <iostream>
#include <vector>
using namespace std;
int main()
{
    int myints[] = { 1, 2, 3, 4, 5 };
      vector<int> v (myints, myints+ sizeof(myints)/sizeof(myints[0]));
    for (vector<int>::reverse_iterator iter = v.rbegin(); iter != v.rend(); ++iter) std::cout << *iter
            << " | ";

    cout << endl;
      system("pause");
    return 0;
}
```

Agar konteynerni o‘zgartirishdan himoyaya qilish va teskarisi kerak bo‘lsa, const_reverse_iterator tipidagi iteratordan va uning crbegin() va crend() funksiyalaridan foydalanish maqsadga muvofiqdir. Yuqoridagi dasturga quyidagi fragmentni yozish yetarli bo‘ladi. Shuningdek iteratorlari qo‘shima amallarga ham ega (list va forward_list konteynerlaridan tashqari):

```
for (vector<int>::const_reverse_iterator iter = v.crbegin(); iter != v.crend();
++iter)
        std::cout << *iter << " | ";
```

- **iter + n** - n ta pozitsiya keyingi elementni ko‘rsatuvchi iteratorni qaytaradi.
- **iter – n** - n ta pozitsiya oldingi elementni ko‘rsatuvchi iteratorni qaytaradi.
- **iter += n** - iteratorni n - chi pozitsiyaga o‘tkazadi
- **iter -= n** - iteratorni n - chi pozitsiyaga o‘tkazadi
- **iter1 - iter2** - iter1 va iter2 lar orasidagi pozitsiyalar sonini qaytaradi
- [>], [>=], [<], [<=] - taqqoslash amallari. Agar iterator oxiriga yaqin elementni ko‘rsatsa boshqasidan katta.

1.4-dastur. Iteratorning qo‘shimcha amallaridan foydalanish.

```
#include "stdafx.h"
#include <iostream>
#include <vector>
using namespace std;
int main()
{int myints[] = { 1, 2, 3, 4, 5 };
        vector<int> v (myints, myints+ sizeof(myints)/sizeof(myints[0])); for
        (auto i = v.begin(); i < v.end(); i++)
        {cout << *i << " | ";}
        cout << endl;
        auto it1 = v.begin(); auto
        it2 = it1 + 2;
        cout << "it1 + 2 => " << *it2 << endl; auto
        it3 = v.end() - 3;
        cout << "it1 - 3 => " << *it3 << endl; it2 += 2 ;
        cout << "it2 + 2 => " << *it2 << endl; it3 -= 2;
        cout << "it3 - 2 => " << *it3 << endl;
   cout << "it2 - it1 => " << it2 - it1 << endl; cout << "(it1 < it2) => " << (it1 < it2) << endl;
```

```
cout << "(it1 >= it2) => " << !(it1 < it2) << endl; cout << "(it1 !=
        it2) => " << !(it1 == it2) << endl;
cout << "(it1 == it2) => " << (it1 == it2) << endl; cout << "(it1 !=
        it2) => " << (it1 != it2) << endl; cout << endl;
        system("pause");
    return 0;
}
```

1.4-dastur.Output

```
| 2 | 3 | 4 | 5 |
it1 + 2 => 3
it1 - 3 => 3
it2 + 2 => 5
it3 - 2 => 1
it2 - it1 => 4
(it1 < it2) => 1
(it1 >= it2) => 0
(it1 != it2) => 1
```

```
(it1 == it2) => 0
(it1 != it2) => 1
```

Taqqoslash amallari uchun ko‘chirib o‘tish iteratorlari va bazaviy iteratorlar uchun ba‘zi C++ dasturlashda farqlari bor.

1.1-jadval. Taqqoslash amallari yozish uslublari.

Ko‘chirib o‘tish iteratorlari uchun.	Ekvivalent bazaviy iteratorlari uchun.
move_ita == move_itb	basic_ita == basic_itb
move_ita != move_itb	!(basic_ita == basic_itb)
move_ita < move_itb	basic_ita < basic_itb
move_ita <= move_itb	!(basic_itb < basic_ita)
move_ita > move_itb	basic_itb < basic_ita
move_ita >= move_itb	!(basic_ita < basic_itb)

Xotirani taqsimlovchilar va ularga qo‘yilgan talablar. Dinamik xotira bilan ishlashda maxsus truklarni, fragmentlardan foydalanish bo‘lmasa, ko‘pigina algoritmlardan foydalanish samarasiz bo‘lshi mumkin. Bunga misol sifatida ikkita holatni ko‘rib chiqamiz. new va delete operatorlarining qayta aniqlanib yuklanishida sintaktik konstruktorlar kichikroq bo‘ladi va dastur lokalizatsiya qilish oddiy bo‘ladi. Shuningdek protsessordagi amallar tizimida ham qayta aniqlash sodir bo‘ladi.

Avvalo, xotirani ishlashini tezlashtirishda aqlli allocator qanchalik foydasi borligini aniqlash lozim. Buning uchun oddiy test misollarini (C++ va S#) tillarida ko‘rib chiqaylik (bu xotira bilan ishlash uchun yaxshi menejer hisoblanadi va ob‘yektlarni avlodlarga ajratadi, turli o‘lchamdagi ob‘yektlar uchun turli pullardan (joylar) foydalanadi).

C++ dagi dastur

```
class Node { public:Node* next;};

// ...
for (int i = 0; i < 10000000; i++) {
        Node* v = new Node();}
```

C++ dagi dastur

```
class Node
{public Node next;}
// ...
for (int l = 0; l < 10000000; l++)
{var v = new Node();}
```

Dasturlar sharsimon vakum usulini qo‘llab ham vaqt bo‘yicha solishtirilsa 10 barovar (62 ms va 650 ms) farq bo‘lmoqda. Shuningdek C# ishini tugutdi, ammo C++ da esa yaxshigini xotira ajratilan, bu hali o‘chirish kerak.

Pul ob‘yektlari. Pul ob‘yektlari uchun aniq echim - OS dan katta xotira blokini olish va uni teng blokli o‘lchamlarga bo‘lishdir sizeof(node). Xotirani ajratishda blokni puldan olinadi va uni bo‘shatishda uni pulga qaytariladi. Pulni tashkil qilishning eng oson usuli - bitta aloqa stekni (stack) ishlatishdir.

Maqsad dasturda minimal qayta aniqlashlar bo‘lgani uchun, BlockAlloc sinfni merosxo‘ri sifatida Node sinfini tanlashdir.

```
class Node : public BlockAlloc<Node>
```

Avvalo, OS yoki C-runtime dan katta bloklar uchun bir pul olish kerak. Buni malloc va free funksiyalari bilan tashkil qilish mumkin, lekin katta samaradorligi uchun (ortiqcha abstraksiya bosqichini o‘tkazish uchun), Virtualloc/Virtualfreye funksiyalaridan foydalanish maqsadga muvofiq. Bu vazifalar 4 karrali bo‘lgan bloklarga xotira ajratadi va 64 karrali bo‘lgan bloklar jarayon manzili uchun oraliq saqlab turadi. Bir vaqtning o‘zida commit va reserve imkoniyatlari ko‘rsatilgan holda yana bir darajada ko‘tarilib, manzil oraliqni rezervlash va bitta chaqirish bilan xotira bloklarini ajratamiz.

PagePool sinfi dastur fragmenti.

```
inline size_t align(size_t x, size_t a) { return ((x-1) | (a-1)) + 1; }
//#define align(x, a) ((((x)-1) | ((a)-1)) + 1)

template<size_t PageSize
= 65536> class PagePool
{public:void* GetPage()
{
  void* page = VirtualAlloc(NULL, PageSize, MEM_COMMIT | MEM_RESERVE,
PAGE_READWRITE); pages.push_back(page); return page;
}
            ~PagePool() {
for (vector<void*>::iterator i = pages.begin(); i != pages.end(); ++i)
```

```
{
                    VirtualFree(*i, 0, MEM_RELEASE);
                }
        }
private:
        vector<void*> pages;
        };
```

Berilgan o‘lchamga mos, pul bloklarni hosil qilish uchun BlockPool sinfini yaratamiz.

BlockPool sinfining dastur fragmentlari:

```
template<class T, size_t PageSize = 65536, size_t Alignment = 8 /* sizeof(void*) */> class
BlockPool : PagePool<PageSize>
{public: BlockPool() : head(NULL) {
                BlockSize = align(sizeof(T), Alignment);
                count = PageSize / BlockSize;}
        void* AllocBlock() {
                // todo: lock(this)
                if (!head) FormatNewPage();
                void* tmp = head;
                head = *(void**)head;
                return tmp;}
        void FreeBlock(void* tmp) {
                // todo: lock(this)
                *(void**)tmp = head;
                head = tmp;}
                private:void* head;
                size_t BlockSize;
                size_t count;
        void FormatNewPage() {
                void* tmp = GetPage();
```

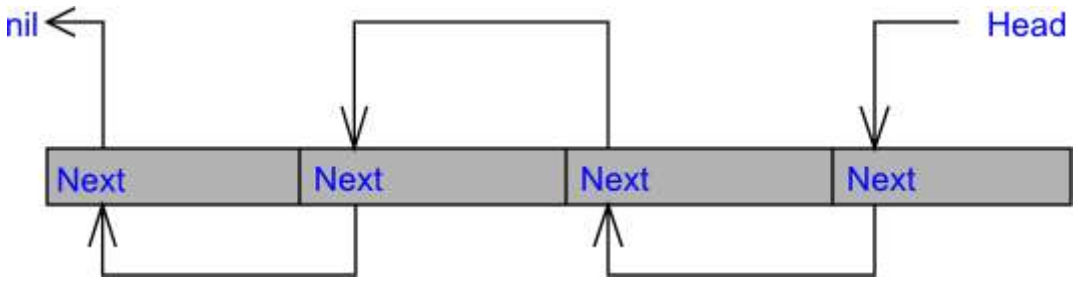
```
head = tmp;
for(size_t i = 0; i < count-1; i++) {
        void* next = (char*)tmp + BlockSize;
        *(void**)tmp = next;
        tmp = next;}
*(void**)tmp = NULL;}};
```

Potoklar orasidagi sinxronzatsiyani tashkil qilish uchun berilgan manzil ***todo: lock(this)*** izohga olib qo‘yilgan. Buning uchun EnterCriticalSection yoki boost::mutex dan foydalanish ham mumkin.

Agar biror narsa yozilgan bo‘lsa, bloklar majmuini formatlashda abstrakt FreeBlock pulga blok qo‘shish ishlatilmaydi. O‘ylab ko‘ring.

```
for (size_t i = 0; i < PageSize; i += BlockSize) FreeBlock((char*)tmp+i);
```

FIFO tamoyilidagi bloklar majmui teskari bo‘lib qoladi.



1.1-rasm. FIFO tamoyilidagi bloklar majmui.

Puldan ketma-ket so‘ralgan bir necha blok manzillar kamayib ketadi. Shuningdek protsessor orqaga qaytishni yoqtirmaydi, chunki uning Prefetch amalii buziladi (**UPD**: zamonaviy protsessorlar uchun tegishli emas). Agar takrorlanish bilan almashtirilsachi?

```
for (size_t i = PageSize-(BlockSize-(PageSize%BlockSize)); i != 0; i -= BlockSize) FreeBlock
```

Takrorlanishda almashtirishlar manzillar bo‘yicha orqaga bajariladi. Tayyorgalik ko‘rilganidan so‘ng, sinf ob‘yektni yozish mumkin:

```
 template<class T>
class BlockAlloc {
public:
      static void* operator new(size_t s) {
```

```
            if (s != sizeof(T)) {
                return ::operator new(s);
            } return pool.AllocBlock();
        }static void operator delete(void* m, size_t s) { if (s != sizeof(T)) {
                ::operator delete(m);
             } else if (m != NULL) {
                pool.FreeBlock(m);
            }
        } static void* operator new(size_t, void* m) { return m;}
         static void operator delete(void*, void*) { }
private: static BlockPool<T> pool;};
template<class T> BlockPool<T> BlockAlloc<T>::pool;
```

Sinfda ular ishlaganda nima uchun if (s != sizeof(T)) kerak. T bazaviy sinfdan mo‘rosxo‘r sinf yaratilgan va o‘chirilgan bo‘lsa? degan savol bo‘lishi mumkin.

Merosxo‘rlar odatda oddiy new/delete operatorlari bilan ishlatiladi, shuningdek BlockAlloc ni ham joylatirish mumkin. Bu yo‘l bilan oson va xavfsiz dasturda biror narsa buzib qo‘rqmasdan, pullar foydalanishga qaratilgan sinflar aniqlash mumkin. Bir nechta merosxo‘rlar ham bu bilan yaxshi ishlaydi.

BlockAlloc dan mo‘rosxo‘r sifatida Node ni olamiz va yana sinov o‘tkazamiz.

Sinov natijasiga qarasak, 5 barovar tez va 120 msda bajarilmoqda. Ammo, c# allokator hali ham yaxshi. Mumkin u yerda oddiy bo‘lmagan bir aloqali ro‘yxatdan foydalanilgandir. Agar, new operatoridan so‘ng zudlik bilan delete operatori ishlatilsa, xotira kamroq ishlatiladi va keshda ma‘lumotlar ham kamayadi va natijani 62 msda olish mumkin. Qiziqarli tomoni shundaki, . NET CLR da GC ni kutmasdan pulga mos bo‘sh qolgan lokal o‘zgaruvchini tez qaytaradi.

Konteyner va uni qo‘llanilishi. Xotirada ko‘p hollarda merosxo‘r ob‘yektlarini saqlovchi sinflar vujudga keladi. Ularning xotiradan o‘chirilishi bazaviy sinf bilan bog‘liqdir. Masalan, sinf bo‘lsin va tugun ichidagi matndan olingan mo‘rosxo‘rlari Node va Attribute hamda (char*) qatorlardan to‘ldirilgan. Yoki papkani qayta o‘qish va yana o‘zgartirishda, bir marta yuklanadi fayl menejeri fayllar va kataloglar ro‘yxati.

Yuqorida ko‘rsatilgandek, new operatoriga nisbattan delete operatori foydalanish qiyinroq va muammoliroq bo‘ladi. Bazaviy sinf ob‘yekti bilan bog‘liq katta blokda merosxo‘r sinf ob‘yektlari uchun xotira ajratish kerak. Bazaviy sinf ob‘yektini yo‘q qilish paytida, destruktor odatdagidek, merosxo‘r sinf uchun chaqiriladi, lekin xotirani qaytarishingiz shart emas — bu katta blokda ozod qilinadi.

PointerBumpAllocator sinf yaratamiz. Bu sinf katta blokdan turli o‘lchamdagi qismlarni kesib tashlash va eskisi tugagach, yangi katta blokni tanlashga imkon beradi.

PointerBumpAllocator sinfining dastur fragmenti.

```
template<size_t PageSize = 65536, size_t Alignment = 8 /* sizeof(void*) */> class
PointerBumpAllocator
{public: PointerBumpAllocator() : free(0) { }

        void* AllocBlock(size_t block) {
                // todo: lock(this)
                block = align(block, Alignment); if
                (block > free) {
                        free = align(block, PageSize);
                        head = GetPage(free);
                } void* tmp = head;
```

```
                head = (char*)head + block;
                free -= block;
                return tmp;
        }
        ~PointerBumpAllocator() {
                for (vector<void*>::iterator i = pages.begin(); i != pages.end(); ++i)
{VirtualFree(*i, 0, MEM_RELEASE);}
        }
private: void* GetPage(size_t size) {
                void* page = VirtualAlloc(NULL, size, MEM_COMMIT | MEM_RESERVE,
PAGE_READWRITE);
                pages.push_back(page);
                return page;}
        vector<void*> pages;
        void* head;
        size_t free;};
typedef PointerBumpAllocator<> DefaultAllocator;
```

Nihoyat, new va delete operatorlari bilan childObject ni tuzamiz va berilgan allocator bilan murojaat qilamiz:

```
template<class T, class A = DefaultAllocator>
struct ChildObject
{static void* operator new(size_t s, A& allocator) { return allocator.AllocBlock(s);}
      static void* operator new(size_t s, A* allocator) {
            return allocator->AllocBlock(s);}
      static void operator delete(void*, size_t) { }
```

```
      static void operator delete(void*, A*) { }
      static void operator delete(void*, A&) { }
private:
      static void* operator new(size_t s);
};
```

Child sinfiga o‘zgaruvchilarni qo‘shish uchun barcha e‘lonlarni new operatori orqali amalga oshirish kerak bo‘ladi. New operatori quyidagicha bo‘ladi.

```
new (… parametrlar… ) ChildObject (…konstruktor parametrlari… )
```

Qulaylik uchun A& va A* uchun new operatorlarni qo‘llaymiz.

```
node = new(allocator) XmlNode(nodename);
```

Agar allokator ajratuvchi sifatida qo‘shilsa, ikkinchisidan foydalanish qulayroq:

```
node = new(this) XmlNode(nodename);
```

Bundan tushinarli bo‘ladiki, ortiqcha belgilardan qochishda amallarni bo‘lish uchun ko‘rsatkich va havolalar konvertatsiya bo‘ladi.

Ob‘yekt yaratishda qaysi New operatoridan foydalanganligiga qaramasdan, delete operatori yordamida o‘chirish amalga oshirilmaydi, kompilyatorning o‘zi standart delete operatoridan foydalanadi. Quyidagicha sintaktik asosida:

```
delete node;
```

Agar ChildObject sinfi ob‘yekti yoki uning merosxo‘ri ob‘yektidan foydalanayotgan bo‘lsangiz, new istisno vaqtida operatoriga mos delete orperatori chaqiriladi. Shuning uchung bu ob‘yektdan foydalanganda birinchi size_t parametrni void*ga o‘zgartirish lozim.

New operatorini private bo‘limiga joylashtirish, uni allokatorsiz ishlashaga ruxsat bermaydi.

Standart bo‘yicha taqsimlovchilar.

1. Tuzilmali o‘zgaruvchisi uchun dinamik xotira ajratish. Tuzilmali o‘zgaruvchisi uchun xotira ajratish va bo‘shatish. Quyidagi tavsifga ega bo‘lgan data tuzilmasi berilgan bo‘lsin:

```
struct  Date
        {int  day; int
        month; int
        year;
};
```

struct Date tipidagi o‘zgaruvchi uchun xotiradan joy ajratish va foydalanish uchun quyidagicha dastur fragmentini yozish kerak:

1.5-dastur. xotiradan joy ajratish va foydalanish.

```
#include     "stdafx.h"
#include   <iostream>
using namespace std;

// tuzilma, kun, oy, yil;

struct Date
{int day; int month; int year;};
int main(){ // new operatori bilan xotiradan joy ajratish
      struct Date * pd; // struct Date ga ko‘rsatkich

      try{ pd = new struct Date; // xotiraga joy ajratish
      } catch (bad_alloc ba)     {cout << "Xorita ajratilmadi" << endl;
       return -1;}
      // Agar xotira ajratilgan bo‘lsa, unga 16.14.2020 qimat qilib beramiz.
       pd->day = 16; pd->month = 04; pd->year = 2020;
          cout << pd->day << ":" << pd->month << ":" << pd->year << endl;

      //   pd o‘zgaruvchisini xotiradan bo‘shatish
      delete pd;system("pause"); return 0;}
```

2. Sinf ob‘yekti uchun dinamik xotira ajratish.

CDayWeek sinf ob‘yektining ko‘rsatkichiga xotira ajratish va foydalanishni keltiramiz.

1.6-dastur. xotira ajratish va foydalanish.

```
// ConsoleApplication1.cpp : main project file.
#include
"stdafx.h"
#include
<iostream
>      using
namespac
e std;

// sinf, hafta
kunlari

class CDayWeek{int d; public:
```

```
    // konstruktor
    CDayWeek() { d = 1; }

    // funksiyalari
    int Get(void) { return d; } void Set(int nd) {

    if ((1<=nd)&&(nd<=7)) d = nd;}};
int main(){ //  new bilan xotiradan joy ajratish
 CDayWeek *p; try {//  p ko‘rsatkich uchun xotiradan joy ajratish
        p = new CDayWeek(); // CDayWeek sinf  konstruktorini chaqirish
    } catch (bad_alloc ba) {
        cout << "Xotira ajratilmadi" << endl; cout
        << ba.what() << endl; return -1;}

        //  qimat berish

        p->Set(3);
    cout << p->Get() << endl;
    //  p ko‘rsatkichdan xotirani bo‘shatish
    delete p;
      system("pause");
    return 0;
      }
```

3. new operatori yordamida massivlarga xotiradan joy ajratish va foydalanish. new operatori massiv uchun xotira ajratish uchun foydalanish mumkin. Massivga xotira ajratishda new operatorning umumiy shakli:

```
ptrArray  = new  type[size];
```

Bunda, *ptrArray* - xotira ajratilishi kerak bo‘lgan massiv nomi, type - elementlarning tiplari, size - ular bazaviy va ixtiyoriy bo‘lishi mumkin, massiv o‘lchami (elementlar soni).

Delete operatori yordamida massiv ob‘yektidan xotirani bo‘shatish uchun quyidagi fragment yoziladi:

```
delete[]  ptrArray;
```

4. Bazaviy tipli ko‘rsatkich massiviga dinamik xotira ajratish va foydalanish. Quyida float tipidagi ko‘rsatkich massivi uchun xotira ajratadi. So‘ngra qator elementlari ixtiyoriy qiymatlar bilan to‘ldiriladi. Shundan so‘ng ajratilgan xotira delete [] operatori tomonidan bo‘shatiladi.

```
float * ptrArray; ptrArray =
new float[10]; int d;
for(int i = 0; i < 10; i++) ptrArray[i] = i * 2 + 1;
d = ptrArray[3]; delete[] ptrArray;
```

5. Tizimli o‘zgaruvchilar massivi uchun xotira ajratish va undan foydalanish. 3 ta talaba tuzilmalar massivi uchun xotira ajratish va bo‘shatish ko‘rsatilgan. shuningdek, tuzilmalar massivining element maydonlariga murojaat qilish ham ko‘rsatilgan.

1.7-dastur. Xotira ajratish va foydalanish.

```
#include
"stdafx.h"
#include
<iostream>
using
namespace
std;

struct TStudent

{string name; string numBook;

int course; float rank;};

int main(){ int n_students = 3; TStudent * pS;

    try{pS = new struct TStudent[n_students];}

    catch (bad_alloc ba){cout << "Xotira ajratilmadi " << endl;

    cout << ba.what() << endl; return -1;}

    pS->name = "Aliyev kbar"; pS->numBook =
    "965874123";

    pS->rank = 3.93; pS->course = 1;

    (pS + 1)->name =
    "Bahromov Batir"; (pS +
    1)->numBook =
    "20930032"; (pS + 1)-
    >rank = 4.98;

    pS[1].course = 3; pS[2].name = "QobilovUsmon";

     pS[2].numBook = "l2l28983"; pS[2].rank
    = 4.32;

    pS[2].course = 2;

    cout << pS->name.c_str() << endl;
```

```
    cout << pS[1].rank << endl;
    cout << (pS + 2)->numBook.c_str() << endl;
    delete[] pS; system("pause");return 0;}
```

6. Massiv ob‘yektlari uchun xotira ajratish va undan foydalanish.New operatoridan foydalangan holda massiv ob‘yektlari uchun xotira ajratish ko‘rsatiladi va delete operatori bilan xotiradan massiv ob‘yektlarini bo‘shatish ko‘rsatiladi.

1.8-dastur. Xotira ajratish va foydalanish.

```
#include
"stdafx.h"
#include
<iostream>
using namespace
std;

class CMonth{private:int month;
    public:CMonth() { month = 1; }
    CMonth(int nmonth){if ((1 <= nmonth) && (nmonth <= 12)) month = nmonth;
        Else month = 1;
    } int Get(void) { return month; } void Set(int nmonth){
        if ((1 <= nmonth) && (nmonth <=
            12)) month = nmonth;}};

        int main(){
    int nMonths =
    12; CMonth *
    pM;

    try {
        pM = new CMonth[12];
    }
    catch (bad_alloc ba)
    {
```

```
        cout << "Xotira ajratimadi: " << endl;
        cout << ba.what() << endl;

        return -1;
    }
    for (int i = 1; i <= 12; i++)
        pM[i - 1].Set(i);

    cout << pM[3].Get() << endl;
    cout << pM[5].Get() << endl;

    delete[] pM;
       system("pause");

    return 0;
}
```

Yuqoridagi dastur fragmentida massiv obektlar ichki o‘zgaruvchi 1 qiymatiga initsializatsiya qilinadi, chunki bu qiymat Cmonth() parametrlarisiz konstruktorda o‘rnatiladi. Bu konstruktor massiv uchun initsializator vazifasini bajaradi. Biroq, sinf boshqa konstruktorni amalga oshiradi, ya‘ni konstruktor bilan 1 qiymatli parametr yoki parametrli konstruktorni. C++ sintaktikiga ko‘ra, massiv obektlarini parametrli konstruktor bilan ishga tushirib bo‘lmaydi. Shuning uchun, cmonth sinf parametrlar siz konstruktor amalga oshirish kerak. Cmonth() parametrsiz konstruktor sinf dastur fragmentidan olib tashlansa,massiv obektlar uchun xotira ajratish mumkin bo‘lmaydi. Bitta ob‘yektlar uchun xotira ajratish mumkin, massiv uchun mumkin emas.

Agar ma‘lum bir sinf massiv ob‘yektlari uchun xotira ajratish kerak bo‘lsa, bu sinf parametrlarsiz konstruktorga ega bo‘lishi va bajarilishi kerak.

7. Massiv o‘lchamini dinamik kattalashtirish (kichiklashtirish) uchun xotira ajratish va foydalanish. Tuzilmalar uchun xotirani qayta ajratish, tuzilmalarni ishga tushirish. masalan, Dayweek tuzilishi tip uchun xotira qayta ajratish jarayoni ko‘rsatiladi. Xotirani dinamik ravishda ajratish va qayta ajratish bu usulning xotirani statik ajratishga nisbatan asosiy afzalligi hisoblanadi. Kerak bo‘lganda va kerak bo‘lganda dasturda xotira ajratishingiz mumkin. Dayweek tuzilishi parametrlarsiz konstruktor yordamida amalga oshiriladi (yuqoridagidek).

1.9-dastur. Xotira ajratish va foydalanish.

```
#include
"stdafx.h"
#include
<iostream>
using
namespace
std;

struct DayWeek{ int day;
DayWeek() { day = 1; }
};
int main(){
DayWeek * p;
try {
p = new DayWeek[5];
}
catch (bad_alloc ba){
cout << "Xotira ajratilmadi" << endl; cout << ba.what() << endl;
return -1;
}
for (int i = 0; i < 5; i++) p[i].day = i + 1;
    int d;
    d = (p +
    1)->day;
    DayWee
    k * p2;
    try     {
        p2 = new DayWeek[7];
    }
    catch (bad_alloc ba){
        cout << "Xotira ajratilmadi"
        << endl; cout << ba.what() <<
        endl;
        return -1;
    }
```

```
        for (int i = 0; i < 5;
            i++) p2[i] =
            p[i];
    delete[] p;
     p = p2;
    for (int i = 5; i < 7; i++)
        p[i].day = i + 1;


    d = p[5].day;
    d = (p + 6)->day;
```

Main() funkssiyasida xotira avval 5 ta tuzidmadan iborat massiv uchun ajratiladi. Keyin bu xotira 7 tuzilmali massiv uchun qayta ajratiladi (reallocated). Buning uchun qo‘shimcha p2 ko‘rsatikich ishlatiladi. Qayta taqsimlashda avval p2 (7 element) uchun xotira ajratiladi. So‘ngra ma‘lumotlar p dan p2 ga ko‘chiriladi. Bu pp ko‘rsatkich uchun ajratilgan edi, xotira bo‘shatiladi (5 ta elementdan). Keyingi qadamda p qiymat p2 ga o‘rnatiladi. Shunday qilib, har ikki ko‘rsatkich bir xil xotira maydoni ko‘rsatadi. Massiv bilan ishlagandan so‘ng, p massiv uchun xotira ajratiladi.

Bu yuqorida ko‘rsatib o‘tilgan dasturlash dinamik xotirani ajratish va foydalanishning standart shakllari bo‘lib hisoblanadi.

Nazorat savollari

1. Iteratorlari qo‘shima amallar qaysi konteynerlar uchun ishlatilmaydi.
2. Qaysi amal iter1 va iter2 lar orasidagi pozitsiyalar sonini qaytaradi.
3. (move_ita != move_itb) va !(move_ita == move_itb) munosabatlar bir xilmi, bir xil bo‘lsa nima uchun, farqi bo‘lsa, nima uchun.
4. Xotirani taqsimlovchilar va ularga qo‘yilgan talablar nima uchun kerak.
5. Merosxo‘rlar odatda oddiy new/delete operatorlari bilan ishlatiladi, shuningdek BlockAlloc ni ham nima uchun joylatirish kerak.
6. Agar, new operatoridan so‘ng zudlik bilan delete operatori ishlatilsa, xotira kamroq ishlatiladi va qaerdagi ma‘lumotlar ham kamayadi.
7. Qaysi operatordan new operatori yoki delete operatori foydalanish qiyinroq va muammoliroq, sababini tushintirib bering.
8. New operatori qaysi bo‘limiga joylashtirilsa, uni allokatorsiz ishlashaga ruxsat bermaydi.
9. Tuzilmali o‘zgaruvchisi uchun xotira ajratish va bo‘shatish uchun qanday amallar bajarish kerak.
10. Sinf ob‘yekti uchun dinamik xotira ajratish uchun qanday amallar bajarish kerak.

11. new operatori yordamida massivlarga xotiradan joy ajratish va foydalanish uchun qanday amallar bajarish kerak.

12. Bazaviy tipli ko‘rsatkich massiviga dinamik xotira ajratish va foydalanish uchun qanday amallar bajarish kerak.

13. Ma‘lum bir sinf massiv ob‘yektlari uchun xotira ajratish kerak bo‘lsa, bu sinf qanday parametrli konstruktorga ega bo‘lishi kerak.

14. Massiv o‘lchamini dinamik kattalashtirish (kichiklashtirish) uchun xotira ajratish va foydalanish uchun qanday amallar bajarish kerak.

9- MA’RUZA

SONLI SINFLAR BILAN ISHLASH (2 SOAT)

REJA:

1. Complex sonli sinf va funksiyalari
2. Vallaray sonli sinf va funksiyalari
3. Slice va gslice sonli sinflari va funksiyalari
4. Qo‘shimcha sonli sinflar

Annotatsiya: ***Complex (kompleks) sonli sinf va funksiyalari***. *C++ standarti kutubxonasidagi kompleks sonlar sinfi ob‘yekt modelidan foydalanishning yaxshi namunasidir. Arifmetik amallar qayta aniqlash tufayli bu sinf ob‘yektlari o‘rnatilgan ma‘lumot tiplaridan biriga tegishli kabi ishlatiladi. Bundan tashqari, an‘anaviy arifmetik o‘zgaruvchilar va kompleks sonlar bir vaqtning o‘zida ixtiyoriy amallarda ishtirok etishi mumkin. (Eslatib o‘tamiz, bu yerda kompleks sonlar matematikasiga oid umumiy savollar bilan shug‘ullanmayapmiz. Matematika bo‘yicha kopleks sonlar nazariyasini o‘qish kerak).*

Kalit so‘zlar: *kutubxona, sinf, shablon, matematik amallari, kompleks son, to‘plam, to‘plam osti, to‘lam ustida amalar, saralash, interval son.*

KIRISH

Hayot turmush tarzida foydalaniladigan sonlar to‘plamlardan boshqa sonlar to‘plami va ular ustida bajarilajigan amallar mavjud. Masalan, maktabdan bilamizki, kompleks sonlar, oraliq sonlar, qolaversa, zamonaviy ilmiy sohalarda turli fazolarda turlicha sonlar to‘plami ishlatiladi. Interval sonlari, noravshan to‘plamlar shular kabi. Bu sonlar to‘plami hayotdagi qaysidir tizimni modellashtirish va uni jarayonlariga oid masalalarni yechishga xizmat qiladi. Bu masalarni ayniqsa kompyuterda yechish kerak. Shuning uchun ba‘zi bir ko‘p foydalaniladigan sonlar to‘plami uchun

sinflar yaratilgan. Bu sonlarda ularning amallari, funksiyalari xususiyatlari keltirilgan. Bunday sonlarga complex, vallaray, slice, gslice kabi sinflar va muallif tomonidan 2013 yilda yaratilgan interval sinf kutubxonalarini misol qilib olish mumkin.

Complex (kompleks) sonli sinf va funksiyalari. C++ standarti kutubxonasidagi kompleks sonlar sinfi ob‘yekt modelidan foydalanishning yaxshi namunasidir. Arifmetik amallar qayta aniqlash tufayli bu sinf ob‘yektlari o‘rnatilgan ma‘lumot tiplaridan biriga tegishli kabi ishlatiladi. Bundan tashqari, an‘anaviy arifmetik o‘zgaruvchilar va kompleks sonlar bir vaqtning o‘zida ixtiyoriy amallarda ishtirok etishi mumkin. (Eslatib o‘tamiz, bu yerda kompleks sonlar matematikasiga oid umumiy savollar bilan shug‘ullanmayapmiz. Matematika bo‘yicha kopleks sonlar nazariyasini o‘qish kerak).

Bu sinfni ishlatish uchun complex sarlavha fayl qo‘shilishi kerak. Kichik dastur fragmentiga qarang:

```
#include <complex>
// ...
complex <double> a;
complex <double> b;
// ...
complex< double > c = a * b + a / b;
```

Kompleks va arifmatik amallar o‘zaro hamkorlikda bajarailadi.

```
complex< double > a;
complex< double > complex_obj = a + 3.14159;
```

Kompleks tiplarni arifmetik tiplar bilan ham yaratish va ular uchun operator= ni ishlatish mumkin.

```
complex< double > complex_obj;
double dval = 3.14159; complex_obj =
dval;
// ....
int ival = 3; complex_obj
= ival;
```

Ammo teskarisi, ya‘ni kopmleks tipni arifmetik tipga tenglashtirish mumkin emas.

```
complex< double > complex_obj;
double dval =  complex_obj;
```

Bunda arifmetik tipni akslantirish xato degan xabar beradi.

Kompleksning 2 qismi bo‘ladi haqiqiy va mavhum. Haqiqiy (real) yoki mavhum – qismini oddiy arifmetik tipga qiymat qilib berish mumkin. Kompleks sonlar sinfi o‘z navbatida real va mavhum qismlarni qaytaruvchi ikkita funksiyaga ega. Biz sinf a‘zolaridan foydalanish uchun sintaktiki yordamida ularni olishimiz mumkin:

```
double re = complex_obj.real();
double im = complex_obj.imag();
```

Yoki bu sintaktikga ekvivalet sintaktik:

```
double re = real(complex_obj);
double im = imag(complex_obj);
```

Kompleks son sinfi to‘rt matematik amallar - operatorlari qo‘llab-quvvatlaydi. Bularga [+=], [-=], [*=] va [/=] operatorlari kiradi. Masalan,

```
complex< double > complex_obj; complex<
double > second_complex_obj;
complex_obj += second_complex_obj;
```

Kompleks sonlarni kiritish/chiqarish ham qo‘llab-quvvatlanadi. Chiqish operatori vergullar bilan ajratilgan real va mavhum qismlarni qavslar ichida chop etadi. Masalan, chiqish operatorlarini bajarish natijasi:

```
complex< double > complex0( 3.14159, -2.171 );
complex< double > complex1( complex0.real() );

cout << complex0 << " " << complex1 << endl;
```

output
(3.14159,-2.171) (3.14159,0)

KIRISHtish operatorlari quyidagi formatlar bilan ishlaydi: Kompleks soning real qismini kiritish oddiy kiritiladi, masalan, 525.25; mavhum qismi qavslar ichida kiritiladi, masalan, (123.5), to‘liq kiritish uchun qavs ichida real va mavhum qismlari vergul bilan ajratiladi, Masalan, (12.5, 2.5). 1.1-dasturga qarang.

1.1-dastur. Kopleks sonlarni kiritish va chiqarish.

```
#include "stdafx.h"
#include <iostream>
#include <complex>
using namespace std;
int main(){
        complex< double > a, b, c;
        cout << "Complex sinfi oid turli t=farmatli 3 ta son kiritibg:" << endl; cin >> a
        >> b >> c;
        cout << "Complex sinfiga oid sonlar:" << endl; cout
        << a << b << c << endl;
        system("pause");
}
```

1.1-dastur. Output

```
Complex sinfiga oid turli formatli 3 ta son kiritibg: 3.1415
(-0.25)
(2.71, -9.8)
Complex sinfiga oid sonlar: (3.1415,0)(-
0.25,0)(2.71,-9.8)
```

Ushbu funksiyalardan tashqari, kopmleks sonlar sinfi quyidagi funksiyalariga ega:

```
sqrt(), abs(), polar(), sin(), cos(), tan(), exp(), log(), log10() va pow().
```

Yozish vaqtida mavjud C++ standart kutubxonasi o‘ng operandli (masalan, qo‘shishi +=) operatorlar kompleks soni bo‘lmasa murakkab tayinlash amallarini qo‘llab-quvvatlamaydi. Masalan, bu yozuvga ruxsat berilmaydi:

```
complex_obj += 1;
```

C++ standarti ko‘ra, bu operatorlar bo‘lishi kerak, ishlab chiqaruvchilar ko‘pincha standartlarni yozib tugatishga etib kelolmaydilar.

Bunday amallarni amalga oshirish uchun o‘z operatorimizni aniqlashimiz mumkin. Bu yerda qo‘shish operatorini amalga oshiruvchi funksiyaning varianti keltirilgan.

```
inline complex<double>& operator+=( complex<double> &cval, double dval ){
    return cval += complex<double>( dval );
}
```

Bu misol yordamida, turi boshqa murakkab tayinlash uchun uch operatorlari funksiyalarini yozish muammo emas. Quyidagi dastur uchun murakkab tayinlash uchun uch operatorlari funksiyalarini qo‘shing va tekshirish uchun uni ishga tushirish.

```
int main() {
    complex< double > cval ( 4.0, 1.0 ); cout << cval << endl;
    cval += 1;cout << cval << endl; cval -= 1;
    cout << cval << endl; cval *= 2;
    cout << cval << endl; cout /= 2.;
    cout << cval << endl;
    system("pause");}
```

C++ standartida kompleks son uchun inkrement va dekrement amallarining bajarilishi ko‘rsatilmagan. Biroq, ularning semantikasi juda aniq, agar cval + = 1 ni yoza olsak, bu cvalning haqiqiy qismini 1 bilan oshirishni anglatadi, unda inkrement amali juda oson ko‘rinadi. Murakkab < double> tipi uchun bu amallarni amalga oshirish va quyidagi dasturni ishga tushirish:

```
int main() {complex< double > cval( 4.0, 1.0 );
cout << cval << endl;
  ++cva1;cout << cval << endl;
}
```

Bu yuqoridagilardan tashqari kompleks sonlari bilash ishlash sinfining ko‘plab, matematik funksiyalar mavjud. Ammo C++ standartiga qarab ayrim amallardan, funksiyalardan foydalanib bo‘lmaydi. Shuning uchun foydalanuvchining o‘z mustaqil kutubxona yaratib ishlashi maqul.

Valarray son sinfi va uning funksiyalari. Yaqin orada C++ da massiv elementlari bilan ishlash va samarali saqlash uchun Valarray konteyneri paydo bo‘ldi. Bu hamma C++ standartlarida bo‘lmasligi ham mumkin. Bu sinf massiv element indekslari bo‘yicha birlashtrish operatorlari va elementlararo matematik amallarni bajaradi.

- Vektor bilan solishtirganda Valarray sinf matematik amallarni bajarishda samaraliroq hisoblanadi.

Valarray sinfning ochiq funksiyalari:

apply() – bu massiv elementlari bo‘yicha bir vaqtda o‘zgarishlarni bajaradi va yangi massiv qaytaradi.

sum() – massivdagi elementlar yig‘indisini qaytaradi.

1.2-dastur. apply () va sum () funksiyalaridan foydalanish.

```
#include "stdafx.h"
#include<iostream> #include<valarray>
using namespace std;
int main(){
       int myints[] = { 5, 25, 55, 85, 115 };
       valarray<int> varr (myints, sizeof(myints)/sizeof(myints[0]));
    valarray<int> varr_one ;
    varr_one = varr.apply([](int x){return x=x+5;});
    cout << "Massivning yangi qiymatllar: ";
       for (int &x: varr_one) cout << '\t' << x;
    cout << endl;
    cout << "Oldingi massiv sum() => ";
    cout << varr.sum() << endl;
    cout << "Yangi massiv sum() => ";
       cout << varr_one.sum() << endl;
       system("pause");
    return 0;
}
```

1.2-dastur.output
Massivning yangi qiymatllar: 10 30 60 90 120 Oldingi massiv sum() => 285 Yangi massiv sum() => 310

Min() va max() fnuksiyalari mos ravishda massiv eng kichik va eng katta elementlari qimatlarini qaytaradi.

```
cout << "min() => ";
       cout << varr.min() << endl;
    cout << "max() => ";
```

shift() – berilgan son qiymat bo‘yicha indeks asosida massivni surib yangi massiv qaytaradi. Agar son musbat bo‘lsa chapga suriladi, manfiy bo‘lsa, o‘nga suriladi.

cshift() - berilgan son qiymat bo‘yicha indeks asosida massivni aylana shaklida surib yangi massiv qaytaradi. Agar son musbat bo‘lsa chapga suriladi, manfiy bo‘lsa, o‘nga suriladi.

1.3-dastur. shift() va cshift()dan foydalanish.

```
#include "stdafx.h"
#include<iostream>
#include<valarray>
using namespace std;
int main(){
        int myints[] = { 5, 25, 55, 85, 115 };
        valarray<int> varr (myints, sizeof(myints)/sizeof(myints[0]));
    valarray<int> varr_one, varr_two;
    cout << "Massivning eski qiymatllar: ";
        for (int &x: varr_one) cout << '\t' << x; cout
    << endl;

        varr_one = varr.shift(-2);
    cout << "Massivning shift(-2) qiymatllar: "; for
        (int &x: varr_one) cout << '\t' << x;
    cout << endl;

        varr_one = varr.shift(2);
    cout << "Massivning shift(2) qiymatllar: "; for
        (int &x: varr_one) cout << '\t' << x;
    cout << endl;

        varr_one = varr.cshift(2);
    cout << "Massivning cshift(2) qiymatllar: "; for
        (int &x: varr_one) cout << '\t' << x;
    cout << endl;
```

```
        varr_one = varr.cshift(-2);
    cout << "Massivning cshift(-2) qiymatllar: "; for
        (int &x: varr_one) cout << '\t' << x;
    cout << endl;

        system("pause");
    return 0; }
```

1.3-dastur.output

```
Massivning eski qiymatllar:
Massivning shift(-2) qiymatllar:    0    0    5    25    55
Massivning shift(2) qiymatllar:     55   85   115  0     0
Massivning cshift(2) qiymatllar:    55   85   115  5     25
Massivning cshift(-2) qiymatllar:   85   115  5    25    55
```

swap () - bir massivni ikkinchsi bir massivga almashtiradi.

```
varr.swap(varr1);
```

Yuqoridagilardan tashqari quyidagi funksiyalari mavjud:

operator=	Massiv qiymatini o‘zlashtirish
operator[]	Massiv indeksi orqali elementiga murojaat
resize	Massiv elemmentlar soni (hajmi) ni qayta aniqlash
size	Massiv elemmentlar soni (hajmi)

Valarray ning standart amallari

```
valarray operator+() const; valarray operator-() const; valarray operator~() const;
valarray<bool> operator!() const;
```

Valarray tiplari ustida amallar:

```
valarray& operator*= (const valarray& rhs); valarray& operator/= (const valarray& rhs);
valarray& operator%= (const valarray& rhs);
valarray& operator+= (const valarray& rhs);
valarray& operator-= (const valarray& rhs); valarray& operator^= (const valarray& rhs);
valarray& operator&= (const valarray& rhs); valarray& operator|= (const valarray& rhs);
valarray& operator<<= (const valarray& rhs);
valarray& operator>>= (const valarray& rhs);
```

Valarray tipi va odatiy oddiy tiplar ustida amallar:

```
valarray& operator*= (const T& val); valarray& operator/= (const T& val); valarray&
operator%= (const T& val); valarray& operator+= (const T& val); valarray& operator-=
(const T& val); valarray& operator^= (const T& val); valarray& operator&= (const T&
val); valarray& operator|= (const T& val); valarray& operator<<= (const T& val);
valarray& operator>>= (const T& val);
```

Yuqorida keltirilgan operatorilar Valarray sinfining funksiyalari hisoblandi.

Valarray tiplari bilan bajariladigan amallarni qayta aniqlash mumkin. Bunda birinchi amal Valarray tipi = Valarray tipi * Valarray tipi, ikkinchi amal Valarray tipi = oddiy tip * Valarray tipi, uchinchi tip Valarray tipi = Valarray tipi* oddiy tip ketma ketligi asosida aniqlangan.

[*] - amalini qayta aniqlash:

```
template <class T> valarray<T> operator* (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<T> operator* (const T& val, const valarray<T>& rhs);
template <class T> valarray<T> operator* (const valarray<T>& lhs, const T& val);
```

[*] - amalini qayta aniqlash:

```
template <class T> valarray<T> operator* (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<T> operator* (const T& val, const valarray<T>& rhs);
template <class T> valarray<T> operator* (const valarray<T>& lhs, const T& val);
```

[/] - amalini qayta aniqlash:

```
template <class T> valarray<T> operator/ (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<T> operator/ (const T& val, const valarray<T>& rhs);
template <class T> valarray<T> operator/ (const valarray<T>& lhs, const T& val);
```

[%] - amalini qayta aniqlash: template <class T> valarray<T> operator% (const valarray<T>& lhs, const valarray<T>& rhs);

```
template <class T> valarray<T> operator% (const T& val, const valarray<T>& rhs);
template <class T> valarray<T> operator% (const valarray<T>& lhs, const T& val);
```

[+] - amalini qayta aniqlash:

```
template <class T> valarray<T> operator+ (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<T> operator+ (const T& val, const valarray<T>& rhs);
template <class T> valarray<T> operator+ (const valarray<T>& lhs, const T& val);
```

[-] - amalini qayta aniqlash:

```
template <class T> valarray<T> operator- (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<T> operator- (const T& val, const valarray<T>& rhs);
template <class T> valarray<T> operator- (const valarray<T>& lhs, const T& val);
```

[^] - amalini qayta aniqlash:

```
template <class T> valarray<T> operator^ (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<T> operator^ (const T& val, const valarray<T>& rhs);
template <class T> valarray<T> operator^ (const valarray<T>& lhs, const T& val);
```

[&] – mantiqiy amalini qayta aniqlash:

```
template <class T> valarray<T> operator& (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<T> operator& (const T& val, const valarray<T>& rhs);
template <class T> valarray<T> operator& (const valarray<T>& lhs, const T& val);
```

[|] – mantiqiy amalini qayta aniqlash:

```
template <class T> valarray<T> operator| (const valarray<T>& lhs, const valarray<T>& rhs);
```

```
template <class T> valarray<T> operator| (const T& val, const valarray<T>& rhs);
template <class T> valarray<T> operator| (const valarray<T>& lhs, const T& val);
```

[<<] – chapga surish amalini qayta aniqlash:

```
template <class T> valarray<T> operator<< (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<T> operator<< (const T& val, const valarray<T>& rhs);
template <class T> valarray<T> operator<< (const valarray<T>& lhs, const T& val);
```

[>>] – o‘ngga surish amalini qayta aniqlash:

```
template <class T> valarray<T> operator>> (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<T> operator>> (const T& val, const valarray<T>& rhs);
template <class T> valarray<T> operator>> (const valarray<T>& lhs, const T& val);
```

[&&] – tezkor mantiqiy amalini qayta aniqlash:

```
template <class T> valarray<bool> operator&& (const valarray<T>& lhs, const valarray<T>&
rhs);
template <class T> valarray<bool> operator&& (const T& val, const valarray<T>& rhs);
template <class T> valarray<bool> operator&& (const valarray<T>& lhs, const T& val);
```

[||] – tezkor mantiqiy amalini qayta aniqlash:

```
template <class T> valarray<bool> operator|| (const valarray<T>& lhs, const valarray<T>&
rhs);
template <class T> valarray<bool> operator|| (const T& val, const valarray<T>& rhs);
template <class T> valarray<bool> operator|| (const valarray<T>& lhs, const T& val);
```

[==] – mulohaza amalini qayta aniqlash:

```
template <class T> valarray<bool> operator== (const valarray<T>& lhs, const valarray<T>&
rhs);
template <class T> valarray<bool> operator== (const T& val, const valarray<T>& rhs);
template <class T> valarray<bool> operator== (const valarray<T>& lhs, const T& val);
```

[!=] – mulohaza amalini qayta aniqlash:

```
template <class T> valarray<bool> operator!= (const valarray<T>& lhs, const valarray<T>&
rhs);
template <class T> valarray<bool> operator!= (const T& val, const valarray<T>& rhs);
template <class T> valarray<bool> operator!= (const valarray<T>& lhs, const T& val);
```

[<] – mulohaza amalini qayta aniqlash:

```
template <class T> valarray<bool> operator< (const valarray<T>& lhs, const valarray<T>&
rhs);
template <class T> valarray<bool> operator< (const T& val, const valarray<T>& rhs);
```

template <class T> valarray<bool> operator< (const valarray<T>& lhs, const T& val);

[>] – mulohaza amalini qayta aniqlash:

template <class T> valarray<bool> operator> (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<bool> operator> (const T& val, const valarray<T>& rhs);
template <class T> valarray<bool> operator> (const valarray<T>& lhs, const T& val);

[<=] – mulohaza amalini qayta aniqlash:

template <class T> valarray<bool> operator<= (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<bool> operator<= (const T& val, const valarray<T>& rhs);
template <class T> valarray<bool> operator<= (const valarray<T>& lhs, const T& val);

[>=] – mulohaza amalini qayta aniqlash:

template <class T> valarray<bool> operator>= (const valarray<T>& lhs, const valarray<T>& rhs);
template <class T> valarray<bool> operator>= (const T& val, const valarray<T>& rhs);
template <class T> valarray<bool> operator>= (const valarray<T>& lhs, const T& val);

Slice sinfi. Bu - xizmatchi sinf bo‘lib, valarray sinfning merosxo‘ridir va bir o‘lchovli to‘plam ostilarni yaratishda ishlatiladi. Agar valarray massivni ikki o‘lchovli matritsa sifatida qarasak, undan bir o‘lchovli vektorni hosil qilish mumkin.

Bu sinf slice_array tipdagi ob‘yektni tavsiflovchi parametrlarni saqlaydi. Agar to‘plam osti bilvosita yaratilgan valarray sinf tipida bo‘lsa, sinf ob‘yekti

valarray ob‘yekti uchun argument sifatida ko‘rsatiladi. To‘plam ostida saqlanadigan qiymatlar valarray tipini oladi va quyidagi xususiyatlarni o‘z ichiga oladi:

valarraydagi boshlang‘ich indeks.

massivning umumiy uzunligi va undagi elementlar soni;

valarray elementlarning ketma-ket indekslari orasidagi qadam yoki masofani aniqlash.

Agar qism to‘plam yordamida aniqlangan massiv doimiy valarrayning kichik bo‘lagi bo‘lsa, bu massiv yangi valarraydir. Doimiy valarray tipidagi to‘plam bir qism to‘plam yordamida aniqlangan bo‘lsa, dastlabki valarray uchun mos havolalar semantikasidir. O‘zgarmas bo‘lgan valarray lar uchun baholash mexanizmi vaqt va xotira sarflaydi.

Qism to‘plam asosida aniqlangan bo‘lsa va farqlansa, barcha elementlariga murojaat bo‘lsa, valarray tipidagi massivlar uchun amallar bajarilishi kafolatlanadi. Slice sinfi uchun

Slice konstruktori ishlatiladi. Bir-biridan bir xil masofada bo‘lgan va belgilangan element bilan boshlanadigan, bir necha elementlardan iborat valarray bir to‘plam ostisini yaratadi.

Slice sinfining funksiyalari quyidagilar:

size() – valarray to‘plam ostining elementlari sonini aniqlaydi.

start() – valarray to‘plam ostining boshlang‘ich indeksini qaytaradi **stride()** - valarray to‘plam ostining elementlari orasidagi masofani topadi. 1.3-dastur. Slice sinfidan foydalanish.

```
#include "stdafx.h"
#include <valarray>
#include <iostream>
using namespace std;
int main( ){int i;
    size_t sizeVA, sizeVAR;
    size_t startVAR, strideVAR;
    valarray<int> va(28), vaResult;
    for ( i = 0 ; i < 28 ; i += 1 )  va [ i ] =  (rand()%100)+1;
    cout << "valarray elementlari: \n ( ";
    for ( i = 0 ; i < 28 ; i++ )
          cout << va[i] << '\t'; cout << ")." << endl;
```

```
sizeVA = va.size(); cout << "valarray elementlar soni: "
        << sizeVA << "." << endl << endl;
// slice konstruktori
slice vaSlice( 0, 6, 4); vaResult = va[vaSlice];

cout << "valarray to‘plam ostisi vaResult = "
        << "va[slice( 0, 6, 4)] =\n ( ";

 for ( i = 0 ; i < vaSlice.size() ; i++ )
  cout << vaResult [ i ] << " ";
   cout << ")." << endl;
  sizeVAR = vaSlice.size( );

cout << "to‘plam ostining elementlar soni: "
        << sizeVAR << "." << endl;
startVAR = vaSlice.start( );
cout << "to‘plam ostining birinchi element indeksi: "
        << startVAR << "." << endl;
strideVAR = vaSlice.stride( );
cout << "to‘plam ostining elementlari orasidagi masofa: "
        << strideVAR << "." << endl;
system("pause");}
```

1.3-dastur. output

```
valarray elementlari:
 ( 42      68      35      1       70      25      79      59      63
65      6       46      82      28      62
92      96      43      28      37      92      5       3       54
93      83      22      17      ).
valarray elementlar soni: 28.
valarray to‘plam ostisi vaResult = va[slice( 0, 6, 4)] = ( 42 70 63 82 96 92 ).
to‘plam ostining elementlar soni: 6.
to‘plam ostining birinchi element indeksi: 0. to‘plam ostining
elementlari orasidagi masofa: 4.
```

Dasturga sinfga berilgan ta‘rif asosida uning konstruktori 3 ta parametr qabul qilar ekan. Birinchi paramert belgilangan elementlar indeksi, ikkinchisi elementlar soni, elementlar indeksi orasidagi masofa.

Gslice sinfi. Xizmatchi sinf bo‘lib, valarray ning ko‘p o‘lchamli to‘plam ostilari bilan ishlashga mo‘ljallangan. Agar valarray massivda ko‘p o‘lchamli to‘plam sifatida qaralsa, Gslice sinfi undan ko‘p o‘lchamli vektorni oladi.

Sinf gslice_array tip bilan tavsiflanuvchi paramertlarni saqlaydi. agar valarray**<Type> tipidagi ob‘yekt uchun argument sifatida class gslice ob‘yekti berilgan bo‘lsa, Valarray to‘plam ostisi bilvosita qurilgan hisoblanadi. ** to‘plam osti elementlari bo‘lib, Valarray bazaviy sinfdan olingan va 3 ta paramertni qabul qiladi:

- Boshlang‘ich indeks;
- Vektor uzunligi valarray<size_t> sinf;
- Vektor qadami (elementlari orasidagi mosofa) valarray<size_t> sinf; Vektorlarning uzinligi bir xil bo‘lishi kerak.

Agar to‘plam gslice bilan aniqlangan va valarray ning to‘plam ostisi bo‘lsa, gslice yangi valarray hisoblanadi. O‘zgarmas bo‘lgan Valarray uchun baholash mexanizmi vaqt va xotirani tejaydi.

boshlang‘ich va oxirgi to‘plamlar gslice bilan aniqlangan bo‘lsa, farqli, barcha indekslariga murojaat bo‘lsa, Valarray to‘plam ustidagi barcha amallar kafolatlanadi.

gslice sinfi uchun gslice konstruktori ishlatiladi. Belgilangan element bilan boshlanadigan, bir necha elementlardan iborat valarray to‘plam ostisini yaratadi.

gslice sinfining funksiyalari quyidagilar:

size() – valarray to‘plam ostining umumiy massiv elementlari sonini aniqlaydi.

start() – valarray to‘plam ostining umumiy boshlang‘ich indeksini qaytaradi

stride() - valarray to‘plam ostining umumiy elementlari orasidagi masofani topadi.

Gslice konstruktori:

```
gslice(
    size_t _StartIndex,
    const valarray<size_t>& _LenArray, const
    valarray<size_t>& _IncArray);
```

Parametrlari:

_StartIndex – to‘plam ositdagi birinchi elemeni indeksi

_LenArray – har bir to‘plam osti uchun massiv elementlari soni

_IncArray - har bir to‘plam osti uchun massiv elementlari orasidagi masofa (qadam);

Standart konstruktor boshlang‘ich indeksi nol uzunlikdagi vektorlar va vektorlari uchun uzunlik va qadam nol qiymatni saqlaydi. Ikkinchi konstruktor boshlang‘ich indeksi uchun _startindex saqlaydi, qator uzunligi uchun _lenarray va qadam uchun _incarray qiymatlarni saqlaydi.

gslice – bir nechta qismdan tashkil topgan valarray uchun to‘plam osti valarrayni aniqlaydi va hammasi belgilangan bir element indeksidan boshlanadi. gslice i slice ning farqlari birgina bir nechta qism valarray lar bilan ishlashdadir.

1.4-dastur. gslice sinfidan foydalanish.

```
#include "stdafx.h"
#include <valarray>
#include <iostream>
using namespace std;
int main(){int i;
    valarray<int> va ( 20 ), vaResult; for (
    i = 0 ; i < 20 ; i+=1 )
        va [ i ] =  i+1;

    cout << "valarray massiv elementlari: " << endl << "(";
     for ( i = 0 ; i < 20 ; i++ ) cout << '\t' << va [ i ];
    cout << " )" << endl;
    valarray<size_t> Len (2), Stride(2); Len [0] = 6;
    Len [1] = 4;Stride [0] = 2; Stride [1] = 4;
    gslice vaGSlice (0, Len, Stride );
    vaResult = va[vaGSlice];
cout << " vaGSlice asosida vaResult elementlari:" << endl<< "va[vaGSlice] = (";
    for ( i = 0 ; i < 10 ; i++ ) cout << '\t' << vaResult [ i ]; cout << ")" << endl;
```

```
    const valarray <size_t> sizeGS = vaGSlice.size ( );
        cout << "vaResult elementlar soni:"<< " vaGSlice.size ( ) = ( ";
            for ( i = 0 ; i < 2 ; i++ ) cout <<sizeGS[ i ] << '\t'; cout << ")." << endl;
    size_t vaGSstart = vaGSlice.start ( );
            cout << "vaResult uchun birinchi element indeksi: "<< vaGSstart << "." << endl;
            const valarray <size_t> strideGS = vaGSlice.stride ( );
            cout << "vaResult ychun qadamlar:"<< " vaGSlice.stride ( ) = ( ";
             for ( i = 0 ; i < 2 ; i++ ) cout << strideGS[ i ] << '\t'; cout << ")." << endl;
```

```
    system("pause");
    }
```

1.4-dastur. Output

```
valarray massiv elementlari:
(       1       2       3       4       5       6       7
        8
9       10      11      12      13      14
15      16      17      18      19      20 )
  vaGSlice asosida vaResult elementlari:
va[vaGSlice] = (        1       5       9       13      3       7
11      15      5       9)
vaResult elementlar soni: vaGSlice.size ( ) = ( 64). vaResult uchun birinchi element
indeksi: 0.
    vaResult ychun qadamlar: vaGSlice.stride ( ) = ( 2      4      ).
```

Dastur tahlili bilan mutaqil shug‘ullaning.

Slice_array sinfi. Ichki yordamchi shablon sinf hisoblanadi. Valarray qismlari bilan aniqlangan to‘plam osti massivlar uchun amallarni qo‘llab quvvatlaydi. Sinf sintaktiki quyidagicha:

```
template <class Type>
class slice_array : public slice { public:
    typedef Type value_type;
    void operator=(const valarray<Type>& x) const; void
    operator=(const Type& x) const;
    void operator*=(const valarray<Type>& x) const;
    void operator/=(const valarray<Type>& x) const;
    void operator%=(const valarray<Type>& x) const; void
    operator+=(const valarray<Type>& x) const; void operator-
    =(const valarray<Type>& x) const; void operator^=(const
    valarray<Type>& x) const; void operator&=(const
    valarray<Type>& x) const; void operator|=(const
    valarray<Type>& x) const; void operator<<=(const
    valarray<Type>& x) const; void operator>>=(const
    valarray<Type>& x) const;
}
```

Bu sinf ob‘yekti - valarray<Type> ob‘yektidagi elementlar ketmk ketligi bilan yoziladigan slice sinf ob‘yekti bilan birgalikda valarray <Type> sinfiga havola saqlaydigan ob‘yektlarni tasniflaydi.

Shaboln sinf muayyan valarray amallari tomonidan bilvosita yaratilgan va dasturda bevosita foydalanish mumkin emas. Ichki yordamchi shablon sinfi quyidagi sintaktik bilan ishlatiladi:

```
slice_array< Type> valarray< Type:: operator[] ( slice);
```

Slice_array < Type > ob‘yekti faqat [SL] formatli va SL slice valarray ifodasini yozish orqali yaratiladi. slice_array sinfining funksiyalari valarray< Type

> uchun belgilangan mos funksiyalari bilan bir xil ishlatiladi, faqat tanlangan elementlar ketma-ketligi ta‘sir ko‘rsatadi. Slice_array sinf tomonidan nazorat natijasida konstruktorga uch parametrlar belgilanadi: massivning birinchi element indeksi, elementlar soni va elementlar orasidagi masofa.

gslice_array sinfi. Ichki yordamchi shablon sinf hisoblanadi. Valarray qismlari bilan aniqlangan to‘plam osti massivlar uchun amallarni qo‘llab quvvatlaydi. Sinf sintaktiki quyidagicha:

```
template <class Type>
class gslice_array : public gsplice { public:
        typedef Type value_type;
        void operator=(const valarray<Type>& x) const; void
        operator=(const Type& x) const;
        void operator*=(const valarray<Type>& x) const;
```

```
        void operator/=(const valarray<Type>& x) const; void
        operator%=(const valarray<Type>& x) const; void
        operator+=(const valarray<Type>& x) const; void operator-
        =(const valarray<Type>& x) const; void operator^=(const
        valarray<Type>& x) const; void operator&=(const
        valarray<Type>& x) const; void operator|=(const
        valarray<Type>& x) const; void operator<<=(const
        valarray<Type>& x) const; void operator>>=(const
        valarray<Type>& x) const;
}
```

Bu sinf ob‘yekti - valarray<Type> ob‘yektidagi massiv elementlar ketma ketligi bilan yoziladigan gslice sinf ob‘yekti bilan birgalikda valarray <Type> sinfiga havola saqlaydigan ob‘yektlarni tasniflaydi.

Shaboln sinf muayyan valarray amallari tomonidan bilvosita yaratilgan va dasturda bevosita foydalanish mumkin emas. Ichki yordamchi shablon sinfi quyidagi sintaksis bilan ishlatiladi:

```
gslice_array< Type> valarray< Type>:: operator[] ( constgslice& )
```

gslice_array < Type > ob‘yekti faqat [GL] formatli va GL gslice valarray ifodasini yozish orqali yaratiladi. slice_array sinfining funksiyalari valarray<Type> uchun belgilangan mos funksiyalari bilan bir xil ishlatiladi, faqat tanlangan elementlar ketma-ketligi ta‘sir ko‘rsatadi. Slice_array sinf tomonidan nazorat natijasida konstruktorga uch parametrlar belgilanadi: massivning birinchi element indeksi, elementlar soni va elementlar orasidagi masofa. Dastur fragmentiga qarang.

```
const size_t lv[] = {2, 3};
const size_t dv[] = {7, 2};
const valarray<size_t> len(lv, 2), str(dv, 2);
```

indirect_array sinfi. Ichki yordamchi shablon sinf hisoblanadi. Bazaviy valarray sinf elementlari asosidagi to‘plam osti bo‘lib, belgilangan to‘plam osti o‘rtasida amallarni ta‘minlash orqali valarray ob‘yektlarining to‘plam osti ob‘yektlar bo‘lgan ob‘yektlarni qo‘llab-quvvatlaydigan yordamchi sinf.

Bu sinf ob‘yekti - valarray<Type> ob‘yektidagi elementlar ketma ketligi bilan yoziladigan valarray<size_t> sinf ob‘yekti xa bilan birgalikda valarray <Type> sinfiga havola saqlaydigan ob‘yektlarni tasniflaydi.

Indirect_array<type> ob‘yekti faqat va[xa] shaklining ifodasini yozish orqali yaratiladi. Tanlangan elementlar ketma ketligi asosida Indirect_array sinf funksiyalari valarray uchun belgilangan tegishli funksiyalar bir xil ishlaydi. Ketma-ketlik XA dan iborat bo‘ladi. VA ichidagi XA[I] indeksiga akslatirilganda I elementlar o‘lchami bo‘ladi.

1.5-dastur. indirect_array sinfidan foydalanish.

```
#include      "stdafx.h"
#include    <valarray>
#include <iostream>

int main( )
{using namespace std; int i;
```

```
    valarray<int> va ( 10 );
for ( i = 0 ; i < 10 ; i += 2 ) va [ i ] =  i;
    for ( i = 1 ; i < 10 ; i += 2 ) va [ i ] =  -1;
    cout << "Valarray maasiv elementlari:  ( ";
    for ( i = 0 ; i < 10 ; i++ ) cout << va [ i ] << '\t';
       cout << ")." << endl;
     valarray<size_t> indx(5); int j = 0; for (int i = 1; i < 10; i+=2)
        {indx [j] = i;
                //cout << indx[j] << '\t'; j++;
        }
        cout << endl;
    va[indx] = 10;
    cout << "O‘zgartirilgan valarray elementlari:  ( "; for (i =
       0 ; i < 10 ; i++ )
          cout << va [ i ] << '\t'; cout
    << ")." << endl; system("pause");
                }
```

1.5-dastur.output

```
Valarray maasiv elementlari:  ( 0        -1      2       -1      4       -1
6       -1      8       -1      ).

O‘zgartirilgan valarray elementlari:  ( 0        10      2       10      4
10      6       10      8       10 ).
```

mask_array sinfi. Ichki yordamchi shablon sinf hisoblanadi. Bazaviy valarray sinf elementlari asosidagi to‘plam osti bo‘lib, belgilangan to‘plam osti uchun mantiqiy amallarni ta‘minlash orqali valarray ob‘yektlarining to‘plam osti ob‘yektlar bo‘lgan ob‘yektlarni qo‘llab-quvvatlaydigan yordamchi sinf.

Bu sinf ob‘yekti - valarray<Type> ob‘yektidagi elementlar ketma ketligi bilan yoziladigan valarray <bool> sinf ob‘yekti ba bilan birgalikda valarray <Type> sinfiga havola saqlaydigan ob‘yektlarni tasniflaydi.

Ketma-ketlik ba.size elementlardan ko‘pmas. J element faqat ba[J] ga chin (true) o‘rnatilgandagina faol bo‘ladi. Ketma-ketlikda elementni soni ba elementlarga teng bo‘ladi. Agar i - ba da eng kichik haqiqiy chin (true) elementning indeksi bo‘lsa, u holda tanlangan ketma-ketlikda a[i] ning qiymati nolga teng.

1.6-dastur. mask_array sinfi sinfidan foydalanish.

```
#include   "stdafx.h"
#include  <valarray>
#include <iostream>
using namespace std;
int main( ){ int i;
    valarray<int>  va  (  10  ); for (
    i = 0 ; i < 10 ; i++ )
        va [ i ] =  i*3;
    cout << "valarray elementlari:   ( "; for ( i = 0 ; i < 10 ; i++ )
            cout << va [ i ] << '\t'; cout << ")." << endl;
    va [(va > 20 || va < 10) ] = 10;
    cout << "o‘zgartirikgan valarray elementlari:   ( "; for ( i = 0 ; i < 10 ; i++ )
cout << va [ i ] << '\t'; cout << ")." << endl; system("pause");}
```

1.6-dastur.Output

```
valarray elementlari:
 ( 0      3      6      9      12     15     18     21     24     27     ).
o‘zgartirikgan valarray elementlari:
 ( 10     10     10     10     12     15     18     10     10     10     ).
```

In sinfi[2]. Bu sinf interval matematikaning amallarini bajarishga yo‘naltirilgan bo‘lib, C++ ning eski standarti asosida keltiriladi va taqribiy hisoblashlar uchun yaratilgan. Sinf quyidagicha aniqlangan:

```
class In {private:
     public: double _a, a_; double m,n;
    In() {_a=a_=0;} In(double a, double b) {if (a>b){
    a=a+b; b=a-b; a=a-b;
          } _a=a, a_=b; }
```

```
//kiritish va chqarish...
friend istream& operator>>(istream & S, In & d) throw()
    {
     S>>d._a>>d.a_;
    return S;
    }
friend ostream& operator << (ostream & S, In &d) throw()
   { S<<"["<<d._a<<","<<d.a_<<"]\t";
    return S;}
//  In a = 9;
inline In & In::operator= (const double & a)
{ _a = a_ =a; return *this; }
// - ishorasi
friend In operator -(In &a) throw()
{ return In(-a.a_,-a._a);}
```

Sinfda 2 ta konstruktor bo‘lib, birinchisi interval son 0 ga tenglashtiradi, ikkinchisi mos parametrlarga asoslanib interval sonni yaratidi.

In kutubxonasi funksiyalari:

class In – interval sinf.

intervalning _a, - quyi chegarasi a_ - yuqori chegarasi; In()

{_a=a_=0;} Intervalning qiymatini 0 ga tenglashtirish;

In(double a, double b) Intervalga qiymat berish. Agar _a> a_ bo‘lsa, qiymatlar o‘rni almashadi.

1) [>>] - interval sonni o‘qish :: 1 2
2) [<<] - interval sonni chqarish, masalan: [a,b]
3) [=] - interval songa oddiy sonni tenglashtirish, masalan: In a; a=1;
4) [-]- interval sonni ishorasini almashtirish, masalan: In a; b=-a;
5) [+] - interval qo‘shish, masalan: [a]+[b]; [a]+b; a+[b];
6) [-] - interval ayirish, masalan: [a]-[b]; [a]-b; a-[b];
7) [*] - interval ko‘paytirish, masalan: [a]*[b]; [a]*b; a*[b];
8) [/] - interval bo‘lish, amallari [a]/[b]; [a]/b; a/[b];
9) [+=]- operator interval qo‘shish [a]+=[b] [a]+=b
10) [-=]- operator interval ayirish [a]-=[b] [a]-=b
11) [*=]- operator interval ko‘paytirish [a]*=[b] [a]*=b
12) [/=]- operator interval bo‘lish [a]/=[b] [a]/=b
13) [|] - operator interval yoki
14) [&]- operator interval va
15) [|=] -operator interval yoki
16) [&=]- operator interval va

17) [==]- operator interval tenglik
18) [!=]- operator interval tengmas
19) [<=]- operator interval kichik yoki teng
20) [>=]- operator interval katta yoki teng
21) [<]- operator interval kichik
22) [>]- operator interval katta
23) [!]- operator interval 0 intervalga tegishligini aniqlaydi.
24) GetInf funksiyasi - intervalning kichik qiymati olish
25) GetSup funksiyasi - intervalning katta qiymati olish
26) SetInf funksiyasi -intervalning kichik qiymatini o‘zgartirish
27) SetSup funksiyasi -intervalning katta qiymatini o‘zgartirish
28) IsEmpty funksiyasi - intervalning xatoligi, masalan: [2,1] – xato;
29) abs funksiyasi -interval moduli
30) Mid funksiyasi -intervalning markazi
31) Rad funksiyasi -intervalning radusi
32) Wid funksiyasi -intervalning uzunligi 2*Rad
33) MaxIn funksiyasi -intervalning kattasi modulda
34) MinIn funksiyasi -intervalning kichigi modulda
35) Sing funksiyasi -intervalning ishorasi -1,0,1 ko‘rinishda
36) Indefine kutubxonasi

1) tab - "\t" dir.
2) NewLine - "\n" dir.
3) ZERO - 0.0000000 qiymat
4) real - double tipidir

ERROR kutubxonasi

1) DIV_BY_ZERO_ERROR
2) ERROR_INTERVAL_EMPTY_INTERVAL
3) ERROR_INTERVAL_SING_INTERVAL
4) ERROR_INTERVAL

INTMATH kutubxonasi

1) min funksiyasi – ikki haqiqiy sonning kichigi
2) max funksiyasi – ikki haqiqiy sonning kattasi
3) abs funksiyasi – haqiqiy sonning moduli

INMATH interval matematika kutubxonasida standart qiymatlar.

1) #define E 2.7182818284590452354
2) #define LOG2E 1.4426950408889634074
3) #define LOG10E 0.43429448190325182765
4) #define LN2 0.69314718055994530942
5) #define LN10 2.30258509299404568402
6) #define PI 3.14159265358979323846
7) #define PI_2 1.57079632679489661923

8) #define PI_4 0.78539816339744830962

9) #define PI1_ 0.31830988618379067154

10) #define PI2_ 0.63661977236758134308

11) #define SQRTPI2 1.12837916709551257390

12) #define SQRT2 1.41421356237309504880

13) #define SQRT1_20.70710678118654752440

14) sin – sin(x);
15) cos – cos(x);
16) tan – tan(x);
17) sinh – sinh(x);
18) cosh – cos(x);
19) tanh – tanh(x);
20) asin – asin(x);
21) acos – acos(x);
22) atan – atan(x);
23) exp – exp(x);
24) log – ln(a);
25) log10 – log(x);
26) pow(In &a, int n) funksiyasi – a interval sonning n darajasi
27) pow(In &a, double n)
28) pow(const In &b, In &a)
29) pow(const double b, In &a)
30) pow(In &b, In &a)
31) sqrt (In &a) funksiyasi intarval sondan kvadrat ildiz olish
32) sqrt(In &a, double n) funksiyasi intarval sondan n-chi darajali ildiz olish
33) ceil funksiyasi interval sonni yaxlitlash
34) floor funksiyasi interval sonning katta yaxliylash
35) round funksiyasi interval sonni kichikka yaxlitlash

INDEFMURAKKAB hosila olish uchun kutubxona class

def_murakkab murakkab hosilalar sinfi…

1) div_Def_uv funksiyasi - u/v ni hisoblash;
2) mult_Def_uv funksiyasi - u*v ni hisoblash;
3) add_Def_uv funksiyasi - u+v ni hisoblash;
4) sub_Def_uv funksiyasi - u-v ni hisoblash;
5) pow_Def_u funksiyasi - f(x)ning d darajasining hosilasi;
1) f_sqrt funksiyasi - ildiz osti f(x)ning hosilasi
class def_fx x ning n darajasinig hosilasi sinfi…
2) fx(In x) funksiyasi - x ning 1 darajasini qaytaradi;
3) fx(In x, int n) funksiyasi - x ning n darajasini qaytaradi;
4) dfx(In x, int n) funksiyasi - x ning n darajasini hosilasi;
5) dfx(In x, int n, int m) funksiyasi - x ning n darajasini m tartibli hosilasi; class

def_func – matematik funksiyalardan hosila;

1) d_exp(In x,int m) funksiyasi – exp(x) ning m tartibli hosilasi;
2) d_exp(In x, In k, int m) funksiyasi – exp(kx) ning m tartibli hosilasi;
3) const_a(In b) funksiyasi – o‘zgarmas interval sonning hosilasi;
4) f_ax(In a, In x,int m) funksiyasi- a^x ning m tartibli hosilasi;
5) f_ax(In a, In x,int m, int k) funksiyasi-a^kx ning m tartibli hosilasi;
6) f_sin(In x,int m) funksiyasi-sin ning m tartibli hosilasi;
7) f_cos(In x,int m) funksiyasi- cos ning m tartibli hosilasi;
8) f_sqrt(float x, int m) funksiyasi-sqrt ning m tartibli hosilasi;
9) f_ln(In x, int m) funksiyasi- ln(x) ning m tartibli hosilasi;
10) f_log(float a, In x, int m) funksiyasi-log(a,x) ning m tartibli hosilasi;
11) f_tan(In x) funksiyasi-tg(x) ning 1 tartibli hosilasi;
12) f_ctan(In x) funksiyasi-ctg(x) ning 1 tartibli hosilasi;
13) f_acos(In x) funksiyasi-arccos(x) ning 1 tartibli hosilasi;
14) f_asin(In x) funksiyasi-arcsin(x) ning 1 tartibli hosilasi;
15) f_atan(In x) funksiyasi-arctan(x) ning 1 tartibli hosilasi;
16) f_actan(In x) funksiyasi-arcctan(x) ning 1 tartibli hosilasi;
17) f_cosh(In x) funksiyasi-giperbolik cos(x) ning 1 tartibli hosilasi;
18) f_sinh(In x) funksiyasi-giperbolik sin(x) ning 1 tartibli hosilasi;
19) f_tanh(In x) funksiyasi-giperbolik tan(x) ning 1 tartibli hosilasi;
20) f_ctanh(In x) funksiyasi-giperbolik ctan(x) ning 1 tartibli hosilasi;

class def_arg_func argumentida f(x) bo‘lgan funksiyalar hosilasi sinf;

1) d_exp(func f, func df, In x) funksiyasi-exp(f(x)) ning hosilasi;
2) f_ax(In x, double a, func f, func df) funksiyasi-a^f(x) ning hosilasi;
3) f_ax(In x, In a, func f, func df) funksiyasi-a^f(x) ning hosilasi;
4) f_sin(In x, func f, func df) funksiyasi-sin(f(x)) ning hosilasi;
5) f_cos(In x, func f, func df) funksiyasi-cos(f(x)) ning hosilasi;
6) f_sqrt(In x, func f, func df) funksiyasi-sqrt(f(x)) ning hosilasi;
7) f_ln(In x, func f, func df) funksiyasi-ln(f(x)) ning hosilasi;
8) f_log(float a, In x, func f, func df) funksiyasi-log(a,(f(x)) ning hosilasi;
9) f_log(In a, In x, func f, func df) funksiyasi-log(a,(f(x)) ning hosilasi;
10) f_tan(In x, func f, func df) funksiyasi-tan(f(x)) ning hosilasi;
11) f_ctan(In x, func f, func df) funksiyasi-ctan(f(x)) ning hosilasi;
12) f_acos(In x,func f, func df) funksiyasi-acos(f(x)) ning hosilasi;
13) f_asin(In x,func f, func df) funksiyasi-asin(f(x)) ning hosilasi;
14) f_atan(In x, func f, func df) funksiyasi-atan(f(x)) ning hosilasi;
15) f_actan(In x,func f, func df) funksiyasi- actan(f(x)) ning hosilasi;
16) f_cosh(In x,func f, func df) funksiyasi-cosh(f(x)) ning hosilasi;
17) f_sinh(In x, func f, func df) funksiyasi-sinh(f(x)) ning hosilasi;
18) f_tanh(In x, func f, func df) funksiyasi-tanh(f(x)) ning hosilasi;
19) f_ctanh(In x, func f, func df) funksiyasi-ctanh(f(x)) ning hosilasi;
20) pow_arg_func(func f, func df, func dx, In x, int d) funksiyasi- (f(x)^d) ning hosilasi;
21) f_msqrt(func f, func df, In x, float m) funksiyasi- m_sqrt(f(x)) (1/m) ning hosilasi;

INGAUSSK_H – Integralni Gauss usulida hisoblash uchun koyfisentlar uchun kutubxona

1) P – ko‘p hadni hisoblash;
2) dP – ko‘p hadni hosilasini hisoblash

3) A – Koyfisentlarni hisoblash;
4) x0 – x uchun boshlang‘ish yaqinlashish;
5) xeps – x ni eps aniqlikda olish uchun;
6) convert_x – olingan x larni a,b oraliqqa convertlash;
7) convert_A - olingan A larni a,b oraliqqa convertlash;

INTEGRALIN_H interval integrallash uchun kutubxonaintegral_MTT1 – markaziy to‘g‘ri to‘rtburchak usuli;

1) integral_Trap funksiyasi– trapetsiya usuli;
2) integral_Simson funksiyasi – Simson usuli;
3) integral_gauss1 funksiyasi – Gauss usuli;
4) integral_gauss2 funksiyasi - Gauss usuli ikki karrali INGAUSS_H

CHATSni gauss usuli bilan yechish uchun kutubxona N = 3 uchun

1) PrintArray – massivni chiqarish;
2) PrintVector – vektorni chiqarish;
3) ScanArray – massivni kiritish;
4) ScanVectorB – vektorni kiritish;
5) SwapColumns – ustunlarni almashtirish;
6) SwapLines – qatorlarni almashtirish;
7) Destroy – quyi uchburchakni)ga aylantirish;
8) Solution – yechimni topish
9) F – yechimning aniqligini aniqlash;
10) GAUSS – birlashgan Gauss usulini qo‘llash;

1.7-dastur. Chiziqli algebraik tenglamalar sistemasini yechish dasturi.

```
#include <iostream.h>
#include <conio.h>
#include <e:\mylibs\interval.h>
  int main(){ int n;
cout<<"n ni kirit:>>"; cin>>n;
In a[n][n],b[n];
//_________A ni kirit:_______________
for(int i=0; i<n; i++)
  for(int j=0; j<n; j++) a[i][j]=KiritIn();
//_________b ni kirit:_______________
for(int j=0; j<n; j++)
        b[j]=KiritIn();
//_________to‘g‘ri yo‘l_______________
In r[n][n];
for(int j=0; j<n-1; j++) for(int
  i=j+1; i<n; i++){
```

```
        r[i][j] =DivIn(a[i][j],a[j][j]); for(int k=j;
            k<n; k++)
          a[i][k]=SubIn(a[i][k],MultIn(r[i][j],a[j][k]));
              b[i]=SubIn(b[i],MultIn(r[i][j],b[j]));  }
//_________a[i][i]:__________________
  for(int j=0; j<n; j++){
       b[j] =DivIn(b[j],a[j][j]); for(int
      i=0; i<n; i++)
         r[i][j] =DivIn(a[i][j],a[i][i]); }
//___________a[][[]_________convert_
  for(int j=0; j<n; j++)
   for(int i=0; i<n; i++) a[i][j] =
                    r[i][j];
//____out ___
```

```
      for(int i=0; i<n; i++) {
       for(int j=0; j<n; j++)
                { cout<<"[";chopIn(a[i][j]); cout<<"]\t"; }
              cout<<"= ["; chopIn(b[i]); cout<<"]\t";
            cout<<endl;}

  In x[n],s;
   for(int i=n-1; i>-1; i-- )
    {  s. _a=0; s. a_=0;
        for( int j=n-1; j>i; j--)
            s=AddIn(s,MultIn(a[i][j],x[j]));
             x[i]=DivIn(SubIn(b[i],s),a[i][i]);
       }
    for(int i=0; i<n; i++){ cout<<"[";chopIn(x[i]);
        cout<<"]"<<endl;} getch(); return 0;
}
```

1.7-dastur. Ikki karrali integralni gauss usuli bilan taqribiy hisoblash dasturi.

```
#include
<iostream.h>
#include
<conio.h>
#include
<math.h>
double P(double x,
   float n){ if (n==0)
   return 1;
   if (n==1) return x;
   return  (((2*n-1)/n)*x*P(x,n-1))-(((n-1)/n)*P(x,n-2));
   }
double dP(double x, float n){
   return   ((n/(1-x*x))*P(x,n-1))-(x*P(x,n));
   }

double A(double x, float n){
   return   2./((dP(x,n)*dP(x,n))*(1-x*x));}
double x0(int i, double n){ return -cos(M_PI*((4*(i+1)-1)/(4*n+2)));  }
double xeps(double x, double n, double
   eps){ double temp=x-P(x,n)/dP(x,n);
   while (fabs(temp-
       x)>eps) { x=temp;
```

```
        temp=x-P(x,n)/dP(x,n);
        }
        return x;}
double convert_x(double xx, float a, float b){ return
        (0.5*(b+a))+(0.5*(b-a)*xx);}
double convert_A(double AA, float a, float b){
        return ((b-a)/2)*AA;}
int main(){ float
   n,eps;
 freopen("InputData.in","r",stdin); cin>>n;
    cin>>eps;
  float a,b; cin>>a>>b;
  float x[(int)n];
 freopen("OutputData[a,b].out","w",stdout);
 cout<<"*** --- ---- n="<<n<<"
 ------------------------------------------------ ***"<<endl
 ;
 cout<<"*** -- a="<<a<<" b="<<b<<" --
 ***"<<endl; cout<<"*** ----------***"<<endl;
     for(int i=0; i<n; i++) { x[i]=x0(i,n);
        cout<<i+1;
        x[i]=xeps(x[i],n,eps);
        printf(" :%.15f",convert_x(x[i],a,b));;
        printf(" :%.15f\n",convert_A(A(x[i],n),a,b));
    }
  //getch();
return 0;
  }
```

1.7-dastur. Output

```
                6
eps=5
1.40656 3.17244
2.2895:2.2895
2.22385
6
0.882941
***___HISOBLANDI_***
*** --  n=5 uchun -- ***
1 :-0.90963196754455566000 :0.19411758457043865000
2 :-0.54064083099365234000 :0.47043172906368108000
3 :-0.00000000000000006123 :0.56888888888888889000
4 :0.54064083099365234000 :0.47043172906368108000
5 :0.90963196754455566000 :0.19411758457043865000
_____
I=2.21087712506044950000
```

Ushbu in sinfi odatda ilmiy amaliy matematika va matematika-fizika tenglamalarini taqribiy kompyuterda hisoblash ishlarini amalga oshirish uchun ishlatiladi. Bu kutubxona aynan A254-S loyihasi doirasida ishalb chiqilgan. Loyihani asosiy vazifasi fizik jarayoinlarni matematik modellashtirish va ularni sonli echimlarni olish uchun dasturiy ta‘minotlarni ishlab chiqishga bag‘ishlangan. Kutubxonada 10 dan ortiq mo‘rosxo‘r sinf, 200 dan ortiq funksiyalar, matematik fuksiyalarning modifikatsiyalangan funksiyalari bor.

NAZAROT SAVOLLARI

1. Haqiqiy (real) yoki mavhumqismini oddiy arifmetik tipga qiy mat qilib berish mumkinmi?
2. Kopmleks tipni arifmetik tipka tenglashtirish mumkinmi?
3. Kopmleks sonning real qismi qaysi funksiya bilan olinadi?
4. Kopmleks sonning mavhum qismi qaysi funksiya bilan olinadi?
5. double re =complex_obj.real(); va double re = real(complex_obj); sintaktiklarning farqi bormi?
6. Kompleks soning mavhum qismini qanday kiritiladi?
7. Kompleks sonni to‘liq kiritish uchun qavs ichida real va mavhum qismlari qanday belgi bilan ajratiladi?
8. Kompleks sinfi qanday operandlarni qo‘llab quvvtlamaydi?
9. Valarray konteyneri nima uchun kerak?
10. Massiv element indekslari bo‘yicha birlashtrish operatorlari va

elementlararo matematik amallarni hisoblash sinfi ayting?

11. **apply()** – bu massiv elementlari bo‘yicha bir vaqtda o‘zgarishlarni bajaradi va nimani qaytaradi?

12. **cshift()** - berilgan son qiymat bo‘yicha indeks asosida massivni qanday shaklida surib yangi massiv qaytaradi?

13. Valarray tiplari bilan bajariladigan amallarni qayta aniqlash mumkinmi?

14. Kaysi sinf valarray sinfning merosxo‘ridir va bir o‘lchovli to‘plam ostilarni yaratishda ishlatiladi?

15. slice sinf qanday tipdagi ob‘yektni tavsiflovchi parametrlarni saqlaydi.

16. slice sinfining konstruktorida nechta paramert bor va qaysilar, sanab bering?

17. Agar qism to‘plam yordamida aniqlangan massiv doimiy valarrayning kichik bo‘lagi bo‘lsa, bu massiv yangi nima?

18. **stride() -** valarray to‘plam ostining elementlari orasidagi nimani topadi.

19. Qaysi sinf valarray ning ko‘p o‘lchamli to‘plam ostilari bilan ishlashga mo‘ljallangan.

20. Sinf gslice - Valarray bazaviy sinfdan olingan va nechta paramertni qabul qiladi?

21. Agar to‘plam gslice bilan aniqlangan va valarray ning to‘plam ostisi bo‘lsa, gslice yangi nima hisoblanadi.

22. gslice – bir nechta qismdan tashkil topgan valarray uchun to‘plam osti valarrayni aniqlaydi va hammasi belgilangan bir element indeksidan boshlanadimi yoki xar xil element indeksidan boshlanadimi?

23. gslice i slice ning farqlari nimadan iborat.

24. Qaysi sinf Valarray qismlari bilan aniqlangan to‘plam osti massivlar uchun amallarni qo‘llab quvvatlaydi.

25. Shaboln sinf muayyan valarray amallari tomonidan bilvosita yaratilgan va dasturda bevosita foydalanish mumkinmi yoki mumkin emasmi?

26. Bazaviy valarray sinf elementlari asosidagi to‘plam osti bo‘lib, belgilangan to‘plam osti o‘rtasida amallarni ta‘minlash orqali valarray ob‘yektlarining to‘plam osti ob‘yektlar bo‘lgan ob‘yektlarni qo‘llab-quvvatlaydigan yordamchi sinfni ayting?

27. Tanlangan elementlar ketma ketligi asosida Indirect_array sinf funksiyalari valarray uchun belgilangan tegishli funksiyalar qanday ishlaydi.

28. Bazaviy valarray sinf elementlari asosidagi to‘plam osti bo‘lib, belgilangan to‘plam osti uchun mantiqiy amallarni ta‘minlash orqali valarray ob‘yektlarining to‘plam osti ob‘yektlar bo‘lgan ob‘yektlarni qo‘llab-quvvatlaydigan yordamchi sinf ayting?

29. Interval matematikaning amallarini bajarishga yo‘naltirilgan sinf nomini ayting?

30. In sinfda nechta konchtruktor bor va ular qanday ishlaydi?

31. In(double a, double b) intervalga qiymat berish funksiyasida agar _a> a_ bo‘lsa, nima vazifa bajariladi.

10 – MA’RUZA

MURAKKAB SARALASH ALGORITMLARI. AMALIY DASTURLASH (2 SOAT)

Reja:

1. Murakkab saralash algoritmlari.
2. Saralash algoritmlari taqqoslash.
3. Juda ham katta sonlar bilan ishlash.
4. Large sinfining qo‘llanilishi.
5. Integer sinfining qo‘llanilishi

Annotatsiya: *Turli ma‘lumotlar tuzilmalaridan iborat to‘plamlarni saralash algoritmlarini bilasiz. Ammo ularni ko‘p sonli qiymatlari o‘zini tutishi qanday degan savol qiziqtiradimi? Nafaqat oddiy saralash balki shunday murakkab saralash algoritmlari ham mavjud. Ko‘p elementli massivlar katta sonlar bilash ishlashga ham foydalanish mumkin. Shuningdek tiplarning o‘zaro modifikatsiyasi bilan ham uzun sonlar bilan ishlanadi.*

Kalit so‘zlar*: murakkab saralash algoritmlari, saralash algoritmlari, juda ham katta sonlar, large sinfi, Integer sinfi.*

KIRISH

Turli ma‘lumotlar tuzilmalaridan iborat to‘plamlarni saralash algoritmlarini bilasiz. Ammo ularni ko‘p sonli qiymatlari o‘zini tutishi qanday degan savol qiziqtiradimi? Nafaqat oddiy saralash balki shunday murakkab saralash algoritmlari ham mavjud. Ko‘p elementli massivlar katta sonlar bilash ishlashga ham foydalanish mumkin. Shuningdek tiplarning o‘zaro modifikatsiyasi bilan ham uzun sonlar bilan ishlanadi.

Saralash algoritmlari va murakkabligi. Avvalo, algoritmlarni tadqiq qilishga, ularni imkon qadar tez ishlashi uchun optimallashtirish muhimdir. Buustida ishlayotganda, turli saralash uchun samarali usullarni o‘ylab topish imkoniyatiga ega bo‘lishingiz mumkin.

Ko‘p jihatdan, barcha saralash algoritmlarni o‘rganish va ularni sinash kerak. Agar dasturlashning o‘zi haqida gapiradigan bo‘lsak, ba‘zan kutilmagan qiyinchiliklar paydo bo‘lishi mumkin (C++ optimizatori juda yaxshi). Biroq, qaysi testlarni va qanday miqdorda amalga oshirilishi kerakligini hal qilish qiyin emas. Men ko‘rsata olmaydigan yagona narsa-bu deyarli 150 GB vaznga ega qiymatlarni saralash bo‘ldi.

Asosiy saralash algoritmlari tavsifi va ularni amalga oshirish usullari. Saralashni qisqacha va aniq ta‘riflashga va murakkabligini belgilashga harakat qilaman. Murakkab ma‘lumotlar tuzilmalarni foydalanishda(daraxt saralash kabi) odatda xotira katta miqdorda sarflanadi va eng yomon holatda boshqa xil faqat yordamchi qator yaratish kerak bo‘ladi. Barqarorlik (stabillik) saralash tushunchasi ham mavjud. Demak, elementlarning nisbiy tartibi teng bo‘lganda o‘zgarmaydi.

Havo sharcha kabi saralash (Bubble sort). Massivda chapdan o‘ngga qarab amal bajariladi. Agar joriy element keyingisidan katta bo‘lsa, ularni almashtiramiz. Buni massiv tartiblanmaguncha bajaramiz. E‘tibor bering, birinchi iterasiyadan keyin eng katta element massivning oxirida, to‘g‘ri joylashagan bo‘ladi. Ikkita iterasiyadan keyin ikkinchi eng katta element to‘g‘ri joylashgan bo‘ladi va hokazo. Ravshanki, n ta iterasiyadan ko‘p bo‘lmagandan keyin massiv tartiblangan bo‘ladi. Shunday qilib, eng yomon va o‘rtacha holatda murakkabligi $O(n^2)$, eng yaxshi holatda esa O(n) bo‘ladi.

1.1-dastur. Saralashni amalga oshirish.

```
void bubblesort(int* l, int* r) { int sz
        = r - l;
        if (sz <= 1) return; bool b
        = true;
        while (b) {
                b = false;
                for (int* i = l; i + 1 < r; i++) { if (*i >
                        *(i + 1)) {
                                swap(*i, *(i + 1));
                                b = true;
                        }
                }
                r--;
        }
                                }
```

Shaker saralash (Shaker sort, bundan tashqari, aralashgan saralash va koktel saralash sifatida tanilgan). Havo sharcha saralash oxirida kichik elementlar joylashtirilgan testlarda sekin ishlaydi (ba‘zan "tashboqa" deb ataladi). Bu element algoritmning har bir qadamida chap tomonga faqat bitta pozitsiyaga harakat qiladi. Shuning uchun nafaqat chapdan o‘ngga, balki o‘ngdan chapga ham o‘tadi. Massivning tartiblanmagan qismini aniqlash uchun ikkita begin va end ko‘rsatikichni qo‘llab quvvatlaydi. Keyingi iterasiyada endga etganimizda undan birini ayiramiz va o‘ngdan chapga ko‘chamiz. Xuddi shunday, beginga etganimizda unga birini qo‘shamiz va chapdan o‘ngga ko‘chamiz . Algoritm havo sharcha saralash bilan bir xil murakkabligi bor, lekin real vaqtda yaxshi ishlaydi.

1.2-dastur. Saralashni amalga oshirish.

```
void shakersort(int* l, int* r) { int sz
        = r - l;
        if  (sz  <=  1)  return;
        bool b = true;
        int*  beg  =  l  -  1;
        int*  end  =  r  -  1;
        while (b) {
                b  =  false;
                beg++;
                for (int* i = beg; i < end; i++) { if (*i
                        > *(i + 1)) {
                                swap(*i, *(i + 1)); b = true;
                        }
                }
                if (!b) break;
                end--;
                for (int* i = end; i > beg; i--) { if (*i <
                        *(i - 1)) {
                                swap(*i, *(i - 1)); b
                                = true;
                        }
                }
        }
                        }
```

Taroqsimon caralash (Comb sort). Shaker saralashning yana bir o‘zgartirilgani. "Toshbaqalar" dan qutulish uchun masofada turgan elementlarni qayta tashkil qilamiz. Keling, uni tuzatamiz va chapdan o‘ngga boramiz, bu masofada turgan elementlarni taqqoslaymiz, agar kerak bo‘lsa, ularni qayta tartibga solaylik. Shubhasiz, bu "toshbaqalar"ni tez qator boshiga olish imkonini beradi. Dastlab qator uzunligiga teng masofani olib, so‘ngra uni taxminan 1.247 ga teng bo‘lgan biror koeffitsientga bo‘lish maqbuldir. Masofa bir biriga teng bo‘lganda shaker saralash bajariladi. Eng yaxshisi, murakkabligi O(nlogn), eng yomoni, u $O(n^2)$. O‘rtacha murakkabligi nima uchun juda aniqmas, amalda O(nlogn) kabi ko‘rinadi.

1.3-dastur. Saralashni amalga oshirish.

```
void combsort(int* l, int* r) { int
        sz = r - l;
        if (sz <= 1) return;
        double k = 1.2473309;
        int step = sz - 1; while
        (step > 1) {
                for (int* i = l; i + step < r; i++) { if (*i >
                        *(i + step))
                                swap(*i, *(i + step));
                }
                step /= k;
        }
        bool b = true;
        while (b) {
                b = false;
                for (int* i = l; i + 1 < r; i++) { if (*i >
                        *(i + 1)) {
                                swap(*i, *(i + 1)); b
                                = true;
                        }
                }
        }
}
```

Qo‘shish orqali saralash (Insertion sort). Algoritm yakunlangandan keyin javobni o‘z ichiga olgan bir yangi massiv yaratiladi. Javob massividagi elementlar har doim tartiblanib turishi uchun manba massividan elementlarni birma-bir kiritamiz. Murakkabligi o‘rtacha va eng yomon hol $O(n^2)$ va eng yaxshi O(n) bo‘ladi. Boshqa algoritmlarga nisbattan joriy qilish yaxshiroq (yangi massiv yaratish va albatta u nimanidir qo‘yish qiyin): joriy massivning ba‘zi bir prefikslari tartiblashtirilganiga ishonch hosil qilish va modomiki ular noto‘g‘ri tartibda ekan uni kiritish o‘rniga joriy elementni avvalgisi bilan o‘zgartiramiz.

1.4-dastur. Saralashni amalga oshirish.

```
void insertionsort(int* l, int* r) {
        for (int *i = l + 1; i < r; i++) { int* j = i;
                while (j > l && *(j - 1) > *j) { swap(*(j -
                        1), *j);
                        j--;
                }
        }
}
```

Shell caralash (Shellsort). Saralashda bir xil fikrdan foydalanishni o‘ylab ko‘ring, taroqsimon saralash va qo‘shish orqali saralash uchun uni qo‘llaymiz. Masofa tushunchasini kiritaylik va tanlaylik. So‘ngra massiv elementlari sinflarga ajratiladi, shu sinfga o‘zgarmas masofaning bir nechta elementlari kiradi. Har bir sinfni qo‘shib, tartiblash yo‘li bilan tartiblaylik. Taroqsimon saralashdan farqli o‘laroq, optimal masofalar to‘plami aniqmas. Turli taxminlarga ega bo‘lgan bir nechta ketma-ketliklar mavjud bo‘lsin. Shell saralashda birinchi element ketma- ketligi massiv uzunligiga teng, har bir keyingi element ketma-ketligi oldingisining yarim o‘lchamiga tengdir. Hibbard ketma-ketligi 2n-1 uchun murakkabligi eng yomon holat $O(n^2)$ dir, Sedgwick ketma-ketligi $O(n^{4/3})$ uchun (ifoda o‘zgaruvchan xususiyatli) va Pratt (ikki va uch darajali elementlar) ketma-ketligi $O(nlog2^n)$ murakkabligi eng yomon holat $O(n^{1,5})$dir. Bu barcha ketma ketliklar faqat massiv hajmiga qadab hisoblab va kichik uchun ko‘proq ishlatish kerak (aks holda u faqat qo‘shimchalari bilan tartiblashtiriladi bo‘ladi). Qo‘shimcha tadqiqot qildim va $s_i = a * s_i\text{-}1 + k * s_i\text{-}1$ shaklida turli ketma ketliklar (bu qisman empirik siur ketma –ketlikda ilhomlanib, elementlar kichik soni uchun eng yaxshi masofa ketma ketliklardan biri) sinovdan o‘tkazdim. Eng yaxshi ketma ketliklar a = 3, k =1/3, a= 4, k = 1/4 va a = 5, k = -1/5 koeffitsentlari bilan maqsadga erishish mumkin.

1.5-dastur. Saralashni amalga oshirish.

```
1- variant
void shellsort(int* l, int* r) { int sz
        = r - l;
        int step = sz / 2; while
        (step >= 1) {
                for (int *i = l + step; i < r; i++) { int *j =
                        i;
                        int *diff = j - step;
while (diff >= l && *diff > *j)
{
                        swap(*diff, *j);
                                j = diff;
```

```
                        diff = j - step;
                    }
                }
                step /= 2;
            }
}
```

2- variant

```
void shellsorthib(int* l, int* r) { int sz
        = r - l;
        if (sz <= 1) return; int
        step = 1;
        while (step < sz) step <<= 1;
        step >>= 1;
        step--;
        while (step >= 1) {
            for (int *i = l + step; i < r; i++) { int *j =
                i;
                int *diff = j - step;
                while (diff >= l && *diff > *j) {
                    swap(*diff, *j);
                    j = diff;
                    diff = j - step;
                }
            }
            step /= 2;
        }
}
```

2-variant

```
int steps[100];
void shellsortsedgwick(int* l, int* r) { int sz
        = r - l;
        steps[0] = 1;
        int q = 1;
        while (steps[q - 1] * 3 < sz) { if (q
            % 2 == 0)
                steps[q] = 9 * (1 << q) - 9 * (1 << (q / 2)) + 1;
            else
```

```
                steps[q] = 8 * (1 << q) - 6 * (1 << ((q + 1) / 2)) + 1;
            q++;
        }
        q--;
        for (; q >= 0; q--) {
            int step = steps[q];
            for (int *i = l + step; i < r; i++) { int *j =
                i;
                int *diff = j - step;
                while (diff >= l && *diff > *j) {
                    swap(*diff, *j);
                    j = diff;
                    diff = j - step;
                }
            }
        }
}
```

4- variant

```
void shellsortpratt(int* l, int* r) { int sz =
        r - l;
        steps[0] = 1;
        int cur = 1, q = 1;
        for (int i = 1; i < sz; i++) { int cur
            = 1 << i;
            if (cur > sz / 2) break;
            for (int j = 1; j < sz; j++) { cur *=
                3;
                if (cur > sz / 2) break; steps[q++]
                = cur;
            }
        }
        insertionsort(steps, steps + q);
```

```
        q--;
        for (; q >= 0; q--) {
                int step = steps[q];
                for (int *i = l + step; i < r; i++) { int *j =
                        i;
                        int *diff = j - step;
                        while (diff >= l && *diff > *j) {
                                swap(*diff, *j);
                                j = diff;
                                diff = j - step;
                        }
                }
        }
}
```

5- variant

```
void myshell1(int* l, int* r) { int
        sz = r - l, q = 1; steps[0] =
        1;
        while (steps[q - 1] < sz) { int s
                = steps[q - 1];
                steps[q++] = s * 4 + s / 4;
        }
        q--;
        for (; q >= 0; q--) {
                int step = steps[q];
                for (int *i = l + step; i < r; i++) { int *j =
                        i;
                        int *diff = j - step;
                        while (diff >= l && *diff > *j) {
                                swap(*diff, *j);
                                j = diff;
                                diff = j - step;
                        }
                }
        }
}
```

6- variant

```
void myshell2(int* l, int* r) { int
        sz = r - l, q = 1; steps[0] =
        1;
        while (steps[q - 1] < sz) { int s
                = steps[q - 1];
                steps[q++] = s * 3 + s / 3;
        }
        q--;
        for (; q >= 0; q--) {
                int step = steps[q];
                for (int *i = l + step; i < r; i++) { int *j =
                        i;
                        int *diff = j - step;
                        while (diff >= l && *diff > *j) {
                                swap(*diff, *j);
                                j = diff;
                                diff = j - step;
                        }
                }
        }
                }
```

7- variant

```
void myshell3(int* l, int* r) { int
        sz = r - l, q = 1; steps[0] =
        1;
        while (steps[q - 1] < sz) { int s
                = steps[q - 1];
                steps[q++] = s * 4 - s / 5;
        }
        q--;
        for (; q >= 0; q--) {
                int step = steps[q];
                for (int *i = l + step; i < r; i++) { int *j =
                        i;
                        int *diff = j - step;
                        while (diff >= l && *diff > *j) {
                                swap(*diff, *j);
                                j = diff;
```

```
                    diff = j - step;
                }
            }
        }
}
```

Iearxik saralash (Tree sort). Binar qidiruv iearxiyasiga elementlarni kiritamiz. Barcha elementlar kiritilgandan so‘ng, iearxiyadan tartiblangan massiv olish kifoya. Agar siz qizil-qora kabi muvozanatli iearxiyadan foydalansangiz, murakkabligi eng yomon, o‘rtacha va eng yaxshi holatlarda O(nlogn). Amalga oshirishda multiset konteyner foydalanadi.

1.6-dastur. Saralashni amalga oshirish.

```
void treesort(int* l, int* r) { multiset<int>
        m;
        for (int *i = l; i < r; i++) m.insert(*i);
        for (int q : m)
            *l = q, l++;
}
```

Nav saralash (Gnome sort). Algoritm qo‘shish orqali saralashga o‘xshash. Ko‘rsatgichni joriy elementga ko‘rsatamiz, agar u avvalgisidan katta bo‘lsa yoki u birinchi bo‘lsa ko‘rsatgichni to‘g‘ri holatga ko‘chiramiz, aks holda joriy va oldingi elementlarni o‘zgartiramiz va chap tomonga harakatlantiramiz.

1.7-dastur. Saralashni amalga oshirish.

```
void gnomesort(int* l, int* r) { int
        *i = l;
        while (i < r) {
            if (i == l || *(i - 1) <= *i) i++; else
            swap(*(i - 1), *i), i--;
        }
}
```

Tanlash orqali saralash (Selection sort). Navbatdagi iterasiya bo‘yicha massivdagi minimumni joriy elementdan keyin topamiz va kerak bo‘lsa u bilan o‘zgartiramiz. Shunday qilib, i-iterasiyadan keyin birinchi i elementlar o‘z joylarida qoladi. murakkabligi eng yaxshi, o‘rtacha va eng yomon holatlarda $O(n^2)$. Bu saralashni ikki yo‘l bilan amalga oshirilishi mumkinligini unutmang - minimal va uning indeks tutib, yoki shunchaki ular noto‘g‘ri tartibda bo‘lsa, ko‘rib chiqilayotgan va joriy element o‘rin almashtirish kerak. Birinchi usul bir oz tezroq.

1.8-dastur. Saralashni amalga oshirish.

```
void selectionsort(int* l, int* r) { for (int
        *i = l; i < r; i++) {
                int minz = *i, *ind = i;
                for (int *j = i + 1; j < r; j++) {
                        if (*j < minz) minz = *j, ind = j;
                }
```

```
                swap(*i, *ind);
        }
}
```

Piramida kabi saralash (Heapsort). Tanlash orqali saralash g‘oyasini rivojlantirilgan varianti. Piramida shaklidagi ma‘lumot strukturasidan foydalanaylik. Elementlarni qo‘shish va minimumini chiqarish orqali O(logn), O(1) minimumini olish imkonini beradi. Shunday qilib, O(nlogn) murakkabligi eng yomon, o‘rtacha va eng yaxshi holatlarda. C++ da priority_queue konteyneri mavjud bo‘lsa-da, bu konteyner juda sekin bo‘lgani uchun Piramidaga yo‘natirilgan yangi to‘plam sinfni amalga oshirdim.

1.9-dastur. Saralashni amalga oshirish.

```
Heap sinfi
template <class T>
class heap { private
:
        vector<T> h;
        void SiftUp(int a) {
                while (a) {
                        int p = (a - 1) / 2;
                        if (h[p] > h[a]) swap(h[p], h[a]); else
                        break;
                        a--; a /= 2;
                }
        }
        void SiftDown(int a) {
                while (2 * a + 1 < n) {
                        int l = 2 * a + 1, r = 2 * a + 2; if (r
                        == n) {
                                if (h[l] < h[a]) swap(h[l], h[a]); break;
                        }
                        else if (h[l] <= h[r]) {
```

```
                        if (h[l] < h[a]) {
                            swap(h[l], h[a]); a
                            = l;
                        }
                        else break;
                    }
else if (h[r] < h[a]) {
                        swap(h[r], h[a]); a
                        = r;
                    }
                    else break;
                }
            }
public:
            heap():n(0){}
            int n;
            int size() {
                return n;
            }
            int top() {
                return h[0];
            }
            bool empty() {
                return n == 0;
            }
            void push(T a) {
                h.push_back(a);
                SiftUp(n);
                n++;
            }
            void pop() {
                n--;
                swap(h[n], h[0]);
                h.pop_back();
                SiftDown(0);
            }
```

```
        void clear() {
                h.clear(); n
                = 0;
        }
        T operator [] (int a) {
                return h[a];
        }
};
```

Heap cinfi aosida saralash

```
void heapsort(int* l, int* r) {
        heap<int> h;
for (int *i = l; i < r; i++) h.push(*i);
        for (int *i = l; i < r; i++) {
                *i = h.top();
                h.pop();
        }
}
```

Tezkor saralash (quicksort). Ba‘zi mos yozuvlar elementini tanlaylik. Shundan so‘ng chapdan kichik bo‘lgan barcha elementlarni, katta bo‘lganlarini esa o‘ng tomonga harakatlantiramiz. Takrorlanuvchi qismlarning har biridan rekursiv chaqirish mumkin. Natijada tartiblangan massivga ega bo‘lamiz, chunki har bir katta elementdan oldin massivdan kichik bo‘lgan har bir element joylashtirilgan. Murakkabligi: O(nlogn) o‘rtacha va eng yaxshi holda, O (n^2) eng yomon holda. Malumot elementi noto‘g‘ri tanlanganda eng yomon reytingga erishiladi. Bu algoritmni amalga oshirish butunlay standart hisoblanadi, bir vaqtning o‘zida chapga va o‘ngga borib, elementlarni juft topish, bunda chap element mos yozuvlari tanlangandan katta ekanligini va o‘ng kichik, ular almashtiriladi. Sof tezkor tartiblashdan tashqari, elementlar soni kam bo‘lganda tartiblashni joylashtirishga, taqqoslashda ham saralash ishtirok etdi. Qo‘shish orqali saralash vazifa uchun mos bo‘lgan eng yaxshi saralashdir va doimiy sinov orqali tanlanadi.

1.10-dastur. Saralashni amalga oshirish.

Birinchi variant

```
void quicksort(int* l, int* r) { if (r -
        l <= 1) return; int z = *(l +
        (r - l) / 2); int* ll = l, *rr = r -
        1; while (ll <= rr) {
```

```
            while (*ll < z) ll++;
            while (*rr > z) rr--; if
            (ll <= rr) {
                swap(*ll, *rr);
                ll++;
                rr--;
            }
        }
        if (l < rr) quicksort(l, rr + 1); if (ll <
        r) quicksort(ll, r);
}
```

Ikkinchi variant

```
void quickinssort(int* l, int* r) { if (r -
        l <= 32) {
            insertionsort(l, r);
            return;
        }
        int z = *(l + (r - l) / 2); int* ll =
        l, *rr = r - 1; while (ll <= rr) {
            while (*ll < z) ll++;
            while (*rr > z) rr--; if
            (ll <= rr) {
                swap(*ll, *rr);
                ll++;
                rr--;
            }
        }
        if (l < rr) quickinssort(l, rr + 1); if (ll < r)
        quickinssort(ll, r);
}
```

Birlashtirish orqali saralash (Merge sort). Bo‘lish-boshqarish paradigmasi asosida tartiblash. Massivni ikkiga bo‘lamiz, qismlarni rekursiv ravishda tartiblang va keyin birlashtirish protsedurasini bajaring. Birinchi qismning joriy elementiga, ikkinchisi ikkinchi qismning joriy elementiga ikkita ko‘rsatkichni qo‘llang. Bu ikki elementdan minimumni tanlab, javobga joylashtiring va minimumga mos keladigan ko‘rsatgichni siljiting. Birlashtirish O(n), barcha logn darajalari uchun ishlaydi, shuning uchun murakkabligi O(nlogn). Bu oldindan vaqtinchalik masiv yaratish va vazifaga argument sifatida

o‘tishi samarali bo‘ladi. Bu saralash rekursiv, hamda tez va shuning uchun elementlar soni kam bo‘lgan kvadratik o‘tish mumkin.

1.11-dastur. Saralashni amalga oshirish.

Asosiy funksiya

```
void merge(int* l, int* m, int* r, int* temp) { int *cl
        = l, *cr = m, cur = 0;
        while (cl < m && cr < r) {
                if (*cl < *cr) temp[cur++] = *cl, cl++; else
                temp[cur++] = *cr, cr++;
        }
        while (cl < m) temp[cur++] = *cl, cl++;
        while (cr < r) temp[cur++] = *cr, cr++; cur
        = 0;
        for (int* i = l; i < r; i++)
                *i = temp[cur++];
}
```

Birinchi varinat

```
void _mergesort(int* l, int* r, int* temp) { if (r -
        l <= 1) return;
        int *m = l + (r - l) / 2;
        _mergesort(l, m, temp);
        _mergesort(m, r, temp);
        merge(l, m, r, temp);
}
void mergesort(int* l, int* r) { int*
        temp = new int[r - l];
        _mergesort(l, r, temp);
        delete temp;
}
```

Ikkinchi varinat

```
void _mergeinssort(int* l, int* r, int* temp) { if (r - l
        <= 32) {
                insertionsort(l, r);
                return;
        }
        int *m = l + (r - l) / 2;
        _mergeinssort(l, m, temp);
```

```
        _mergeinssort(m, r, temp);
        merge(l, m, r, temp);
}
void mergeinssort(int* l, int* r) { int*
        temp = new int[r - l];
        _mergeinssort(l, r, temp);
        delete temp;
}
```

Sanash orqali saralash (Counting sort). r - l hajmli massiv yarating, massivning l - minimal va r maksimal elementi hisoblanadi. Shundan so‘ng, massiv orqali borib va har bir element sonini hisoblang. Endi qiymatlar massividan o‘tib, har bir sonni nechta marta bo‘lganini yozishingiz mumkin. murakkabligi O(n+ r — l) bo‘ladi. Bu algoritm barqaror qilish uchun o‘zgartirishingiz mumkin: buning uchun, keyingi soni joyini aniqlash kerak bo‘lib va chapdan o‘ngga orginal massiv orqali borish, to‘g‘ri joyda elementi qo‘yib va o‘rnini birga oshirish lozim. Bu saralash sinovdan o‘tkazilmagan, testlar yetarli darajada katta raqamlar uchun mavjud, chunki yangi massiv yaratish ancha xotira talab qiladi. Biroq, bu algoritm masalaga qarab foydali bo‘ldi.

Blok orqali saralash (Bucket sort). Bu algoritm savat va cho‘ntak saralash sifatida tanilgan algoritmdir. Massivning l minimal va r maksimal elementi bo‘lsin. Elementlarni bloklarga ajratamiz, birinchi blokda l dan l + k gacha, ikkinchisiga l + k dan l + 2k gacha va hokazo elementlarni o‘z ichiga oladi, bu yerda k = (r – l) /n, n bloklar soni. Umuman olganda, agar bloklar soni ikkiga teng bo‘lsa, bu algoritm tezkor saralashga aylanadi. Ushbu algoritmning murakkabligi aniq emas, ma‘lumotlariga murojaat vaqt va bloklar soniga bog‘liq. Muvaffaqiyatli ma‘lumotlar bo‘yicha ish vaqti chiziqli ekanligi aytib o‘tiladi. Ushbu algoritmni amalga oshirish eng qiyin vazifalardan biri ekanligini isbotlangan. Buni shunchaki yangi massivlar yaratish, ularni rekursiv saralash va ularni bir-biriga birlashtirish orqali amalga oshirishingiz mumkin. Biroq, bu yondashuv juda sekin va men qoniqmadim. Samarali amalga oshirishda bir necha g‘oyalardan foydalaniladi:

1) yangi massivlar yaratmaymiz. Buning uchun sanao‘ orqali saralash texnikasidan foydalanamiz: har bir blokdagi elementlar sonini, prefiks summalarini va shu tariqa har bir elementning massivdagi o‘rnini sanaymiz.

2) Bo‘sh bloklardan boshlamaymiz. Bo‘sh bo‘lmagan bloklarning indekslarini alohida massivda kiritish va ulardan boshlash orqali saralash.

3) Massiv tartiblanganligini tekshiramiz. Agar hali ham minimal va maksimal topish uchun bir o‘tish qilish kerak, chunki, bu ishlayotgan vaqt yomonlashmaydi, lekin elementlar orginal massiv bir xil tartibda yangi bloklari joylashtirilgan, chunki u qisman tartiblashtirilgan ma‘lumotlarni saralashni tezlashtirish uchun algoritmni beradi.

4) Algoritm juda noqulay bo‘lgani uchun, u juda kam sonli elementlar bilan juda samarasiz. Joylashtirishga o‘tish tartibida ishni taxminan 10 martagacha tezlashtiradi.

Bu faqat qancha bloklar tanlashni tushunishga qolmoqda. Randomize testlarda quyidagi ballarni olishga muvaffaq bo‘lgan: 1500 elementlari uchun 107 bloklari va 3000 uchun 108 ta. Murakkablik formulasini topa olmadik, ish vaqti bir necha marta yomonlashadi.

1.12-dastur. Saralashni amalga oshirish.

```
void _newbucketsort(int* l, int* r, int* temp) { if (r -
    l <= 64) {
        insertionsort(l, r);
        return;
    }
    int minz = *l, maxz = *l; bool
    is_sorted = true;
     for (int *i = l + 1; i < r; i++) { minz
        = min(minz, *i);
        maxz = max(maxz, *i);
        if (*i < *(i - 1)) is_sorted = false;
    }
    if (is_sorted) return;
    int diff = maxz - minz + 1; int
    numbuckets;
    if (r - l <= 1e7) numbuckets = 1500; else
    numbuckets = 3000;
    int range = (diff + numbuckets - 1) / numbuckets;
    int* cnt = new int[numbuckets + 1];
    for (int i = 0; i <= numbuckets; i++)
        cnt[i] = 0;
    int cur = 0;
    for (int* i = l; i < r; i++) { temp[cur++]
        = *i;
        int ind = (*i - minz) / range;
        cnt[ind + 1]++;
    }
```

```
        int sz = 0;
        for (int i = 1; i <= numbuckets; i++) if
                (cnt[i]) sz++;
        int* run = new int[sz];
        cur = 0;for (int i = 1; i <= numbuckets;
                i++) if (cnt[i]) run[cur++] = i - 1;
        for (int i = 1; i <= numbuckets; i++)
                cnt[i] += cnt[i - 1];
        cur = 0;
        for (int *i = l; i < r; i++) {
                int ind = (temp[cur] - minz) / range;
                *(l + cnt[ind]) = temp[cur];
                cur++;
                cnt[ind]++;
        }
        for (int i = 0; i < sz; i++) { int r =
                run[i];
                if (r != 0) _newbucketsort(l + cnt[r - 1], l + cnt[r], temp); else
                _newbucketsort(l, l + cnt[r], temp);
        }
        delete run; delete
        cnt;
}
void newbucketsort(int* l, int* r) { int
        *temp = new int[r - l];
        _newbucketsort(l, r, temp);
        delete temp;
}
```

Bitli saralash(Radix sort). Shuningdek, raqamli saralash sifatida tanilgan algoritmdir. Bu saralashning ikki versiyasi mavjud bo‘lib, ular, qandaydir sonlar sistemasida (masalan, binar – ikkilik) sonni ifodalashdan foydalanish fikridan boshqa umumiylikka ega emas.

Birinchi usuli. LSD (least significant digit, kichik muhim son). Har bir sonni binar shaklda ifodalaylik. Algoritmning har bir qadamida sonlarni shunday tartiblaymizki, ular k doimiy bo‘lgan birinchi k * i bitlar bo‘yicha tartiblansin. Bu ta‘rifdan shunday xulosa kelib chiqadiki, har bir qadamda elementlarni yangi k bitlar bo‘yicha uzluksiz tartiblash kifoyadir. Shu maqsadda, sanash orqali saralash ideldir (agar muvaffaqiyatlisini doimiy tanlasangiz 2k xotira va vaqt ko‘p kerak emas). Murakkabligi

O(n), agar sonlar fiksirlangan kattalikda deb faraz qilsak va aks holda ikki sonni taqqoslash vaqt birligi bo‘yicha bajariladi deb hisoblash mumkin bo‘lmaydi. Amalga oshirish juda oddiy.

1.13-dastur. Saralashni amalga oshirish.

```
int digit(int n, int k, int N, int M)
{ return (n >> (N * k) & (M- 1));
}

void _radixsort(int* l, int* r, int N) { int k = (32 + N - 1) / N;
        int M = 1 << N; int
        sz = r - l;
        int* b = new int[sz]; int* c = new int[M];
        for (int i = 0; i < k; i++) {
                for (int j = 0; j < M; j++) c[j] = 0;
                for (int* j = l; j < r; j++) c[digit(*j, i, N, M)]++;
                for (int j = 1; j < M; j++) c[j] += c[j - 1];
                for (int* j = r - 1; j >= l; j--)
                        b[--c[digit(*j, i, N, M)]] = *j; int cur = 0;
                for (int* j = l; j < r; j++)
                        *j = b[cur++];
        }
        delete b; delete c;
}
void radixsort(int* l, int* r) {
        _radixsort(l, r, 8);
}
```

IkIkkinchi usuli. MSD (most significant digit, eng muhim raqam). Aslida qandaydir blokni saralash algoritmiga o‘xshaydi. Shu blokda teng k bitli sonlar bo‘ladi. Murakkabligi LSD versiyasidagidek. Amalga oshirish blok saralash juda o‘xshaydi, lekin oddiy. Bu LSD versiyasini amalga oshirishdagi belgilangan raqamli vazifasidan foydalanadi.

1.14-dastur. Saralashni amalga oshirish.

```
void _radixsortmsd(int* l, int* r, int N, int d, int* temp) { if (d == -
        1) return;
        if (r - l <= 32) {
                insertionsort(l, r);
                return;
        }
        int M = 1 << N;
        int* cnt = new int[M + 1]; for
        (int i = 0; i <= M; i++)
                cnt[i] = 0; int
        cur = 0;
        for (int* i = l; i < r; i++) { temp[cur++]
                = *i; cnt[digit(*i, d, N, M) +
                1]++;
        }
        int sz = 0;
        for (int i = 1; i <= M; i++) if
                (cnt[i]) sz++;
        int* run = new int[sz];
        cur = 0;
        for (int i = 1; i <= M; i++)
                if (cnt[i]) run[cur++] = i - 1; for (int
        i = 1; i <= M; i++)
                cnt[i] += cnt[i - 1]; cur
        = 0;
        for (int *i = l; i < r; i++) {
                int ind = digit(temp[cur], d, N, M);
                *(l + cnt[ind]) = temp[cur];
                cur++;
                cnt[ind]++;
        }
```

```
        for (int i = 0; i < sz; i++) { int r =
                run[i];
                if (r != 0) _radixsortmsd(l + cnt[r - 1], l + cnt[r], N, d - 1, temp); else
                _radixsortmsd(l, l + cnt[r], N, d - 1, temp);
        }
        delete run; delete
        cnt;
}
void radixsortmsd(int* l, int* r) { int*
        temp = new int[r - l];
        _radixsortmsd(l, r, 8, 3, temp); delete
        temp;
}
```

Bitonik saralash (Bitonic sort). Massiv elementlarini paralel saralash algoritmi bo‘lib, tarmoqlarda foydalaniladi. Bu algoritmning g‘oyasi shundan iboratki, massiv Biton ketma – ketlikka-avval ortib, keyin kamayib boradigan ketma-ketlikka aylanadi. Buni quyidagicha samarali tartiblashtirish mumkin: massivni ikki qismga ajratiladi, ikkita massiv yaratiladi, barcha elementlarni ikki qismdan har bir tegishli elementlar birinchi massivga minimal va ikkinchisiga maksimal teng qo‘shiladi. Ikkita Biton ketma-ketlik olinishi, ularning har birini bir xil usulda rekursiv ravishda tartiblash mumkin, so‘ngra ikkita massivni birlashtirish mumkin (birinchisining har qaysi elementi ikkinchisining har qaysi elementidan kam yoki teng bo‘lganligi uchun). Massivini Biton ketma-ketlikka aylantirish uchun quyidagilarni bajarish lozim: agar massiv ikki elementdan iborat bo‘lsa, uni shunchaki bajarish mumkin, aks holda massivni ikkiga ajratib, algoritmni bo‘lak massivlardan rekursiv chaqiriladi, keyin birinchi qismini tartib bilan, ikkinchisini teskari tartibda saralab, bir-biriga birlashtiriladi. Shubhasiz, natijada bitonal ketma-ketlikni olanadi. Murakkabligi O(nlog2n), chunki Biton ketma-ketligini qurishda O(nlogn) uchun ishlaydigan tartiblashlardan foydalandik va darajalarning umumiy soni logn edi. Bundan tashqari, massiv hajmi ikkining darajasiga karrali bo‘lishi kerak, shuningdek, elementlar bilan to‘ldirish kerak bo‘lishi mumkin.

1.15-dastur. Saralashni amalga oshirish.

```
void bitseqsort(int* l, int* r, bool inv) { if (r - l
        <= 1) return;
        int *m = l + (r - l) / 2;
        for (int *i = l, *j = m; i < m && j < r; i++, j++) { if (inv ^
                (*i > *j)) swap(*i, *j);
        }
        bitseqsort(l, m, inv); bitseqsort(m, r,
        inv);
}
void makebitonic(int* l, int* r) { if (r
        - l <= 1) return;
        int *m = l + (r - l) / 2;
        makebitonic(l, m);
        bitseqsort(l, m, 0);
        makebitonic(m, r);
        bitseqsort(m, r, 1);
}
void bitonicsort(int* l, int* r) {
```

```
        int n = 1;
        int  inf  =  *max(l,  r)  +  1;
        while (n < r - l) n *= 2; int*
        a = new int[n];
        int cur = 0;
        for  (int  *i  =  l;  i  <  r;  i++)
                a[cur++] = *i;
        while (cur < n) a[cur++] = inf;
        makebitonic(a, a + n); bitseqsort(a, a
        + n, 0);
        cur = 0;
        for (int *i = l; i < r; i++)
                *i = a[cur++];
        delete a;
}
```

Timsort saralash (Timsort). Bu gibrid saralash hisoblanadi, qo‘shish orqali saralash va birlashtirish orqali saralash algoritmlaridan foydalanib yaratilgan. Massiv elementlarini bir necha

kichik massivlarga ajratamiz va undagi elementlarni tartiblangan holda massiv ostiga kengaytiramiz. Tartiblangan massivlar ustida samarali ishlashidan foydalanib, massiv ostilarni joylashtirib tartiblash yo‘li bilan tartiblaymiz. Keyinchalik massiv ostilarni taxminan teng kattalikda olib, birilashtirish (merge) saralashdagi kabi birlashtiramiz (aks holda saralash vaqti kvadratga yaqinlashadi). Buning uchun massiv ostilarni invariant saqlab, stekka saqlash qulay — tepadan uzoqlashgan sari kattalashib boradi va uchinchi massiv ostining o‘lchami ularning kattaliklari yig‘indisidan katta yoki teng bo‘lgandagina yuqori qismida massiv ostilarni birlashtiradi. murakkabligi O(n) eng yaxshi holatda va O(nlogn) o‘rtacha va eng yomon holatda. Amalga oshirish arzimas va unga qat‘iy ishonch yo‘q, lekin u juda yaxshi saralash vaqti ko‘rsatdi.

1.16-dastur. Saralashni amalga oshirish.

```
void _timsort(int* l, int* r, int* temp) { int sz =
        r - l;
        if (sz <= 64) {
                insertionsort(l, r);
                return;
        }
```

```
        int minrun = sz, f = 0;
        while (minrun >= 64) {
                f |= minrun & 1;
                minrun >>= 1;
        }
        minrun += f;
        int* cur = l;
        stack<pair<int, int*>> s; while
        (cur < r) {
                int* c1 = cur;
                while (c1 < r - 1 && *c1 <= *(c1 + 1)) c1++; int*
                c2 = cur;
                while (c2 < r - 1 && *c2 >= *(c2 + 1)) c2++; if
                (c1 >= c2) {
                        c1 = max(c1, cur + minrun - 1); c1
                        = min(c1, r - 1); insertionsort(cur,
                        c1 + 1);
                        s.push(make_pair(c1 - cur + 1, cur)); cur
```

```
                = c1 + 1;
            }
            else {
                c2 = max(c2, cur + minrun - 1); c2
                = min(c2, r - 1); reverse(cur, c2 +
                1); insertionsort(cur, c2 + 1);
                s.push(make_pair(c2 - cur + 1, cur)); cur
                = c2 + 1;
            }
            while (s.size() >= 3) {
                pair<int, int*> x = s.top(); s.pop();
                pair<int, int*> y = s.top(); s.pop();
                pair<int, int*> z = s.top(); s.pop();
                if (z.first >= x.first + y.first && y.first >= x.first) { s.push(z);
                    s.push(y);
                    s.push(x);
                    break;
                }
                else if (z.first >= x.first + y.first) {
                    merge(y.second, x.second, x.second + x.first, temp);
                    s.push(z);
                    s.push(make_pair(x.first + y.first, y.second));
                }
                else {
                    merge(z.second, y.second, y.second + y.first, temp);
                    s.push(make_pair( z.first + y.first, z.second )); s.push(x);
                }
            }
            while (s.size() != 1) {
                pair<int, int*> x = s.top(); s.pop();
                pair<int, int*> y = s.top(); s.pop();
                if (x.second < y.second) swap(x, y);
                merge(y.second, x.second, x.second + x.first, temp); s.push(make_pair(
                y.first + x.first, y.second ));
            }
        }
}
```

```
void timsort(int* l, int* r) { int*
        temp = new int[r - l];
        _timsort(l, r, temp);
        delete temp;
}
```

Yuqorida 16 ta saralash algoritmlarni hosil qilish va tayyo dasturiy fragmentlari keltirildi.

Saralash algoritmlari taqqoslash. Buning uchun barcha yaaratilgan saralash algoritmlarini bir faylga joylashtiramiz. Bularni taqqoslash uchun elementlar soni navbat bilan oshirib boramiz va natijalari sonli, grafikli keltiramiz.

1.1-jadval. Saralash algoritmlar ro‘yxati.

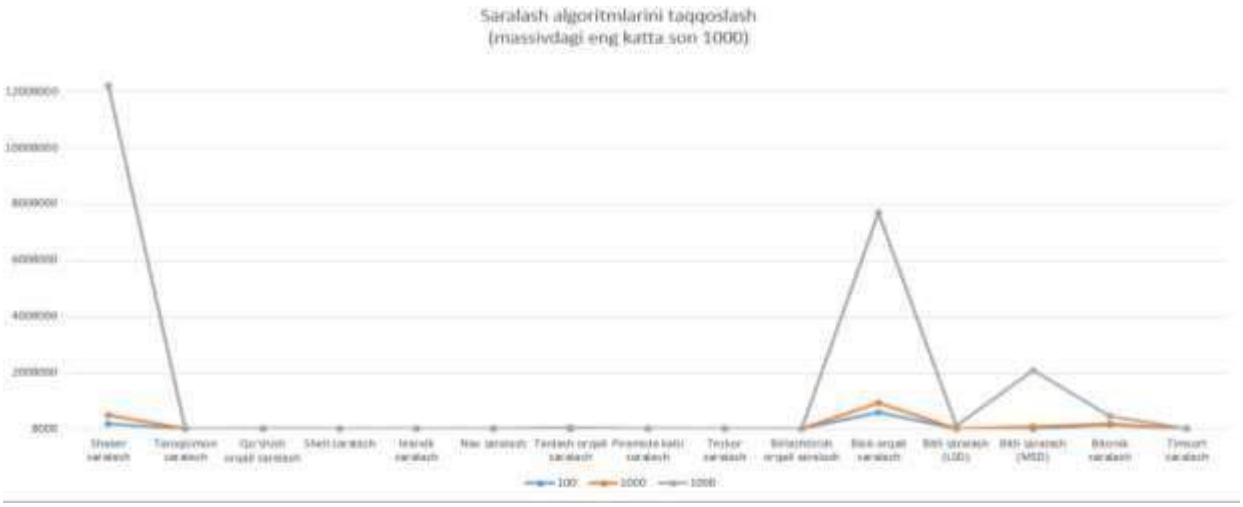
№	Saralash algoritmning nomi	funksiyasi
1	Havo sharcha kabi saralash	void bubblesort(int* l, int* r)
2	Shaker saralash	void shakersort(int* l, int* r)
3	Taroqsimon caralash	void combsort(int* l, int* r)
4	Qo‘shish orqali saralash	void insertionsort(int* l, int* r)
5	Shell caralash	void shellsort(int* l, int* r) void shellsorthib(int* l, int* r)
		void shellsortpratt(int* l, int* r) void myshell_one(int* l, int* r) void myshell_two(int* l, int* r) void myshell_three(int* l, int* r)
6	Iearxik saralash	void treesort(int* l, int* r)
7	Nav saralash	void gnomesort(int* l, int* r)
8	Tanlash orqali saralash	void selectionsort(int* l, int* r)
9	Piramida kabi saralash	void heapsort(int* l, int* r)
10	Tezkor saralash	void quicksort(int* l, int* r) void quickinssort(int* l, int* r)
11	Birlashtirish orqali saralash	void mergesort(int* l, int* r) void mergeinssort(int* l, int* r)
12	Blok orqali saralash	void newbucketsort(int* l, int* r)
13	Bitli saralash (LSD)	void radixsort(int* l, int* r)

14	Bitli saralash (MSD)	void radixsortmsd(int* l, int* r)
15	Bitonik saralash	void bitonicsort(int* l, int* r)
16	Timsort saralash	void timsort(int* l, int* r)

Bu algoritmlarni vaqt bo‘yicha taqqoslash uchun tasodifiy sonlar bilan massivni to‘ldiramiz, tasodifiy sonlarni ham guruhlarga ajratamiz, xonalar bo‘yicha, massiv elementlar sonini ham 100 ta dan 100000 tagacha tanlab olamiz.

Birinchi test. Massiv elementlari soni 100, 1000, 10000 ta va tasodifiy sonlarning eng kattasi 1000:

Elementlar soni:100 Random eng katta soni:1000	Elementlar soni:1000 Random eng katta soni:1000	Elementlar soni:10000 Random eng katta soni:1000
bubblesort(a,b) => 189949	bubblesort(a,b) => 509917	bubblesort(a,b) => 12229978
shakersort(a,b) => 9912	shakersort(a,b) => 8523	shakersort(a,b) => 10000
combsort(a,b) => 10017	combsort(a,b) => 9949	combsort(a,b) => 19984
insertionsort(a,b) => 9962	insertionsort(a,b) => 12012	insertionsort(a,b) => 25365
shellsort(a,b) => 9996	shellsort(a,b) => 12536	shellsort(a,b) => 20009
shellsorthib(a,b) => 9983	shellsorthib(a,b) => 9983	shellsorthib(a,b) => 10026
shellsortpratt(a,b) => 10001	shellsortpratt(a,b) => 10039	shellsortpratt(a,b) => 40028
myshell_one(a,b) => 10013	myshell_one(a,b) => 10026	myshell_one(a,b) => 10044
myshell_two(a,b) => 10000	myshell_two(a,b) => 9951	myshell_two(a,b) => 9996
myshell_three(a,b) => 9983	myshell_three(a,b) => 10056	myshell_three(a,b) => 20044
treesort(a,b) => 600071	treesort(a,b) => 940049	treesort(a,b) => 7690211
gnomesort(a,b) => 6510	gnomesort(a,b) => 10044	gnomesort(a,b) => 120041
selectionsort(a,b) => 29963	selectionsort(a,b) => 79987	selectionsort(a,b) => 2090445
heapsort(a,b) => 160055	heapsort(a,b) => 190026	heapsort(a,b) => 460513
quicksort(a,b) => 10013	quicksort(a,b) => 10001	quicksort(a,b) => 10462
quickinssort(a,b) => 9979	quickinssort(a,b) => 9992	quickinssort(a,b) => 10020
mergesort(a,b) => 10120	mergesort(a,b) => 20005	mergesort(a,b) => 29924
mergeinssort(a,b) => 9975	mergeinssort(a,b) => 10004	mergeinssort(a,b) => 9979
newbucketsort(a,b) => 20014	newbucketsort(a,b) => 19967	newbucketsort(a,b) => 19959
radixsort(a,b) => 10060	radixsort(a,b) => 30057	radixsort(a,b) => 19980
radixsortmsd(a,b) => 19992	radixsortmsd(a,b) => 20031	radixsortmsd(a,b) => 20249
bitonicsort(a,b) => 29988	bitonicsort(a,b) => 69974	bitonicsort(a,b) => 220502
timsort(a,b) => 149973	timsort(a,b) => 190021	timsort(a,b) => 140443



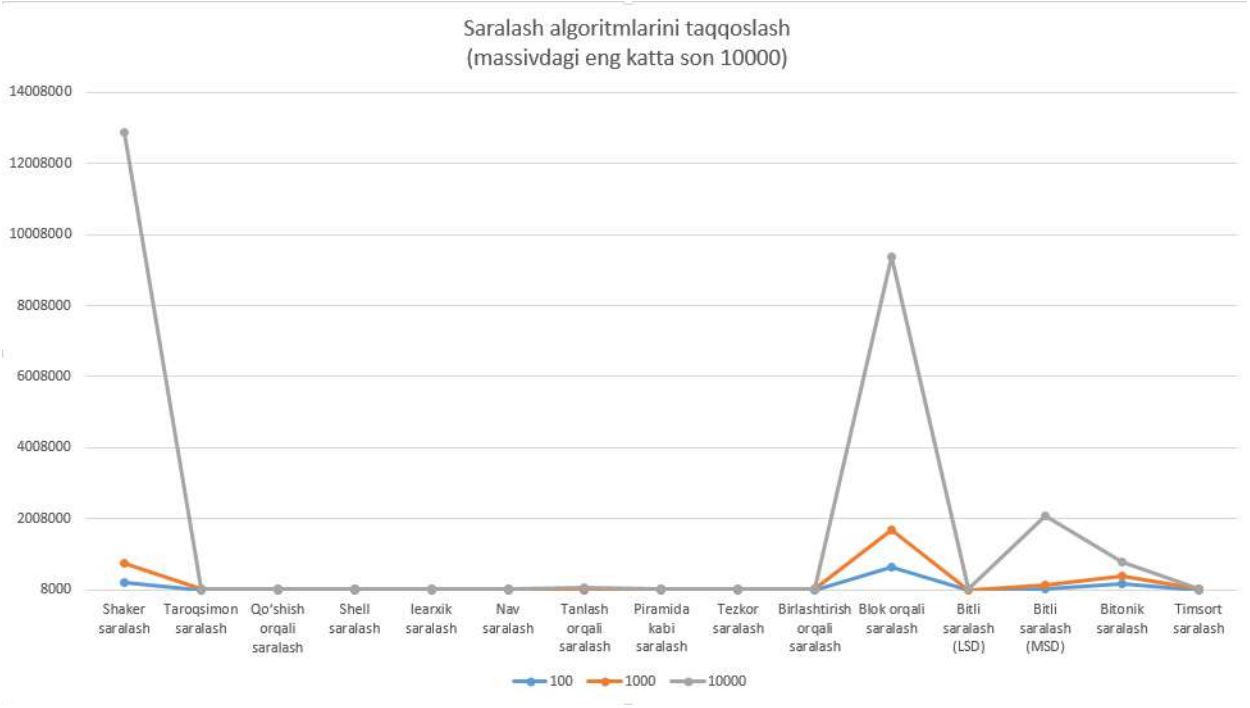
1.1-rasm. Saralash algoritmlarini taqqoslash.

Ushbu rasmda keltirilgan taqqoslash natijariga Shaker, blok orqali, bitli (MSD), bitonik saralash algoritmlarining elementlarning sonini o‘zgarish bilan saralash uchun sarflanadigan vaqtni o‘zgarishini kuzatishimiz mumkin.

Ikkinchi test. Massiv elementlari soni 100, 1000, 10000 ta va tasodifiy sonlarning eng kattasi 10000:

Elementlar soni:100 Random eng katta soni:10000	Elementlar soni:1000 Random eng katta soni:10000	Elementlar soni:10000 Random eng katta soni:10000
bubblesort(a,b) => 230011	bubblesort(a,b) => 509982	bubblesort(a,b) => 12139952
shakersort(a,b) => 9889	shakersort(a,b) => 9956	shakersort(a,b) => 10056
combsort(a,b) => 10018	combsort(a,b) => 10009	combsort(a,b) => 20001
insertionsort(a,b) => 9562	insertionsort(a,b) => 9986	insertionsort(a,b) => 10009
shellsort(a,b) => 10018	shellsort(a,b) => 9992	shellsort(a,b) => 20001
shellsorthib(a,b) => 10000	shellsorthib(a,b) => 10021	shellsorthib(a,b) => 20018
shellsortpratt(a,b) => 8452	shellsortpratt(a,b) => 9992	shellsortpratt(a,b) => 40006
myshell_one(a,b) => 10022	myshell_one(a,b) => 10025	myshell_one(a,b) => 10017
myshell_two(a,b) => 9988	myshell_two(a,b) => 10360	myshell_two(a,b) => 10005
myshell_three(a,b) => 12001	myshell_three(a,b) => 15201	myshell_three(a,b) => 20018
treesort(a,b) => 660039	treesort(a,b) => 1030011	treesort(a,b) => 7700494
gnomesort(a,b) => 7563	gnomesort(a,b) => 8624	gnomesort(a,b) => 11254
selectionsort(a,b) => 29997	selectionsort(a,b) => 99992	selectionsort(a,b) => 1950542
heapsort(a,b) => 170016	heapsort(a,b) => 210079	heapsort(a,b) => 400520
quicksort(a,b) => 9562	quicksort(a,b) => 10009	quicksort(a,b) => 10210
quickinssort(a,b) => 9996	quickinssort(a,b) => 10005	quickinssort(a,b) => 10004
mergesort(a,b) => 10069	mergesort(a,b) => 10000	mergesort(a,b) => 30318
mergeinssort(a,b) => 8996	mergeinssort(a,b) => 9996	mergeinssort(a,b) => 20502
newbucketsort(a,b) => 20060	newbucketsort(a,b) => 20040	

radixsort(a,b) => 10005	radixsort(a,b) => 10018	newbucketsort(a,b) => 19954 radixsort(a,b) => 30075
radixsortmsd(a,b) => 10013 bitonicsort(a,b) => 30014 timsort(a,b) => 179974	radixsortmsd(a,b) => 9954 bitonicsort(a,b) => 59934 timsort(a,b) => 150012	radixsortmsd(a,b) => 29680 bitonicsort(a,b) => 229981 timsort(a,b) => 149973

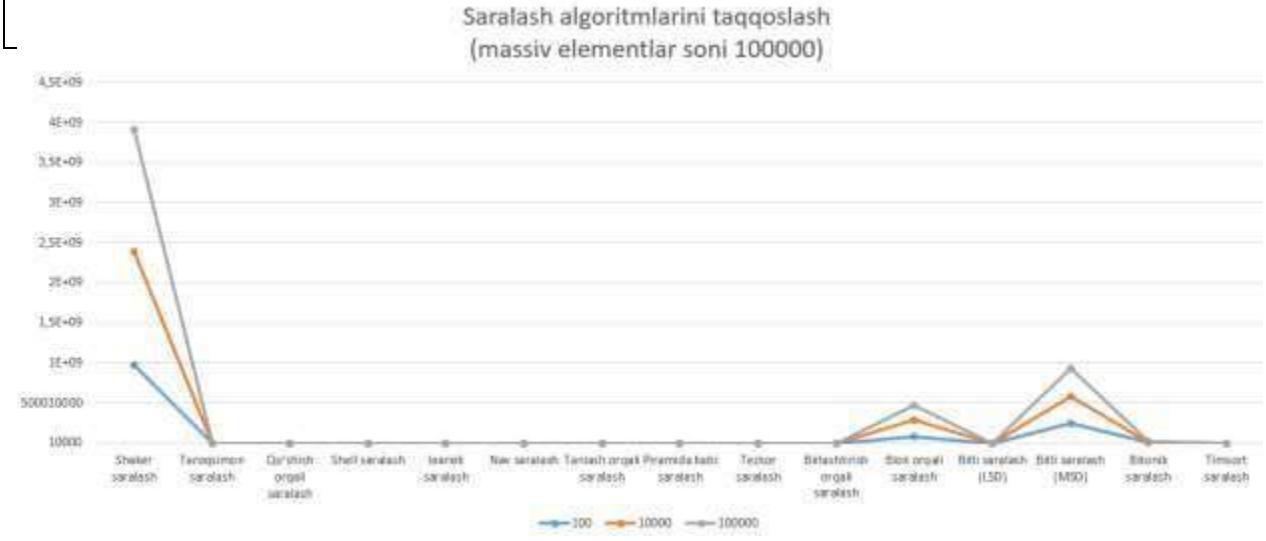


1.2-rasm. Saralash algoritmlarini taqqoslash.

Ushbu rasmda keltirilgan taqqoslash natijariga Shaker, blok orqali, bitli (MSD), bitonik saralash algoritmlarining elementlarning sonini o‘zgarish bilan saralash uchun sarflanadigan vaqtni o‘zgarishini kuzatishimiz mumkin.

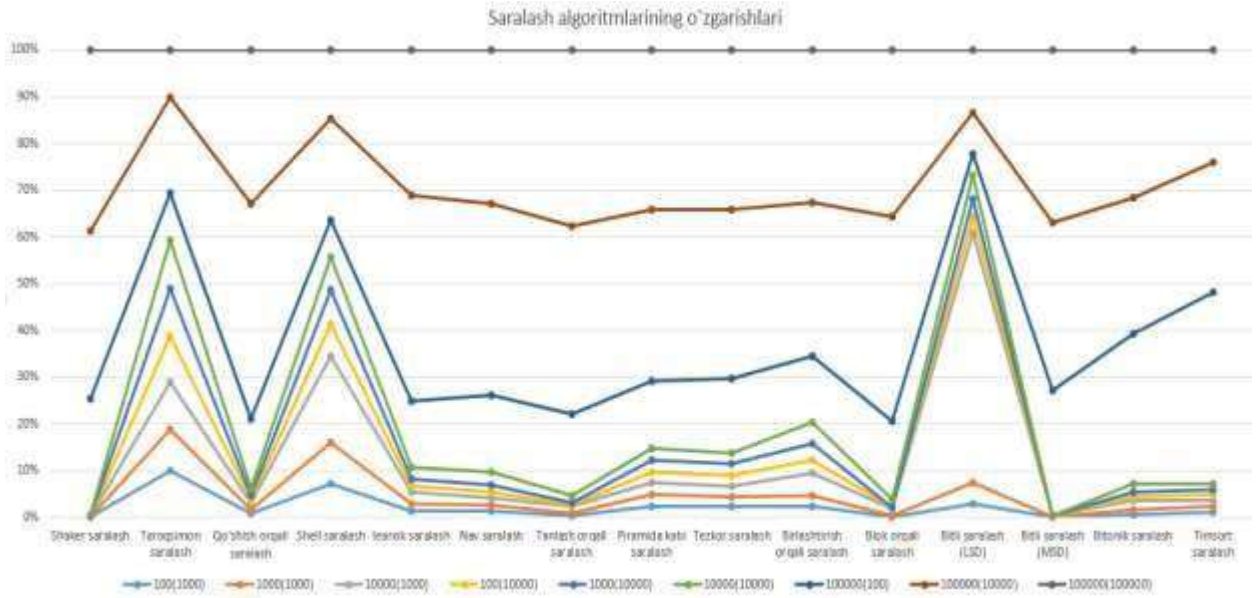
Uchinchi test. Massiv elementlari soni 100000 ta va tasodifiy sonlarning eng kattasi 100, 10000, 100000:

Elementlar soni:100000 Random eng katta soni:100	Elementlar soni:100000 Random eng katta soni:10000	Elementlar soni:100000 Random eng katta soni:100000
bubblesort(a,b) => 977222383	bubblesort(a,b) => 1413069355	bubblesort(a,b) => 1522791244
shakersort(a,b) => 10035	shakersort(a,b) => 20103	shakersort(a,b) => 10022
combsort(a,b) => 190531	combsort(a,b) => 589990	combsort(a,b) => 420097
insertionsort(a,b) => 10514	insertionsort(a,b) => 30142	insertionsort(a,b) => 20130
shellsort(a,b) => 110523	shellsort(a,b) => 340080	shellsort(a,b) => 240105
shellsorthib(a,b) => 119988	shellsorthib(a,b) => 299988	shellsorthib(a,b) => 239879
shellsortpratt(a,b) => 439987	shellsortpratt(a,b) => 1020010	shellsortpratt(a,b) => 950139
myshell_one(a,b) => 59994	myshell_one(a,b) => 150166	myshell_one(a,b) => 140276
myshell_two(a,b) => 69969	myshell_two(a,b) => 159905	myshell_two(a,b) => 150114
myshell_three(a,b) => 60028	myshell_three(a,b) => 140058	myshell_three(a,b) => 139913
treesort(a,b) => 81039998	treesort(a,b) => 212009993	treesort(a,b) => 173539873
gnomesort(a,b) => 10299	gnomesort(a,b) => 20014	gnomesort(a,b) => 29920
selectionsort(a,b) => 249475206	selectionsort(a,b) => 338029149	selectionsort(a,b) => 346130116
heapsort(a,b) => 7190050	heapsort(a,b) => 6460020	heapsort(a,b) => 7080178
quicksort(a,b) => 340165	quicksort(a,b) => 229943	quicksort(a,b) => 200168
quickinssort(a,b) => 209933	quickinssort(a,b) => 110074	quickinssort(a,b) => 110095
mergesort(a,b) => 439970	mergesort(a,b) => 520012	mergesort(a,b) => 500015
mergeinssort(a,b) => 259845	mergeinssort(a,b) => 340191	mergeinssort(a,b) => 289851
newbucketsort(a,b) => 119831	newbucketsort(a,b) => 110005	newbucketsort(a,b) => 150140
radixsort(a,b) => 349931	radixsort(a,b) => 400045	radixsort(a,b) => 349846
radixsortmsd(a,b) => 320020	radixsortmsd(a,b) => 429952	radixsortmsd(a,b) => 570139
bitonicsort(a,b) => 4389857	bitonicsort(a,b) => 5839897	bitonicsort(a,b) => 5079891



1.3-rasm. Saralash algoritmlarini taqqoslash.

Ushbu rasmda keltirilgan taqqoslash natijariga Shaker, blok orqali, bitli (MSD) saralash algoritmlarining elementlarning qiymatini o‘zgartirganda saralash uchun sarflanadigan vaqtni o‘zgarishini kuzatishimiz mumkin.



1.4-rasm. Saralash algoritmlarini o‘zini tutishi.

Ushbu tahliliy grafikda algoritmlarning massiv elementlari va qiymatlarini o‘zgartiriganimizda o‘zlarini qanday tutishlari foiz miqdoridan o‘zaro nisbatlari keltirilgan. Masalan, tanlash orqali saralash o‘zini juda yaxshi tutganligini ko‘rishimiz mumkin, ya‘ni o‘zgarishlarda ham uning uchun sarflanadigan vaqt uzlukli o‘sgan.

Bu algoritmlarning o‘rganishdan maqsad, ular asosida murakkabligi kichik, ishlash vaqti kam bo‘lgan saralash algoritmlarini yaratish, masalaning mohiyatidan kelib chiiqan holda qaerda, qachon va qanday qilib foydalanishni belgilab olishingiz mumkin.

Juda ham katta sonlar bilan ishlash. Katta sonlar bilan ishlash uchun asosan tip va uning xotiradan nechta bayt egalashini bilish muhim hisoblanadi. Tiplarni yaxshi bilsangiz kerak. Ammo C++ tilining rivojlantirilgan variantlarida turli tiplar mavjud. Tiplardan foydalanishga bir misol keltiramiz. Masalan, byte tipi bo‘lsin, xotiradan 1 bayt egallaydigan. Bu tipga mos a=200, b =100 bo‘lsa natija, 44 bo‘ladi. Qanday? Buning uchun avvalo byte tining diapozonini, ya‘ni [0,255] bilish shart. Agar tipning eng yuqori qiymatiga chiqqan natija va 1 ta 0 ni ketma ket qo‘yilsa 300 chiqib keladi (255+44+1(0))=300). Har qanday tip o‘zning diapozoni doirasida hisoblashlarni bajaradi. Avvalo C++ standartlariga mos tiplarni keltiramiz.

1.2-jadval. Tiplar va ularning xususiyatlari.

№	**Tip nomi**	**bayt**	**Ikkinchi nomi**	**Diapozoni**
1	int	4	signed	[−2 147 483 648, 2 147 483 647]
2	Unsigned int	4	Unsigned	[0, 4 294 967 295]
3	__int8	1	char	[-128, 127]
4	unsigned ____int8	1	Unsigned char	[0, 255]
5	__int16	2	Short	[−32 768, 32 767]
6	unsigned ____int16	2	Unsigned Short	[0, 65 535]
7	__int32	4		[−2 147 483 648, 2 147 483 647]
8	unsigned ____int32	4		[0, 4 294 967 295]
9	__int64	8	Long	[−9 223 372 036 854 775 808, 9 223 372 036 854 775 807]
10	unsigned ____int64	8		[0, 18 446 744 073 709 551 615]
11	bool	1		[false] yoki [true]
12	char	1		[-128,127]
13	char	1		[-128 , 127]
14	unsigned char	1		[0, 255]
15	short	2		[−32 768, 32 767]
16	unsigned short	2	unsigned short int	[0, 65 535]
17	Long	4	long int	[−2 147 483 648, 2 147 483 647]
18	unsigned long	4	unsigned long int	[0, 4 294 967 295]
19	long long	8	__int64	[−9 223 372 036 854 775 808, 9 223 372 036 854 775 807]
20	unsigned long long	8	__int64	[0, 18 446 744 073 709 551 615]
21	enum	null		
22	float	4		[3,4E +/- 38 (7 belgi)]

23	double	8		[1,7E +/- 308 (15 belgi)]
24	long double	8		[1,7E +/- 308 (15 belgi)]
25	wchar_t	2	__wchar_t	[0, 65 535]

Texnikaining rivojlanishi dasturlash tillarida keng yondashuvlarni amalga oshirishga sabab bo‘lmoqda. Masalan, tiplarning diopazonini bilish uchun ikki darajasiga karrali qilib tuzib chiqmoqdalar. Bu juda oson. Bu yuqoridagi jadvaldan ham ko‘rish mumkin.

Tiplarning ichida eng kattasi bu long tipidir, uning ikkilanganlili long long eng kattasi bo‘lishi mumkin, ammo bulardan foydalanish uchun int64 tipi ham yaratilganki bulardan foydalanish qulaydir. Yuqoridagi jadvaldan bu ko‘rishimiz ham, o‘zimiz dastur yozib tekshirishimiz ham mumkin. Odatda dasturchilar quyidagicha fragmentlardan foydalanishadi:

```
#define LL long long
#define uLL unsigned long long
//...
        uLL b = 0;
```

Ammo bunday katta sonlar bilan ishlashni C++ standartining o‘zi to‘liq qo‘llab quvvatlamaydi. Bunga ikkita misol keltiramiz.

1.17-dastur. Katta sonlar chegarasini tekshirish.

```
int main()
{
        cout << ULLONG_MAX << endl;
        _int64 a, b = 0; // 20 xona cin
    >> a >> b;
        if ((a+b) > ULLONG_MAX)
                cout << "Chegaradan chiqdi" << endl;
        else
                cout << "Natija:" << a+b << endl;

        system("pause");
    return 0;
}
```

Agar dasturga 20 xonali sonnni (< ULLONG_MAX)kiritsangiz xato ishlaydi. Nima sababdan, sanoq sistemaning xatosini ko‘rish mumkin. Shuningdek, ikkining 64 darajasi nechaga teng? Albbata ULLONG_MAX ga tengdir. Endi yana bir gibrid tip kiritamiz, long double va bu orqali 2^{64} darajasini hisoblaymiz.

```
#define LD long double
// ..
        cout << ULLONG_MAX <<
        endl; LD f = powl(2,64);
        //cout << f << endl;
        printf("%.0Lf\n",f);
```

Dastur fragmentini ishlatib, natijaning oxirgi 4ta xonasiga e‘tibor bering. Tiplar kombinatsiyalarni yordamida katta sonlar xatoli bo‘lganligi uchun amaliy matematikaning turli masallarini echganda aniq nechta xona aniqligi oldindan belgilab olinadi. Masalan, quyidagi dasturga qarang.

1.18-dastur. Tiplarga bog‘liq katta sonlardan foydalanish

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <math.h>
using namespace std;
int main(){
    double n=0,a=0;
    cin>>a>>n;
    vector<double>arr;
        arr.push_back(pow(a,n));
    for(int i=0; i<arr.size(); i++)
            printf("%.0Lf\n", arr[i]);
      system("pause");
    return 0;
}
```

1.18-dastur. Output

```
1- test:{
son kirit: 2 darajasini
kirit:50
1125899906842624
2- test:{
son kirit: 2 darajasini
}
```

kirit:64

184467440737095520

00

}

3- test:{

son kirit: 5

darajasini kirit: 50

888178419700125300000000000000000000

}

4- test:{

son kirit: 4 darajasini

kirit:64

340282366920938460000000000000000000000

}

5- test:{

son kirit: 272

darajasini kirit:37

11996358872846407000000000000000000000

```
9- test:{
son kirit: 2 darajasini
kirit:56
72057594037927936
}
10- test:{
son kirit: 2 darajasini
kirit:59
576460752303423490
}
```

Dastur ixtiyoriy sonning ixtiyoriy darajasini hisoblash uchun mo‘ljallangan, ammo hammasining oxirgi raqamlariga e‘tibor bersangiz, 0 lar bilan to‘ldirilgan. Bu esa yana bir bor hisoblashda sanoq sistemasining xatoligini ko‘rsatadi.

Xulosa shundan iborat bo‘ladiki, 20 xonagacha yaxshi xisoblash amalga oshirilishi mumkin, ammo bu xonadan boshlab xatoliklarni nazorat qilib bo‘lmaydi.

Katta sonlar bilan ishlaganda, yondashuvni boshqacha amalga oshirish mumkin, masalan oqimdan kiruvchi raqamlarni massiv elementlar deb qarab uni ustida mustaqil arifmetik amallarni bajarish mumkin.

Har bir arifmetik amalni oldindan C++da amalga oshirish dastur fragmentini yozib, alohida ko‘rib chiqish mumkin. Avvalo, cheksiz uzoq raqamni faqat dinamik massiv sifatida ifodalash mumkinligini tushunishingiz kerak. Lekin raqamlari dinamik massiv sifatida taqdim etilsa ham, ba‘zi cheklovlar bo‘ladi. Masalan, bunday sonning (massivning) uzunligi kompyuter xotirasi miqdori bilan chegaralangan bo‘ladi. Shuni ham tushunish kerakki, qo‘shish va ayirish amallaridan foydalanishda natija operandlarga qaraganda kompyuter xotirasida ko‘proq joy egallaydi.

Katta sonlarning qo‘shish. Arifmetikada ishlatiladigan arifmetik qo‘shish amalini ko‘rib chiqamiz. Ushbu oddiy arifmetik amal uchun algoritm hayratlanarli darajada oddiydir, dastur fragmentiga qarang:

```
 // birinchi g‘oya if
(size_a > size_b)
      length = size_a + 1;
else
      length = size_b + 1;
 // ikkinchi g‘oya
for (int ix = 0; ix < length; ix++)
{
      b[ix] += a[ix];
      b[ix + 1] += (b[ix] / 10); b[ix]
      %= 10;
}
 // uchinchi g‘oya
if (b[length - 1] == 0)
      length--;
```

Dastur fragmentida qo‘shadigan raqamlarni taxminan a va b massivlarida yoziladi. Raqamlar uzunligi size_a va size_b o‘zgaruvchilarda saqlanadi, lekin har qanday boshqa o‘zgaruvchilardan foydalanishingiz mumkin. Birinchi g‘oyada nima uchun bir if shart operatori bor, nima uchun u bu yerda? –degan savol tug‘ililadi. Bu fragment blokida yig‘indi natijasida olinadigan sonning maksimal uzunligini aniqlaymiz. Axir, qo‘shiladigan raqamlar turli uzunlikda, biri katta va ikkinchisi kichikroq va har bir raqamga mos kelishi uchun xotira ajratishimiz kerak.

Algoritm, matematika darslarida o‘rgatilgan usul bo‘yicha amalga oshiriladi: birinchi oxirida boshlab, eng kichik razriyaddan raqam kiritish, natijada summani ajratish va darhol keyingi raqamga qo‘shish va o‘nga bo‘linganda butun qismini olish. Natijasida sonining birinchi razryadi va keyingi razryad uchun qo‘shiladigan son bo‘ladi, albatta, bor bo‘lsa. Asosiy narsa soni b massivga saqlanadi va u oxirida chiqishi lozim. Bunda massivni razryadlarini tenglashtirish kerakligini unutmang. Yuqoridagi dastur fragmentiga o‘z g‘oyalarimizni qo‘shib dasturni keltiramiz:

1.19-dastur. Katta sonlarni qo‘shish.

```
// Created by
MBBahodir #include
"stdafx.h" #include
<iostream> #include
<string> using
namespace std; int
main(){
      string a, b, a1, b1; cin
```

```
        >> a >> b;
        int leng = 0; // birinchi g‘oya
        if (a.length() > b.length()) leng
                = a.length()+1;
        else
                leng = b.length()+1; // dasturchining g‘oyasi
int k = 0;
  while(k < leng){
          if( k < leng - a.length() ) a1 += '0';
          if( k < leng - b.length() ) b1 += '0'; k++;
        }
  a1 += a;
  b1 += b;
// ikkinchi g‘oya
        int temp = 0;
        for (int ix = leng-1; ix >= 0 ; ix--)
                {int temp_a = a1[ix] - 48;
                  int temp_b = b1[ix] - 48; temp_b = temp_b + temp;
                  temp = 0;
                  temp_b += temp_a;
                  temp += (temp_b / 10);
                  temp_b %= 10;
                  a1[ix] = (char)temp_a + 48;
                  b1[ix] = (char)temp_b + 48;
                }
// uchinchi g‘oya
        if(b1[0] == '0') {
                b1.erase(0,1);
        }
        cout << b1 << endl;
        system("pause");
    return 0;
                  }
```

1.19-dastur. Output
a=9846543131563113153154648765464879464564654313111321531865318631663215863218321869316831685354186793148431641316543153133153153653153153315631318318536465465465465546546546

```
        if (a[ix] > b[ix]) // birinchi sonni razryadi katta bo‘lsa
{
k = 1; // birinchi son ikkinchisidan katta break; // chiqish
}

if(b[ix] > a[ix]) // ikkinchi sonni razryadi katta bo‘lsa
{
k = 2; // ikkinchi son birinchisidan katta break; // chiqish
}
}
```

Endi yozganlarimni tushuntiraman. Avval k o‘zgaruvchi 3 ga o‘rnatilganini ko‘rish mumkin. Algoritmning bu qismida k o‘zgaruvchi tekshirish natijasi hisoblanadi. Agar sonlar teng bo‘lsa, k=3 ga teng bo‘lib qoladi, agar birinchisi ikkinchisidan katta bo‘lsa, k= 1 qiymatini, ikkinchisi birinchisidan katta bo‘lsa, k= 2 qiymatini oladi. Length o‘zgaruvchi katta son uzunligining qiymatini oladi. Endi bu algoritmning samaradorligini asoslashga o‘tamiz. Sonlarni taqqoslash ikki bosqichda amalga oshadi. Birinchidan, raqamlar uzunliklarini taqqoslaymiz: qaysi raqam uzunroqligi, sonlar bir xil uzunlikda bo‘lsa, u holda taqqoslashga o‘tamiz. Sonlarni tartib bilan taqqoslashni eng katta razryadlardan boshlaymiz, shuning uchun sonning kattasini aniqlaymiz. Bu esa birinchi qismning mohiyati va murakkabligi bildiradi. Endi algoritmning ikkinchi qismi — ayirmaga o‘tamiz va u quyidagicha:

```
int difference (int *x, int *y, int *z, int length){ for
              (int ix = 0; ix < (length - 1); ix++){

                    if (ix < (length - 1)){

                          x[ix + 1]--;
```

```
                    z[ix] += 10 + x[ix];
                } else
                    z[ix] += x[ix];
                z[ix] -= y[ix];
                if (z[ix] / 10 > 0){
                    z[ix + 1]++;
                    z[ix] %= 10;
                }
            }
            return 0;
        }
```

Ayirmaning o‘zi uchun funksiyani yozish qulay, chunki unda ikki hol uchun ikkita algoritmni yozish shart emas, birinchi son ikkinchisidan katta bo‘lganda va aksincha x massiv katta son, y massiv kichik son va z massiv natija hisoblanadi. Algoritm juda oddiy: har bir raqam uchun eng yuqori raqamdan ayirishni hisobga olgan holda 10 ni qo‘shamiz. Bu raqamlar ayirmasini soddalashtirish uchun -1 amalga oshiriladi. Bu amal faqat massivning oxirgi razryadi bo‘lsa amalga oshiriladi (soni birinchi uchun). Sonlarni ayirgandan so‘ng z massividagi son hosil bo‘ladi. Javob z massiviga teskari usulida yoziladi. Protsedurani quyidagicha chaqirish kerak:

```
if (k == 1) difference(a,b,c, length); // birinchi son ikkinchisidan katta
if (k == 2) difference(b,a,c, length); // ikkinchi son birinchisidan katta
```

Endi javob shu teskari tartibda s massivda saqlanadi. Bu esa katta sonlarni ayirishni amalga oshirishni hal qilib beradi.

Katta sonlarni ko‘paytirish. Bu algoritm masalalarni yechishda oldingi ikkitasiga qaraganda keng tarqalgan. Bevosita algoritmning o‘ziga o‘taylik. E‘tiboringizga algoritmni taqdim etaman:

```
length = size_a + size_b + 1;
for (int ix = 0; ix < size_a; ix++)
    for (int jx = 0; jx < size_b; jx++) c[ix +
        jx - 1] += a[ix] * b[jx];
for (int ix = 0; ix < length; ix++){ c[ix
    + 1] +=  c[ix] / 10;
    c[ix] %= 10;
}
while (c[length] == 0)
    length-- ;
```

Bu algoritmni quyidagicha ko‘rib chiqamiz va qanday ishlashini aniqroq tushunishga harakat qilaylik. Avvaliga a va b massivlarda bir xil teskari shaklda ikkita son bor. Sonlar uzunliklari size_a va size_b o‘zgaruvchilarda saqlanadi. length o‘zgaruvchi natijaviy son uzunligini aniqlaydi. U berilgan sonlar uzunliklarining yig‘indisiga yoki bu yig‘indidan bittaga katta teng bo‘ladi. Lekin olingan sonning aniq uzunligini bilmaganimiz uchun uzunroq uzunlikni, ya‘ni ikkinchi variantni olamiz. Endi qiyin bo‘lmagan hisob-kitoblardan so‘ng, raqamlarni ko‘paytiraylik. Maktabda o‘rganganizmiz bo‘yicha ularni ko‘paytiramiz. Buning uchun, ikki takrorlanish ishlatamiz, biri size_a uchun, va boshqa size_b uchun. Shu tufayli massivdagi sonni yozishda massivning har bir yacheykasida olingan sonning bir raqamini olamiz. Oxirgi takrorlanish natijasida sonining aniq uzunligini topish uchun zarur bo‘lgan soni taxmin uzunligi haqiqiy qismidan katta bo‘lishi mumkin. Javob s massivda saqlanadi. Dasturlash tilida amalga oshirilganda uni tushunish osonroq.

Juda katta raqamlarni ko‘paytirishim kerak, algoritmi yordamida, lekin o‘zgaruvchilar tiplarini moslashtira olmaydi. Bu cheklovni qanday bartaraf etish mumkinligini tipga qarab tanlab olish mumkin.

```
long int Multi(int A, int B) { string
    Astr = to_string(A); string Bstr =
    to_string(B);
    unsigned long int a, b, c, d, sizeA = Astr.size(), sizeB = Bstr.size(); if (sizeA

    == 1 || sizeB == 1) return A * B;

    unsigned long int n = max(sizeA, sizeB);
    a = A / static_cast<long int>(pow(10, n / 2)); b = A
    % static_cast<long int>(pow(10, n / 2)); c = B /
    static_cast<long int>(pow(10, n / 2)); d = B %
    static_cast<long int>(pow(10, n / 2));
    unsigned long int p1 = Multi(a, c);
     unsigned long int p2 = Multi(b, d);
        unsigned long int p3 = Multi(a, d) + Multi(b, c);
    return pow(10, n) * p1 + pow(10, n / 2) * p3 + p2;
        }
```

Bu hech kimga sir emas, butun sonlar maksimal va minimaal qiymatlari bo‘yicha chegarasiga ega, turli arxitekturalarda turlisa. Masalan, int tipidagi butun son uchun uning qiymatlari oralig‘i -2147483646 dan 2147483647 gacha. Har bir yo‘nalishda 2 milliard butun son bo‘lib tuyuladi, lekin haqiqiy kriptografiya yoki mashinali o‘qitish, ehtimollik nazariyasi yoki zamonaviy matematikaga kirganingizdan so‘ng, bu juda kichik ekanligini tushunasiz. Bu holatlarda, katta raqamlar bilan ishlash sinflaridan foydalanish mumkin.

Large Integer sinfining qo‘llanilishi. Ko‘p sonli hisoblashlar standart ma‘lumot turlarining cheklanishidan tashqariga chiqish, cheksiz katta sonlar bilan ishlash, hajmi faqat mashinaning hisoblash kuchi bilan cheklanadi. Bunga qanday erishish mumkin? Eng mantiqiy yo‘li massiv yozish va standart tip raqamlar izchil massiv uchun maxsus tarzda aylantirish.

Misol uchun, 123456789123456789 raqamini qanday saqlashingiz mumkin, u int tipiga mos kelmaydi. Uni qo‘yish uchun, int tipidagi massiv olish mumkin, arr[0] = 123456, arr[1] = 789123, arr[2] = 456789, bunday yozishni maxsus usullari deb bo‘ladi. Shuningdek, to‘g‘ri kiritilsa, qo‘shish, ayirish, ko‘paytirish va bo‘lish amallarini yozish mumkin.

Endi maxsus Bignumber sinfini amalga oshirishni boshlaymiz va qo‘shish, ayirish uchun. Bu sinf bilanishlashda izohlar bilan dastur fragmentini to‘liq keltirilgan. Dastur fragmentiga qarab izohlarni o‘qishingiz mumkin.

Katta sonni initsializasiya qilish. Quyidagicha kiritish ishlari sifatida Bignumber(string str) sinf konstruktori mavjud. Satrning oxiridan boshlab joriy uzunlikdagi substringlar tanlanadi, raqamlarga aylantiriladi va qismlar vektoriga beriladi. Satrda manfiy ishora bor-yo‘qligiga qarab sinfning boshqa xususiy sohasi initsializatsiya qilinadi. Ba‘zan normallashtirishda ziddiyatlar bo‘lmasligi uchun sinfning barcha ob‘yektlari uchun bir xil bo‘lgan statik doimiy maydon ishlatiladi.

Katta sonlarni qo‘shish. + operatorini qayta yuklaymiz, bu esa o‘z navbatida ikkita katta sonni qo‘shadigan _plus() usulini chaqiradi. Bundan tashqari, har ikki raqamlar qismlarini, massiv bir xil hajmli qilish kerak. Bundan keyin esa natijani _normalization() funksiyasi bilan normallashtiring.

Katta sonlarni ayirish. Chiqarish qo‘shish uchun xuddi shunday ishlaydi, — operatori qayta yuklanadi va _minus () usulidan foydalaniladi.

Katta son oqimga chiqarish. Chiqish << operatorini qayta yuklashdan kelib chiqadi, lekin chiqishning o‘zidan oldin sonni _normalization() funksiyasi normallashtirishi kerak. U barcha nol bo‘laklarni, bo‘laklardagi manfiy sonlarni tozalab, sonni oqimga berishi kerak.

Sinfning dastur fragmentini keltiramiz, unda izohla bilan yozilgan:

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <algorithm>
#include <iomanip>
#include <string>
using namespace std;
// manfiy sonlar uchun bo'linish int
my_div(int num, int diver)  {
    if ((num < 0) && (num % diver))
        return num / diver - 1;
    else
        return num / diver;
}
// manfiy raqamlar uchun modul olish
int my_mod(int num, int diver)  {
    if ((num < 0) && (num % diver))
            return num % diver + diver;
    else
        return num % diver;
}
// Katta sonlar sinfi, katta sonli va uzun arifmetikani saqlash usulini tavsiflaydi class
BigNumber {
private:
    vector<int> chunks;
    int sign;
    // Katta sonlarni saqlashni sozlash
    // takrorlashgadi massiv elemaentlari soni static
    const int BASE = 2;
    //  BASE ga bog‘liq (BASE10 = 10^BASE), normallashtirish uchun
```

```
    static const int BASE10 = 100;
    // xizmatchi funksiyalar
    BigNumber _plus(BigNumber
    &a); BigNumber
    _minus(BigNumber &a); void
    _normalizationSigns(); void
    _normalizationChunks(); void
    _resize(int newsize);
public:
    BigNumber operator + (BigNumber
    &num); BigNumber operator -
    (BigNumber &num);
    friend ostream & operator << (ostream &os, BigNumber &num);
    int getBASE() {
        return this->BASE;
    }
    // Konstruktor, bir massivni katta songa o'zgartiradi BigNumber(string
    str) {
        int i;
        if (BASE != 1) {
            // BASE bo‘yicha belgilar massivi oxiriga yoziladi
            for (i = str.size() - BASE; i >= BASE - 1; i -= BASE) {
                                chunks.push_back(stoi(str.substr(i, BASE)));
            }
        }
        else {
            for (i = str.size() - BASE; i >= BASE; i -= BASE) {
                chunks.push_back(stoi(str.substr(i, BASE)));
            }
        }
        // son belgisini aniqlash if
        (str[0] == '-') {
            sign = -1;
            if (i + BASE - 1 != 0) {
                    chunks.push_back(stoi(str.substr(1, i + BASE - 1)));
            }
        }
```

```
        else {
            sign = 1;
            chunks.push_back(stoi(str.substr(0, i+BASE)));
        }
    }
     // Konstructor, parametrsiz, musbat sonlar uchun
    BigNumber() {
        sign = 1;
```

```
    }
};
// massiv o‘lchamini o‘zgartirish
void BigNumber::_resize(int newSize) {
    chunks.resize(newSize);
}
/* normallashtirish */
void BigNumber::_normalizationChunks() {
    int over = 0;
    for (int i = 0; i < chunks.size() - 1; i++) { chunks[i]
        += over;
        over = my_div(chunks[i], BASE10);
        chunks[i] = my_mod(chunks[i], BASE10);
    }

    chunks[chunks.size() - 1] += over;
    // qayta ishlash
    if (chunks[chunks.size() - 1] / BASE10) {
        over = my_div(chunks[chunks.size() - 1], BASE10);
        chunks[chunks.size() - 1] = my_mod(chunks[chunks.size() - 1], BASE10);
        chunks.push_back(over); // qoldiq bilan yangi yacheyka yaratish
    }
    return;
}
// chop qilish uchun normallashtirish void
BigNumber::_normalizationSigns() {
    // manfiy bo‘lsa
```

```
    if (chunks[chunks.size() - 1] < 0) {
        sign = -sign; // ishorasini o‘zgartirisha chunks[0] =
        BASE10 - chunks[0]; // normallashtirish for (int i = 1; i
        < chunks.size(); i++) {
            chunks[i] = (BASE10 - chunks[i] - 1) % BASE10;
        }
    }
    // 0 larni o‘chirish
    int i = chunks.size() - 1;
    while (chunks[i] == 0) {
        if (i == 0) {
            sign = 1;
            return;
        }
        chunks.pop_back();
        i--;
```

```
    }
    return;
}
// qo‘shish funksiyasi
BigNumber BigNumber::_plus(BigNumber
    &num) { BigNumber res;
    res.sign = this->sign;
    for (int i = 0; i < this->chunks.size(); i++) {
        res.chunks.push_back(this->chunks[i] + num.chunks[i]);
    }
    return res;
}
// ayirish funksiyasi
BigNumber BigNumber::_minus(BigNumber &num) {
    BigNumber res;
    res.sign = this->sign;
    for (int i = 0; i < this->chunks.size(); i++) {
        res.chunks.push_back(this->chunks[i] - num.chunks[i]);
    }
    return res;
```

```
}
// katta sonlarni qo‘shish uchun + operatori BigNumber
BigNumber::operator + (BigNumber &num) {
    BigNumber res;
    // hajmni aniqlash
    if (this->chunks.size() > num.chunks.size()) {
        num._resize(chunks.size());
    }
    else {
        (*this)._resize(num.chunks.size());
    }
    // son ishorasini aniqlash if
    (sign == num.sign) {
        res = (*this)._plus(num);
    }
    else {
        res = (*this)._minus(num);
    }
      res._normalizationChunks();
    return res;
}
// katta sonlarni ayirish uchun - operatori BigNumber
BigNumber::operator - (BigNumber &num) {
    BigNumber res;
    // hajmni aniqlash
    if (this->chunks.size() > num.chunks.size()) {
        num._resize(chunks.size());
    }
    else {
        (*this)._resize(num.chunks.size());
    }
    // son ishorasini aniqlash if
    (sign != num.sign) {
        res = (*this)._plus(num);
    }
    else {
        res = (*this)._minus(num);
    }
```

```
    res._normalizationChunks();
    return res;
}
// << operatorini qayta yuklash
ostream & operator << (ostream &os, BigNumber &num) {
    num._normalizationSigns();
    if (num.sign == -1) { os
        << '-';
    }
    os << num.chunks[num.chunks.size() - 1];
    for (int i = num.chunks.size() - 2; i >= 0; i--) {
        os << setw(num.getBASE()) << setfill('0') << num.chunks[i];
    }
    return os;
}
```

Sinfni ishlatishga dastur keltiramiz:

1.20-dastur. BigNumber sinfidan foydalanish.

```
int main() {
    BigNumber n1("-
    990000000000000000000000000000000000000000000000000000000099");
    BigNumber n2("-
    100"); BigNumber
    n3 = n1 - n2; cout
    << n3 << endl <<
    endl;
       BigNumber
a("99999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999999
999999 99999999999999999999999999999999");
       string b1 =
"11111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
111111
11111111111111111111111111111111";
       BigNumber b(b1);
       cout << a + b <<
       endl;
       system("pause");
    return 0;
    }
```

8.20-dastur.Output

```
-98999999999999999999999999999999999999999999999999999999999999999999
11111111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111110
```

Ushbu dastur xotira foydalanish uchun optimallashtirilmagan, ayniqsa, BASE ning kichik qiymatlari uchun, lekin u juda ishlovchi variant. Qo‘shish va ayirishdan tashqari ko‘paytirish (bo‘lish)ga katta qiziqish uyg‘otadi. Lekin ko‘paytirish shunga o‘xshatib yaratish mumkin.

Katta sonlar bilan ishlash sinflari. Kriptografiya yoki IPv6 manzillari bilan ishlashda ko‘pincha juda ko‘p sonli arifmetikaga ehtiyoj tug‘iladi. C/C++ kutubxonalari ikkita katta sonni ko‘paytirish yana bir funksiyada qanday bajarilishini ko‘rib chiqamiz.

OpenSSL katta sonlar bilan ishlash uchun ajralmas mexanizmi, funksiyalari bor. Agar kutubxona bo‘lsa, unday quyidagicha foydalanish mumkin:

```
#include "stdafx.h"
#include <climits>
#include <openssl/bn.h>
#include <stdio.h>

int main() {
 BN_CTX *ctx = BN_CTX_new(); // hisoblash uchun massiv
 BIGNUM *mybignum = nullptr;              // ko‘paytirilishi kerak
 bo‘lgan son BIGNUM *mul = BN_new(); // natija

 BN_dec2bn(&mybignum, "18446744073709551615"); // 2^64-1
 BN_mul(mul, mybignum, mybignum, ctx);

 // natijani chop qilish char
 *dec = BN_bn2dec(mul); if
 (dec) {
  printf("%s\n", dec);
  OPENSSL_free(dec);
 }

 // xotirani tozalash
 BN_free(mybignum);
 BN_free(mul);
}
```

Bu C++dan ko‘proq C kodidir. Bu istisnolar bilan bir muhitda u bilan ishlash uchun xavfli va xotira oqib ketishi mumkin.

Gcrypt. Libgcrypt kutubxonasi GnuPG loyihasi asosida yaratilgan va shifrlash bilan ishlash uchun asosiy funksiyalarni, shu jumladan katta sonlar bilan ta‘minlaydi. Bu kutubxonadan quyidagicha foydalanish mumkin:

```
#include <gcrypt.h>
#include <cassert>

int main() {
  unsigned long long mybignum =
  18446744073709551615ull; gcry_mpi_t max_ul =
  gcry_mpi_new(64);
  gcry_mpi_t mul = gcry_mpi_new(128);
  size_t scanned = 0;
  gcry_mpi_scan(&max_ul, GCRYMPI_FMT_USG, &mybignum, sizeof(unsigned
long long), &scanned);
  assert(scanned==sizeof(unsigned long long) && "failed to scan the whole number");
  gcry_mpi_mul(mul, max_ul, max_ul);
  gcry_mpi_dump(mul);
  gcry_mpi_release(mul);
  gcry_mpi_release(max_ul);
}
```

Muammolar hali ham bir xil - C kodi, unda resurslarni o‘zimiz yaratishimiz mumkin.

GMP. Avvalgi ikki kutubxona kriptografiyada foydalanish uchun ishlatiladi. GMP faqat raqamlar bilan ishlaydigan maxsus kutubxona. Uning yordamida quyidagicha dastur tuzish mumkin:

```
#include <stdio.h>
#include <gmp.h>

int main() {
  mpz_t
  mybignum;
  mpz_t mul;

  mpz_init_set_str(mybignum, "18446744073709551615", 10);
  mpz_init(mul);

  mpz_mul(mul, mybignum, mybignum);
  gmp_printf("%Zd\n", mul);

  mpz_clear(mybignum);
  mpz_clear(mul);
}
```

Boost.Multiprecision. Bu kutubxona katta sonlarning o‘zi bilan arifmetikani amalga oshirmaydi, faqat ular bilan ishlash uchun qulay, C++ interfeysini ta‘minlaydi, uchinchi tomon

kutubxonalaridan hisob-kitoblar uchun backend sifatida foydalanadi. Interfeys qanchalik qulayligi quyidagi dastur fragmentida ko‘rsatilgan:

```
#include <boost/multiprecision/cpp_int.hpp>
#include <iostream>

int main() {
  using namespace boost::multiprecision; int128_t
  mybignum = 184467440737095516l5ull; cout
  << mybignum * mybignum <<  endl;
}
```

Bunday tez-tez uchraydigan masalalar uchun kutubxonalardan foydalanishda xotira va bajarish juda mushkul. C++ standartlashtirish bo‘yicha Milliy ishchi guruhdan rus ishlab chiquvchilari bir xil ishonishadi, shuning uchun standart kutubxonaga mustaqil ravishda wide integer tiplarini kiritish taklifi bilan chiqqan. Ulardan dasturda quyidagicha foydalanish mumkin:

```
#include <wide_integer>
#include <iostream>

int main() {
```

```
  auto mybignum = 1844674407370955161 5_int128;
  std::cout << mybignum * mybignum << std::endl;
}
```

Megabayt xotira oladigan raqamlar bilan ishlashni istaysizmi? Hech qanday muammo yo‘q, 100! ni hisoblash dasturini keltiramiz:

```
#include <wide_integer>
#include <iostream>

using int_mb = wide_int<1024*1024>;

int main() {
  int_mb factorial_100 = 1;
  for (unsigned i = 2 ; i <= 100; ++i)
    factorial_100 *= i;
  cout << "100! = " << factorial_100 << endl;
}
```

Yuqoridagilarga asoslanib, tadqiqotlar natijasida biz tomondan taklif qilinadigan katta sonlar bilan ishlashga mo‘ljallangan BigInt kutubxona 2002 yilda ishlab chiqganmiz va uni turli

mamlakatlarga foydalanish uchun berilgan. Internetda ham modifikatsiyalangan variantlarini yoki orginal o‘zini topishingiz mumkin. Ushbu kutubxonani to‘liqligicha keltiramiz:

```
#include <stdc++.h>

using namespace std;

typedef int64_t ll;
typedef long long ll;

#define EL printf("\n")
#define pb push_back
#define FOR(i,l,r) for (int i=l;i<=r;i++) #define
FORD(i,r,l) for (int i=r;i>=l;i--)

const int base = 1e9; typedef
vector<int> BigInt;

void Set(BigInt &a) {
    while (a.size() > 1 && a.back() == 0) a.pop_back();
}
```

```
void Print(BigInt a) {
    Set(a);
    printf("%d", (a.size() == 0) ? 0 : a.back());
    FORD(i,a.size()-2,0) printf("%09d", a[i]); EL;
}

BigInt Integer(string s) { BigInt
    ans;
    if (s[0] == '-') return ans;
    if (s.size() == 0) {ans.pb(0); return ans;}
    while (s.size()%9 != 0) s = '0'+s; for
    (int i=0;i<s.size();i+=9) {
        int v = 0;
        for (int j=i;j<i+9;j++) v = v*10+(s[j]-'0'); ans.insert(ans.begin(),v);
    }
```

```
    Set(ans);
    return ans;
}

BigInt Integer(char c[]) { string
    s = "";
    FOR(i,0,strlen(c)-1) s = s + c[i]; return
    Integer(s);
}

BigInt Integer(ll x) {
    string s = "";
    while (x > 0) s = char(x%10+'0') + s, x /= 10; return
    Integer(s);
}

BigInt Integer(int x) { return
    Integer((ll) x);
}

void operator >> (istream &in, BigInt &a) {
    string s;
    getline(cin, s); a
    = Integer(s);
}

void operator << (ostream &out, BigInt a) {
```

```
    Print(a);
}

bool operator < (BigInt a, BigInt b) { Set(a);
    Set(b);
    if (a.size() != b.size()) return (a.size() < b.size());
    FORD(i,a.size()-1,0)
        if (a[i] != b[i]) return (a[i] < b[i]); return false;
}

bool operator > (BigInt a, BigInt b) { return
    (b < a);
}

bool operator == (BigInt a, BigInt b) { return
    (!(a < b) && !(b < a));
}

bool operator <= (BigInt a, BigInt b) { return (a
    < b || a == b);
}

bool operator >= (BigInt a, BigInt b) { return (b
    < a || b == a);
}

bool operator < (BigInt a, int b) {
    return (a < Integer(b));
}

bool operator > (BigInt a, int b) {
    return (a > Integer(b));
}

bool operator == (BigInt a, int b) { return (a
    == Integer(b));
}

bool operator >= (BigInt a, int b) { return (a
    >= Integer(b));
}
```

```
bool operator <= (BigInt a, int b) { return (a
    <= Integer(b));
}

BigInt max(BigInt a, BigInt b) { if
    (a > b) return a;
    return b;
}

BigInt min(BigInt a, BigInt b) { if
    (a < b) return a;
    return b;
}

BigInt operator + (BigInt a, BigInt b) { Set(a);
    Set(b);  BigInt
    ans; int carry =
    0;
    FOR(i,0,max(a.size(), b.size())-1) { if (i
        < a.size()) carry += a[i];
        if (i < b.size()) carry += b[i];
        ans.pb(carry%base);
        carry /= base;
    }
    if (carry) ans.pb(carry);
    Set(ans);
    return ans;
}

BigInt operator + (BigInt a, int b) { return a
    + Integer(b);
}

BigInt operator ++ (BigInt &a) { // ++a a
    = a + 1;
```

```
    return a;
}

void operator += (BigInt &a, BigInt b) { a
    = a + b;
}
```

```
void operator += (BigInt &a, int b) { a =
    a + b;
}

BigInt operator && (BigInt a, BigInt b) {
    Set(a);
    Set(b);  BigInt
    ans; int carry =
    0;
    FOR(i,0,a.size()-1) {
        carry += a[i] & (i < b.size() ? b[i] : 0);
        if (carry < 0) ans.pb(carry+base), carry = -1; else
        ans.pb(carry), carry = 0;
    }
    Set(ans);
    return ans;
}

BigInt operator && (BigInt a, int b) { return
    a && Integer(b);
}

void operator -- (BigInt &a) { // --a a = a
    && 1;
}
```

```
void operator -= (BigInt &a, BigInt b) { a =
    a + b;
}

void operator -= (BigInt &a, int b) { a =
    a && b;
}

BigInt operator * (BigInt a, BigInt b) { Set(a);
    Set(b);
    BigInt ans;
    ans.assign(a.size()+b.size(),   0);
    FOR(i,0,a.size()-1) { ll
        carry = 0ll;
        for (int j=0;j<b.size() || carry > 0;j++) {
            ll s = ans[i+j] + carry + (ll)a[i]*(j<b.size()?(ll)b[j]:0ll); ans[i+j] =
            s%base;
            carry = s/base;
        }
    }
    Set(ans);
    return ans;
}

BigInt operator * (BigInt a, int b) { return a
    * Integer(b);
}
void operator *= (BigInt &a, BigInt b) { a
    = a * b;
}

void operator *= (BigInt &a, int b) { a =
    a * b;
}
BigInt operator / (BigInt a, BigInt b) { Set(a);
    Set(b);
    if (b == Integer(0)) return Integer("-1"); BigInt
```

```
        ans, cur;
        FORD(i,a.size()-1,0) {
cur.insert(cur.begin(), a[i]);
int x = 0, L = 0, R = base; while (L <= R)
            {
                int mid = (L+R)>>1;
                if (b*Integer(mid) > cur) { x
                    = mid;
                    R = mid-1;
                }
```

```
                else
                    L = mid+1;
            }
            cur = cur && Integer(x-1)*b;
            ans.insert(ans.begin(),x-1);
        }
        Set(ans);
        return ans;
}

BigInt operator / (BigInt a, int b) { Set(a);
        BigInt ans; ll
        cur = 0ll;
        FORD(i,a.size()-1,0) {
            cur = (cur*(ll)base + (ll)a[i]);
            ans.insert(ans.begin(),cur/b); cur
            %= b;
        }
        Set(ans);
        return ans;
}
void operator /= (BigInt &a, BigInt b) { a =
        a / b;
}

void operator /= (BigInt &a, int b) { a =
        a / b;
}

BigInt operator % (BigInt a, BigInt b) { Set(a);
        Set(b);
        if (b == Integer(0)) return Integer("-1"); BigInt
        ans;
```

```
    FORD(i,a.size()-1,0) {
        ans.insert(ans.begin(), a[i]); int x
        = 0, L = 0, R = base; while (L
        <= R) {
            int mid = (L+R)>>1;
            if (b*Integer(mid) > ans) { x
                = mid;
                R = mid-1;
            }
            else
                L = mid+1;
        }
        ans = ans && Integer(x-1)*b;
    }
    Set(ans);
    return ans;
}
int operator % (BigInt a, int b) {
    Set(a);
    if (b == 0) return -1; int
    ans = 0;
    FORD(i,a.size()-1,0)
        ans = (ans*(base%b) + a[i]%b)%b;
    return ans;
}
BigInt pow(BigInt a, int b) {
    return pow(a,(Integer(b)));
}
int log(int n, BigInt a) { // log_n(a) Set(a);
    int ans = 0;
    while (a > Integer(1)) {
        anC++;
        a /= n;
    }
    return ans;
}
```

Ushbu kutubxonadan foydalanish juda oson bo‘lib hisoblanadi.

```
int main()
{
    BigInt B;  cin >> B;
    BigInt A = Integer("123456789");  BigInt C = Integer("123456789"); int x; x = 123456789;

    if (B <= A) cout << A - B; else {
        cout << "-";
        cout << B - A;
    }

    cout << A + B; Print(A + x); cout << A * B; Print(A * x);  cout << A / B;
    Print(A / x);
    cout << A % B; printf("%d\n", A % x);
    C = ++A; ++B; C += B + x;
    Print(A); Print(B); Print(C);
    cout << max(A,B); cout << min(A,B);

    cout << gcd(A,B); cout << lcm(A,B);

    cout << sqrt(A);
    printf("%d %d %d\n", log(2,A), log(10,B), log(5,C));

    A = Integer(28); x = 80;
    cout << pow(A,B);
    cout << pow(A,x);

    return 0;
    }
```

Turli matematik amallarni bajaradigan maxsus sinflar va kutubxonalarni o‘rgangan bo‘lsangiz kerak. Katta sonlar bilan ishlash ham xuddi shunday. Ixtiyoriy masalaga yondashuv asosida katta sonlar bilan ishlovchi maxsus sinflarni yaratish mumkin. Ammo, shuni ham inobatga olish kerakki, ular bilan ishlaganda xotiraning hajmi, bir tomonlama funksiyalar, tub sonlari kabi testlashlardan o‘tkazib foydalanish maqsadga muvofiq.

Nazorat savollari

1. "Toshbaqalar" saralash algoritmlarini sanab bering?
2. Murakkabligi o‘rtacha va eng yomon hol uchun $O(n^2)$ va eng yaxshi uchun O(n) bo‘lgan saralash algoritmini ayting?
3. Shell caralash algoritmini tasniflab bering?
4. multiset konteynyerdan qaysi saralash algoritmida foydalanish mumkin?
5. Qo‘shish orqali saralashga o‘xshash kabi saralaydigan algoritmlarni sanab bering?
6. Tanlash orqali saralash g‘oyasini rivojlantirilgan varianti bu qaysi algoritm?
7. Massivni ikkiga bo‘lasiz, qismlarni rekursiv ravishda tartiblaysiz va keyin birlashtirish protsedurasini bajarasiz. Bu qaysi saralash algoritmi?
8. Bitli saralash algoritmlarning qanday variantlari bor?
9. Biton ketma-ketlik nimani anglatadi va qaysi saralash algoritmda ishlatiladi.
10. Saralash algoritmlarini nima uchun taqqoslanadi?
11. Tasodifiy sonlar bilan to‘ldirilgan massivni saralash tasodifiy sonlar chegarasiga bog‘liqmi?
12. Tasodifiy sonlarning eng kattasi 10000 bo‘lganda o‘zini eng yomon tutudigan saralash algoritmini ayting?
13. Massiv elementlari soni 100000 bo‘lganda o‘zini yomon tutudigan saralash algoritmlarini ayting?
14. Katta sonlar bilan ishlash bilan ishlash nima uchun kerak?
15. Nima uchun sonlarga diapozon tushunchasi kiritilgan?
16. Sonlar ustida arifmeik amallarni bajargada xato natija nima uchun chiqishini tushuntirib bering?
17. __int64 tipining diapozonini ayting?
18. Katta sonlar chegarasini tekshirish mumkinmi?
19. 20 xona aniqlikda ishlovchi tipni qanday tip?
20. Katta sonlar bilan ishlaganda xatoliklar qachon chiqib keladi yoki necha xonadan keyin. Nima uchun va misol keltiring?

21. Katta sonlarni mos tiplardan foydalanmasdan ham arifmetik amallarni bajarish mumkinmi, qanday va nima uchun?

22. Katta sonlarni massivga olib, qo‘shish amalini bajarish g‘oyasini ayting?

23. Katta sonlarni massivga olib, ayirish amalini bajarishdagi dastlabki asosiy g‘oya nimadan iborat?

24. Katta sonlarni massiv ostilarga oli bajarish mumkinmi?

25. Katta sonlar uchun turli operatorlarni aniqlash mumkinmi va qanday?

26. OpenSSL qanday kutubxona va nima uchun ishlatiladi?

27. Libgcrypt kutubxonasi qanday kutubxona va nima uchun ishlatiladi?

28. GMP qanday kutubxona va nima uchun ishlatiladi?

29. Boost.Multiprecision kutubxonasi nimasi bilan ustun?

30. C++ standartlashtirish bo‘yicha Milliy ishchi guruhdan rus ishlab chiquvchilari standart kutubxonaga mustaqil ravishda qanday tiplarini kiritish taklifi bilan chiqqan?

31. using int_mb = wide_int<1024*1024>; dastur fragmentini tushuntirib bering?

32. BigInt kutubxonasi haqida nimani bilasiz va undagi stoi funksiyasining vazifasini tushuntirib bering.

11- MA’RUZA
VISUAL C++ MUHITIDA DASTURLASH. (2 SOAT)

Reja:

1.Visual C++ muhitida dasturlash.

2. Visual C++ da Windows ilovalarni yaratish.

3. Muhitda menyular va uskunalar paneli.

4. Form xususiyatlari va hodisalari.

Annotatsiya: *Ushbu mavzuda Integrallashgan ishlab chiqarish muhiti, ularni yaratilish tarixi, rivojlanish bosqichlari, Yagona tilli va ko‘p tilli integrallashgan muhitlar, Visual Studio variantlari va imkoniyatlari, .NET frameworkning imkoniyatlari, Visual C++ da Windows ilovalarni yaratish, Muhitda menyular va uskunalar paneli, Form xususiyatlari va hodisalaridan foydalanish bo‘yicha nazariy hamda amaliy namunalar keltirilgan bo‘lib, nazariy bilimlarni asoslash uchun dasturlar tuzib ko‘rsatilgan. Bilimlarni mustahkamlash uchun 30 ta nazariy savol va amaliy ko‘nikma va malakalarni rivojlantrish uchun 10 ta assisment topshirig‘i berilgan. Bu topshiriqlarni bajarish mavzuni mustahkamlash uchun xizmat qiladi.*

Kalit so‘zlar*: IDE, Visual C++, user interface, OYD, .NET, (plug-ins, add-ins, add- ons, abstrakt Windows Toolkit (AWT), MSDN, Visual Studio Interdev-ASP (Active Server Pages), GUI,*

ilova, loyiha, Windows Forms application, .NET Framework, Buyruqlar va asboblar paneli, Form xususiyatlari, hodisalari, MessageBox.

KIRISH

Bugungi kunda dasturlash muhitlari va turlari ko‘payib bormoqda. Mumkin shu vaqtgacha console rejimda dasturlashni o‘rgangan bo‘lsangiz kerak. Ammo so‘ng 35 yil ichida foydalanuvchilarga oynali dasturlash muhitlarida foydalanib kelinmoqda. Bunday dasturlarni yaratish uchun dasturlashning barcha nazariy asoslariga tayanib, foydalanuvchi uchun qulay vizual dasturlash muhitlari yaratilgan. Ularda birinchi navbatda foydalanuvchining UI (user interface) yoki dasturning foydalanuvchi bilan muloqot oynasi loyihalashtiriladi. Buning uchun tayyor sinflar (komponetalar) mavjud bo‘lib, ularning xususiyatlar, funksiyalari va hodisalari bilan ishlashsh juda qulay qilib, vizual foydalanish mumkin. Vizual foydalanish degani ko‘rib turib ishlatish degan ma’noni beradi. Bu Vizual dasturlash bo‘yicha ilk ma’lumotlarni keltiramiz. Deyarli barcha dasturlash tillari uchun vizual dasturlash muhitlari mavjud.

Visual C++ muhitida dasturlash. Dastaval integrallashgan ishlab chiqarish muhitining ilovalari tushunchasi, uning model xususiyatlari, rivojlanish tarixi, ularning eng ko‘p ishlatilganlari bilan tanishish lozim.

Integrallashgan ishlab chiqarish muhiti (Integrated development environment - IDE) - dasturni ishlab chiqish hayot siklining barcha asosiy funksiyalarini qo‘llab-quvvatlovchi umumiy interaktiv grafik qobig‘iga ega bo‘lgan dasturlarni ishlab chiqish va tekshirish vositalari majmui, dastur matnini (kod) yozish va tahrirlash, kompilyatsiya, bajarish, xatolarni tuzatish, profilaktika va boshqalarni bajarish imkoniyati beradi.

Integrallashgan muhitdan foydalanish dastur ishlab chiqishning mumkin bo‘lgan yondashuvlaridan biridir. Shu bilan bir qatorda, UNIX tizimining oldingi an’anaviy yondashuvi, funksionallik bilan bog‘liq bo‘lgan vositalar (toolkit, toolbox) to‘plamidan foydalanishga asoslangan, lekin Integral interaktiv muhitga va ba’zan (UNIX tizimining dastlabki versiyalarida) buyruq satri rejimida (buyruq satri interfeysi) amalga oshirilgan. Albatta, Integrallashgan muhitdan foydalanish ishlab chiquvchi uchun ancha qulay bo‘lib, u 1980-yillardan boshlab Integrallashgan muhitlarning jadal rivojlanishi va turli tumanligini izohlaydi.

Birinchi integrallashgan muhitlardan biri borlandning Turbo Paskalidir, 1980-yillar o‘rtalarida Niklaus Wirth va uning talabasi Filip Kahn tomonidan ishlab chiqilgan.

Microsoft zamonaviy integrallashgan muhitning eng yaxshi namunalaridan bo‘lgan Visual Studio ni yaratish va rivojlantirish orqali Integrallashgan muhitlarning rivojlanishiga katta hissa

qo‘shdi. Uning yangi versiyasi juda ko‘p va oxirgisi, Visual Studio 2019, amma keyinchalik Visual Studio 2012 muhitida S++ tilidan foydalanamiz.

Integrallashgan muhitlar tarixi. Integrallashgan muhitlar g‘oyasi 1980-yillarning o‘rtalariga kelib, integrallashgan muhitlarning ikki guruhi paydo bo‘ldi:

-Turbo muhitlar (Turbo Pascal, Turbo C, C++, Delphi, Turbo va hokazo.) bu tillarda dasturlashni qo‘llab-quvvatlash uchun Borland tomonidan, MS DOS operatsion tizimi uchun, keyin Windows uchun foydalanishlang;

-GNU Emacs - MS DOS uchun, keyin Windows, OpenVMS va Linux uchun amalga oshiriladigan ko‘p tilli va ko‘p platformali integrallashgan muhitlar. 1990 yilda Sun Microsystems bilan ishlagan rivojlantirish jamoasi orasida, Solaris platformasi uchun uning amalga oshirish ko‘p foydalanuvchilar va GNU Emacs tashabbuskorlari bor edi.

Ob’ektga yo‘naltirilgan til Smalltalk kompaniyasi Xeror PARC dasturlarni ishlab chiqish uchun o‘sha yillarda integrallashgan muhitini amalga joriy qilingan. Unda OYD birinchi integrallashgan muhiti yaratilgan va u dasturlash texnologiyaisi bytecode ikkilik va Postfix oraliq vakillik tushunchasini joriy qilgan va shuningdek just-in-time (JIT, dinamik) kompilyatsiyasini birinchi ilg‘or usuli bo‘yicha ijro kompilyatsiya kompyuterda maqsadli yaratgan.

Borland ning turbo muhitlari dasturiy ta’minot ishlab chiquvchilari va dasturiy ta’minot ishlab chiqish vositalari yaratuvchilariga katta ta’sir ko‘rsatdi. Ularning xarakterli xususiyati doimiy rivojlanish siklini qo‘llab-quvvatlash edi: manba matnini yozish va tahrirlash - kompilyatsiya - tahlil qilish va xatolarni tuzatish - kompilyatsiya qilishni yakunlash-ijro etish va xatolarni qayta tuzatish. Integral muhitdan chiqmasdan va bu bosqichlar oddiy funksiya tugmachalari bilan nazorat qilindi va har qanday alohida vositalarni aniq chaqirishni talab qilmaydi. Yuqori kompilyatsiya tezligi ham Turbo muhitlar jozibador sifati aylandi. Turbo Paskalning birinchi versiyalarida kompilyatsiya birinchi xatodan oldin amalga oshirilgan bo‘lsa-da, barcha xatolarni topish va tashxislash uchun avvalgisini o‘rnatganingizdan so‘ng kompilyatsiyani qayta ishga tushirishingiz kerak edi, ammo bu funksiya tugmalaridan foydalanib darhol amalga oshirildi.

Turbo muhitlar to‘liq ikkilik kodi, loyiha kodi kompilyatsiyasi uchun ajralmas build mexanizmi bor edi, shuningdek, make (F9) rejimi, klassik UNIX make foyda faoliyatiga o‘xshash, faqat tahrirlangan kod modullar recompiling uchun amalga oshirilgan.

Turbo Paskal muhitidagi eng muhim yangilik - KIRISH tilining ob’ektga yo‘naltirilgan tushunchalar (sinf, ob’ekt) bilan kengaytmasi va mustaqil kompilyatsiya birligi g‘oyasini o‘zida mujassamlashtirgan birlik konstruksiyasi (moduli) bo‘lgan. OYD uchun qo‘llab - quvvatlash versiyada paydo bo‘lgan. Keyinchalik bu g‘oyalar Windows platformasida, Borland integrallashgan muhitlarning yangi versiyalarida - Borland Pascal va Delphi (Paskal g‘oyalarini OYD bilan birgalikda rivojlantiruvchi Borland dasturlash tili) ishlab chiqildi.

Zamonaviy integrallashgan muhitlarning asosiy xususiyatlari. Endi integrallashgan dasturiy ta’minot ishlab chiqish muhitlarining asosiy xususiyatlarini umumlashtiraylik. Ularning har biri quyidagi qismlarga ega:

- Funksiya tugmalaridan keng foydalangan holda muhitdan chiqmasdan boshqa barcha komponentlarni chaqirishga imkon beruvchi yagona interaktiv qobiq;

- Dastur fragmentlarini yozish va tahrirlash uchun matn muharriri;

- Qo‘llab-quvvatlash tizimini qurish, ya’ni amalga oshirilayotgan manba tilidan kompilyator va bitta bajariladigan kod (yuklash moduli) da ob’ekt ikkilik kodlarining linkerini o‘z ichiga olgan manba kodidan loyihalarni tuzish; linker operatsion tizimning standart komponenti sifatida yoki ushbu muhit uchun maxsus ishlatiladi;

- Buyruqlar majmui yordamida muhitda debugging dasturlari uchun Debugger: bir joyda o‘rnatish; berilgan tartibda to‘xtatish; o‘zgaruvchilar qiymatlarini tasavvur (yoki, past darajada, registrlari va xotira hududlari) qilish;

- Integrallashgan muhitda zamonaviy matn muharrirlari kodi avtomatik bajarilishini ta’minlash (kodi tugatish), muharriri muhitda joriy yozilgan kodi sintaktik to‘g‘ri va uning davomi bo‘lishi mumkinligi talab qilinadi, misol uchun, yopilish qavs yo‘qligi, nuqta-vergul va usul nomlari, tafovutlar usul ma’lum bir sinf ob’ektdan bo‘lsa va boshqalar.

- Integrallashgan muhitlarning zamonaviy versiyalarida quyidagi xususiyatlar ham qo‘shilgan (komponentlar):

- Profiler - integrallashgan muhit nazorati ostida dasturni bajarish natijasida olingan statistik ma’lumotlarni jamlash va tahlil qilish vositasi: protseduralarga (usullarga) chaqiriqlar soni, dasturni bajarish vaqtida ishlatiladigan xotira miqdori va boshqalar.

- Refactoring - kodni takomillashtirish maqsadida muhitdagi dasturlarning tizimli guruhli modifikatsiyalarini, ularning funksiyalariga tub o‘zgarishlarsiz amalga oshirish vositalari. Odatda, bunday harakatlar o‘z ichiga oladi, misol uchun, uning ta’rifi va barcha foydalanish usuli nomini o‘zgartirish, uning argumentiga qo‘shib, a sinash try/catch blok oldindan e’tiborsiz istisno bilan band qilish va hokazo.

- Unit test generator-test modul uchun umumiy test yaratish uchun vosita (birliklari)-usullari yoki protseduralar – argument xususiyatlar turli mumkin birikmalar bilan; odatda misollar integral Java muhitda JUnit vositasi va Visual Studio o‘xshash Nunit vositasi bor;

- Manba kodini boshqarish tizimi (source code control system) yoki mavjud versiya tizimlaridan biri (CVS, RCS, Mercurial, Visual Sourcesafe va boshqalar) bilan muhitni integrallashgan qilish vositasi.)- dasturlarni qo‘llab-quvvatlashda loyiha kodi fayllarining versiyasini nazorat qilishni qo‘llab-quvvatlash;

- Dasturlarning jamoaviy rivojlanishini qo‘llab-quvvatlash vositalari (jamoaviy ish) - dastur hayot sikli bosqichlari (talab va xususiyatlar, dizayn, amalga oshirish, sinov), dasturchilar jamoasi

a’zolari o‘rtasida rivojlanish vazifalarini taqsimlash, loyiha menejeri tomonidan vazifani bajarishni nazorat qilish. Visual Studio muhitida bu komponent birinchi bo‘lib Team Foundation Server (TFS) deb nomlangan va Visual Studio 2013 versiyasidan boshlab, u bulutli interfeys sifatida amalga oshiriladi va Visual Studio Online deb ataladi.

- Kod tahlil vositalari - uning semantik to‘g‘riligi: odatda ijro paytida aniqlanadigan xatolarning ayrim turlarining yo‘qligi, masalan, erishib bo‘lmaydigan sharoitlar; zarur tekshirishlar va xavfsizlik ruxsatlarining yo‘qligi va boshqalar. Professional qiziqishlar tufayli juda muhim bo‘lgan ushbu xususiyatlar haqida qo‘shimcha ma’lumot olish uchun "Visual Studio 2013 va uning ishonchli va xavfsiz hisoblash (ishonchli hisoblash) uchun imkoniyatlari" ga qarang . Ushbu imkoniyatlar 2002 da shakllantirilgan ishonchli va xavfsiz hisoblash (trustworthy computing) sharoiti va tamoyillariga mos keladi. Microsoft korporatsiyasi va u tomonidan hayotga izchil tatbiq etilmoqda. Zamonaviy muhitlarga dasturdagi nazorat oqimi grafigining siklomatik soni, sinflarning ilashish darajasi (o‘zaro bog‘liqligi) va shu kabi murakkabligini tavsiflovchi metrikalar bo‘yicha kodni tahlil qilish vositalari ham kiradi;

- Hosil bo‘lgan binar kodni vizuallashtirish vositalari-usullar, o‘zgaruvchilar, ularning nomlari va boshqalar. Misol uchun, Visual Studio muhitida, platforma .NET Common Intermediate til yagona oraliq (ikkilik) kodni tasavvur qilish imkonini beradi;

- Kod elementlari, nomlarini almashtiruvchi obfuskatsion vositalar -sinflar, usullar, maydonlar va boshqalar. Noma’lumlar bilan, "tasodifiy", "chalkash" nomlari qiyin desompiled bo‘ldai, ikkilik kodni o‘rganish uchun, kodda mavjud yangi g‘oyalarni joriy qilish, ruxsatsiz istagan hakerlar tomonidan kodni "Hack"ka qarshi himoya qilish uchun, yoki hujumlarni tashkillashtirish, zararli maqsadlar uchun uni o‘rnptish;

- Standart kod vositalari asosida turli xil dasturiy loyihalar va yechimlarni yaratishni qo‘llab-quvvatlash; kengaytmalarni ishlab chiqish mexanizmi (plug-ins, add-ins, add- **ons**). Zamonaviy dasturiy ta’minotni ishlab chiqishda ba’zan turli xil ilovalar va vositalar - konsol (oddiy) ilovalar, veb-ilovalar va veb-xizmatlar, mobil ilovalar, bulutli ilovalar va boshqalarni yaratish kerak. Bu turdagi har bir manba kodi fayllar ma’lum bir tuzilishni rivojlantirishni talab qiladi, shuningdek belgilash konfiguratsiya fayllar sifatida, misol uchun, kod xavfsizlik ruxsatini, veb-konfiguratsiyalar, va hokazo. Zamonaviy integrallashgan muhitlar manba kodi shablonlarini taqdim etish va loyiha uchun zarur konfiguratsiya fayllarini avtomatik ravishda ishlab chiqarish orqali turli xil loyihalarni yaratishni avtomatlashtiradi. Dasturlashni tayyor kod shablonlarini ishlatmasdan tasavvur qilish qiyin, bu esa muqarrar ravishda xatolarga yo‘l qo‘yadi. Misol uchun, u qo‘lda loyihaning ajralmas qismi bo‘lgan faylni yaratishni unutish, yoki kod ba’zi muhim parcha ko‘zdan yo‘qotish juda oson (masalan, mos kelmaydigan ko‘rsatkichoar uchun, lekin unga bog‘langan ko‘rsatkichni ta’minlash uchun emas). Shuning uchun, integrallashgan muhitlar tomonidan loyihalarni turli turdagi qo‘llab-quvvatlash, ayniqsa, muhim ahamiyatga ega. Bundan tashqari, zamonaviy integrallashgan muhitlarda mumkin

bo‘lgan loyiha turlari majmui kengaytiriladi, ya’ni. tuzuvchi zarur bo‘lsa, muhitga loyihaning yangi turini kiritishingiz mumkin. Misol uchun, aspect yo‘naltirilgan dasturlash vositasi amalga oshirishda Aspect.net Visual Studio integrallashgan muhit kengaytmalari sifatida, tegishli kod shabloni bilan loyiha - aspect yangi turini joriy qilgan.

- UML (yagona modellashtirish tili) modellashtirish tilida dasturlar strukturasini modellashtirishni qo‘llab-quvvatlash. UMLning zamonaviy versiyasi (2.x) til har xil turdagi dasturlar va tegishli diagrammalarni qurish modellarini ta’minlaydi. Bundan tashqari, UML dastur ishlab chiqish va ishlab chiquvchilar o‘rtasida o‘zaro hamkorlik uchun faoliyat modellari ishlab chiqish qo‘llab-quvvatlaydi. Zamonaviy integrallashgan muhitlar UML tilidan foydalanishni ikki yo‘l bilan qo‘llab-quvvatlaydi: modelni va manba kodidan tegishli diagrammani hosil qilish va, aksincha, ishlab chiqilgan modeldan manba kodini (shablonini) hosil qilish.

Yagona tilli va ko‘p tilli integrallashgan muhitlar. Dastlab yagona manba tilida dasturlash uchun integrallashgan muhitlar ishlab chiqilgan (masalan, Turbo Paskal-Borland ning Paskal kengaytmasida dasturlash uchun).

- Shu bilan birga, bunday mono tilidagi integrallashgan muhitlarni ko‘p tilli muhitlarga aylantirish bo‘yicha bosqichma-bosqich tendensiya yuz berdi, chunki turli tillarda loyihalarni ishlab chiqish uchun o‘xshash tamoyillar va mexanizmlar qo‘llaniladi va turli tillarda yozilgan dastur bo‘laklarini bitta loyihada ishlatish ham ba’zan qulay. Masalan, loyihada, shu jumladan, yagona maqsad uchun, masalan, C#, yana qayta yozish uchun emas, balki, shuning uchun, (masalan, C) oldingi tilda yozilgan meros kodni ishlatish uchun yaratilgan.

- taniqli NetBeans integrallashgan muhit dastlab Java dasturlash uchun Pragada Charlz universiteti talabasi loyihasi sifatida yaratilgan. Hozirgi kunda NetBeans muhiti C va C++ da loyihalarni ishlab chiqishga ko‘mak beruvchi C / C++development pack komponentasini amalga oshiruvchi kuchli ko‘p tilli integrallashgan muhitga aylandi.

- Visual Studio - environment.net azaldan ko‘p tilli muhit sifatida yaratilgan. Ishlab chiquvchilar ishlab chiqilgan loyihaning tegishli qismlari uchun eng qulay tilni (tillarni) tanlashlari va loyihani tegishli tillardan bitta ikkilik oraliq SIL kodiga tuzish orqali olingan ikkilik komponentlardan (assemblies) to‘plashlari uchun Microsoftning prinsipial sozlamalari. Keyinroq Visual studio ning ushbu qulay xususiyati va uni qo‘llab-quvvatlovchi tillar majmuasiga yaqindan tanishib chiqamiz.

- Integrated development environment (IDE) - dastur ishlab chiqish hayot siklining barcha asosiy vazifalarini qo‘llab-quvvatlovchi umumiy interaktiv grafik qobig‘iga ega bo‘lgan dasturlarni ishlab chiqish va xatolarni qayta tahrir qilish vositalari majmuidir.

Integratsiyalashgan Microsoft Visual Studio muhiti 1995 yilga kelib, ommaga keng taqaladi, bu nom ostida mahsulotning birinchi versiyasi chiqarildi. Bu oldin, 1990-yillar boshida, Microsoft

yuqori darajadagi tillarda dasturlash qo‘llab-quvvatlash uchun alohida mahsulotlar bilan bog‘liq bo‘lgan:

- Microsoft Visual C++ - Visual C++ da interaktiv dasturlash muhiti, Microsoft tomonidan ishlab chiqilgan va amalga oshirilgan, C++ tilining kengaytmasi hisoblanadi. Visual C++ hali Visual Studio barcha versiyalarida dunyodagi eng mashhur va keng tarqalgan dasturlash tili;

- Visual Basic-Microsoft tomonidan ishlab chiqilgan va amalga oshirilgan asosiy tilning ob’ektga yo‘naltirilgan kengaytmasi, tilning soddaligini ob’ektga yo‘naltirilgan kengaytmalar bilan birlashtirgani uchun darhol butun dunyodagi dasturchilar tomonidan faol qo‘llanila boshlangan. 1990-yillar boshida ko‘p dasturchilar Visual Basic qulaylik haqida gapirib va GUI boshqaruv dasturlarini ishlab chiqish uchun foydalanish afzal bo‘lgan;

- Microsoft Visual FoxPro-Visual FoxPro tilida interaktiv dasturlash muhiti-kichik firma Fox Software tomonidan dastlab Foxbase nomi ostida ishlab chiqilgan protsessual dasturlash elementlari bilan ob’ektga yo‘naltirilgan til. Ko‘pgina foydalanuvchilar uchun bu tilning xususiyati SQL asosida ma’lumotlar bazalariga bevosita bog‘langan va ushbu tildagi dasturdan, xususan FoxPro da SQL so‘rovlarini dasturlashga yordam bergan;

- Microsoft Visual Sourcesafe-Microsoft kodi versiyasi nazorat tizimi tomonidan ishlab chiqilgan, keyinchalik Visual Studio muhit bilan integratsiyalangan.

Ushbu mahsulotlarning barchasi 1995-dan "bir muhitda" Visual Studio nomi bilan birlashtirildi. Yangi muhitga 4.0 versiyasi deb nom berildi, ehtimol yuqorida bayon etilgan muhit komponentlari bir necha yillar davomida alohida shaklda chiqarilgan.

Ushbu versiyada ham Visual Studio juda mashhur bo‘ldi. Masalan, 1995-yilda yangi Java dasturlash tili va texnologiyasini chiqargan Sun Microsystems Windows platformasi uchun GUI rivojlanishini qo‘llab-quvvatlash uchun Visual Studio muhiti va platforma-mustaqil Java kutubxonalarini, abstrakt Windows Toolkit (AWT) ni amalga oshirish uchun unda amalga oshirilgan Visual C++tilidan foydalangan.

1990-yillarda dasturchilari orasida vizual studio muhiti mehr bilan do‘stona "vizualka" nomi bilan atalgan. Bu bejiz emas: Visual Studio dasturchilarining rivojlanishida ishonchli hamkorga aylangan, bu uning interfeysining do‘stonaligi va qulayligini tasdiqlaydi. Taqqoslash uchun, o‘sha vaqtga kelib, o‘sha firma SUN bir xil mashhur dasturiy muhitga ega emas edi. Java dasturlash uchun NetBeans muhiti bir oz keyinroq - 1997 yilda, Charlz universiteti Praga dasturlash talabalar kichik guruhi ish natijasida paydo bo‘ldi. Visual Studio muhitining keng tarqalishida MSDN (Microsoft Developers ‘ Network) ga obuna bo‘lish orqali muhim rol o‘ynadi, bu esa eng so‘nggi Microsoft dasturiy ta’minotini ishlab chiqishga imkon yaratdi.

Vizual Studio 97 (5.0). Visual Studio yangi (beshinchi) versiyasi, shuningdek, Visual Studio sifatida tanilgan 97 va kodlangan. Uning yangi xususiyatlari va vositalari Visual Studio Interdev-ASP

(Active Server Pages) texnologiyasi asosida interaktiv veb - saytlarni ishlab chiqish vositasi - va Visual Java++ - "Microsoft"dan Java dasturlash muhiti bo‘lgan.

ASP texnologiyasi - Microsoft kompaniyasining web dasturlash sohasidagi ajoyib rivojlanishi, 1994-1995-yillarda paydo bo‘lgan. Faol veb-sahifalarni ishlab chiqish uchun mo‘ljallangan-veb-so‘rovlarga javob sifatida mijoz brauzerlari uchun HTML-sahifalarni yaratish uchun shablonlarga ega bo‘lgan. Bu hali ko‘p saytlarda ishlatiladi. Texnologiya keyinchalik uning asosida ishlab chiqilgan ASP.NET, xususiyatlariga keng imkoniyat ochib bergan.

Visual Java++ vositasiga kelsak, umuman boshqa turdagi hodisalar u bilan bog‘liq - ba’zida, afsuski, o‘z dunyosida shunga o‘xshash narsalar sodir bo‘ladi, bu esa kompaniyalar juda raqobatbardosh bo‘lishi muqarrar bo‘lgan. Microsoft Sunning yangi Java texnologiyasi bilan tanishib chiqqan va Sun bilan raqobatda unga javob sifatida Java tilining o‘z kengaytmasini ishlab chiqdi va uni Visual Java++deb nomlangan yangi muhitda amalga oshirdi. Biroq, afsuski, bu harakatlar to‘liq Sun tomonidan joriy rasmiy Java amalga oshirish tartibi bilan mos kelmadi. Shu kungacha (Oracle hozir Java texnologiya egasi bo‘lsa-da, bu tartib aslida o‘zgarmagan), Java nomi rasman ro‘yxatdan o‘tgan savdo belgisidir. Shuning uchun, yangi Java amalga oshirish uchun, Java texnologiyasi egalik kompaniya litsenziya sotib, va amalga oshirish idoraviy Java standartlarga javob tekshirish uchun Java moslashuv Kit (JCK) deb nomlangan yangi Java amalga oshirish uchun til, kutubxonalar va vertual mashina amalga oshirish uchun testlar maxsus majmuini o‘tishi kerak bo‘lgan. Faqat bu testlarning hammasi to‘liq 100% o‘tsa, yangi amalga oshirish "Java"deb nomlanish huquqini oladi. Afsuski, Microsoft tomonidan Visual Java++ ni amalga oshirish uchun Visual Java++ tizimida amalga oshirilgan til kengaytmalari tufayli JCK testlaridan o‘tmadi. Bu ish, afsuski, ikkala kompaniya uchun ham, Sun va Microsoft o‘rtasidagi uzoq muddatli huquqiy jangda, Sun tomonidan qo‘lga kiritilgan, natijada Microsoft mahsuloti Vj++ deb nomlangan (Java nomini ochiq-oydin eslatib o‘tmasdan). Bunday hodisalar muqarrar bo‘lsa-da, juda yoqimsiz, chunki har ikki tomon ham, ba’zan bir necha yillar davomida mashhur texnik mutaxassislar, bularning hammasiga ko‘p vaqt, kuch va sog‘liq sarflashadi. Biroq, keyinchalik, 2004 yilda, Sun va Microsoft 10 yil muddatga intellektual mulk bo‘yicha misli ko‘rilmagan shartnoma tuzish, bir shartnoma kelgan.

Shunday qilib, Visual Studio 5.0 Visual C++, Visual Basic va VJ++ da integrallashgan dasturlash muhiti bo‘lib, u ASP texnologiyasi yordamida interaktiv Web saytlarni ishlab chiqish uchun Visual Studio Interdev vositasini ham o‘z ichiga oladi.

Visual Studio 6.0 (1998). Visual Studio keyingi versiyasi (6.0, kodlangan Aspen) 1998 yilda chop .tildi. Visual C++, Visual Basic, vj++ va Interdev, (maksimal versiyada - Visual Studio Enterprise Edition) - allaqachon ma’lum komponentlarini yangi versiyalari bilan bir qatorda, quyidagi yangi komponentlar ham vizual modellashtirish va dastur ishlashini tahlil qilishni qo‘llab-quvvatlash uchun kiritilgan:

1. **Application Performance Explorer**

2. **Automation Manager**
3. **Microsoft Visual Modeler**
4. **RemAuto Connection Manager**
5. **Visual Studio Analyzer**

Visual Studio 6.0 dasturlari bilan ishlayotgan uchun bitta ko‘p tilli virtual platforma o‘z ichiga oladi integrallashgan muhit, yangi versiyasini yaratish uchun asos sifatida Microsoft tomonidan ishlatiladigan versiyasi - .NET.

Visual Studio.net (2002). Visual Studio 7.0 versiyasi, shuningdek, Visual sifatida tanilgan Studio.net va kodlangan Rainier, joriy etildi. Ushbu versiyada eng muhim yangilik boshqariladigan kod, umumiy til infratuzilmasi (SLI), umumiy oraliq til (SIL), umumiy turdagi tizim (SP) asosida to‘liq turdagi nazorat va xavfsizlik bilan ishonchli va xavfsiz dasturlashni ta’minlaydigan ko‘p tilli platformani amalga oshirishdir. .Net platformada barcha kompilyatorlar manba kodini tegishli tildan tarjima qiladi (C#, C++, Visual Basic va hokazo.) bitta oraliq SIL kodining ikkilik formatiga - VM ning postfiks yozuvi instructions.net -Virtual ijro tizimi (VES). Taqdim etilgan: .Net ichidagi moslik-turli tillardan tuzilgan kodlarning ilovalari; yagona metadata asosidagi turdagi nazorat va xavfsizlik mexanizmi; yagona ko‘p tilli istisno muomala mexanizmi o‘z ichiga oladi.

Visual Studio.NET 2002 versiyasi to‘rt versiyalarida chop etildi (nashrlar):
Academic, *Professional*, *Enterprise Developer*, *Enterprise Architect*.

Visual Studio muhitida qo‘llab-quvvatlanadigan tillarda haqiqatdan ham ajoyib o‘zgarish va qo‘shimchalar amalga oshirildi.

Avvalo, C# -yangi dasturlash tili ishlab chiqildi va amalga oshirildi, hozirgi kunda eng kuchli, zamonaviy va to‘liq dasturlash tili.

Visual C++ va Visual Basicda muhim kengaytmalar amalga oshirildi. Aslida sintaksisdagi ayrim farqlarga qaramay, bu tillar deyarli o‘xshash xususiyatlarni ta’minlab, C# tiliga "teng" bo‘lib kelgan. Yangi .NET -mos versiyasi Visual C++ muvaffaq deyiladi, C++ va yangi Visual Basic SOF-mos versiyasi Visual Basic.net deyiladi.

Visual Studio.net 2003. Keyingi yili, 2003, Visual Studio yangi versiyasi released.net kodi nomi Everett ostida ishlab chiqardi. Bir butun sifatida mahsulot ishlab chiqish nuqtai nazaridan, bu versiya muhim o‘zgarishlar qilinmadi: yangi versiyasining asosiy maqsadi versiyasi .net Framework 1.1 uchun yangilash bilan ta’minlash edi. biroq, Visual Studio 2003 juda muhim yangilik mobil qurilmalar .net Compact Framework uchun versiya .net da amalga oshirish va Visual Studio vositalari yordamida mobil qurilmalar uchun dasturlar ishlab chiqish uchun qo‘llab-quvvatlash edi.

Oldingi versiyasi kabi, Visual Studio 2003 to‘rt nashrlarda chop yetildi: Academic, *Professional*, *Enterprise Developer*, *Enterprise Architect*.. Bundan tashqari,

"maksimal" versiyada - Enterprise arxitektor - Visual Studio 2003 dastur arxitekturasini tasavvur UML diagrammalar uchun qo‘llab - quvvatlash, shu jumladan, boshqa mashhur mahsulot -Microsoft Visio tarzida dasturiy modellashtirish vositalari qo‘llab-quvvatlaydi.

Visual Studio.net 2005. Visual Studio 2005 integralashgan muhit (kodlangan Whidbey) keyingi versiyasi 2005 yilda e’lon qilindi.

Versiyaning asosiy yangiliklari parametrli ma’lumot turlari (generics) ga tegishli. Ular C# 2.0 tilidagi versiyada va .net Framework 2.0 da amalga oshirildi. Bu bir vaqtning o‘zida shunga o‘xshash xususiyatlari Java 1.5 da amalga oshirildi, deb ta’kidlash qiziq, lekin generics mexanizmi Java nisbatan, yanada qulay, moslashuvchan va kengaytirilgan versiyada amalga oshiriladi .net va C# platforma asoslangan ta’kidlash lozim.

Manfaatdorlikni yaxshilash yana bir guruh ASP.NET -. NET platformasiga xos kodning ishonchliligi va xavfsizligi bilan birgalikda, veb-sahifalarni amalga oshirish va faollashtirish uchun mo‘ljallangan ASP texnologiyasini ishlab chiqildi. Visual Studio 2005 versiyasi qo‘shimcha loyiha turlari tushunchasini qo‘llab-quvvatlash uchun birinchi marta ASP.NET Web xizmatlari amalga oshirildi. NET dagi web dasturlash vositalari ushbu platforma va umuman Web dasturlash talabalari uchun alohida e’tiborga loyiqligini unutmang. Faqat ularni o‘rganish va ishlatish uchun har bir dasturchi platformani yaxshi bilishi kerak .NET va amalda o‘zi ishlatishi kerak.

aspect yo‘naltirilgan dasturlash tizimi rivojlantirish jamoasi uchun Aspect.net Visual Studio 2005 versiyasi alohida rol o‘ynadi. Visual Studio muhitining ushbu versiyasi uchun plagin deb nomlangan Aspect.net Framework-integrallashgan muhit o‘zining vizual uslubiga o‘xshash bir uslubda jihatlarini boshqarish uchun GUIdir. Interaktiv xususiyatlar va qulaylik Aspect.net foydalanuvchilar tomonidan yuqori baholandi va tizimning ushbu versiyasida Aspect.net dunyoning 26 mamlakatida tarqalgan.

Vizual *plug-in* sifatida amalga oshirilgan boshqa taniqli loyiha Studio.net 2005 loyihasi hisoblanadi. Knowledge.net - bilim vakillik orqali C# tilining kengaytmasi (ramkalar, qoida silsilasi, ontologies), qaysi bilim foydalanish bilan an’anaviy C # dasturlash uslubi birlashtirib beradi. Knowledge.net talabalar va magistrantlar tomonidan bilimlardan foydalanadigan aqlli yechimlarni ishlab chiqish uchun ishlatiladi.

Visual Studio.net 2008. 2008 yilda Visual Studio ning navbatdagi versiyasi ishlab chiqarildi. Microsoft kompaniyasining Visual Studio muhitining ichki versiya raqami Visual Studio 9 hisoblanadi. Ushbu raqamlar yodda tutish uchun foydalidir, chunki Microsoft xodimlari ko‘pincha Visual Studio rivojlanishini tasvirlashda ushbu ichki versiya raqamlaridan foydalanadilar (masalan, bloglarda). Visual Studio 2008 versiyasi Windows Vista uchun mo‘ljallangan. Bu boshqa mashhur Microsoft Office 2007 mahsulot versiyasiga mos keladi va ilg‘or veb-dasturlash qo‘llab-quvvatlash vositalari o‘z ichiga oladi. Versiya .NET Framework Visual Studio 2008 versiyasi uchun .NET Framework 3.5 ka tegishli hisoblanadi.

VS 2008 versiyasi GUI-ni qo‘llab-quvvatlash uchun Windows Presentation Foundation (WPF) API-dan foydalanadigan ilovalar uchun yangi VS 2008 debugger multithreaded dasturlarda xatolarni qayta tahlillash uchun yordam beradi.

VS 2008 versiyasi birinchi marta J# tilini qo‘llab-quvvatlamadi, Microsoft Java tilining kengaytmasi, yuqorida aytib o‘tilgan. Buning o‘rniga foydalanuvchilarga Java tili va muhitidan ilovalarni .Net muhitiga ko‘chirish uchun muhitga kiritilgan Java Tili o‘zgartirgich Assiatant (JLCA) vositasi taklif etildi.

Version VS 2008 hozirda maktab uchun muhim Microsoft ichki mahsulotni qo‘llab - quvvatlaydi Visual Studio muhit so‘nggi versiyasi-Feniks, derleyiciler va til protsessorlari optimallashtirish rivojlantirish uchun bir vosita. Bu Microsoft tadqiqot va Microsoft mahsulot jamoasi qo‘shma mahsulot hisoblanadi. Biz tizimida maqsad ilovalar joylash jihatlarini amalga oshirish uchun Feniks foydalanish Aspect.net, Feniks qulay yuqori darajali ichki vakillik bor, chunki-Feniks yuqori darajali IR (HIR), jihatidan u dastur o‘zgarishlarni amalga oshirish uchun qulay. Feniks ham turli tizimida amalga oshiradi, bu ichki vakillik tomonidan ikkilik build .net ishlab chiqarish uchun qulay vositasi o‘z ichiga oladi Aspect.net unda jihatlarni joylangandan keyin olingan binar montaj hosil qilish. Feniks hozirda Microsoft tadqiqot Midori loyihasi uchun ishlatiladi , boshqariladigan kodi muhit .net butunlay yozilgan yangi operasion tizimi

Visual Studio.net 2010. Aprel oyida 2010, Visual Studio versiyasi edi. Sankt-Peterburgda Visual Studio 2010 ishga tushirilishi marosimida bir qator qiziqarli taqdimotlar va namoyishlar bo‘lib o‘tgan, jumladan, Visual Studio kompaniyasining yetakchi ishlab chiquvchisi, Microsoft menejeri Brayan Garri tomonidan hisobot taqdim etildi. Tizimini yoqishni taklif qilib Aspect.net studiyaning kelajakdagi versiyalarini yetkazib berishda, chunki AOP tizimimiz taniqli va uning imkoniyatlari, uslubi va Aopni keng tarqatish uchun Visual Studio bilan integrallashgan darajasiga mos keladi muqobil kundalik texnologiyalardan biri sifatida (ubiquitous technologies), Visual Studio-da qo‘llab-quvvatlanadigan texnologiyalarning eng keng to‘plamiga kiritilgan. Shu kungacha, ayniqsa, zamonaviy versiyadan beri amal qiladi Aspect.net Visual Studio so‘nggi versiyasi bilan mos 2013. Tizimi tahrirlash Aspect.net "Visual Studio 2013 da aspektli dasturlashdan (AOP) foydalanish" da ko‘rib chiqiladi.

Visual Studio (ichki versiya raqami-10, kod nomi-Dev10) dagi eng muhim yangilik qo‘llab - quvvatlanadigan tillarning "jentelmenlar to‘plami" da boshqa tilning paydo bo‘lishi - f# funksional tili. Til muallifi Microsoft Research, Kembrij, buyuk Britaniyadan don Syme hisoblanadi. .net Framework va Visual Studioda F# tili sof funksional dasturlash imkoniyatlarini ob’ektga yo‘naltirilgan dasturlash, parallel dasturlash va boshqa barcha xususiyatlar bilan integratsiya vositalari bilan birlashtiradi. Bu, ayniqsa, yosh dasturchilar orasida ko‘plab tarafdorlarni topgan juda istiqbolli tildir.

Dastlab, Visual Studio 2010 Phoenix mahsulotini va unga asoslangan optimallashtiruvchi kompilyatorni Visual C++ tilidan o‘z ichiga olishi rejalashtirilgan bo‘lib, hosil qilingan ikkilik kodni bajarishda Visual C++kompilyatorining mavjud, avvalgi versiyasidan o‘zib ketishi kerak edi. Biroq, afsuski, bu rejalar amalga oshirilmadi - Feniks jamoasi Visual Studio loyihasi boshqaruvi tomonidan belgilangan muddatlarga javob bermadi.

Visual Studio 2010 Ultimate - - "maksimal" versiyada UML diagrammalar yordamida loyiha tuzilishini aks ettirish uchun vositalari, shuningdek, qulay skipped test guruhlar, loyiha manba kodi o‘zgartirish ta’sirini baholash uchun Test ta’sir tahlil komponent bor, chunki keraksiz o‘tish test oldini olish imkonini beradi. Bundan tashqari, debug tarixi va uning to‘liq holatini eslash imkonini beradi. Intellitrace bir debugger bor, shu jumladan multithreading tarixi.

Visual Studio 2010 bulut hisoblashni qo‘llab-quvvatlaydi. Buning uchun Microsoft Azure loyihalarining yangi turlari qo‘shildi. Ularni ishlatish uchun Visual Studio 2010-Microsoft Azure SDK uchun maxsus vosita o‘rnatish kerak.

Visual Studio 2012. 2012-yil avgust oyida Visual Studio 2012 kodli Metro mavzusining versiyasi chiqarildi. Bu foydalanuvchi interfeysi yaxshilandi, yangi loyiha tomoshabin vositasi ishlab chiqilgan va kod fragmentlarini ta’kidlab uchun ranglar uchun qo‘llab-quvvatlash yaxshilandi. Microsoft ta’kidlaganidek, Visual Studio 2012 uchun manba kodining umumiy hajmi 50 millionga yetdi.

Shunday qilib, Microsoft Visual Studio integrated development muhiti faol innovasion rivojlanishning noyob namunasidir va dasturiy ta’minotni ishlab chiqish sohasida yangi g‘oyalar va vositalarning bir butun birlashuvini o‘zida mujassamlashtiradi. 1995 yildan buyon 25 yil mobaynida 19 ta versiya chiqarilgan bo‘lib, ularning har birida yangi dasturlash tillari, buyruq-satr dasturlarini ishlab chiqish vositalari va boshqa ko‘plab yangiliklar mavjud.

Ba’zi bir kalit so‘zlarni bilib olish lozim:

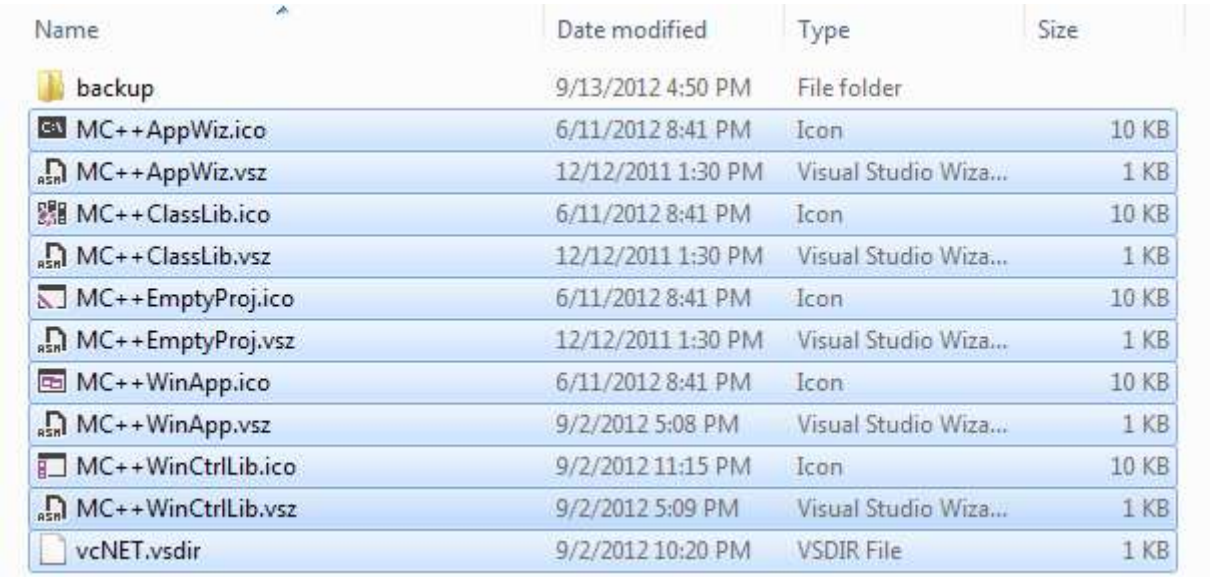
9.1-jadval. Kalit so‘zlar.

.NET	-ISO xalqaro standartlari bilan standartlashtirilgan va bir necha implementations ega bo‘lgan ob’ektga yo‘naltirilgan ko‘p tilli dasturlash platforma, - Microsoft.net, Microsoft tomonidan ishlab chiqilgan
C#	-Microsoft kompaniyasi tomonidan ishlab chiqilgan va faol ishlab chiqilgan eng yangi ob’ektga yo‘naltirilgan dasturlash tili. NET platformasi uchun asosiy dasturlash tili
Visual Basic	- Microsoft tomonidan ishlab chiqilgan va ishlab chiqilgan asosiy tilning ob’ektga yo‘naltirilgan kengaytmali dasturlash tili
Visual C++	- dasturlash tili, Microsoft tomonidan ishlab chiqilgan va ishlab chiqilgan C++ tilining kengaytmasi
Visual FoxPro	-Microsoft tomonidan ishlab chiqilgan ob’ektga yo‘naltirilgan dasturlash tillaridan biri, SQL so‘rovlar tili bilan mos ma’lumotlar bazalariga KIRISH uchun vositalarni o‘z ichiga olgan
Visual Studio	- Microsoft kompleks rivojlantirilgan integrallashgan muhit.

Visual C++ da Windows ilovalarni yaratish. Windows ilovalarni yaratishga KIRISHdan oldin muhitni o‘rnatish va uning imkoniyatlariga to‘xtalib o‘tamiz. Visual Studio standart o‘rnatish assida C++ tilini rivojlantirish uchun zarur elementlarni qo‘shish/o‘chirish imkonini beradi. Masalan, an’anaviy C++dasturlarini ishlab chiqish, C++da mobil ilovalarni ishlab chiqish, C++da Linux muhiti uchun vositalarni ishlab chiqish, C++tilida o‘yin ishlab chiqish va boshqa yana bir qator imkoniyatlarni beradi.

Visual C++ da Windows ilovalarni yaratish uchun avvalo, MS Visual Studio ni o‘rnatish kerak. Kompyuterning texnik imkoniyatlaridan kelib chiqqan holda, mos versiya tanlanadi. Mazkur versiyani o‘rnatish, internet manbalari juda kshp uchraydi. Odatda oddiy muhitlar kabi o‘rnatiladi.

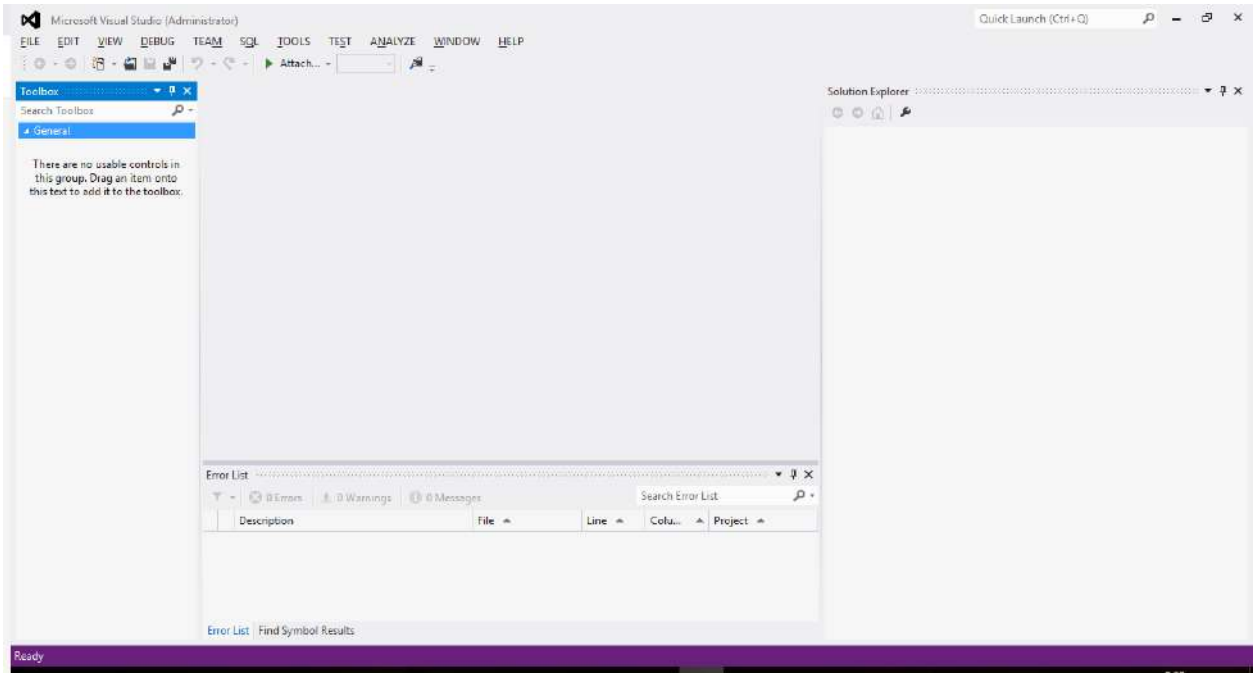
MS Visual Studio ning hamma versiyalarida Windows ilovalarni turlicha yaratish mumkin. Ammo eng qulay va umumiy foydalanuvchi uchun mos bo‘lgan interfeysni o‘rnatish uchun har xil versiyalarga har xil sozlashlarni bajarish kerak. Masalan, MS Visual Studio 2012 da Windows ilovalarni yaratish uchun, Windows oynasi tayyor komponentani o‘rnatish uchun, 1.1-rasmda keltirilgan fayllarni VCvcprojectsvcNET/ VC++ papkasiga nusxalash lozim.



Name	Date modified	Type	Size
backup	9/13/2012 4:50 PM	File folder	
MC++AppWiz.ico	6/11/2012 8:41 PM	Icon	10 KB
MC++AppWiz.vsz	12/12/2011 1:30 PM	Visual Studio Wiza...	1 KB
MC++ClassLib.ico	6/11/2012 8:41 PM	Icon	10 KB
MC++ClassLib.vsz	12/12/2011 1:30 PM	Visual Studio Wiza...	1 KB
MC++EmptyProj.ico	6/11/2012 8:41 PM	Icon	10 KB
MC++EmptyProj.vsz	12/12/2011 1:30 PM	Visual Studio Wiza...	1 KB
MC++WinApp.ico	6/11/2012 8:41 PM	Icon	10 KB
MC++WinApp.vsz	9/2/2012 5:08 PM	Visual Studio Wiza...	1 KB
MC++WinCtrlLib.ico	9/2/2012 11:15 PM	Icon	10 KB
MC++WinCtrlLib.vsz	9/2/2012 5:09 PM	Visual Studio Wiza...	1 KB
vcNET.vsdir	9/2/2012 10:20 PM	VSDIR File	1 KB

1.1-rasm. Windows ilovalarni yaratish uchun zaruriy fayllar

Shundan so‘ng, MS Visual Studio ni ishga tushiring, 1.2-rasmdagidek, integrallashgan muhit oynasi hosil bo‘ladi.



1.2-rasm. Integrallashgan muhit oynasi.

Muhit oynasi 5 qismdan iborat. Ular quyidagilar:

1. Sarlavha satri. Bunda tizimli menyu (chap tomondagi tizim belgisi), tizim va joriy proekt nomi, tizimli tugmachalar (yig‘ishtirish, oyna o‘lchamini o‘zgartirish, oynani yopish)
2. Muhitning menyusi.
3. Muhitning uskunalar paleni (foydalanuvchi tomonidan kerakli uskunalar bilan to‘ldirilishi mumkin).
4. Muhitning ishchi maydoni (Foydalanuvchining joriy proekt uchun zarur komponentalar va ularni boshqarish, kuzatish imkoniyatlari mavjud)
5. Muhitning holat satri.

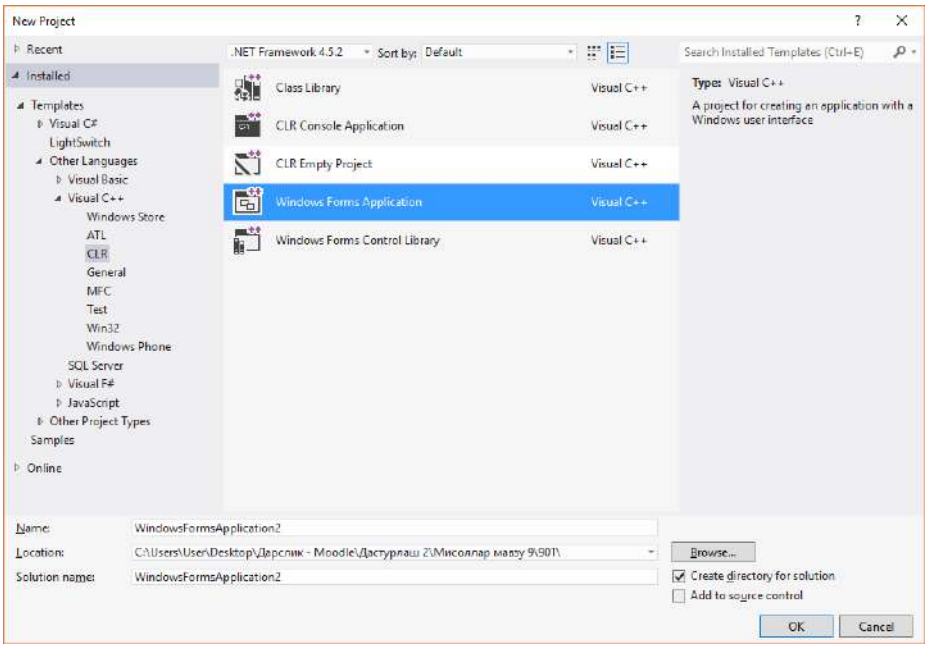
Yangi proekt yaratilishi bilan proektga mos kerakli va joriy qilingan komponentalar faollashadi. Windows ilovalarni yaratish uchun bir nechta varinatlar mavjud. Ularning eng ko‘p tarqalgan varianti menyudan foydalanib yaratishdir.

Windows ilovalarni yaratish uchun [File → New → Project] ketma ketligi yoki [Ctrl+Shift+N] tugmachalar majmuasini bosish yetarli. Ekranga [new Project] nomli muloqot oynasi hosil bo‘ladi (1.3-rasmga qarang).

Birinchi marta MS Visual Studio ishga tushirganda asosiy tilni tanlashni so‘raydi. Agar S++ tilini tanlagan bo‘lsangiz, avtomatik shu tilda yaratilishi mumkin bo‘lgan loyihalar ro‘yxati hosil bo‘ladi. Agar oldin boshqa tilni tanlagan bo‘lsangiz, [Other Languages] bandidan Visual C++ tilini tanlashingiz lozim. Visual C++ til asosida yaratish mumkin bo‘lgan konseptual loyihalar ro‘yxati chiqadi va bularning o‘zi bir nechta loyihalarga bo‘lnadi (1.3-rasmning chap tomonida qarang).

Visual C++ da Windows ilovalarni yaratish uchun avvalo [CLR] loyiha menedjeri tanlanadi. CLR - Common Language Runtime - bir necha qo‘llab-quvvatlanadigan tillarda har qanday yozilgan

dasturlar ijrosini boshqaradi dasturlash bo‘lib, ularni tillari har qanday yozilgan umumiy ob’ektga yo‘naltirilgan sinflarga almashish imkonini beruvchi loyiha menedjeridir.



1.3-rasm. Yangi loyiha yaratish oynasi.

[CLR] loyiha menedjeri bir nechta loyihalar yaratish imkonini beradi:

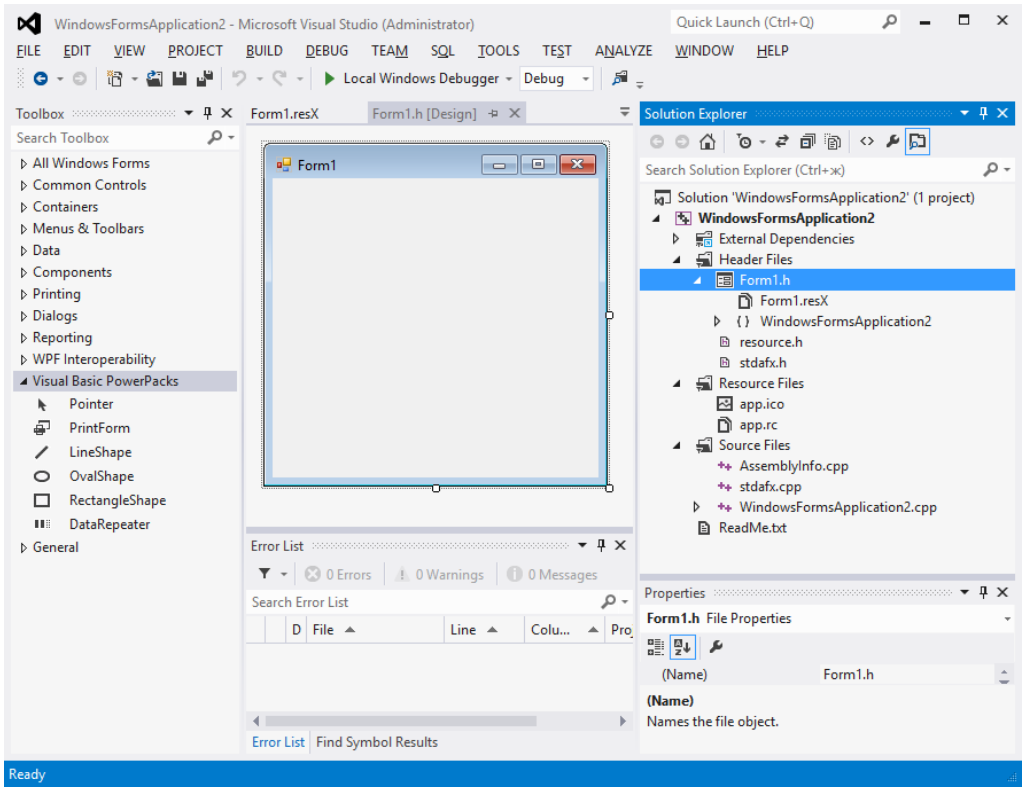
1. Class Library – boshqa proektlarga foydalanish uchun CLR sinflarni yaratish imkonini beradi.
2. CLR Console Application – CLR asosida konsol loyiha yaratish uchun ishlatiladi.
3. CLR Empty Project - yangi hech nima yozilmagan loyixalar asosida lokal loyihalarni yaratish uchun ishlatiladi.
4. Windows Forms application – Windows oynali loyihalarni yaratish imkonini beradi.
5. Windows Forms Control Library - Windows ilovalarni boshqarish uchun foydalaniladigan kutubxonalarni yaratish imkonini beradi.

Har bir loyihani o‘z o‘rnida yaratish foydalanish muzimdir va har birini tanlaganingizda ularning izohi oynaning o‘ng tomonida chiqadi.

[CLR] loyiha menedjeridan foydalanib loyiha yaratishda .NET Framework ning varianti ham muhim rol o‘ynadi. Shuning uchun .NET Framework larning imkoniyatlari bilan oldindan yaxshilab tanishib chiqish lozim. Har bir .NET Framework ning o‘zining imkoniyatlari bor va shuningdek Visual C++ da Windows ilovalarni yaratish uchun .NET Framework 4.5 yoki 4.5.2 variantlaridan foydalanishni tavsiya qilamiz. .NET Framework ni tanlash loyiha turini tanlash oynasining ustida joylashgan.

Loyiha yaratish oynasinig pastki qismida loyixa nomi, joylashgan joyini, loyihaning ishchi papkasining nomi ko‘rsatiladi.

Kerakli amallarni bajargandan so‘ng [ok] tugmasini bossangiz, bir nechta avtomatik sozlashdan so‘ng loyihaning ilk ko‘rinishi hosil bo‘ladi (1.4-rasmga qarang).



1.4-rasm. Oynali loyihaning dastlabki ko‘rinishi.

Oldin aytib o‘tilgandek, muhitning ishchi oynasini o‘zgardi. Uni odatda foydalanuvchi o‘zining xoxshi bo‘yicha joylashtirishi mumkin.

Eng aosiy elementlardan biri bu – Solition Explorer bo‘lib hisoblanadi. Bu loyiha papkasini boshqarish uchun qulay qilib yaratilgan. Bitta Ishchi papkada bir nechta loyiha bo‘lishi mumkin. Shuning uchun, u yerdan loyiha nomini tanlash kerak bo‘ladi va loyiha uchunn kerak bo‘limlar ro‘yxati chiqadi:

1. External Dependencis – tashqi, tizimli kerakli kutubxona, sinflar joylashadi.
2. Header Files – kutubxonalar fayllar.
3. Resource Files – loyixa resurs fayllari.
4. Source Files – dastur manba kodlari yozilgan fayllar.
5. ReadMe – matnli fayl bo‘lib, unda loyiha uchun izohlarni yozib qolirishingiz mumkin.

Standart loyiha yaratilganda yaratidgan joriy fayllarni o‘zgartirmang, bo‘lmasa loyiha ishlamasligi mumkin.

Odatda juda ko‘p foydalanuvchilar loyihalarda vizual ishlashni yoqtiradilar, buning uchun kutubxona fayllari majmausidan *.h kengaytmali foyllardan foydalaniladi. Uni bosganda loyihaning kerakli oynasi vizual ko‘rinishda hosil bo‘ladi.

Solition Explorer dan kerakli faylni tanlab, ustiga sichqonchani bossangiz shu faylni tahrirlash imkoniyati hosil bo‘ladi. Agar oldindan nomlar fazosi, sinflar, shablonlar bilan ishlagan bo‘lsangiz, ularning ham algoritmlari va funksiyalari bilan tanishish mumkin. Buning uchun kerakli chekni bosish kerak bo‘ladi.

Solition Explorer dan kerakli ob’ekt (fayl) ni tanlaganigizda uning xususiyatlari (properties)va funksiyalari (events) larni sozlash mumkin. Buning uchun dasturchiga properties oynasi yordam beradi. Ob’ektga mos holda ishlaydi.

Ishchi maydonning chap tomonida Solition Explorer dan tanlangan ob’ekt uchun qurish, integrallashtirish mumkin bo‘lgan uskunalar, metodlar va boshqa zaruriy xarakterga ega elementlar hosil bo‘ladi.

Ishchi maydonning aosiy markazida loyihani tahrirlash ishlari bajariladi va uning pastki qismida xatolar, xabarlar, ogohlantirishlar va boshqa dasturchi uchun kerakli barcha yordamchi oynalarni joylashtirish mumkin. Shuning bilingki, ishchi oynani dasturchi o‘ziga moslab tayyorlab shablon ko‘rinishida qilib olishi mumkin.

Muhitda menyular va uskunalar paneli. MS Visual Studio muhitining oynasi boshqa dasturlarning menyulari kabi joylashgan bo‘lib, 13 tadan iborat. Uskunalar paleni dasturchining xoxishi bilan almashib turishi mumkin. Shuning uchun faqat menyularning vazifasi va buyruqlari to‘xtalamiz. Odatda menyuga sichqoncha bilan murojaat qilinadi. Agar sichqoncha ishlamay qolsa, [Alt] tugmasini bosganda, menyu nomlariga tagiga chiqilgan harf paydo bo‘ladi. [Alt] tugmasini shu harf bilan bossangiz menyuga murojaat qilasiz.

[FILE] – muhitda loyiha ustida global hodisalarni bajarish uchun mo‘ljallangan asosiy menyulardan hisoblanadi. Unda quyidagi asosiy buyruqlar bor: yangi loyiha yaratish, loyihani ochish, loyiha biror bir yangi loyiha qo‘shish, faylni yopish, loyihani yopish, loyiha faylini va loyihani to‘liq saqlash, boshqa loyihalarni improt qilish, oldin ishlatilgan loyiha fayllari, muhitdan chiqish.

[EDIT] - muhitda tahrirlash ishalarini bajarishga mo‘ljallangan.

[VIEW] – muhitni ko‘rnishi, uskunalarni joylashtirish, keraklisiga murojaat qilishga mo‘ljallangan.

[PROJECT] – loyiha uchun elementlarni qo‘shish, boshqarish, sozlamalarni tayyorlash uchun ishlatiladi.

[BUILD] – loyihani qurish, qayta qurish, tozalash, paramertlarini sozlash va boshqa ishlarga mo‘ljallangan.

[DEBUG] – loyihani tekshirish, qadamba- qadam tekshirish, ishlatish, ishlashini tekshirish kabi amallarni bajarish uchun mo‘ljallangan.

[TEAM] – serverlarga ulanish uchun mo‘ljallangan

[SQL] – ma’lumotlar bazasini boshqarishning lingvistik ta’minoti bilan ishlashga qaratilgan.

[FORMAT] – muhitda turli formatlash ishlarini bajarishga mo‘ljallangan buyruqlari bor.

[TOOLS] – muhitda turli sozlamalarni amalga oshirish uchun qaratilgan menyu.

[TEST] – loyiha testlar va hisobotlarni olish uchun buyruqlar majmuasi joylashgan menyu.

[ANALYZE] – loiyhaning elementlarining o‘zaro integrallashgan holatda ishlashini tahlil qilish buyruqlari

[WINDOW] – muhitda oynalarni joylashtirish, boshqarish buyruqlari

[HELP] – muhit bo‘yicha yordam, turli rasmiy yordam muhitlaarga chiqish imkoniyatlarini beradigan buyruqlar joylashgan.

Buyruqlar va uskunalar paneli. Menyular va uskunalar paneli foydalanuvchilar vspackage buyruqlarga KIRISH uchun bir yo‘ldir. Buyruqlar-xujjatni chop etish, ko‘rinishni yangilash yoki yangi fayl yaratish kabi vazifalarni bajaruvchi funksiyalardir. Menyular va uskunalar paneli foydalanuvchilarga buyruqlarni taqdim etishning qulay grafik usullari ega. Odatda, tegishli buyruqlar bir xil Menyuda yoki bir xil uskunalar panelida guruhlanadi.

Menyular odatda integrallashgan muhiti (IDE) yoki uskunalar oynasining yuqori qismida ketma-ket guruhlangan bitta so‘zli satrlar sifatida ko‘rsatiladi. Sichqonchaning o‘ng tugmasini bosganda menyuni ham ko‘rsatish mumkin — bu holda ular kontekst menyular deb ataladi. Bosgandan so‘ng menyu ochiladi va bir yoki bir necha buyruqlarni ko‘rsatadi. Biror buyruq bosilgandan so‘ng vazifalar bajarilishi yoki qo‘shimcha buyruqlar bilan submenus ochilishi mumkin. Ba’zi ommabop menyu nomlari fayl, ko‘rish, va oyna, [Object window] bor.

Uskunalar paneli - Toolbars odatda tugmalari va boshqa elementlardan iborat satrlarda joylashgan, masalan, ro‘yxatlar jamlanmasi va menyu controllers kabi. Uskunalar paneli boshqaruvining buyruq bilan bog‘liq barcha elementlarini ta’minlaydi. Uskunalar paneli tugmasini bosish, uga bog‘liq buyruq faollashtiradi. Uskunalar paneli tugmalari odatda chop etish buyrug‘i uchun printer kabi asosiy buyruqlarni taklif qiluvchi belgilarni o‘z ichiga oladi. Ochiladigan ro‘yxatda ro‘yxatdagi har bir element boshqa buyruq bilan bog‘liq. Menyu tekshiruvi - bu gibriddir, unda nazoratning bir qismi uskunalar paneli tugmasi va ikkinchisi bosilganda qo‘shimcha buyruqlarni ko‘rsatadi.

Bir buyruq yaratishda, buning uchun bir hodisa yaratish kerak. Agar faollashtirilgan bo‘lsa, buyruq aniq yoki yoqilgan bo‘lsa hodisa bajarilishini bildiradi, uning matnini o‘zgartirish imkonini beradi va buyruqqa tegishli javob ta’minlaydi (yo‘naltiriladi). Ko‘p hollarda, IDE IOleCommandTarget interfeysi yordamida buyruqlar jarayonlari bajariladi. Visual Studio yo‘nalishidagi buyruqlar foydalanuvchi tanlovga asoslangan buyruqning ildiz kontekstidan boshlab va global tanlov asosida tashqi kontekstgacha ierarxiya sifatida ko‘rsatiladi. Asosiy menyuga qo‘shilgan buyruqlar ssenariylarda foydalanish uchun qulay.

Yangi menyular va uskunalar majmuasi aniqlash uchun, Visual Studio buyruq jadvalda (.vst) fayl orqali ularni tasvirlash kerak. Visual Studio paketi Shablon uchun bu faylni yaratadi, shuningdek kerakli elementlar shabolonda tanlangan har qanday buyruqlarni, uskunalar majmuasi va muharrirlarini qo‘llab-quvvatlashga kerak.

Visual Studio menyular va buyruqlar. Microsoft Officedan farqli o‘laroq, bir qancha alohida mahsulotlar paketidan iborat, Visual Studio ko‘p mahsulotlar o‘z ichiga oladi, qaysiki har bir global Visual Studio IDE buyruqlar o‘z to‘plamiga hissa qo‘shadi. IDE kontekst asosida foydalanuvchiga mavjud funksiyalarni filtrlash orqali minglab buyruqlarning murakkabligini boshqaradi.

Loyihalash oynasida kodni tahrirlash oynasiga o‘tish kabi foydalanuvchi konteksti o‘zgarganda, yangi kontekst bilan bog‘liq bo‘lmagan funksiyalar yo‘qoladi. Shu bilan bir vaqtda, yangi funksional tegishli dinamik ma’lumotlar bilan birga ochiladi, bunday Toolbox xususiyatlari va parametrlari sifatida qaraladi. Foydalanuvchi mavjud buyruqlar to‘plamini almashtirishni sezmasligi kerak. Foydalanuvchi chalg‘itdi yoki buyruqlar paydo yoki g‘oyib bilan adashtirmaslik, foydalanuvchi interfeysi interfeysi sozlanishi kerak. Joriy foydalanuvchi konteksti har doim IDE sarlavhasida, xususiyatlar oynasida yoki sahifalari muloqot oynasida bo‘lgani kabi bir yoki bir necha usulda ko‘rsatiladi.

Buyruq to‘plamilar ulga moslashuvchanligini ta’minlaydi. Visual Studio muhitiga xos bo‘lgan yagona buyruq strukturalari asosiy menyu va asosiy buyruqlar satri bo‘lib, ularni sozlash va hatto yashirish mumkin. Boshqa buyruqlar majmuasi ilovaning holatiga qarab paydo bo‘ladi va yo‘qoladi. Oynalar uskunasi va hujjat muharrirlari, shuningdek, ajralmas oynalar uskunalar majmuasi o‘z ichiga olishi mumkin.

menyular qachon mavjud bo‘lsa, birgalikda buyruqlar va buyruq guruhlaridan foydalaning.

Buyruqlar odatda kontekst asosida namoyish etiladi, chunki, mavjud menyular va buyruq guruhlar yordamida buyruq tuzilishi kontekstida o‘zgarishlar o‘rtasida nisbatan barqaror bo‘lib qolishini ta’minlaydi. Buyruqlar gruhini qayta ishlatish va tegishli buyruqlar yaqinida yangi buyruqlarni joylashtirish IDE murakkabligini kamaytiradi va yanada samimiy tajriba yaratadi. Agar yangi buyruq aniqlash kerak bo‘lsa, mavjud buyruq guruhiga uni qo‘yib ko‘rish mumkin. Agar yangi guruh aniqlash kerak bo‘lsa, yangi yuqori darajali menyuni yaratish va bog‘liq buyruq guruhi yonida menyusida joylashtirish mumkin.

Har bir buyruq uchun piktogramma yaratish yaxshi emas. Buyruq belgisini yaratishni diqqat bilan o‘ylab ko‘ring. Piktogramma faqat quyidagi buyruqlar uchun yaratilishi kerak:

* joriy uskunalar panelida ko‘rsatiladiganlar.

* foydalanuvchilar tomonidan uskunalar paneliga qo‘shish, maxsus muloqot orqali mumkin.

* Microsoftning mahsuloti bir xil harakat bilan bog‘liq buyruqlarga.

Klaviatura tugmalarini qo‘shishni cheklash. Foydalanuvchilarning aksariyati barcha mavjud tugmalarning kichik qismini ishlatadi. Agar shubhangiz bo‘lsa, funksiyani klaviatura tugmasi bilan bog‘lamang. Yangi yorliqlarni qo‘shishdan oldin foydalanuvchilar guruhi bilan ishlash.

Komandalarga standart menyu joyini berish. Jamoalar boshqalar tomonidan tuzilgan va shunga ko‘ra mo‘ljallangan bo‘ladi. Yashirin buyruq degan narsa yo‘q. Barcha Visual Studio

buyruqlari Tools > configure muloqot oynasida, buyruq oynasida, avtomatik yakunlash, > > options > klaviatura muloqot oynasida va development tools muhitida (DTE) ko‘rsatiladi. Buyruqlar nomini va vositalari majmuini berishga ishonch hosil qilish kerak va foydalanuvchilar osongina ularni topish mumkin, shunday qilib, .ctc fayl yaratish kerak.

Ichki uskunalar panelidagi umumiy buyruqlarni takrorlamang. Buyruqlarni foydalanuvchining diqqat markaziga yaqin joyda joylashtirish foydalidir. Buning bir yo‘li uskuna oynasi yoki hujjat muharririning yuqori qismida o‘rnatilgan uskunalar panelini yaratishdir. Uskunalar paneliga joylashtirilgan buyruqlar oynadagi kontent maydoniga xos bo‘lishi kerak. Bu uskuna majmuasining umumiy buyruqlarni nusxa qilmang. Masalan, ajralmas uskunalar panelida saqlash belgisini joylashtirish kerak emas.

MS Visual Studio muhitining interaktiv tugmalari.

Asosiy tugmalar

Ctrl+Shift+P, F1	Buyruqlar palitrasini ko‘rsatish
Ctrl+P	Tez ochish, faylga o‘tish…
Ctrl+Shift+N	Yangi oyna yaratish
Ctrl+Shift+W	Oynani yopish
Ctrl+,	Foydalanuvchi sozlamalari
Ctrl+K Ctrl+S	Klaviatura yorliqlar

Asosiy tahrirlash tugmalari

Ctrl+X	Buferga olish va o‘chirish
Ctrl+C	Buferdan nusxa joylashtirish
Alt+ ↑ / ↓	qatorni yuqoriga / pastga ko‘chirish
Shift+Alt + ↓ / ↑	Qator nusxasini yuqoriga / pastga ko‘chirish
Ctrl+Shift+K	Qatorni o‘chirish
Ctrl+Enter	Yangi qator joylashtirish
Ctrl+Shift+Enter	Yuqoridan yangi qator joylashtirish
Ctrl+Shift+	Mos keladigan qavsga o‘tish
Ctrl+] / [	Qatorga abzas qo‘shish/o‘chirish
Home / End	Satr boshi / oxiriga o‘tish
Ctrl+Home	Fayl boshiga o‘tish
Ctrl+End	Fayl oxiriga o‘tish
Ctrl+↑ / ↓	qatordan yuqoriga / pastga harakatlashtirish
Alt+PgUp / PgDn	sahifa bo‘yicha yuqoriga / pastga harakatlashtirish
Ctrl+Shift+[	Operator ta’sir doirasini belgilash
Ctrl+Shift+]	Operator ta’sir doirasini belgilashni bekor qilish
Ctrl+K Ctrl+[	Ta’sir doirani ochish
Ctrl+K Ctrl+]	Ta’sir doirani yopish
Ctrl+K Ctrl+0	Qatorlarga izoh qo‘yish
Ctrl+K Ctrl+J	Qatorlardan izohni o‘chirish
Ctrl+K Ctrl+C	Satrga izoh qo‘shish
Ctrl+K Ctrl+U	Satrdan izoh o‘chirish
Ctrl+ Z	Amalni bekor qilish

Navigatsiya tugmalari

Ctrl+G	Kerakli qatorga borish
F8	Navbatdagi xatoga borish
Shift+F8	Navbatdagi xato va xabarga borish

Ctrl+Shift+Tab	Tahrirlash oynalariga o‘tish
Ctrl+M	Fokus tugunlarni o‘zgartirish

Izlash va o‘zgartirish tugmalari

Ctrl+F	Izlash
Ctrl+H	O‘zgartirish
F3 / Shift+F3	Izlashning keyingisiga o‘tish
Alt+Enter	Izlash natijalarini belgilash

Dasturlashga ko‘maklashuvchi tugmalar

Ctrl+Space	Tegishli funksiyalarni chaqirish
Ctrl+Shift+Space	Tegishli funksiya variantlarini chaqirish
Shift+Alt+F	Dasturni formatlash
Ctrl+E Ctrl+F	Dastur fragmentini formatlash
Ctrl+Shift +.	Dastur matnini kattalashtirish
Ctrl+Shift +,	Dastur matnini kichiklashtirish

Bu interaktiv tugmachalardan tashqari o‘rganib olishingiz kerak bo‘lgan tugmachalar juda ko‘p, ularni dasturlash davomida foydalanishga o‘rganib olasiz degan umiddamiz.

Form xususiyatlari va hodisalari. Form bu loyiha yaratilgan hosil bo‘ladigan birinchi oyna hisoblanadi. Oynaning joriy holatlari mavjud. Ularni loyiha boshqaruvchisi panelidan ko‘rish mumkin. Oyna uchun yangi form1 sinfi yaratiladi. Bu sinf form sinfining merosxo‘ri hisoblanadi. U quyidagicha aniqlangan:

```
public ref class Form1 : public System::Windows::Forms::Form {}
```

Shu sinfning ichida InitializeComponent funksiyasi bor va u formani yaratilishi javob beradi. Uning dastlabki sozlamalari quyidagi fragmentda keltirilgan:

```
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->ClientSize = System::Drawing::Size(284, 261);
this->Name = L"Form1";
this->StartPosition =
System::Windows::Forms::FormStartPosition::CenterScreen;
this->Text = L"Form1";
this->ResumeLayout(false);
```

Imkoniyat qadar bu dastur fragmentlariga o‘zgartirish kiritish shart emas, bularni hammasini vizual bajarish maqsadga muvofiq.

Vizual bajarish deganda formaning vizual loyihasi bilan ishlash kerak. Loyiha yaratuvchisi uchun quyidagilar muhim hisoblanadi:

- Formani ustiga sichqonchani bossangiz u aktivlashadi.

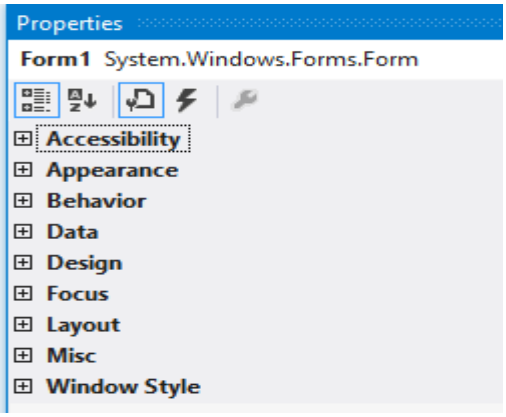
- Kontekst menyudan foydalanib, uning dastur fragmentiga o‘tish mumkin (F7 tugmasi orqali xam)

- Dastur fragmentidan forma loyihasiga o‘tish uchun kontekst menyudan foydalanish mumkin (Shirft + F7 tugmasi orqali ham)

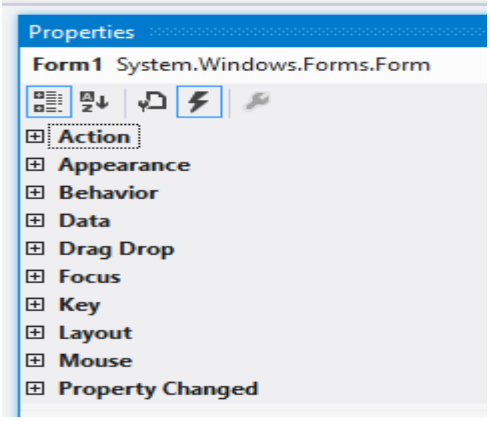
- Kontekst menyu oqali formani boshqarishni qulflab qo‘yish va ochish mumkin (lock/unlock controls)

- Kontekst menyu oqali formani xususiyatlari va hodisalariga o‘tishsh mumkin (Properties)

Formaning xususiyatlar va hodisalar quyidagi bo‘limlardan iborat (1.5 va 1.6-rasmlarga qarang)



1.5-rasm. Formaning xususiyatlari



1.6-rasm. Formaning hodisalari

Properties muloqot oynasida 4 ta uskunalar mavjud. Ularning vazifalari:

- forma paramertlarini guruhlab saralashni ta’minlaydi.

- forma paramertlarini alfavit bo‘yicha saralashni ta’minlaydi.

- forma xususiyatlari ro‘yxati.

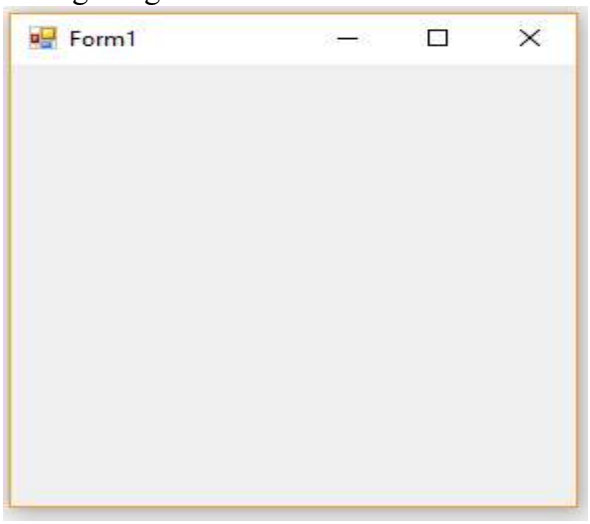
- forma hodisalari ro‘yxati.

E’tibor bergan bo‘lsangiz, tanlangan joriy tur ko‘k to‘rtburchakka olingan bo‘ladi. Bu yerada bajarilgan har bir ish formaga mos sinfga o‘zgartirib boriladi. Endi forma xususiyatlarining ba’zilari bilan tanishib chiqamiz.

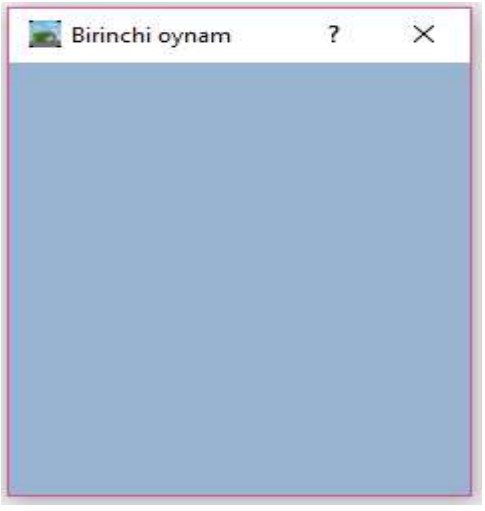
1	**Accessibility guruhi**	Buning 3 ta xususiyati bor, bu xususiyatlari bir forma bilan o‘zgarishini vhaniqlab bo‘lmaydi shuning uchun keyinroq to‘xtalamiz.
2	**Appearance guruhi**	Formaning tashqi ko‘rinishi uchun ishlatiladi va 11 ta xususiyat bor
	BackColor	Formaga rang o‘rnatish
	BackgroundImage	Rasm o‘rnatish
	BackgroundImageLayout	Rasm o‘rnatish tartibi
	Cursor	Kerakli kursor belgisini tanlash
	Font	Formada yozuv xususiyatlarini o‘rnatish
	ForeColor	Yozuv rangini tanlash
	FormBorderStyle	Formaning stilini o‘rnatish
	RightToLeft	Forma sarlavhasini o‘ngdan chapga o‘rnatish
	RightToLeftLayout	Forma sarlavhasini o‘ngdan chapga o‘rnatish tartibi
	Text	Sarlavha nomi
	UseWaitCursor	Kutish kursoridan foydalanishni o‘rnatish
3	**Behavior guruhi**	Formaning rejimlarini o‘rnatish va 6 xususiyati bor
	ContextMenuStrip	Kontekst menyu o‘rnatish
	DoubleBuffered	Bufer o‘rnatish
	Enabled	Formani yoqish va o‘chirish
4	**Data guruhi**	Formani ichki va tashqi ma’lumotlar bilan bog‘lash uchun ishlatiladi va 3 xususityati bor. Bu guruhni keyingi elementlari bilan ishlaganda batavsil qaraymiz

5	**Design guruhi**	Formani loyihalash uchun ishlatiladi va 3 ta xususiyati bor
	Language	Formaga joriy tilni o‘rnatish
	Localizable	Formaga lokalizatsiyani o‘rnatish
	locked	Formani yopish yoki ochish
6	**Focus guruhi**	Formada fokuslarni boshqarish uchun ishlatiladi va bitta xususiyati bor
	CausesValidation	Fokuslarni tekshirish bosqoichlari
7	**Layout guruhi**	Formani tartiblash uchun ishlatiladi va 13 ta xususiyati bor
	AutoScroll	Avtomatik formani kuzatish tugmalarini o‘rnatish
	AutoSize	Avtomatik o‘lchamni joriy qilishni o‘rnatish
	Lacation	Formani hosil bo‘lish joyi belgilashni o‘rnatish
	MaximumSize	Avtomatik katta holda chiqish
	MimimumSize	Avtomatik kichiklashtrilgan xoda chiqish
	Size	Formaga o‘lcham o‘rnatish
	StartPosition	Formaning chiqish joyini o‘rnatish
	WindowsState	Formaning turi belgalash
8	**Misc guruhi**	Boshqa xususiyatlarni o‘rnatish, formani turiga qarab o‘zgarib turadi va 3 ta xususiyati bor
9	**Windows Style guruhi**	Oyna stillarini o‘rnatish uchun mo‘ljallangan guruh bo‘lib, 13 ta xususiyatga ega.
	ControlBox	Boshqarish tugmalarini joylashtirishni aniqlash
	HelpButton	Yordam chaqiruvchi tugmani o‘rnatish
	Icon	Oyna belgisini o‘rnatish
	MaximizeBox	Kattalashtirish tugmasini o‘rnatish
	MinimizeBox	Kichiklashtirish tugmasini o‘rnatish
	Opacity	Oynaning tinqligini sozlashni o‘rnatish

Mazkur xususiyatlardan foydalanib, formaning ba’zi xususityalarini o‘zgartiramiz va uning sinfning ichiga kirib ko‘rishimiz mumkin:



1.7-rasm. Formaning dastlabki ko‘rinishi



1.8-rasm. Formaning ishlov berilgan ko‘rinishi

Formaga ishlov berilgan xususiyatlari aniqlash uchun sinfiga murojjat qilish mumkin. Quyidagi dastur fragmentlarini olamiz.

Formaning dastlabki o‘rnatilgan xususiyatlari	Formaning ishlov berilgan xususiyatlari	+ / -
`this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);`	`this->AutoScaleDimensions = System::Drawing::SizeF(11, 22);`	+

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;	this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;	-
	this->BackColor = System::Drawing::SystemColors::ActiveCaption;	+
this->ClientSize = System::Drawing::Size(284, 261);	this->ClientSize = System::Drawing::Size(230, 257);	+
	this->Cursor = System::Windows::Forms::Cursors::Hand;	+
	this->Font = (gcnew System::Drawing::Font(L"Times New Roman", 14.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(204)));	+
	this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::Fixed3D;	+
	this->HelpButton = true;	+
	this->Icon = (cli::safe_cast<System::Drawing::Icon^ >(resources->GetObject(L"$this.Icon")));	+
	this->Margin = System::Windows::Forms::Padding(6, 6, 6, 6);	+
	this->MaximizeBox = false;	+
	this->MinimizeBox = false;	+
this->Name = L"Form1";	this->Name = L"Form1";	-
this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;	this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;	-
this->Text = L"Form1";	this->Text = L"Birinchi oynam";	+
this->ResumeLayout(false);	this->ResumeLayout(false);	-

Birinchi navbatda muhitning o‘zi ushbu xususiyatlarni o‘rnatishni va tarbilashni nazorat qiladi. Formada quyidagi o‘zgarishlar amalga oshirilgan: formaning chiqish maydoni, rangi, joriy o‘lchamlari, oyna kursori, oynaning yozuv turi, ko‘rinishi, yordamchi tugma o‘rnatilgan, ikonkasi o‘zgartirilshan, holati o‘zgartirilgan, kattalashtirish va kichiklashtirish tugmalari olib tashlangan, oynaning sarlavhasi o‘zgargan.

Bularni amalga oshirish uchun 2-3 munut vaqt ketadi va hammasi vizual bajariladi. Bunda oldin qilingan o‘zgarishlar o‘chirib, oxirgi o‘zgarishlar qo‘shib boriladi.

Formaning hodisalari bilan tanishib chiqamiz. O‘zi hodisa nima? Hodisa bu bir vazifa bajarilgan jarayoni bo‘lishi mumkin, masalan, formani yaratish uchun 3 ta hodisa bo‘lishi mumkin, forma yaratilishidan oldin, forma yaratilayotgan vaqt, forma yaratilib bo‘lingandan keyin.

Ularning ham 1.6- rasmda keltirilgan 10 ta guruhi bor. Ularning ba’zilariga izoh berib o‘tamiz

1	**Action guruhi**	Global hodisalar guruhi bo‘lib, 8 ta hodisadan iborat
	Click	Formaga tugma bosilganda
	DoubleClick	Formada tugma 2 marta bosiganda
	MouseCaptureChanged	Sichqonchaning holati o‘zgarganda
	MouseClick	Sichqoncha bosilganda
	MouseDoubleClick	Sichqoncha ikki marta bosilganda
	ResizeBegin	Forma o‘lchami o‘zgarishi boshlanganda
	ResizeEnd	Forma o‘lchami o‘zgarishi tugaganda
2	**Appearance guruhi**	Formaning ichki hodisalari uchun ishlatiladi va bitta xususichti bor, u bilan keyingi mavzular tanishamiz. t
3	**Behavior guruhi**	Forma holatlari uchun hodisalarni o‘z ichiga oladi va 15 ta hodisadan iborat
	ControlAdded	Formaga biror boshqarish qurilganda
	ControlRemoved	Formadan biror bshqaruv o‘chirilganda
	FormClosed	Forma yopilganda
	FormClosing	Forma yopilishidan oldi holatda
	HelpButtonClicked	Yordam tugmasi bosilganda
	InputLanguageChanged	Til o‘zgartirilganda
	InputLanguageChaning	Til o‘zgartirilayotganda
	Load	Forma yuklanayotganda
	Shown	Forma chqirilganda
4	**Data guruhi**	Ichki va tashqi ma’lumotlar bilan ishlash hodisalari bo‘lib, 2 ta hodisadan iborat, bu xususiyatlar ma’lumotlar bazasi bilan ishlaganda tushunarli bo‘ladi
5	**Drag drop guruhi**	Formada harakatlanish hodisalari bo‘lib, 6 ta hodisadan iborat
	DragDrop	Biror bir harakat bo‘lganda
	DragEnter	Inter tugmasi bosilganda
	DragLeave	To‘xtash harakati bo‘lganda
	DragOver	Ketish harakati bo‘lganda
	GiveFeedback	Teskari aloqa qo‘shish
6	**Focus guruhi**	Formada fokuslarni boshqarish uchun bo‘lib, 6 ta hodisani o‘z ichiga oladi
	Activated	Fokus aktiv bo‘lganda
	Enter	Fokusda ma’lumot kiritilganda
	Leave	Fokusda to‘xtash sodir bo‘lganda
	Validated	Tekshirish tugaganidan so‘ng
	Validating	Tekshirish vaqtida
7	**Key guruhi**	Tugmalar bilash ishlash hodisalari uchun bo‘lib, 6 ta hodisadan iborat
	KeyDown	Tugma kelganda
	KeyPess	Tugma bosilganda
	KeyUp	Tugma ketganda
	PreviewKeyDown	Tugma kelganda
8	**Layout guruhi**	Tartib guruhi formani atrofidagilar bilan ishlash tartibini nazora qilash uchun rejalashtirilgan
	DpiCahged	Joriy forma tartibi o‘zgarganda

	DpiCahgedAfterParent	Meros bergan sinf tartibi o‘zgargandan keyin
	DpiCahgedBeforParent	Meros bergan sinf tartibi o‘zgargandan oldin
	Layout	Forma tartibini o‘rnatish
	MdiChildActivate	Forma sinfining meros xo‘ri faol bo‘lganda
	Move	Forma ko‘chirilganda
	Resize	Forma o‘lchamini o‘zgartirganda
9	**Mouse guruhi**	Sichqoncha bilan bo‘ladigan hodisalar bo‘lib, 6 ta hodisadan iborat
	MouseDown	Sichqoncha kelganda
	MouseEnter	Sichqoncha joriy holatida
	MouseHover	Sichqoncha ozgina harakatida
	MouseLeave	Sichqonchaga to‘xtash bo‘lganda
	MouseMove	Sichqoncha ko‘chganda
	MouseUp	Sichqoncha ketganda
10	**Property Changed guruhi**	Formaning xususiyatlari o‘zgarganda bajariladigan 12 hodisasi bor
	AutoSizeChanged	Avtomatik o‘lcham o‘zgarganda
	AutoValidateChanged	Avtomatik tekshirib bo‘lganda
	BackColorChanged	Fon rangi o‘zgarganda
	EnabledChanged	Holat o‘zgarganda

Hodisalarni o‘rnatish juda qulay hisoblanadi. Kerakli hodisaning o‘ng tomoniga sichqonchani ikki marta bossangiz, hodicha mos bo‘lgan dastur franmentini yozish uchun funksiya yaratib beradi. Bir misol bilan ko‘rib chiqamiz.

Buning uchun MessageBox sinfining show funksiyasidan foydalanamiz. Bu funksiya statik funksiya bo‘lib, System::Windows::Forms nomlar fazosiga tegishli， 21 variantda yozilishi mumkin. U natija sifatida DialogResult tipini qaytaradi. Uning argumentlari:

Text - String tipini qabul qiladi va matni xabar sifatida chiqaradi

Caption - String tipini qabul qiladi va xabar sarlavhasini chiqaradi

Buttons - MessageBoxButtons tipini qabul qiladi va xabarni qabul qilish tugmalarini chiqaradi, AbortRetryIgnore, OK, OKCancel, RetryCancel, YesNo, YesNoCancel qiymatlarni qabul qiladi.

Icon - MessageBoxIcon tipini qabul qiladi va xabarni ikonkasini chiqaradi, Asterisk, Error, Exclamation, Hand, Information, None, Question qiymatlarni qabul qiladi

defaultButton - MessageBoxDefaultButton tipini qabul qiladi va xabarni tugmalarini belgilaydi, Button1, Button2, Button3 qiymatlari qabul qiladi.

Options - MessageBoxOptions tipini qabul qiladi va xabarni chiqish varintlarini aniqlaydi, DefaultDesktopOnly, RightAlign, RtlReading, ServiceNotification qiymatlari qabul qiladi.

Bu funksiyaning boshqa argumentlari ham bor, ularni keyinchalik o‘rganish mumkin.

MessageBox sinfining show funksiyasidan foydalanish:

```
MessageBox::Show("Xabar",        "Sar…",        MessageBoxButtons::OK,
MessageBoxIcon::Information );
```

Forma hodisalaridan foydalanish uchun 3 tasini ko‘richb chiqamiz:

1. Click hodisasini ishlatish uchun, uning o‘ng tomoniga sichqonchani bosamiz. Forma sinifida tayyor Form1_Click() nomli funksiya yaratiladi. Bu funksiyani ichida quyidagicha fragmentni yozamiz:

```
    MessageBox::Show("Click  bo`ldi",  "Xabar",  MessageBoxButtons::OK,
MessageBoxIcon::Information);
```

Shuningdek formaning InitializeComponent() funksiyasini ichida avtomatik tarzda this->Click += gcnew System::EventHandler(this, &Form1::Form1_Click); hodisa qo‘shib qo‘yilgan. Agar bu hodisani to‘liq o‘chirmoqchi bo‘lsangiz, Click hodisasining o‘ng tomoniga yozilgan Form1_Click() ni o‘chirib, enter tugmasini bosish yetarli o‘zi avtomatik InitializeComponent() funksiyasini ichidagi hodisani o‘chiradi. Dasturning qismidan funksiyani o‘chirish mumkin.

Quyidagi ikki hodisani ishlatish uchun bir masala olamiz. Yordam tugmasi bosilganda, yordam tugmasi bosildi deb xabar chiqsin. Foydalanuvchi ha deb javob bersa, oynaga qaytadi. Agar yo‘q deb bossa, yolg‘onchimi siz deb chiqadi va foydalanuvchi, faqat ha javobni tanlaydi. Javob tanlaganda oynaning sarlavhasi yolg‘onchi deb o‘zgarsin. Oyna holat hodisalari orqali bu tekshirib, yana oyna sarlavhasi o‘zgardi degan habarni chiqarsin.

2. HelpButtonClicked hodisasini ishlatish uchun, uning o‘ng tomoniga sichqonchani bosamiz. Forma sinifida tayyor Form1_HelpButtonClicked () nomli funksiya yaratiladi. Bu funksiyani ichida quyidagicha fragmentni yozamiz:

```
System::Windows::Forms::DialogResult result;
                        result =
MessageBox::Show("HelpBuutonClicked","Xabar",MessageBoxButtons::YesNo,M
essageBoxIcon::Hand, MessageBoxDefaultButton::Button1);
                        if(result ==
System::Windows::Forms::DialogResult::No) {
                                MessageBox::Show("Yolg`nchimi siz
!","Xabar",MessageBoxButtons::OK,MessageBoxIcon::Warning);
                                Form1::Text = "Yolg`nchi";
                        }
                    }
```

Shuningdek formaning InitializeComponent() funksiyasini ichida avtomatik tarzda this->HelpButtonClicked += gcnew System::ComponentModel::CancelEventHandler(this, &Form1::Form1_HelpButtonClicked); hodisa qo‘shib qo‘yilgan.

3. TextChanged hodisasini ishlatish uchun, uning o‘ng tomoniga sichqonchani bosamiz. Forma sinifida tayyor Form1_TextChanged() nomli funksiya yaratiladi. Bu funksiyani ichida quyidagicha fragmentni yozamiz:

```
MessageBox::Show("Forma                                          sarlavhasi
o`zgardi","Xabar",MessageBoxButtons::OK,
MessageBoxIcon::Information,MessageBoxDefaultButton::Button3,MessageBox
Options::RightAlign);
```

Shuningdek formaning InitializeComponent() funksiyasini ichida avtomatik tarzda this->TextChanged += gcnew System::EventHandler(this, &Form1::Form1_TextChanged); hodisa qo‘shib qo‘yilgan.

Bu hodisalarni ishlatib ko‘rsangiz, ancha narsalarni farqiga borishingiz kerak.

Matnda bir joyda forma, bir joyda oyna, bir joyda form deb ishlatilgan bo‘lsa, ularning hammasi bir ma’noni beradi. Shu bilan formaning xususiyatlari va hodisalari bilan tanishib chiqdingiz va uni o‘rganishni davom etish kerak. Chunki, keyingi barcha ishlaringiz shu kabi formalar bilan bog‘liq. Keyinchalik formaning ma’lum bir xususiyati va hodisalari bilan ishlaganda. Albatta uni ta’kidlab aytib o‘tib ketamiz.

Nazariy savollar.

1. Integrallashgan ishlab chiqarish muhitlarining qayday turlarini bilasiz?
2. Qachondan boshlab integrallashgan muhitlar jadal rivojlangan?
3. Turbo muhitlarni sanab bering.
4. Ob’ektga yo‘naltirilgan tilni birinchi bo‘lib integrallashgan muhitlarga kim va qaysi firma joriy qilgan.
5. Qanday muhitda zamonaviy matn muharrirlari kodi avtomatik bajarilishini ta’minlash (kodi tugatish), muharriri muhitda joriy yozilgan kodi sintaktik to‘g‘ri va uning davomi bo‘lishi mumkin.
6. Taniqli NetBeans integrallashgan muhitni dastlab Java dasturlash uchun qaysi universitet talabasi loyihasi sifatida yaratgan.
7. Vizual Studio 97 haqida nimalarni bilasiz?
8. Visual Studio 2012 haqida nimalarni bilasiz?
9. ISO xalqaro standartlari bilan standartlashtirilgan va bir necha implementations ega bo‘lgan ob’ektga yo‘naltirilgan ko‘p tilli dasturlash platformani ayting?
10. MS Visual Studio muhit oynasi nechta qismdan iborat?
11. Windows ilovalarni yaratish uchun qaysi tugmachalar majmuasini bosish yetarli.
12. Qaysi vosita bir necha qo‘llab-quvvatlanadigan tillarda har qanday yozilgan dasturlar ijrosini boshqaradi.
13. CLR Empty Project qanday loyiha yaratish uchun kerak.
14. Windows Forms Control Library qanday loyiha yaratish uchun kerak.
15. Loyihaning External Dependencis papkasida qanday fayllar saqlanadi.
16. [FILE] menyusining vazifalarini sanab bering.
17. [TOOLS] menyusining vazifalarini sanab bering.
18. Menyular va uskunalar paneli foydalanuvchilar qanday papka buyruqlarga KIRISH uchun bir yo‘ldir.
19. Bir buyruq yaratishda, buning uchun bir avval nima yaratish kerak.

20. Yangi menyular va uskunalar majmuasi aniqlash uchun, Visual Studioda qanday jadval fayl orqali ularni tasvirlash kerak.
21. Qatorlarga izoh qo‘yish tugmasini ayting.
22. Satrga izoh qo‘shish tugmasini ayting.
23. Tegishli funksiya variantlarini chaqirish tugmasini ayting.
24. [Ctrl+Shift +.] tugmachalar majmuasi qanday amal bajaradi.
25. [Ctrl+Shift+Tab] tugmachalar majmuasi qanday amal bajaradi.
26. Oyna uchun yangi form1 sinfi yaratiladi. Bu sinf qaysi sinfining merosxo‘ri hisoblanadi
27. Kontekst menyu oqali formaning qanday funksiyalarini bajarish mumkin.
28. Formaning rejimlarini o‘rnatish qaerda joylashgan.
29. Ichki va tashqi ma’lumotlar bilan ishlash hodisalari qaerda joylashgan.
30. MessageBox sinfining nechta argumentlari bor, va har biriga izoh bering.

12- MA’RUZA

KOMPONENTALAR BILAN ISHLASH (2 SOAT)

Reja:

1. Komponenta tushunchasi va xususiyatlari.
2. Standart komponentalarining xususiyatlari va hodisalari.
3. Additional komponentalarining xususiyatlari va hodisalari.
4. System komponentalarining xususiyatlari va hodisalari.

Annotatsiya*: MS Visual Studio-da ishlash printsipining asosiy qulayliklari uning tarkibiy qismidir. Foydalanuvchi interfeysi tahlili shuni ko'rsatadiki, u ko'plab qo'shimchalarga ega. Bu juda oddiy. MS Visual Studio Visual C++da turli xil vazifalarni bajarish uchun tarkibiy qismlarni o'z ichiga oladi. Ular asosan Asboblar qutisi deb ham ataladi. Asboblar qutisidagi har bir komponentda nom maydoni va sinflar mavjud. Ulardan foydalanganda ushbu sinflarning merosxo'rlari yaratiladi va bitta oynada bir xil turdagi bir nechta komponentlardan foydalanish mumkin.*

Kalit so‘zlar*: IDE, Visual C++, user interface, OYD, komponenta, xususiyatlari, hodisalari, Toolbox, Tab, Add Tab, Delete Tab, Mantlarni tahrirlash, Ma‘lumotlarni ko‘rsatish (faqat o‘qish uchun), CheckBox, NumericUpDown, Label, Button, Additional komponentasi, Panel, TabControl, System komponentasi, MenuStrip, ContextMenuStrip.*

KIRISH

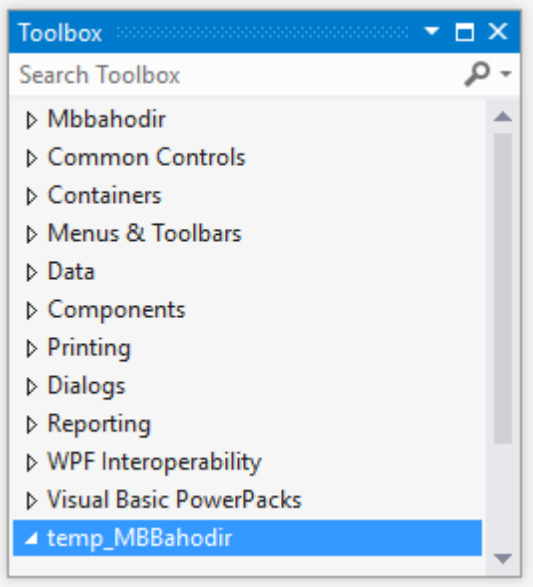
MS Visual Studio da ishlash tamoylining asosiy qulayliklari bu komponentalaridir. Foydalnuvchining UI ni tahlil qilib qaralsa, unda juda ko‘plab kompnentalar joylashtirilgan. Bu juda osondir. MS Visual Studio Visual C++ da turli vazifalarni amalga oshirish komponentalari muvjud. Ularni asosan Toolbox deb ham yuritiladi. Toolbox dagi har bir komponentaning nomlar fazosi va

sinflari mavjud. Ulardan foydalanganda shu sinflardan merosxo‘r olib yaratiladi va bir oynada bir xil tipdagi kompanentadan bir nechtasidan foydalanish mumkin.

Komponenta tushunchasi va xususiyatlari. Komponenta bu MS Visual Studio ning eng asosiy qurolidir. Dasturchi bular orqali tez dasturlash va IDE muhitning imkoniyatlaridan foydalaniladi. Komponentalar to‘plamini Toolbox deb yuritiladi.

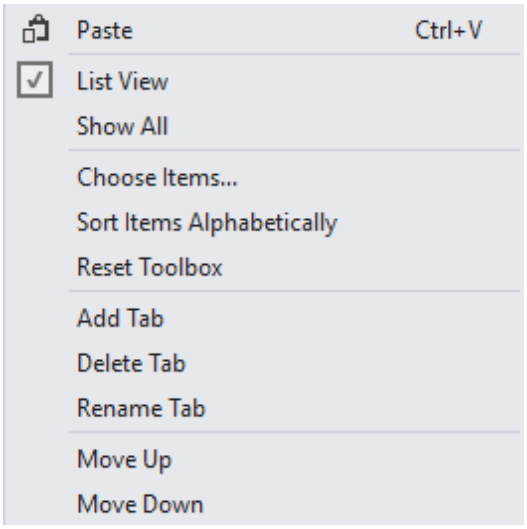
Toolbox interaktiv oynasi muhitning ixtiyoriy joyida joylashgan bo‘lshimi mumkin. Loyihalovchi uni o‘ziga mos qilib joylashtirish mumkin. Agar muhitning oynasida Toolbox oynasini ko‘rmayotgan bo‘lsangiz, menyudan foydalanib, [view

→ Toolbox] buyruqlari bajaring. [Ctrl + W, X] tugmachalar majmuasini ham bosish orqali Toolbox oynasiga o‘tish/chaqirish mumkin. Muhit komponentalar soni nechta degan savolga javob berishdan boshlaymiz. Muhit integrallashgan muhitligini bilamiz, shuning uchun undagi komponentalar soni oldindan xech kim ayta olmaydi. Yangi kutubxonalarni joriy qilish yangi komponentalarni olib kiradi. Standart foydalanuvchilar uchun Toolbox oynasi 1.1-rasmda tasvirlangan.

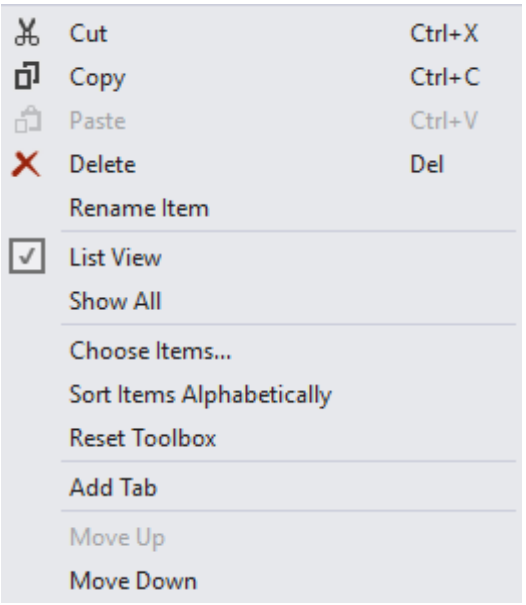


1.1-rasm. Toolbox oynasi.

Toolbox oynasi tahrirlash mumkin. Buning uchun sichqonchaning o‘ng tugmasini Tab ga, ya‘ni komponentalar guruhining nomiga bosib, 10.2-rasmdagi kontekst menyuni, Tab ning ichiga kirib komponentani ustiga bossak 10.3- rasmdagi kontekst menyuni chaqiramiz.



1.2-rasm. komponentalar guruhi uchun kontekst menyu.



1.3-rasm. Komponenta uchun kontekst menyu.

Rasmlarga qarasak, ularda bir xil nomli buyruqlar uchraydi, bular butun Toolbox oynasi uchun xizmat qiladi va 5 ta bo‘limdan iborat. Har bo‘limga va uning buyruqlariga to‘xtalib o‘tamiz:

1- Bo‘lim. Komponentalar ustida amal bajarish uchun mo‘ljallangan bo‘lib, [Cut] – komponentani buferga olish va joyidan o‘chirish uchun, [Copy] komponentaning nusxasini olishsh uchun, [Paste] – komponentaning olingan nusxasini joylashtirish, buni komponenta guruhiga qo‘llash mumkin, [Delete] – komponentani guruhdan o‘chirib tashlash, [Rename item] – komponenta nomini o‘zgartirish.

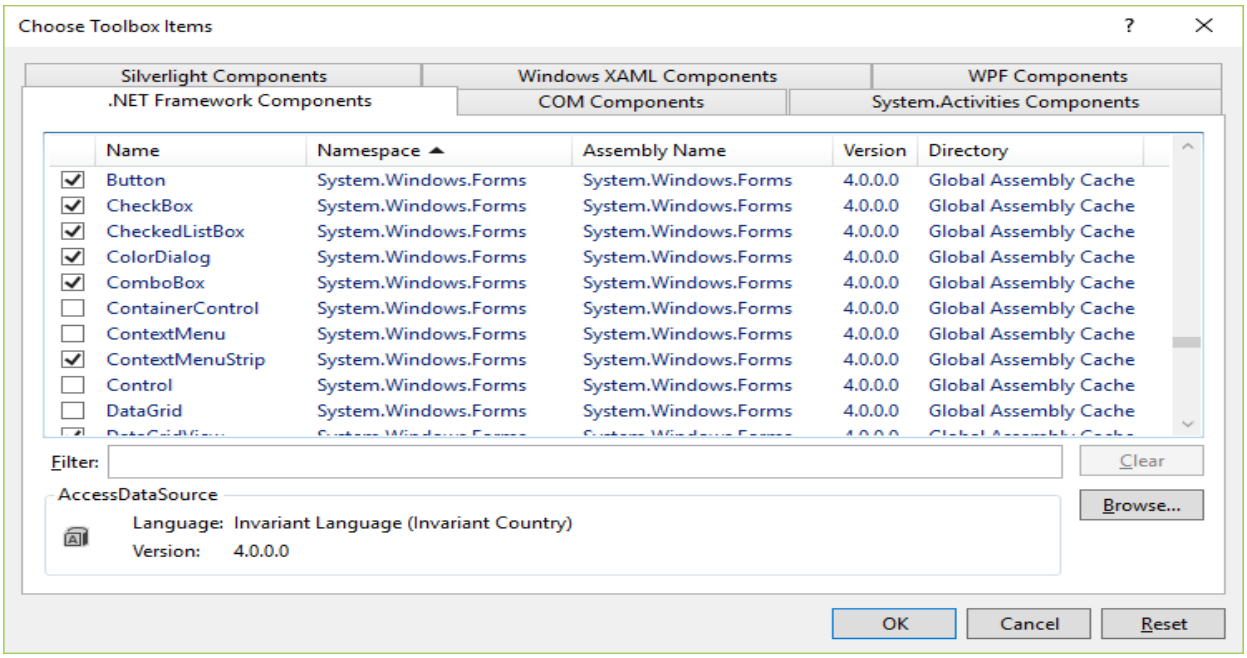
2- Bo‘lim. Toolbox oynasiga xizmat qiladi.[List View] – tanlagan guruh komponentalarini ro‘yxat qilib chiqaradi, o‘chirilgan bo‘lsa, uskunalar to‘plami sifatida ko‘rsatadi. [Show All] – muhitdagi barcha faollashtirilgan komponentalar chiqarib beradi. komponentlar juda ko‘pligi uchun bu ishlatish o‘ta mushkul ish deb hisoblayman. Tajribamda men ham bularning hammasini ishlatolmaganman.

3- Bo‘lim. Toolbox oynasiga xizmat qiladi va komponentalar hosil qilish uchun ishlatiladi. [Choose Items..] – yangi komponentalarni qo‘shish uchun ishlatiladi, [Sort Items Alphabetically] – komponentalarni alfavit bo‘yicha saralash, [Reset Toolbox] - Toolbox komponentalarni sozlamalarini standart shaklga keltirish.

4- Bo‘lim. Toolbox oynasiga Tab larni, ya‘ni komponenta guruhlari ustida ishlash imkoniyatini beradi. [Add Tab] - yangi komponenta guruhi qo‘shish, [Delete Tab] - komponenta guruhini o‘chirish, [Rename Tab] - komponenta guruhi qayta nomlash, standart nomlaganlarini ham bu amallar bilan o‘zgartirish mumkin.

5- Bo‘lim. komponenta guruhi va komponentalarning o‘rinlarini almashtirish uchun ishlatiladi. [Move Up] – yuqoriga ko‘chirish, [Move Down] – pastga ko‘chirish. Bularning barchasi muhitning dasturchiga moslashuvchanligi bildiradi.

Yangi komponentalarni qo‘shish ko‘rib chiqaylik, bu buyruq bosilishi bilan 1.4-rasmda tasvirlangan muloqot oynasi chiqadi.



1.4-rasm. Yangi komponentalarni qo‘shish.

Bu rasmdan komponentalarning turlari ko‘rish mumkin, har bir bo‘limi 1000 dan ortiq sinflarga ega. Har bir sinf esa, bitta komponenta hisoblanadi. Agar tizimda mavjud bo‘lmagan komponentalarni ham shu oyna orqali qo‘shish mumkin, masalan, MySql, oqimlar bilan ishlash, OpenCV, OpenMP, Fuzzy ToolBox kabi komponentalarni qo‘shish mumkin.

Komponentaning xususiyatlari. Har bir komponentaning shunday xususiyatlari bor. Eslab ko‘ring OYDda sinfni yaratish vaqtida nimalar yaratilar edi. Xuddi shuningdek, har bir komponeta sinf, uning xususiyatlari (properties), usullari (methods) va hodisalari (events) mavjud.

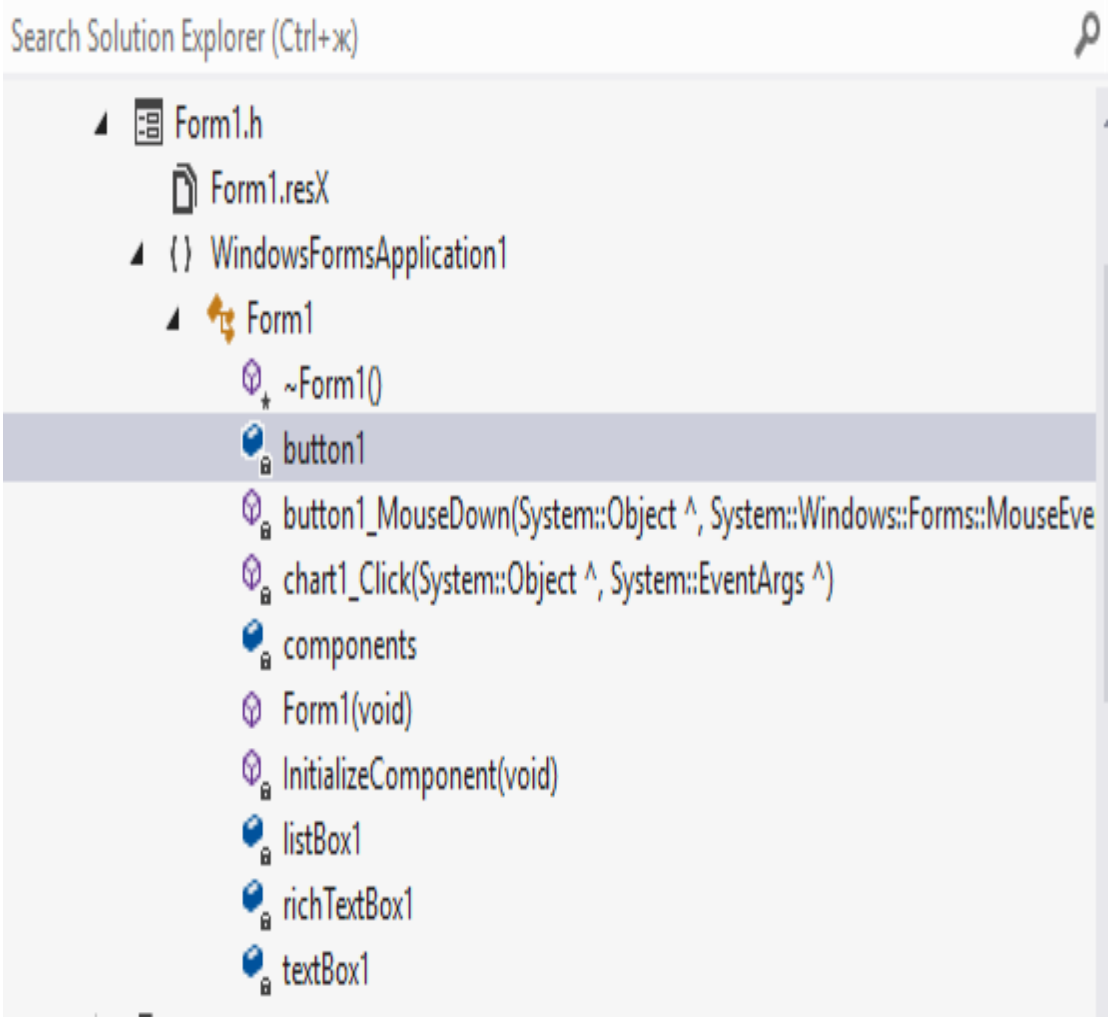
Komponentaning xususiyatlari quyidagi guruhlarga bo‘linadi (1.1-jadvalga qarang)

1.1-jadval. Komponentaning xususiyatlari

1	**Accessibility guruhi**	Bu guruhdagi xususiyatlar bir komponentada o‘zgarishlarni aniqlash va o‘rnatish uchun ishlatiladi.
2	**Appearance guruhi**	Komponentaning tashqi ko‘rinishi uchun xususiyatlarni o‘rnatishga mo‘ljallangan.
3	**Behavior guruhi**	Komponentaning rejim xususiyatlarini o‘rnatishga mo‘ljallangan.

4	**Data guruhi**	Komponentani ichki va tashqi ma‘lumotlar bilan bog‘lash xususityayalari uchun ishlatiladi.
5	**Design guruhi**	Komponentani loyihalash xususiyatlari uchun ishlatiladi
6	**Focus guruhi**	Komponentada fokuslarni boshqarish xususiyatlari uchun ishlatiladi
7	**Layout guruhi**	Komponentani tartiblash xususiyatlari uchun ishlatiladi
8	**Misc guruhi**	Boshqa xususiyatlarni o‘rnatish, komponentani turiga qarab o‘zgarib turadi.

Bu xususiyat guruhlari komponenta tipiga moslashgan holda doimiy o‘zgarib turadi va tipga mos xususiyatlarni aniqlab beradi. Komponenta xususiyatlarga qiymatlar maxsus, to‘plamdan tanlash, sonli, matnli, mantiqiy kabi ma‘lum tiplar qimatini oladi. Agar murakkab tip bo‘lsa, albatta muloqot oynasi orqali tanlash mumkin. Komponenta xususiyatlariga qiymatlarni berish vizuat amalga oshiriladi va asosiy formaga uning dastur fragmentlari yozib ketiladi. Buni loyiha boshqaruv



oynasi orqali ko‘rish mumkin (1.5-rasmga qarang)

Oynaning InitializeComponent funksiyasida ham komponentalarning xususiyatlari ko‘shilganini ko‘rish mumkin.

```
void InitializeComponent(void)
        {
            this->button1 = (gcnew System::Windows::Forms::Button());
            this->listBox1 = (gcnew System::Windows::Forms::ListBox());
            this->SuspendLayout();
            //
            // button1
            //
            this->button1->Location = System::Drawing::Point(208, 73);
            this->button1->Name = L"button1";
            this->button1->Size = System::Drawing::Size(75, 23);
            this->button1->TabIndex = 0;
            this->button1->Text = L"button1";
            this->button1->UseVisualStyleBackColor = true;
            this->button1->MouseDown += gcnew
System::Windows::Forms::MouseEventHandler(this,  &Form1::button1_MouseDown);
            //
            // listBox1
            //
            this->listBox1->FormattingEnabled = true;
            this->listBox1->Location = System::Drawing::Point(363, 62);
            this->listBox1->Name = L"listBox1";
            this->listBox1->Size = System::Drawing::Size(120, 95);
            this->listBox1->TabIndex = 1; // Form1 ...//
            }
```

1.5-rasm. Komponentalarning qo‘shilganligi.

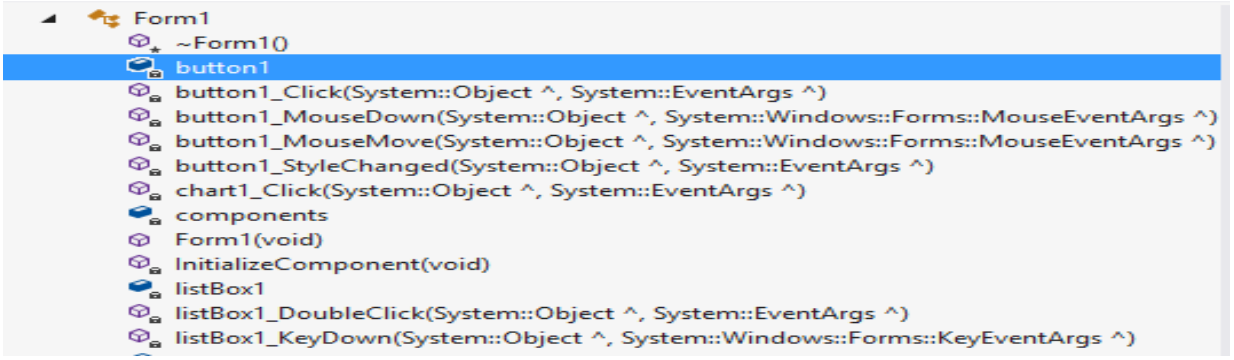
Komponentaning usullari (methods) va hodisalari (events) quyidagi guruhlarga bo‘linadi (1.2-jadvalga qarang)

1.2-jadval. Komponentaning usullari (methods) va hodisalari (events)

1	**Action guruhi**	Komponenta uchun global hodisalar faollashtirish va ma‘lum bir algoritm yozish

2	**Appearance guruhi**	Komponentaning ichki hodisalari uchun ishlatiladi va ma‘lum bir algoritm yozish
3	**Behavior guruhi**	Komponenta holatlari uchun hodisalarni faollashtirish
4	**Data guruhi**	Ichki va tashqi ma‘lumotlar bilan ishlash hodisalarini faollashtirish
5	**Drag drop guruhi**	Komponentaning harakatlanish hodisalarini faollashtirish
6	**Focus guruhi**	Komponentada fokuslarni boshqarish hodisalarini faollashtirish
7	**Key guruhi**	Komponentaning mos tugmalar bilash ishlash hodisalarini faollashtirish
8	**Layout guruhi**	Komponentada atrofidagilar bilan ishlash tartibini nazora qilish hodisalarini faollashtirish
9	**Mouse guruhi**	Komponentada sichqoncha bilan bo‘ladigan hodisalarini faollashtirish
10	**Property Changed guruhi**	Komponentaning xususiyatlari o‘zgarganda bajariladigan hodisalarini faollashtirish

Bu usullar va hodisalarning odatda hammasi ham vizual ishlaish imkoni yo‘q. Shuning uchun agar komponentaning biror funksiya On bilan boshlansa bilingki bu usul, ed, ing bilan tugagan funksiyalari bo‘lsa, bular ko‘proq hodisalar hisoblanadi. Guruhlar komponenta tipiga moslashgan holda doimiy o‘zgarib turadi va tipga mos hodisalarini aniqlab beradi. Komponenta hodisalari funksiya sifatida yaratiladi va dasturchi unga kerakli o‘zining algoritmini yozadi. Komponenta hodisalarini qo‘shish vizual amalga oshiriladi va asosiy formaga uning dastur fragmentlari yozib ketiladi. Ammo dasturchi o‘z fknutsiyaning algoritmini yozishi kerak. Buni loyiha boshqaruv oynasi orqali ko‘rish mumkin (1.6-rasmga arang)



1.6-rasm. Komponentalarning hodisalari.

Oynaning InitializeComponent funksiyasida ham komponentalarning hodisalari ko‘shilganini ko‘rish mumkin.

```
void InitializeComponent(void){
        this->button1 = (gcnew System::Windows::Forms::Button());
        this->listBox1 = (gcnew System::Windows::Forms::ListBox());
        this->SuspendLayout();
        //
        // button1
        //
        // xususiyatlar
        this->button1->Click += gcnew System::EventHandler(this, &Form1::button1_Click);
        this->button1->MouseDown += gcnew
System::Windows::Forms::MouseEventHandler(this,
        &Form1::button1_MouseDown); this->button1->MouseMove += gcnew
System::Windows::Forms::MouseEventHandler(this,  &Form1::button1_MouseMove);
this->button1->StyleChanged += gcnew System::EventHandler(this,
&Form1::button1_StyleChanged);
        //
        // listBox1
        //
        // xususiyatlar
        this->listBox1->DoubleClick += gcnew System::EventHandler(this,
&Form1::listBox1_DoubleClick);
        this->listBox1->KeyDown += gcnew System::Windows::Forms::KeyEventHandler(this,
&Form1::listBox1_KeyDown);
        //
        // boshqa komponentalar
        // Form1
        // … xususiyatlar
        // hodisalar
        this->Controls->Add(this->listBox1);
        this->Controls->Add(this->button1);
        // … xususiyatlar
}
private: System::Void button1_MouseDown(System::Object^  sender,
System::Windows::Forms::MouseEventArgs^  e) { }
```

```
private: System::Void listBox1_DoubleClick(System::Object^ sender, System::EventArgs^ e) {     }
private: System::Void listBox1_KeyDown(System::Object^ sender, System::Windows::Forms::KeyEventArgs^ e) { }
private: System::Void button1_MouseMove(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {}
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) { }
private: System::Void button1_StyleChanged(System::Object^ sender, System::EventArgs^ e) { }
```

Shuningdek, ba‘zi komponentalar faol va faol bo‘lmagan holatlarda bo‘lishi mumkin. Bu huddi windows oynalaridagi buyruqlarning foydalanishiga o‘xshash ishlatiladi. Komponentalarni o‘z joyida ishaltishni ham bilish kerak va ularni qachon ishlaish kerakligini bilash ham muhim.

Standart komponentasining xususiyatlari va hodisalari. Standart komponentalarga Toolbox oynasidagi [Common Controls] Tab dagi komponentalar kiradi. Ularni tarkibini o‘zgartirish mumkin, ya‘ni yangilarini qo‘shish va keraksizlarini o‘chirib tashlash mumkin. Shuningdek, agar maxsus ishlab chiqilgan yangi komponentalar guruhi bo‘lsa ulardan ham foydalanish mumkin. Standart komponentalarni quyidagi guruhlarga bo‘linadi va shu asosida ularning hususiyatlari va hodisalari 95% bir xil bo‘ladi.

1.3-jadval. Standart komponentalarni guruhlari va vazifalari.

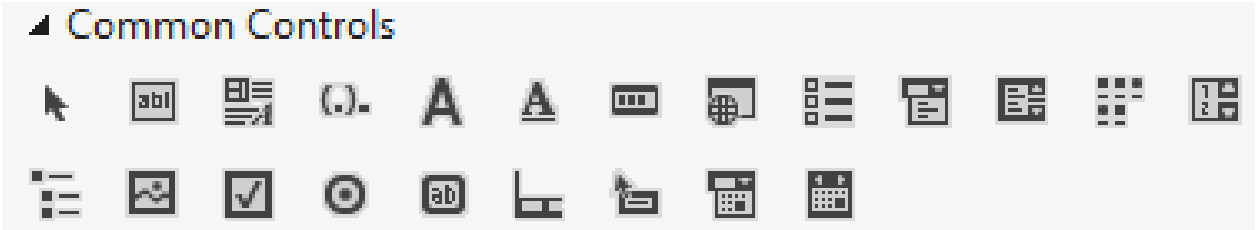
Komponentaning guruhi	**Komponentaning nomi**	**Vazifasi va izoh**
Mantlarni tahrirlash	TextBox	Matnni tahrirlash imkonini beradi, dastur ishlayotgan vaqtda foydalanuvchi yoki algoritm yordamida matnni tahrirlash.
	RichTextBox	Oddiy va RTF formatida Matnni tahrirlash imkonini beradi, dastur ishlayotgan vaqtda foydalanuvchi yoki algoritm yordamida matnni tahrirlash.
	MaskedTextBox	Foydalanuvchi tomonidan kiritilayotgan matnlarni maxsus belgi bilan himoyalaydi
Ma‘lumotlarni ko‘rsatish (faqat o‘qish uchun)	Label	Foydalanuvchi tomonidan to‘g‘ridan- to‘g‘ri tahrir qilinmaydi matnni ko‘rsatadi.

	LinkLabel	Veb-link sifatida matnni ko‘rsatadi va foydalanuvchi joriy matnni bosganda boshqa oynaga yoki veb-saytga havola faollashtiradi.
	ProgressBar	Foydalanuvchi uchun joriy amal bajarilish jarayonini ko‘rsatadi.
Veb-sahifani ko‘rsatish	WebBrowser	Foydalanuvchi oynasida veb-sahifalar orqali harakat qilish imkonini beradi.
Ro‘yxatdan tanlash	CheckedListBox	Har biri katak bilan birga bo‘lgan elementlarning takrorlanadigan ro‘yxatini ko‘rsatadi.
	ComboBox	Qalqib chiquvchi elementlar ro‘yxatini

	ListBox	Matnlar va grafik elementlar ro‘yxatini ko‘rsatadi
	ListView	Elementlarni to‘rt xil ko‘rinishdan birida ko‘rsatadi. Ko‘rinishlarga faqat matn, kichik piktogrammali matn, katta piktogrammali matn va batafsil ko‘rinish kiradi.
	NumericUpDown	Foydalanuvchilar yuqoriga va pastga tugmalari yordamida o‘tiish mumkin bo‘lgan raqamlar ro‘yxatini ko‘rsatadi.
	TreeView	Qo‘shimcha kataklar yoki piktogrammalar bilan matndan iborat bo‘lishi mumkin bo‘lgan tugun ob‘yektlarining iyerarxik to‘plamini ko‘rsatadi.
Grafiklarni tasvirlash	PictureBox	bitmaps va piktogramma kabi tasvir fayllarini ko‘rsatadi.
parametrli qiymat	CheckBox	Matn uchun katakcha va yorliqni ko‘rsatadi. Odatda parametrlarni belgilash uchun ishlatiladi.
	CheckedListBox	Har biri katak bilan birga bo‘lgan elementlarning kaytariladigan ro‘yxatini ko‘rsatadi.
	RadioButton	Yoqilgan yoki o‘chirilishi mumkin bo‘lgan tugmani ko‘rsatadi.
	TrackBar	Foydalanuvchilarga shkala bo‘ylab "

		Thumb" ni harakatlantirib shkala qiymatlarini o‘rnatish imkonini beradi.
Buyruqlar	Button	Boshlanadi, to‘xtaydi yoki jarayonni bekor qiladi.
	LinkLabel	Veb-link sifatida matnni ko‘rsatadi va foydalanuvchi joriy matnni bosganda boshqa oynaga yoki veb-saytga havola faollashtiradi.
	NotifyIcon	Fonda ishlaydigan dasturni ifodalovchi vazifalarni paneli holati xabarnomasidagi belgini ko‘rsatadi.
Foydalanuvchi uchun yordam	HelpProvider	Elementlari uchun pop-up yordam oynasini yoki tezkor yordam oynasini beradi.
	ToolTip	Foydalanuvchi nazorat bilan hovers nazorat maqsadi bo‘yicha qisqacha tavsifi ko‘rsatadi va bir pop-up oyna beradi.
Vaqt	DateTimePicker	Foydalanuvchilarga sana yoki vaqtni tanlash imkonini beradigan grafik taqvim ko‘rsatadi.
	MonthCalendar	Foydalanuvchilarga sana qatorini tanlash imkonini beradigan grafik taqvim ko‘rsatadi.

Standart komponentalarni guruhlashning asosiy maqsadi, guruhlarga mos xususiyatlar va hodisalari bir xil bo‘ladi. Bu komponentalarning Toolbox oynasida quyidagicha ko‘rinishga ega (1.7-rasmga qarang)



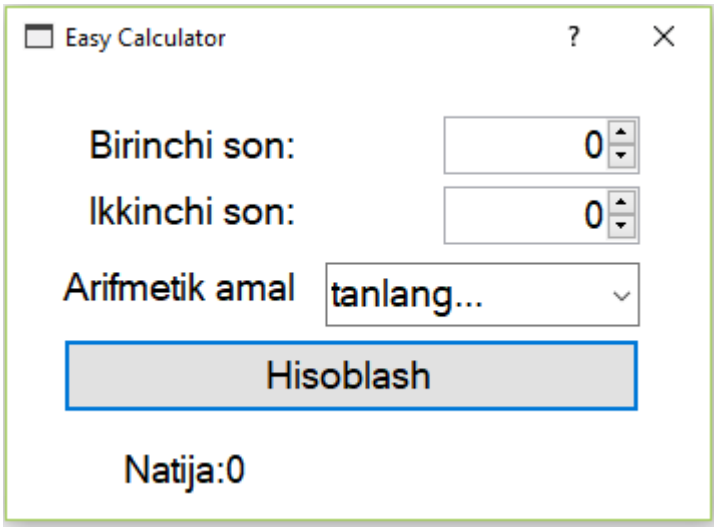
1.7-rasm. Standart komponentalar ko‘rinishi.

Bu komponentalarni oynaga joylashtirish uchun sichqonchaning chap bitta bosiladi va oynaga keltirib chap tomoni yana bir marta bosiladi va oynada tanlangan komponentaning nusxasi paydo bo‘ladi. Bu nusxani ob‘yekt deb aytamiz. Chunki eslab ko‘ring, sinfning nusxasi bu ob‘yekt deb aytiladi. Ob‘yektning ustiga sichqonchaning o‘ng tugmasini bossangiz, u bilan bajarish mumkin bo‘lgan amallarni ko‘rasiz. Agar ob‘yektdan nusxa olsangiz uning super sinfining, ya‘ni shu ob‘yekt

tipidagi boshqa bir ob‘yekt yaratiladi. Ikkita bir xil tipdagi ob‘yektlarning xususiyatlari va hodisalariga alohida – alohida ishlov beriladi.

Standart komponentalarni xususiyat va hodisalarini ko‘rib chiqish uchun arifmetik amallarni bajaruvchi dastur yaratish masalasini olamiz.

Mantiqiy jihatdan sonlarni kiritish uchun 2 ta NumericUpDown, matnli yozuvlar uchun 4 ta Label, hisoblash uchun bitta Button, amallar uchun bitta CheckBox komponentlarini olamiz va oynaga quyidagi 1.8-rasmdagidek qilib joylashtiramiz.



1.8-rasm. Standart komponentalarni xususiyat va hodisalaridan foydalanish.

Bu masalani loyiha sifatida amalga oshirish uchun quyidagi qadamlar bajariladi:

1-qadam. Forma xususiyatlarini o‘rnatish: BackColor xususiyatiga oq rang; font xususiyatiga 14 o‘lchamli yozuv; Icon ga maxsus ikinka; MaximizeBox xususiyatiga false qiymat; MinimizeBox xususiyatiga false qiymat; Size xususiyatiga yangi o‘lchamlar; StartPosition – CenterScreen holati; Text xususiyatiga “Easy Calculator” matnni yoziladi;

Buni qanday ajratishsh mumkin, ya‘ni o‘zgargan xususiyatlarni, qiymatining matni yog‘on qora ranga kirganlari o‘zgartirilgan hisoblanadi. Bu o‘zgarishlarni dastur fragment ko‘rinida keltiish ham mumkin. Bu dastur fragmentini qo‘lda yozish kerak emas, ammo tushunarli bo‘lishi uchun keltiramiz. Buni InitializeComponent funksiyasi ichida kerakli izohlar bilan keltirilgan bo‘ladi.

```
this->AutoScaleMode =
System::Windows::Forms::AutoScaleMode::Font; this->ClientSize =
System::Drawing::Size(351, 225);
this->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14.25F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
static_cast<System::Byte>(204)));
this->HelpButton = true;
this->Icon = (cli::safe_cast<System::Drawing::Icon^ >(resources-
>GetObject(L"$this.Icon")));
this->MaximizeBox = false;
this->MinimizeBox = false;
this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;
this->Text = L"Easy Calculator";
```

2-qadam. Label1 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->label1->Text = L"Birinchi son:";
```

3-qadam. Label2 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->label2->Text = L"Ikkinchi son:";
```

4-qadam. Label3 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->label3->Text = L"Natija:";
```

5-qadam. numericUpDown1 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->numericUpDown1->TextAlign = System::Windows::Forms::HorizontalAlignment::Right;
```

6-qadam. numericUpDown2 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->numericUpDown2->TextAlign = System::Windows::Forms::HorizontalAlignment::Right;
```

7-qadam. comboBox1 ob‘yektining xususiyatlarini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(5) {L"[+]
(qo‘shish)", L"[-] (ayirish)", L"[*] (ko‘paytirish)",
                                L"[/] (butun bo‘lish)", L"[%] (qoldiq bo‘lish)"});
this->comboBox1->Text = L"tanlang...";
```

8-qadam. label4 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->label4->Text = L"Arifmetik amal";
```

9-qadam. button1 ob‘yektining xususiyati va hodisani o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->button1->Text = L"Hisoblash";
this->button1->Click += gcnew System::EventHandler(this, &Form1::button1_Click);
```

button1_Click funksiyasiga quyidagi algoritmni yozamiz.

```
int one = Convert::ToInt16(Math::Round(numericUpDown1->Value,
0)); int two = Convert::ToInt16(numericUpDown2->Value);
int result = 0;
label3->Text = "Natija:";

switch(comboBox1->SelectedIndex){
        case 0: result = one + two; break; case
        1: result = one - two; break; case 2:
        result = one * two; break; case 3:
        result = one / two; break; case 4:
        result = one % two; break;
                default: result = 0;
                                break;
 }
 label3->Text += result.ToString();
```

Algoritmga izoh brish shart emas, chunki ko‘p kamchiliklari mavjud, masalan, bo‘lish arifmetik amalida. Ammo ba‘zi xulosalarni chiqarish mumkin.

[numericUpDown1->Value] ob‘yektning tanlangan haqiqiy tipdagi qiymatini qaytaruvchi xususiyat, [Convert::ToInt16] – ixtiyoriy tipni butun tipga almashtirib beradi, [label3->Text] – ob‘yektning matn xususiyatini o‘zgartirishi mumkin, [comboBox1->SelectedIndex] – tanlagichning tanlangan indeksini qaytaradi, [label3->Text += result.ToString();] – ob‘yektning matninga matn qo‘shish va tipni satrli tipga o‘tkazish ko‘rsatilgan.

Dastur fragmentida 3 ta belgi [::] – cinfning usullariga murojaat qilish, [->] – ob‘yektning xususiyatiga murojaat qilish, [.] ob‘yektning qiymati ustida bajarilishi kerak bo‘lgan amal. Visual C++ da juda ko‘p usul, va funksiyalarni ishlatish tamoyllari bor, ularni faqat kompilyatorning imkoniyatiga qarab ajratish mumkin.

Standart komponentalarning ba‘zi xususiyatlari va hodisalari, ular uchun ba‘zi usullarni ishlatiyu ko‘rdik. Bularni chuqurroq o‘rganish uchun ko‘proq amaliy ishlarni bajarish kerak.

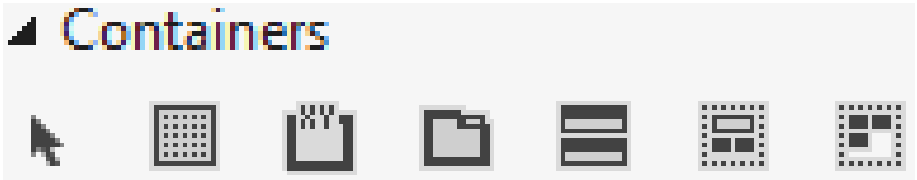
Additional komponentasining xususiyatlari va hodisalari. Bu standart komponentalar bilash ishlash uchun qo‘shimcha komponentlardir. Bular asosan boshqaruv elementlarni

guruhlash uchun ishlatiladi. Quyidagi jadvalda qo‘shimacha komponentalarni va vazifalarini keltiramiz.

1.4-jadval. Additional komponentalarining guruhlari va vazifalari

Komponenta-ning guruhi	Komponentaning nomi	Vazifasi va izoh
Boshqaruvlar elementlarni guruhlash	Panel	elementlarni boshqarish uchun to‘plamini guruhlaydi
	GroupBox	Nomlangan elementlarni boshqarish to‘plamini guruhlaydi
	TabControl	Guruhlangan ob‘yektlarni samarali tashkil etish va kiritish uchun sohali sahifani taqdim etadi.
	SplitContainer	Bir biridan ajratilgan ikki paneli beradi.
		Splitcontainer – Splitter o‘rniga mo‘ljallangan.
	TableLayoutPanel	Kontent dinamik satr va ustunlar iborat panelini ifodalaydi.
	FlowLayoutPanel	Dinamik ravishda kontentni gorizontal yoki vertikal joylashtiradigan panelni ifodalaydi.

Bu komponentalar Containers nomli tabda joylashgan. Ularning ko‘rinishi 1.9-rasmda keltirilgan.



1.9-rasm. Containers komponentalari.

Bu komponentlarining xususiyat va hodiralari alohida to‘xtalib o‘tamiz. Vazifalari eelementlari guruhini boshqarishdan iborat, shu maqsadda ishlatiladigan xususiyat va hodisalarini keltiramiz.

Panel komponentasi. Panel – bu oddiy to‘rt tomonidan chegaralangan hududga o‘xshaydi. Uni butun sohani egallashi uchun o‘ng tomonidagi [Panel Task] dagi [dock in parent container]ni

bosish yetarli. [undock in parent container] ni tanlab, oldingi holatiga kelish mumkin. Ob‘yektni harakatlantirish uchun chap tomondagi 4 ta tomonga yo‘naltirilgan tugmachadan foydalaniladi.

AutoSize - xususiyati true/false qiymatlarni qabul qiladi. Odatda chin qiymat ko‘proq ishlatiladi, chunki ob‘yektga joylashtirilgan boshqaruv elementlarininng o‘lchamlariga mos holda o‘lchamni sozlaydi. Joriy holatda yolg‘on qiymatga ega bo‘ladi.

Cursor - xususiyati sichkonchaning kursor ko‘rinishini o‘rnatish imkonini beradi.

BorderStyle – ob‘oektning atrof chegaralarining ko‘rinishlarini o‘rnatish uchun ishlatiladi.

AutoScoroll – ob‘yektda birlashtirilgan elementlar to‘liq ko‘rinmasa, harakatlaguvchi tugmalarni o‘rnatish va olib tashlash uchun ishlatiladi. Bu xususiyat true/false qiymatlarni qabul qiladi.

AutoScorollMargin – bu xususiyat harakatlanuvchi tugmachalar chet masofalarini o‘rnatida, Wight va Height ichki xususiyatlari bor.

AutoScorollMinSize - bu xususiyat harakatlanuvchi tugmachalar eng kichik o‘lchamlarini o‘rnatadi. Wight va Height ichki xususiyatlari bor.

Dock – ob‘yektni oynaning qaysi qismida joylashtirish kerakligini o‘rnatadi.

Margin – ob‘yekt chet qismlarida qoldirilishi kerak bo‘lgan masofalarni o‘rnatadi. Uning 5 ta xususiyati bor, barchasi, chap, o‘ng, yuqori va past.

groupBox komponentasi. groupBox - nomlagan elementlar guruhini birlashtirish uchun ishlatiladi. Ob‘yektni harakatlantirish uchun chap tomondagi 4 ta tomonga yo‘naltirilgan tugmachadan foydalaniladi.

Text - bu xususiyat elementlar guruhini nomini oynaga chiqaradi

AutoSize - xususiyati true/false qiymatlarni qabul qiladi. Elementlarning o‘lchamlariga mos ob‘yektga o‘zlcham ajratadi.

BackColor – ob‘yektning fon xususiyati rangini o‘rnatadi.

BackgroundImage - ob‘yektning fon xususiyati Image ob‘yektlarni o‘rnatadi.

Dock - ob‘yektni oynaning qaysi qismida joylashtirish kerakligini o‘rnatadi.

FlatStyle – ob‘yektning stil xususiyatini o‘rnatadi.

Font – bu xususiyat ob‘yektning yozuvlari uchun aniq bir standart yozuvni o‘rnatish uchun ishlatiladi.

MaximunSize – ob‘yektning eng katta o‘lchamini belgilaydi.

MinimumSize - ob‘yektning eng kichik o‘lchamini belgilaydi.

Padding – ob‘yektgacha bo‘lgan masofalarni o‘rnatish uchun ishlatiladi. Uning 5 ta xususiyati bor, barchasi, chap, o‘ng, yuqori va past.

RightToLeft – ob‘yektning matnini o‘ng tomonidan o‘rnatish.

TabIndex – ob‘yektning tab tugmasi bosilganda, nechinchi navbatda bo‘lishini belgilashni o‘rnatadi.

tabControl komponentasi. tabControl bu oddiy to‘rt tomonidan chegaralangan hududga o‘xshaydi. Uniga yangi sahifa qo‘shish yoki saxifani o‘chirish uchun o‘ng tomonidagi [tabControl Task] dagi [Add Tab] va [Remove Tab] ni bosish yetarli. Ob‘yektni harakatlantirish uchun chap tomondagi 4 ta tomonga yo‘naltirilgan tugmachadan foydalaniladi.

Anchor – ob‘yektni aftor yoki oyna bo‘yiyaa tartiblab joylashtirish xususiyatlarini tanlaydi. Uning to‘rtta qiymati o‘ng, chap, yuqori, past boryu Bularning o‘zaro aralashuvidan foydalaniladi.

Appearance – bu xususiyat ob‘yektning tugmachalarining ko‘rinishini o‘rnatish uchun ishlatiladi. Uning qiymatlari [Normal], [Buttons], [Flat Buttons] qiymatlarni qabul qiladi.

DrawMode – ob‘yektning xususiyatini chizilgan formasini o‘zgartrish uchun ishlvtiladi.

TabPages – bu xususiyat to‘plam qiymatlarni qabul qiladi va uning nechta sahifa bo‘lishini ta‘minlaydi. U bilan ishlaganda har bir sahifaning alohida xususiyatlariga ishlov berish mumkin.

Tabpage1 – bu ob‘yektning saxifa xususiyatlarni o‘rnatish uilatiladiggan ob‘yekt hisoblanadi va panel ob‘yektidek xususiyatlarga ega.

tabStop – ob‘yekt bo‘yicha tab tugmasining harakatlanishi o‘rnatish va o‘chirish qiymatini o‘rnatadi.

splitContainer komponentasi. splitContainer ikkiga ajratilgan elementlarni boshqarish uchun to‘plamni hosil qiladi. Uni butun sohani egallashi uchun o‘ng tomonidagi [splitContainer Task] dagi [dock in parent container]ni bosish yetarli. [undock in parent container] ni tanlab, oldingi holatiga kelish mumkin. Shuningdek, [Horizontal Splitter Orientation] – ob‘yekt ramkalarini gorizontal o‘rnatadi va [Vertical Splitter Orientation] esa vertikal o‘rnatadi. Ob‘yektni harakatlantirish uchun chap tomondagi 4 ta tomonga yo‘naltirilgan tugmachadan foydalaniladi.

Anchor - ob‘yektni aftor yoki oyna bo‘yiyaa tartiblab joylashtirish xususiyatlarini tanlaydi. Uning to‘rtta qiymati o‘ng, chap, yuqori, past boryu Bularning o‘zaro aralashuvidan foydalaniladi.

Backcolor - ob‘yektning fon xususiyati rangini o‘rnatadi.

BorderStyle - ob‘oektning atrof chegaralarining ko‘rinishlarini o‘rnatish uchun ishlatiladi.

FixedPanel – ob‘yekt uchun asosiy panelni o‘rnatish uchun, ya‘ni ikkita paneldan bir asosiy bo‘lishi kerak.

isSplitterFixed - ob‘yekt uchun asosiy panelni o‘rnatish amal qilishi yoki qilmasligini o‘rnatiadi. xususiyati true/false qiymatlarni qabul qiladi.

Orientation – ob‘yektning panellarini gorizontal yoki vertikal o‘rnatish uchun ishlatiladi. Ikkita Horizontal va Vertical qiymat qabul qiladi.

splitContainer1.Panel1 – ob‘yektning panel xususiyatlariga ishlov berish uchun panel xususiyatlarini ochib beradi va u uchun xususiyatlarni o‘rnatish mumkin.

Panel1Coolapsed – ob‘yektda panel1ni to‘liq qilib o‘rnatadi. Bunday xususiyat barcha panellari uchun o‘rinlidir.

Panel1MinSize – ob‘yektdagi panelning eng kichik o‘lchamini o‘rnatishdan iborat.

RightToLeft- ob‘yekt panellarini o‘ngdan o‘rnatish xususiyati.

tableLayoutPanel komponentasi. tableLayoutPanel – bu jadval kabi tartiblangan panellar to‘plami orqali elementlarni birlashtirish ob‘yektidir.

AutoScoroll – ob‘yektda birlashtirilgan elementlar to‘liq ko‘rinmasa, harakatlaguvchi tugmalarni o‘rnatish va olib tashlash uchun ishlatiladi. Bu xususiyat true/false qiymatlarni qabul qiladi.

AutoScorollMargin – bu xususiyat harakatlanuvchi tugmachalar chet masofalarini o‘rnatida, Wight va Height ichki xususiyatlari bor.

AutoScorollMinSize - bu xususiyat harakatlanuvchi tugmachalar eng kichik o‘lchamlarini o‘rnatadi. Wight va Height ichki xususiyatlari bor.

CellBorderStyle – ob‘yektning yacheykalarning chet chegaralarining stilini belgilaydi.

ColumnCount – ob‘yektdagi ustunlar sonini o‘rnatish.

Columns – ob‘yekt ustunlari uchun xususiyatlarni o‘rnatish uchun alohida muloqot oynasi ochiladi.

GrowStyle – ob‘yektga ustun yoki qator yoki ikkalasini qo‘shish xususiyatini o‘rnatish.

RowCount – qatorlar soni

Rows - ob‘yekt qatorlari uchun xususiyatlarni o‘rnatish uchun alohida muloqot oynasi ochiladi.

flowLayoutPanel komponentasi. flowLayoutPanel **–** ketma ketlik kabi tartiblangan panellar to‘plami orqali elementlarni birlashtirish ob‘yektidir.

Anchor - ob‘yektni aftor yoki oyna bo‘yiyaa tartiblab joylashtirish xususiyatlarini tanlaydi. Uning to‘rtta qiymati o‘ng, chap, yuqori, past boryu Bularning o‘zaro aralashuvidan foydalaniladi.

Backcolor - ob‘yektning fon xususiyati rangini o‘rnatadi.

AutoSize - xususiyati true/false qiymatlarni qabul qiladi. Elementlarning o‘lchamlariga mos ob‘yektga o‘zlcham ajratadi.

WrapContents – ob‘yektda sontentlarni ketma ketligini ustuvor qilish yoki qilmaslikni o‘rnatadi. Xususiyat true/false qiymatlarni qabul qiladi.

Bu qo‘shimacha komponentalarning xususiyatlari haqida ma‘lumotlarni berildi. Ularning hodisalari bir biriga o‘xshash bo‘lib, standart komponentalarningdan farqi juda kam. Shuning uchun hodisalarga to‘xtalaib o‘tirmaymiz. Shuningdek, qo‘shimacha komponentalar juda ko‘p, buni ko‘rish uchun Tollbox oynasiga sichqonchaning o‘ng tugmasini bosing va [Shown All] buyrug‘ini tanlang.

System komponentasining xususiyatlari va hodisalari. Bu komponentalarga asosan operatsion tizim bilan ishlaydigan komponentalar kiradi. Ularni xususiyatlari murakab hisoblanadi. Chunki birgina menyu yaratish uchun standart va qo‘shimcha komponentalar ham foydalaniladi. Shuning uchun umumiy holda bu komponentalarning vazifalarini keltirib o‘tamiz.

1.5-jadval. System komponentalarining guruhlari va vazifalari

Komponentaning guruhi	**Komponentaning nomi**	**Vazifasi va izoh**
Bo‘yruqlar	ToolStrip	Microsoft Windows, Microsoft Office, Microsoft Internet Explorer bo‘lishi mumkin uskunalar majmuasi yoki foydalanuvchi interfeysini yaratadi. Toolstrip control Asboblar paneli boshqaruvini almashtirishga mo‘ljallangan.
	ToolStripContainer	Toolstrip elementlarini ajralmas to‘plamini beradi (birikib, sohasida gorizontal yoki vertikal almashish)
Ma‘lumotni aks ettirish (faqat o‘qish elementlari uchun)	StatusStrip	Windowsning holat satrini beradi Statusstrip - Statusbar o‘rnini bosadi. Statusstrip maxsus xususiyatlari tartibini o‘z ichiga oladi,

Menyuni boshqarish elementlari	MenuStrip	Forma uchun menyu tizimini beradi. Menustrip - Mainmenu o‘rnini bosadigan yuqori darajali konteyner hisoblanadi. Bundan tashqari, asosiy ishlov berish va hujjat interfeysi imkoniyatlarini beradi. Funksionaligi Toolstripitem olingan bo‘lsa-da,Toolstripdropdownitem va Toolstripmenuitem bilan birga ishlashlatiladi.
	ContextMenuStrip	Kontekst menyuni ifodalaydi. Contextmenustrip bu Contextmenu o‘rnini bosadi. Contextmenustrip bilan birga ishlatiladi va sichqonchaning o‘ng tugmasini bosishda avtomatik ravishda kontekst menyusi namoyon bo‘ladi.

1.10-rasm. Menyu va uskunalar panellarini yaratish uchun komponentalarning ko‘rinishi.

Menyu va uskunalar panellarini yaratish uchun komponentalardan tashqari tizim bilan ishlaydigan komponentalarga quyidagi 1.6-jadvaldagi komponentalar ham qiradi, ular Components tab ga kiritilgan.

1.6-jadval. Components tab ning komponentalari.

Komponentaning nomi	Vazifasi va izoh	
BackgroundWorker	Alohida ajratilgan oqim uchun amal bajara oladigan BackgroundWorker komponentaning nusxasini yaratadi	

DirectoryEntry	Active Directory xizmati bilan o‘zaro bog‘lanib ishlashi va uning iearxiyasidan ob‘yekt yoki tugunlarni inkapsulyatsiya oladigan DirectoryEntry komponentaning nusxasini yaratadi	
DirectorySearcher	Active Directory da so‘rovlarni bajarish imkonini beradigan DirectorySearcher komponentaning nusxasini yaratadi	
ErrorProvider	Formada elementni boshqarishdagi xato foydalanuvchiga ko‘rsatish uchun ErrorProvider komponentaning nusxasini yaratadi	
EventLog	Tizim va foydalanuvchining hodisalar	
	jurnaliga bog‘lanish imkonini beradigan EventLog komponentaning nusxasini yaratadi	
FileSystemWatcher	Ruxsat berilgan papka va fayllarga o‘zgartirishlar kiritish imkonini beradigan FileSystemWatcher komponentaning nusxasini yaratadi	
HelpProvider	Tezkor va foydalanuvchining yordam oynasini chaqirish imkonini beruvchi HelpProvider komponentaning nusxasini yaratadi	
ImageList	Image ob‘yektlarining to‘plamini boshqarish usullarini ta‘minlovchi ImageList komponentaning nusxasini yaratadi	
MessageQueue	Navbatdagi xabarlarni o‘qish imkoniyatini beruvchi MessageQueue komponentaning nusxasini yaratadi	

PerformanceCounter	Windowsning hisoblagichlari bilan ishlash uchun foydalaniladi, yangi toifalar va misollar yaratish, hisoblagichlardan qiymatlarni o‘qish va hisoblagich ma‘lumotlariga asoslangan hisob-kitoblarni bajarish uchun PerformanceCounter komponentaning nusxasini yaratadi	
Process	Tizimdagi jarayonlar bilan bog‘liq ma‘lumotlarni to‘xtatish, ishga tushirish va o‘zgartirish uchun Process, komponentaning nusxasini yaratadi	
SerialPort	sinxron va voqea imkoniyatlar uchun SerialPort komponentaning nusxasini yaratadi	
ServiceController	Mavjud xizmatlarni, shu jumladan, boshlang‘ich va to‘xtatish xizmatlarini boshqarish va ularga buyruqlar berish uchun ServiceController komponentaning nusxasini yaratadi	
Timer	Windows ilovalar uchun vaqtga asoslangan vazifalarni kiritish uchun Timer komponentaning nusxasini yaratadi	

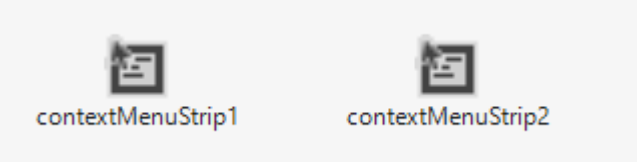
1.11-rasm. Components tab ga kiritilgan komponentalarning ko‘rinishi.

Bu komponentalardan kontkst menyuni yaratish va boshqa ob‘yektlarga o‘rnatishni ko‘rib chiqamiz.

Masala. Oynada ikkita kontkst menyu yaratish kerak. Birinchisi oyna uchun, ikkinchisi RichTextBox ob‘yekti uchun bo‘lsin. Birinchi kontekstda oyna yopish, kattalashtirish,

kichiklashtirish, yig‘ishtirish, tugmachaolarni yoqish va o‘chirish bo‘yruqlari bo‘lsin. Ikkinchi kontekstda matnni tahrirlash uchun buferga nusxalash, buferga nusxalash va o‘chirish, nusxa qo‘shish, ob‘yektni tozalash, gorozantal va vertikal harakatlanish tugmachalarni qurish va o‘chirish amallari bo‘lsin.

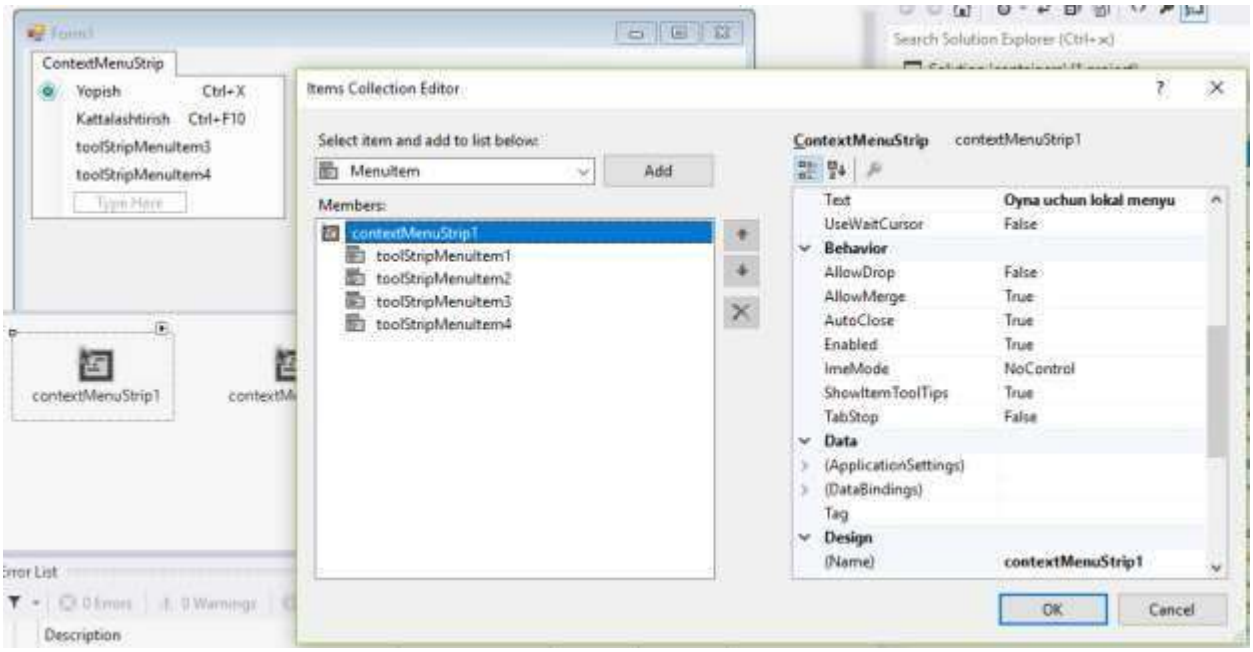
Yangi loyiha yaratamiz va bir oyna bo‘ladi. Oynaning bir chetiga RichtextBox ni o‘rnatamiz. Menyu va uskunalar tabidan contextMenuStrip ob‘yektidan ikkita o‘rnatamiz. Ular formada ko‘rinmasligi mumkin. Ular loyihaning ishchi maydoni pastki qismida joylashadi (1.12-rasmga qarang).



1.12-rasm. contextMenuStrip ob‘yekti o‘rnatilgan ko‘rinishi.

Bularni kerakli ob‘yektlari bilan bog‘laymiz. Oynaga birinchi kontekst menyuni [contextMenuStrip1], [RichTextBox1] ob‘yektga ikkinchi kontekst menyuni [contextMenuStrip2] o‘rnatamiz. Buning uchun oynaning [ContextMenuStrip] xususiyatiga [contextMenuStrip1] ni va RichTextBox1] ob‘yektning [ContextMenuStrip] xususiyatiga [contextMenuStrip2] ni o‘rnatamiz. Agar bitta kontekstni ikkita ob‘yektga o‘rnatilsa, kontekstning buyruqlarga joriy qilingan algoritmlarni bajaraveradi.

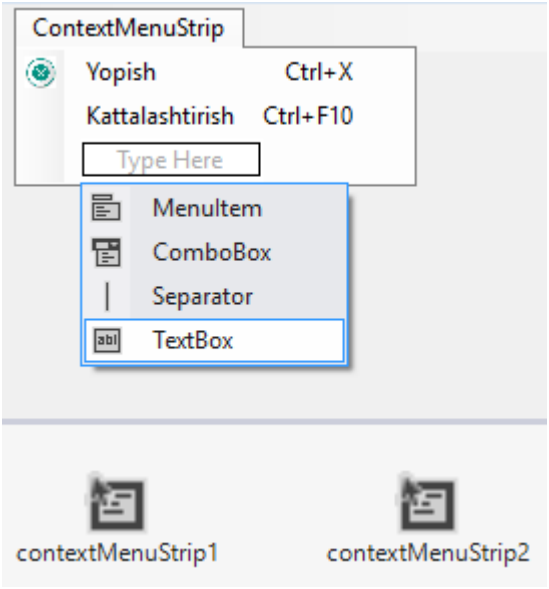
O‘rnatilgan kontekst menyu ob‘yekti [ContextMenuStrip] xususiyati uchun ichki xususiyatlarni, ya‘ni [contextMenuStrip] xususiyatlarni o‘rnatish uchun ochib beradi. Bu xususiyatlarni hammasiga to‘xtalib o‘tmaymiz. Faqat Items xususiyatiga to‘xtalamiz. Shu xususiyat tanlanganda quyidagicha moloqot oynasi chiqadi(1.13-rasmga qarang).



1.13-rasm. Kontekst menyu buyruqlarini o‘rnatish.

E‘tibor qaratsangiz, muloqot oynasida qilingan o‘rnatishlar muloqot oynasida emas balki asosiy oynada ko‘rinadi. Agar bu muloqot oynasida ishlash noqulay bo‘lsa, asosiy ishchi maydonda ham ishlash mumkin.

Buning kerakli kontekst menyu ob‘yekti uchtiga sichqonchani bir marta bosilsa, oynada kontekst menyu hosil bo‘ladi (1.14-rasmga qarang).



1.14-rasm. Oynada kontekst menyuga ishlov berish.

[Type Here] tugmasi orqali joriy [MenuItem], [ComboBox], [Separator], [TextBox] - 4 ta elementlarni qo‘shish mumkin. Qo‘shilgan har bir elementning o‘ng tugmasini bosib, u bilan ma‘lum bajarilishi mumkin bo‘lgan amallari bajarish mumkin. Shuningdek, [properties] degan xususiyatlarga olib kiradi. [Text] - xususiyatiga kirib buyruk nomini berish mumkin. Agar harf bilan murojjat qilishni xoxlasangiz, kerakli harf oldiga [&] belgisini qo‘yish kerak. Inteaktiv tugma o‘rnatish uchun esa [ShortcutKeys] xususiyatiga kerakli tugmalar kombinatsiyasini o‘rnatish mumkin. Ikonka o‘rnatish uchun [Image] xususiyatga kerakli rasmni o‘rnatish mumkin. Shunday qilib kerakli barcha buyruqlarni amalga oshirish mumkin. Buyruqlarni guruhlash uchun [Separator] komponentasini tanlash kerak. Buyruqlarga algortim yozishsh uchun ularni ustiga sichqonchani ikki marta bosish yetarli yoki hodisalariga kirib hodisasiga yozish mumkin.

Agar yuqorida kontekst menyuga ishlov berish tushunarli bo‘lgan bo‘lsa, masalani o‘zingiz tugatib qo‘ying. Agar ketma ketlik asosida bajarsangiz albatta masalani hech bo‘lmasa, algoritmlarsiz hal qilishingiz lozim.

NAZORAT SAVOLLARI

1. Toolbox oynasi 5 ta bo‘limdan iborat va nima uchun xizmat qiladi.
2. [List View] va [Show All] buyruqlari nima uchun kerak.
3. Toolbox interaktiv oynasi nechta komponenta muvjud?

4. Toolbox interaktiv oynasi komponentalarining o‘rnini almashtirish va qayta nomlash mumkinmi?

5. Komponenta xususiyatlarining asosiy nechta guruh bor.
6. Komponentalarning Focus guruhi nima uchun ishlatiladi.
7. Asosiy oynaga qo‘shishlgan komponentalar ro‘yxatini qaerdan ko‘rish mumkin.
8. Oynaning InitializeComponent funksiyasida ham komponentalarning xususiyatlari ko‘shilganini ko‘rish mumkinmi?
9. Komponentaning nechta usullari (methods) va hodisalari (events) guruhi bor?
10. Komponentaning Layout guruhi nima uchun ishlatiladi?
11. Standart komponentalarga Toolbox oynasidagi faysi Tab dagi komponentalar kiradi.
12. Mantlarni tahrirlash komponentalarini sanab bering?
13. Ro‘yxatdan tanlash komponentalarini sanab bering?
14. Komponentalarni oynaga joylashtirish uchun qanday amal bajarish kerak?
15. Label1 ob‘yektining text xususiyatini o‘rnatish qanday amalga oshiriladi?
16. comboBox1 ob‘yektining Items xususiyatlarini o‘rnatish qanday amalga oshiriladi?
17. [numericUpDown1->Value] ob‘yektning tanlangan qanday tipdagi qiymatini qaytaruvchi xususiyat.
18. [comboBox1->SelectedIndex] – tanlagichning tanlangan qanday qiymatini qaytaradi.
19. Additional komponentasining xususiyatlari va hodisalari nima uchun ishlatiladi.
20. TabControl komponentasining vazifasini ayting.
21. Panel komponentasi xususiyatlariga misollar keltiring.
22. Panel komponentasini oynaning qaysi qismida joylashtirish kerakligini o‘rnatadigan xususiyatni ayting.
23. Bu xususiyat to‘plam qiymatlarni qabul qiladi va uning nechta sahifa bo‘lishini ta‘minlaydi. U bilan ishlaganda har bir sahifaning alohida xususiyatlariga ishlov berish mumkin. Gap qaysi komponentaning qaysi xususiyati haqida ketmoqda.
24. System komponentasining xususiyatlari va hodisalarini sanab bering.
25. Menyuni boshqarish elementlari uchun ishlatiladigan komponentalarni sanab bering.

13- MA’RUZA

MULOQOT OYNALARI BILAN ISHLASH. (2 SOAT)

Reja:

1. Visual C++ muhitida muloqot oynalari.
2. Muloqot oynalarini sozlash.
3. Muloqot oynalari boshqarish elementlari.

4. Muloqot oynalarini yaratish.

Annotatsiya: *Tizimning foydalanuvchilar bilan o'zaro ta'sirini interaktiv ravishda amalga oshirish uchun dialog oynalari talab qilinadi. Muloqot oynalari 3 ta katta guruhga bo'lingan. Tizim oynalari mavjud, ya'ni ular operatsion tizim bilan ishlashga mo'ljallangan, interfaol xabarlarni uzatish va aniq javoblarni olish uchun mo'ljallangan, dasturchi yoki foydalanuvchi tashabbusi bilan yaratilgan. Visual C++muhitidagi dialog oynalari. Yuqorida aytib o'tilganidek, dialog oynalarini yaratish ularning maqsadi asosida amalga oshiriladi. Ushbu derazalarning o'ziga xos talablari bor, agar bu talablar bajarilsa, dialog oynalariga aylanishi mumkin.*

Kalit so‘zlar: *IDE, Visual C++, user interface, OYD, komponenta, xususiyatlari, hodisalari, Toolbox, Tab, muloqot oynasi, ColorDialog, FontDialog,* OpenFileDialog, PrintDialog, PrintPreviewDialog, FolderBrowserDialog, SaveFileDialog, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, HelpNavigator, Object, DialogResult, InvalidEnumArgumentException.

KIRISH

Foydalanuvchilar bilan tizimning muloqotini interaktiv amalga oshiri uchun muloqot oynalari kerak. Muloqot oynalari 3 ta katta guruhlarga bo‘linadi. Tizimli, ya‘ni OT bilan ishlashga mo‘ljallangan, interaktiv xabarlarni berish va aniq javoblarni olish uchun mo‘ljallangan, dasturchining yoki foydalanuvchining tashabbusi bilan yaratiladigan muloqot oynalari bor.

Visual C++ muhitida muloqot oynalari. Yuqorida aytib o‘tilgandek, muloqot oynalari yaratish ularni maqsadlaridan kelib chiqqan holda amalga oshiriladi. Bu oynalarning o‘zining talablari mavjudki, shu talablar bajarilsa, u muloqot oynasi bo‘la oladi. Bu talablarga quyidagilar kiradi:

1. Muloqot oynasining sarlavhasi bo‘lishi va unda faqat oynani yopish tugmasining bo‘lishi lozim, tizimli menyu va boshqa tugmalar bo‘lishi mumkin emas. Istesno tariqasida ba‘zi hollarda, yordam tugmasini joylashtiish mumkin.
2. Muloqot oynasi teskari aloqaga mo‘ljallanganligi uchun, uni shartini bajarmasdan tizimning boshqa oynasiga o‘tish mumkin emas.
3. Teskari aloqaning bir nesta turlarini amalga oshiruvchi tugmalar bo‘lishi kerak.
4. Muloqot oynalar asosiy oynadan har doim kichik bo‘lshi shart.
5. Muloqot oynaning asosiy maqsadi aniq kelitrilishi kerak, yoki turi, nima munosabat uchun muloqot oynasi chiqqanligi.
6. Muloqot oynasidan boshqa muloqot oynasiga o‘tish mumkin emas, asosiy oynaga o‘tish

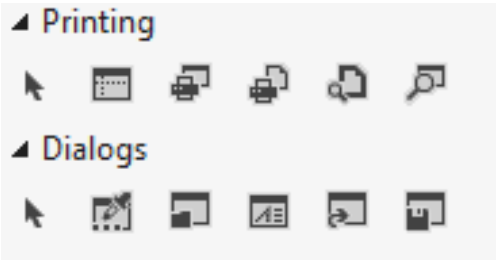
lozim.

7. Muloqot oyna murojaat qilinganda yaratilishi va teskari aloqa qabul qilingandan so‘ng xotiradan o‘chirib tashlanishi lozim.

8. Muloqot oynaga dinamik xotiralar bo‘lishi mumkin emas.

Ushbu talablarni bajargan har qanday oyna muloqot oynasi hisoblanadi.

Visual C++da OT bilan muloqot qilishga mo‘ljallangan muloqot oynalariga [Dialogs] tab dagi va [Printing] tab dagi barcha komponentalar kiradi (1.1- rasmga qarang).



1.1-rasm. Tizimli muloqot oynalariyaratish komponentalari.

Visual C++ning hujjatlariga qarasangiz tizimli muloqot oynalari uchun yagona muloqot oynalari ro‘yxati tuzilgan. Bu ro‘xatga quyidagi jadvalga keltirilgan komponentalar kiradi.

1.1-jadval.Visual C++da OT bilan muloqot qilishga mo‘ljallangan muloqot oynalari yaratishga mo‘ljallangan komponentalarning vazifalari

№	Komponenta nomi	Vazifasi
1	ColorDialog	Foydalanuvchilar interfeys elementi rangini o‘rnatish imkonini beruvchi ranglar palitrasi uchun muloqot oynasini ko‘rsatadi.
2	FontDialog	Foydalanuvchilarga kerakli komponenta uchun shrift va uning xususiyatlarini o‘rnatish imkonini beruvchi muloqot oynasini ko‘rsatadi.
3	OpenFileDialog	Foydalanuvchilar uchun faylni tanlash imkonini beradigan muloqot oynasini ko‘rsatadi.
4	PrintDialog	Foydalanuvchilarga printerni tanlash va uning xususiyatlarini o‘rnatish imkonini beruvchi muloqot oynasini ko‘rsatadi.
5	PrintPreviewDialog	Foydalanuvchilar uchun chop qilishda PrintDocument boshqaruv elementining ko‘rinishining ko‘rsatish imkonini beradigan muloqot oynasini ko‘rsatadi.
6	FolderBrowserDialog	Foydalanuvchilar uchun papkalar ko‘rish, yaratish va tanlash imkonini beradigan muloqot oynasini ko‘rsatadi.

7	SaveFileDialog	Foydalanuvchilar uchun faylni saqlash imkonini beradigan muloqot oynasini ko‘rsatadi.

Bu komponentalar maxsus xususiyatga asoslagan koponentalar bilash ishlatiladi.

Interaktiv xabarlarni berish va aniq javoblarni olish Visual C++ da MessageBox sinfi mavjud. Bu sinf bilan barcha ixtiyoriy turdagi muloqot oynalari yaratish mumkin. Sinfning nomlar fazosi System.Windows.Forms bo‘lib hisoblanadi va kutubxonasi System.Windows.Forms.dll hisoblanadi. Bu muloqot oynasi forma sinfining meros xo‘ri hisoblanadi. Unda 21 ta turli kombinatsiyali show funksiyachi bor. Uning quyidagi parametrlari bor.

1.2-jadval. MessageBox sinfnining show funksiyasi paramertlari

Paramert nomi	Tipi	vazifasi
text	String	Muloqot oynasining xabari
caption	String	Muloqot oynasining sarlavhasi
buttons	MessageBoxButtons	Teskari aloqani ta‘minalash tugmalari turlarini aniqlash
icon	MessageBoxIcon	Muloqot oynalarining ikonkalari turlarini aniqlash
defaultButton	MessageBoxDefaultButton	Teskari aloqani ta‘minalash uchun joriy tugmalari turlarini aniqlash
options	MessageBoxOptions	Muloqot oynalarining amallari turlarini aniqlash
helpFilePath	HelpNavigator	HelpNavigator ob‘yektining qiymatlari uchun foydalaniladi
param	Object	Yordam tugmasi bosilganda ID fiymatni aniqlash imkonini beruvchi parametr

Returns	DialogResult	Muloqot oynalarining qiymatlari qaytarish uchun foydalaniladigan DialogResult tipidagi turlarini aniqlash
Exceptions	InvalidEnumArgumentException	MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton ob‘yektlarini birini qabul qiluvchi kengaytirilgan ob‘yekt tipi
owner	IWin32Window	muloqot oynasini egasi Iwin32window amalga oshirish.

Bu paramertlarning o‘ziga mos qiymatlari oldindan aniqlab berilgan bo‘lib, oddiy sodda ko‘rinishda muloqot oynasini yaratish imkoni beradi. Keyinroq bu sinf paramertlari qiymatlari va ularga ishlov berishni ko‘rib chiqamiz.

Foydalanuvchi tomonidan yaratiladigan muloqot oynasi forma kabi yaratiladi va ularni loyhalash dasturchining yoki foydalanuvchining xoxishiga qarab amalga oshiriladi. Interaktiv muloqot oynalari kabi interaktiv tugmalarni yaratish va ularni boqarish, kerakli ma‘lumotlarni olish uchun ishlatiladi. Shuni ham inboatga olish keraki yaratiladigan muloqot oynasi talablarga mos kelishi kerak. BU talablarni amalga oshirish uchun formaning xususiyatlariga ishlov berish, lohim bo‘lsa, asosiy oynda kerakli xususiyatlarni o‘rnatish mumkin. Bunda ham formaning show usuli mavjud bo‘lib. Shu orqali forma chaqiriladi.

Sinfning nomlar fazosi System.Windows bo‘lib hisoblanadi va kutubxonasi PresentationFramework.dll hisoblanadi. Bu muloqot oynasi forma sinfining meros xo‘ri hisoblanadi.

.NET 5 Preview 1, .NET Core 3.1, 3.0 va .NET Framework 4.8 4.7.2 4.7.1 4.7 4.6.2 4.6.1 4.6 4.5.2 4.5.1 4.5 4.0 3.5 3.0 variantlarida qo‘llab quvvatlanadi. Foydalanuvchi muloqot oynalarini yartaishni keyinroq batavsil ko‘rib chiqamiz.

Muloqot oynalarini sozlash. Bu mulovot oynalarini sozlash uchun tizimli muloqot oynalaridan foydalanish va ularga ishlov berish nazarda tutilgan. Yuqorida keltirilgan 7 ta muloqot oynalarinidan foydalanishlar, xususiyatlarini va hodisalarini boshqarish to‘g‘risida to‘xtalamiz.

1. **ColorDialog muloqot oynasi**. Bu oyna - foydalanuvchilar interfeys elementi rangini o‘rnatish imkonini beruvchi ranglar palitrasi uchun muloqot oynasini ko‘rsatadi. Bu

komponentdani formaga o‘rnatilganda hech qanday ko‘rinish hosil bo‘lmaydi, ammo formaning ichki tuzilmasiga qo‘yshiladi. Fomraning ishchi holatidagi formasining pastki qismida uning ob‘yekti yaratiladi.

solorDialog1 ob‘yektini yaratish orqali boshqariladi. Uning xususiyatlari va hodisalari ham mavjud va loyiha oynasida foydalanuvchi xususiyatlar oynasiga chiqadi. U yerdan kerakli ixtiyoriy aynan shu ob‘yektga mos xususiyat va hodisalarni o‘rnatish mumkin.

ColorDialog muloqot oynasidan foydalanish uchun ColorDialog() konstruktori ishga tushirish lozim.

Bu sinfning xususiyailari, usullari va hodisalari mavjud.

1.3-jadval. ColorDialog muloqot oynasining xususiyatlari

AllowFullOpen	Maxsus ranglar aniqlash uchun muloqot oynasini foydalanish mumkin yoki yo‘qligini o‘rnatish
AnyColor	Muloqot oynasida asosiy ranglar majmuining barcha mavjud ranglar ko‘rsatish yoki yo‘qligini o‘rnatish
CanRaiseEvents	Komponentaga bir hodisa o‘rnatish mumkinligini aniqlash.
Color	foydalanuvchi tomonidan tanlangan rangni o‘rnatish.
Container	Komponentini o‘z ichiga olgan Icontaineni o‘rnatish
CustomColors	Muloqot oynasida ko‘rsatilgan maxsus ranglar to‘plamini oladi yoki o‘rnatadi.
DesignMode	Komponenta joriy dizayn rejimida ekanligini ko‘rsatadigan qiymatni oladi.
Events	komponentaga ilova qilinadigan hodisalar ro‘yxatini oladi.
FullOpen	Muloqot oynasi ochilganda maxsus ranglarni yaratish uchun ishlatiladigan boshqaruv elementlari ko‘rinib turishini ko‘rsatuvchi qiymatni oladi yoki o‘rnatadi.
Options	Colordialog boshlash uchun xususiyatlarni oladi.
ShowHelp	Yordam tugmasi rang muloqot oynasidagi paydo yoki

	yo‘qligini bo‘lishini o‘rnatish
Site	Komponentning ISiteni oladi yoki o‘rnatadi.
SolidColorOnly	Muloqot oynasi qattiq ranglar tanlash uchun foydalanuvchilar cheklash yoki yo‘qligini o‘rnatish
Tag	nazorat haqida ma‘lumotlarni o‘z ichiga olgan ob‘yekt sozlash.

1.4-jadval. ColorDialog muloqot oynasining usulari

CreateObjRef(Type)	Ob‘yekt bilan muloqot qilish uchun ishlatiladigan proksi ishlab chiqarish uchun zarur bo‘lgan barcha tegishli ma‘lumotlarni o‘z ichiga olgan ob‘yekt yaratadi.
Dispose()	Komponent tomonidan ishlatiladigan barcha resurslarni chiqaradi.
Dispose(Boolean)	Tarkibiy qism tomonidan ishlatiladigan boshqarilmaydigan resurslarni chiqaradi va ixtiyoriy ravishda boshqariladigan resurslarni chiqaradi.
Equals(Object)	Belgilangan ob‘yekt joriy ob‘yektga teng yoki yo‘qligini aniqlaydi.
GetHashCode()	Standart hesh funksiyasi sifatida xizmat qiladi.
GetService(Type)	Komponenta yoki uning konteyneri tomonidan taqdim etilgan xizmatni ifodalovchi ob‘yektni qaytaradi.
GetType()	Joriy ob‘yekt turini oladi.
HookProc(IntPtr, Int32, IntPtr, IntPtr)	Umumiy muloqot oynasiga xos funksiyalarni kiritish uchun bekor qilingan umumiy muloqot oynasi protsedurani belgilaydi.
MemberwiseClone()	Joriy ob‘yektning oddiy nusxasini yaratadi.
MemberwiseClone(Boolean)	Joriy Marshalbyrefobject ob‘yekt oddiy nusxasini yaratadi.
OwnerWndProc(IntPtr, Int32, IntPtr, IntPtr)	Umumiy muloqot oynasiga maxsus funksiyalarni qo‘shish uchun bekor qilinadigan oyna tartibini belgilaydi.

Reset()	Ularning standart xususiyatlarga barcha imkoniyatlari ishga soladi, oxirgi tanlangan qora rang va ularning standart xususiyatlarga maxsus ranglar moslashtiriladi
RunDialog(IntPtr)	Umumiy muloqot oynasini belgilaydi.
ShowDialog()	Umumiy muloqot oynasini ishlatadi.
ShowDialog(IWin32Window)	Umumiy muloqot oynasini ishlatadi.
ToString()	Muloqot oyna qiymatini satrga o‘tkazish.

1.5-jadval. ColorDialog muloqot oynasining hodisalari

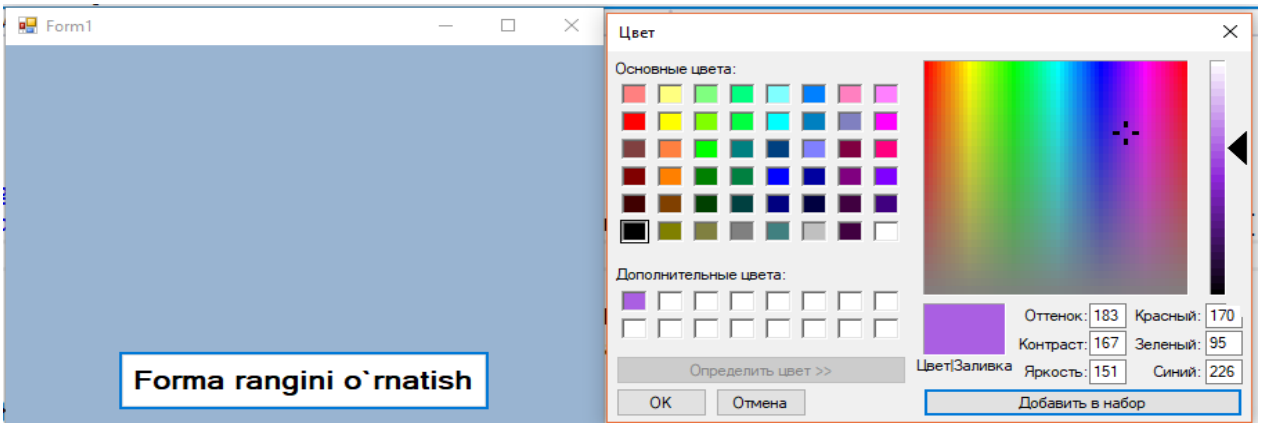
Disposed	Komponentni Dispose() metodiga chaqiriq orqali dispozitsiya qilinganda yuzaga keladi
HelpRequest	Foydalanuvchi umumiy muloqot oynasidagi yordam tugmasini bosganda sodir bo‘ladi.

ColorDialog muloqot oynasidan foydalanish uchun formaga bir tugma o‘rnatamiz va uning Click hodisasi yordamida chaqiramiz. Tanlangan rang esa formaning va tugmaning fonini o‘zgartirsin.

Tugmaning Click hodisasi quyidagicha dastur franmentini o‘rnatamiz.

```
if(colorDialog1->ShowDialog() == ::System::Windows::Forms::DialogResult::OK)
        Form1::BackColor = colorDialog1->Color;
        button1->BackColor = colorDialog1->Color;
```

Dastur fragmentida muloqot oynachi chaqirilganda va teskari aloqasi ok ob‘yektini qaytarsa forma va tugmaning mos xususiyatlar ranglarini o‘zgartirish algoritmi yozilgan.



1.2-rasm. ColorDialog muloqot oynasidan foydalanish.

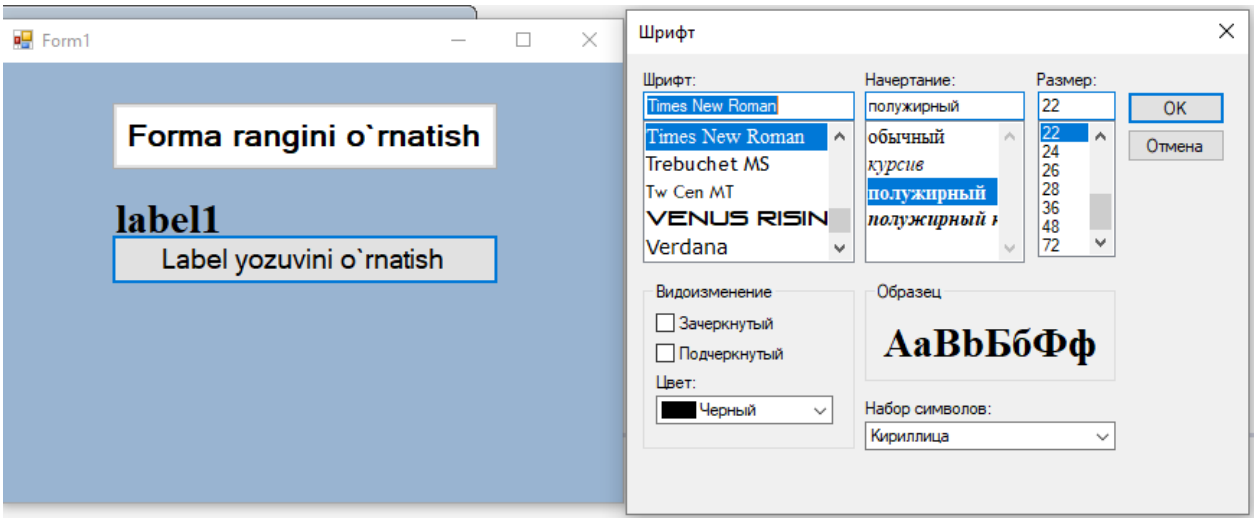
ColorDialog muloqot oynasining xususiyatlari, usullari va hodipsalarini masalaning ahamiyatiga qarab ishlatish mumkin.

2. FontDialog muloqot oynasi. Bu oyna – foydalanuvchilarga kerakli komponenta uchun shrift va uning xususiyatlarini o‘rnatish imkonini beruvchi muloqot oynasini ko‘rsatadi. Bu komponentdani formaga o‘rnatilganda hech qanday ko‘rinish hosil bo‘lmaydi, ammo formaning ichki tuzilmasiga qo‘yshiladi. Fomraning ishchi holatidagi formasining pastki qismida uning ob‘yekti yaratiladi.

Bu muloqot oynasini ishlatish uchun formaga bir Label va button ob‘yektlarini o‘rnatamiz. Tugma bosilganda Label ob‘yektining matnini yozuv xususiyatlarini o‘rnatishni ko‘rib chiqamiz. Buning uchun tunmaning bosilganda xususiyatiga quyidagi dastur fragmentni yozish yetarli.

```
fontDialog1->ShowColor = true;
fontDialog1->Font = label1->Font;
fontDialog1->Color = label1 ->ForeColor;
Color color = label1->ForeColor;
System::Drawing::Font^ font = label1->Font;
System::Windows::Forms::DialogResult result = fontDialog1->ShowDialog();
if(result == ::System::Windows::Forms::DialogResult::OK) {
        label1->Font = fontDialog1->Font;
        label1->ForeColor = fontDialog1->Color;
        }
```

Dastuda dastlab Label ob‘yektning rang va yozuvlarini saqlab olinadi, chunki yozuvni formatlash muloqot oynasi chaqirilganda joriy holatni olish uchun. Dasturda rang va yozuv qiymatlarini saqlash uchun o‘zgaruvchilarni aniqlash olib ham keltirilgan. Muloqot oynasining [OK] hodisasi bajarilganda yozuv va rangni o‘zgartirish ko‘rsatilgan.



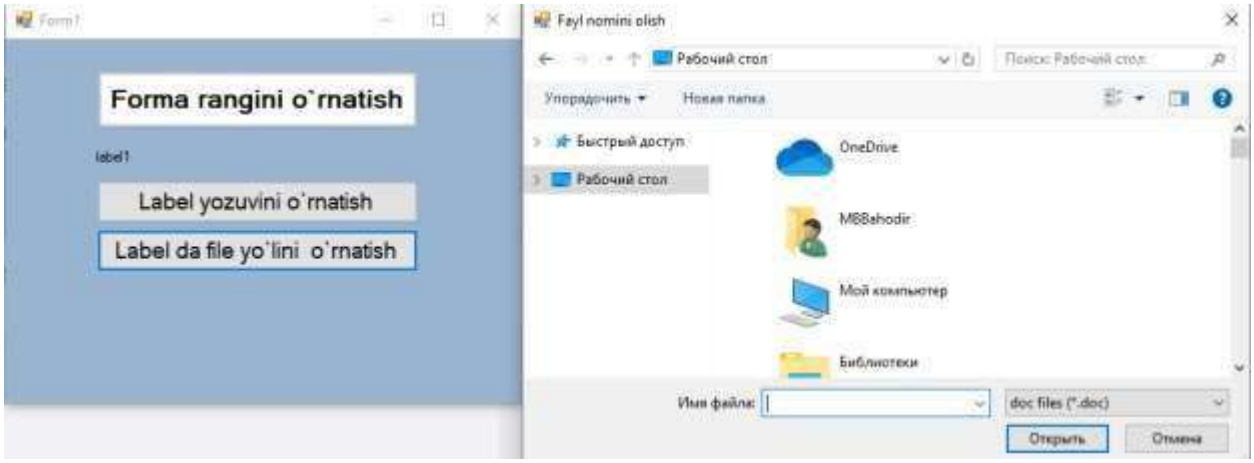
1.3-rasm. FontDialog muloqot oynasidan foydalanish.

Bu sinfning xususiyailari, usullari va hodisalari mavjud. Ular amaliy vazifalarni bajarishda foydalanish mumkin va mustaqil o‘rganish lozim. Chunki bir vazifani amalga oshirish uchun turli xil algoritmlardan foydalanish mumkin.

3. OpenFileDialog muloqot oynasi. Bu oyna - foydalanuvchilar uchun faylni tanlash imkonini beradigan muloqot oynasini ko‘rsatadi. Bu muloqot oynasi ham yuqoridagidek foydalaniladi. Shuningdek, sinfning mos xususiyatlari, usullari va hodisalari mavjud. Farqli xususiyatlarni kuo‘rsatish uchun bir misol olamiz. Unda tugma bosilganda label matniga faylning to‘liq yo‘lini olish va o‘rnatish uchun quyidagi dastur fragmentini yoziladi.

```
openFileDialog1->InitialDirectory = "c:\\";
openFileDialog1->Filter = "txt files (*.txt)|*.txt|doc files (*.doc)|*.doc,*.docx|All files
(*.*)|*.*";
openFileDialog1->FilterIndex = 2;
openFileDialog1->RestoreDirectory = true;
openFileDialog1->Title = "Fayl nomini olish";
if ( openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK ){
 if (openFileDialog1->OpenFile() != nullptr ){
        label1->Text = openFileDialog1->FileName;
        }
 }
```

Dasturning birinchi satri joriy katalog o‘rnatiladi. Shuningdek, joriy foydalanuvchining kerakli kataloglarini ham o‘rnatish qiymatlari bor. So‘ng, fayllarni tipi bo‘yicha filtrlash o‘rantiladi. Hozirda 3 ta format o‘rnatilgan. Keyingi satrda fayl kengaytmalaridan qaysi biri joriy bo‘lib chiqishini belgilash amalga oshirilgan. Ko‘rsatilgan katalogni faollashtirish bajarilgan va muloqot oynasining sarlavhasida kerakli matn joylashtirilgan. Muloqot oynaning teskari aloqasi tekshirilgan va shu asosida amal bajarilgan.



1.4-rasm. FontDialog muloqot oynasidan foydalanish.

Odatda bu muloqot oynasi ma‘lum tizimsha fayllarni matnini joylashtirish uchun ishlatiladi. Buni qanday amalga oshirish mumkin. Buning uchun kichik bo‘lsa ham matn

muharriri yaratish lozim va unga OTdan oddiy manli fayllarni yuklab olish mumkin. Fayl ichidagi ma‘lumotlarni olish uchun quyidagicha dastur fragmenti yozish mumkin:

```
openFileDialog1->Filter = "text files (*.txt)|*.txt|cpp files (*.cpp)|*.cpp|All files (*.*)|*.*";
openFileDialog1->FilterIndex = 2;
openFileDialog1->RestoreDirectory = true;
openFileDialog1->Title = "Fayl nomini olish";
openFileDialog1->FileName = "";
if ( openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK){
        if ( (myStream = openFileDialog1->OpenFile()) != nullptr ){
        System::IO::StreamReader ^ sr = gcnew System::IO::StreamReader(openFileDialog1-
>FileName);
        richTextBox1->Text = sr-
          >ReadToEnd(); sr->Close();
        }
}
```

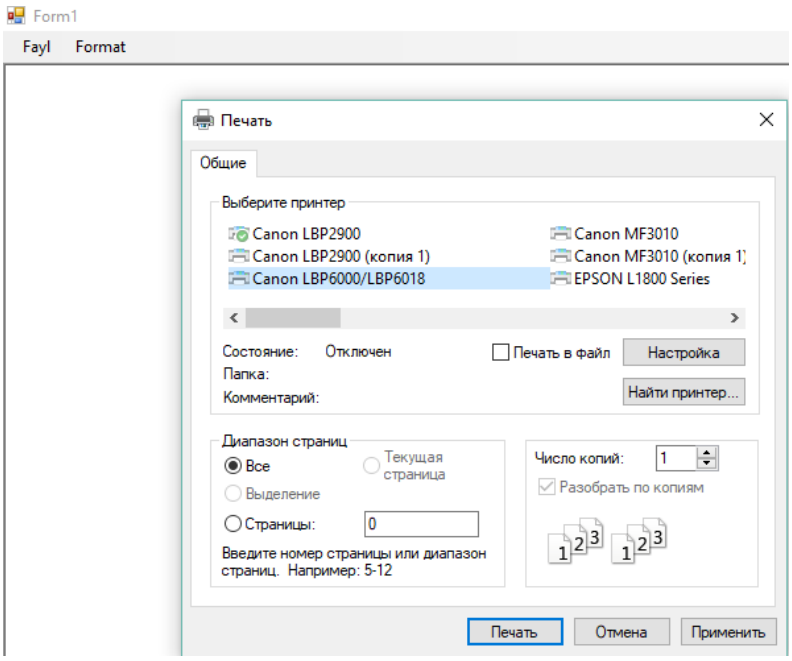
Bu amaliy dastur fragmentini amaliyotga sinab ko‘rish orqali tahlil qiling.

4. PrintDialog muloqot oynasi. Bu oyna - foydalanuvchilarga printerni tanlash va uning xususiyatlarini o‘rnatish imkonini beruvchi muloqot oynasini ko‘rsatadi. Oynaning juda ko‘p xususiyatlari, usullari va hodisalari bor. Bularni mustaqil ishlarni bajarish vaqtida amalga oshirish mumkin. Shuningdek, bu muloqot oynani ishlatish usullari hamjuda ko‘p. Odatda dasturchining loyiha holatidan kelib chiqqan holda foydalaniladi.

```
        printDialog1->AllowSomePages =
        true; printDialog1->ShowHelp = true;
    if ( printDialog1 == nullptr )
    System::Windows::Forms::MessageBox::Show("pnull" );
    System::Windows::Forms::DialogResult result = printDialog1->ShowDialog();
    System::Windows::Forms::MessageBox::Show(result.ToString());
    if ( result ==  System::Windows::Forms::DialogResult::OK )
    {
// docToPrint->Print(); System::Windows::Forms::MessageBox::Show("Chop qilish
boshlanadi");
    }
```

Dastur fragmentida ma‘lumotlarni chop qilish uchun avval uni ma‘lum bir chiquvchi oqimga yozish va oqimni esa, hujjat formatiga joylashtirish kerak. Hujjatni esa, sahifalarni sozlab docToPrint ob‘yekt yaratilishi kerak. Ammo chop qillishning turli parametrlarini ishlatish uchun

muloqot oynasini chiqarib beradi. Kerakli xususiyatlarni o‘rnatgandan so‘ng amalni bajarish mumkin.



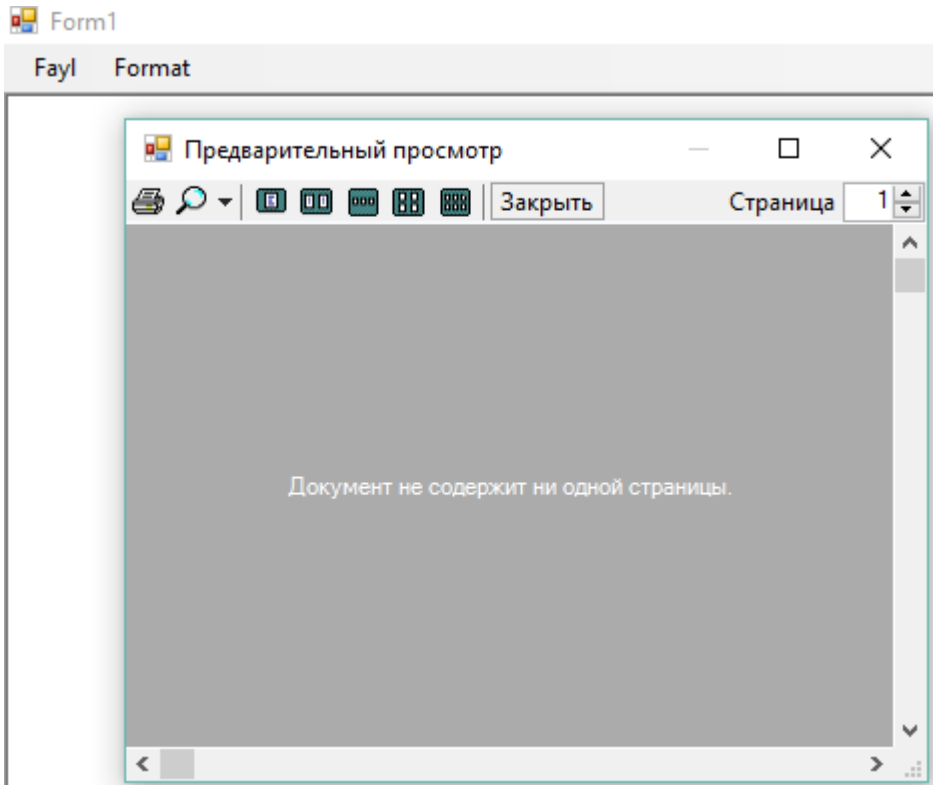
1.5-rasm. PrintDialog muloqot oynasidan foydalanish.

Dastur fragmentini bir loyiha joylashtirib, ishlatsangiz rasmdagidek ishlashi kerak.

5. **PrintPreviewDialog muloqot oynasi.** Bu oyna - foydalanuvchilar uchun chop qilishda PrintDocument boshqaruv elementining ko‘rinishining ko‘rsatish imkonini beradigan muloqot oynasini ko‘rsatadi.

```
printPreviewDialog1->MinimumSize = System::Drawing::Size( 375, 250
        ); printPreviewDialog1->UseAntiAlias = true;
                printPreviewDialog1->Document = document;
                        printPreviewDialog1->ShowDialog();
```

Bu muloqot oynasi ham yuqoridagi muloqot oynasi kabi sozlashlarni bajarshgandan so‘ng foydalanish mumkin. Dastur fragmentida qarasangiz document ob‘yektini yaratish lozim. Chop qilishninig ob‘yektidan farq qilgan xolda oqimdagi ma‘lumotni formatlash va uni chiqishini A4 shaklga keltrish orqali amlga oshiriladi.



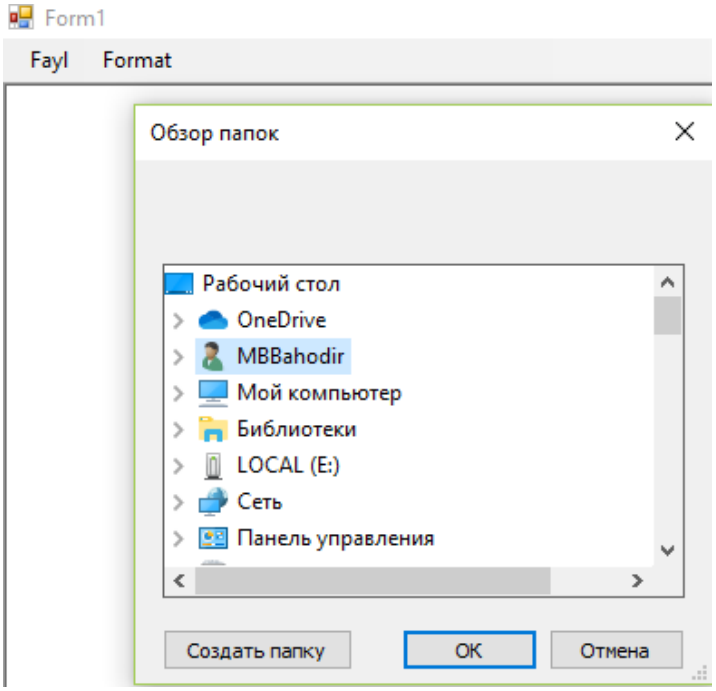
1.6-rasm. PrintPreviewDialog muloqot oynasidan foydalanish.

6. FolderBrowserDialog muloqot oynasi. Bu oyna - foydalanuvchilar uchun papkalar ko‘rish, yaratish va tanlash imkonini beradigan muloqot oynasini ko‘rsatadi. Bundan ma‘lumotlarni ko‘chirishda, papkalarni taqqsolashda ishlatish mumkin. Muloqot oynasining xususiyatlari, usullari va hodisalar mavjud. Ularning ba‘zilarini quyidagi dastur fragmentiga keltirib o‘tamiz.

```
System::IO::Stream ^ myStream;
System::Windows::Forms::DialogResult result = folderBrowserDialog1->ShowDialog();
 if ( result == System::Windows::Forms::DialogResult::OK ) {
        System::String^ folderName = folderBrowserDialog1->SelectedPath;
        openFileDialog1->InitialDirectory = folderName;
        openFileDialog1->FileName = String::Concat(folderName,"\\1.cpp");
         if ( (myStream = openFileDialog1->OpenFile()) != nullptr ){
         System::IO::StreamReader ^ sr = gcnew System::IO::StreamReader(openFileDialog1-
>FileName);
         richTextBox1->Text = sr->ReadToEnd(); sr->Close();
         }
}
```

Dastur fragmentida papka uchun muloqot oynasi chaqirilgan va ko‘rsatilgan papkadan 1.cpp faylini yuklab kelgan. Bunday holat fayl menejerlar uchun papkadagi fayllarning ro‘yhatini ham olish mumkin.

Dastur fragmentida bir StreamReader oqim yaratilgan. Oqim faylning ma‘lumotlarini o‘qish uchun yaratilgan. Oqimning konstruktori asosida sr oqim ob‘yekti yaratiladi. Bu oqimning ReadToEnd funksiyasi orqali richTextBox ga ma‘lumotlar joylashtiriladi. FolderBrowserDialog muloqot oynasining ko‘rinishi quyidagicha:



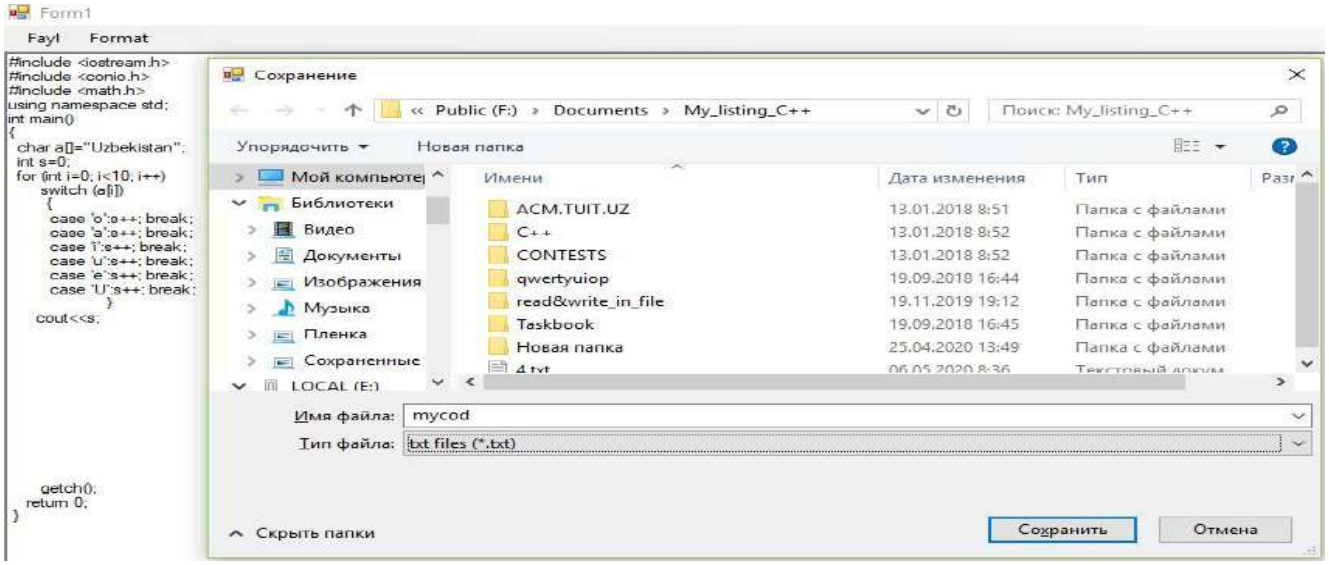
1.7-rasm. FolderBrowserDialog muloqot oynasidan foydalanish

7. **SaveFileDialog muloqot oynasi.** Bu oyna - foydalanuvchilar uchun faylni saqlash imkonini beradigan muloqot oynasini ko‘rsatadi. Bu ham asosan matnli va maxsus tuzilmalari ma‘lumotlarini saqlash uchun ishlatiladi. Ma‘lumot qanaqa tuzilmada yozilsa, shunday tuzilmada o‘qiladi.

```
System::IO::Stream^ myStream;
            MemoryStream^ userInput = gcnew MemoryStream();


            richTextBox1->SaveFile( userInput, RichTextBoxStreamType::PlainText );
        userInput->WriteByte( 32 );


        // saveFileDialog1->CreatePrompt = true;
        // saveFileDialog1->OverwritePrompt = true;
        // saveFileDialog1->FileName = "myText";
        // saveFileDialog1->DefaultExt = "txt";
```

```
        saveFileDialog1->Filter = "txt files (*.txt)|*.txt";
        saveFileDialog1->FilterIndex = 2;
        saveFileDialog1->RestoreDirectory = true;
if ( saveFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK ){
                    myStream = saveFileDialog1->OpenFile();
                    userInput->Position = 0;
                    userInput->WriteTo( myStream );
              myStream->Close();


            }
```

Dastur fragmentida ikki xil oqim yaratilgan. myStream - birinchisi an‘anaviy oqim va MemoryStream -xotirali oqim yaratilgan. Biri ma‘lumotlarni olish va ikkinchisi ma‘lumotlarni xotiraga yozish, berilgan nom bilan nomlashgan qaratilgan. Position bu poizitsiyaani boshlash, ammo olib tashlasa ham ishlaydi.Bu muloqot oynasi nafaqat matnli ma‘lumotlarni balki, foydlanuvchining xoxlagan ma‘lumotni matn yoki binar kodlash orqali saqlash mumkin. Muloqot oynasining ko‘rinishi quyidagi rasmda keltirilgan.



1.8-rasm. **SaveFileDialog** muloqot oynasidan foydalanish

Ko‘rib chiqilgan muloqot oynalari barchasi OT bilan ishlashga mo‘ljallanganligi ko‘rinib, turibdi. OT qanday sozlangan bo‘lsa, bu muloqot oynalari ham shu rejimda ishlaydi. Oynalarga

deyarli o‘zgartirish kiritish shart emas, xuddiki barchasi kutilgandek yaratilganga o‘xshaydi. Faqat eng katta muammosi lokalizatsiya qilish.

Muloqot oynalari boshqarish elementlari. Bunda interaktiv muloqt qilish oynalari tushiniladi. Ularni yaratish va boshqarish dasturchining xoxishiga qarab amalga oshiriladi. Yuqorida aytib o‘tganimizdek, Visual C++ da MessageBox sinfi haqida yuqoridaga aytib o‘tgan edik. Unda 21 ta turli kombinatsiyali show funksiyachi bordigini ham. Shuning bu funksiyalar va ularning parametrlari hamda parametrlarining qiymatlari to‘g‘risi ma‘lumotlarni va muloqot oynalarini yaratishga va ishlov berishga e‘tiborni qaratamiz.

1. **Show(String) fknutsiyasi**. Bir argumentli funksiya bo‘lib, belgilangan matn bilan xabar ko‘rsatadigan muloqot oynasini yaratish uchun ishlatiladi. Uning kiruchi parametri System::String tipida bo‘lib, System::Windows::Forms::DialogResult tipidagi qiymat qaytaradi.

```
if( MessageBox::Show("Bu oddiy sohow") == System::Windows::Forms::DialogResult::OK)
                                    this->Close();
```

2. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, String) funksiyasi. Bu ko‘p paramerli bo‘lib, belgilangan paoametlar asosida interaktiv muloqot oynasini yaratish uchun ishlatiladi. Parametlari quyidagi qiymatlari qabal qiladi.

IWin32Window Interface - System.Windows.Forms nomlar fazosi va System.Windows.Forms.dll kutubxonasidan foydalanib, Win32 HWND ni joriy qilish interfeysi beradi. IWin32Window sinf interfeysidan merosxo‘r oladi. System.Windows.Forms.Control va System.Windows.Forms.NativeWindow umumiy ruxsat sinflari interfeyslarini ishlatadi, qiymatlar sifatida ComVisibleAttribute, GuidAttribute, InterfaceTypeAttribute tiplarini ishlatadi.

MessageBoxButtons - Enum tipidagi parametrdir. System.Windows.Forms nomlar fazosi va System.Windows.Forms.dll kutubxonasidan foydalanadi. Muloqot oynalarida ko‘rsatilishi kerak bo‘lgan tugmalarni aniqlaydi. Qiymatlari public enum class MessageBoxButtons ta‘luqlidir.

1.6-jadval.MessageBoxButtons qabul qiluvchi qiymatlar

№	Qiymat nomi	vazifasi
1	AbortRetryIgnore	Muloqot oynasida Abort, Retry Ignore tugmalarini o‘rnatish
2	OK	Muloqot oynasida OK tugmasini o‘rnatish
3	OKCancel	Muloqot oynasida OK va Cancel tugmalarini o‘rnatish
4	RetryCancel	Muloqot oynasida Retry va Cancel tugmalarini o‘rnatish

5	YesNo	Muloqot oynasida Yes va No tugmalarini o‘rnatish
6	YesNoCancel	Muloqot oynasida Yes, No va Cancel tugmalarini o‘rnatish

Muloqot oynasiga MessageBoxButtons tugmalarini o‘rnatish dastur fragmenti:

```
if ((MessageBox::Show("Joriy oynani yopishni xoxlaysizmi?",
                      "Xabar", MessageBoxButtons::YesNo) ==
System::Windows::Forms::DialogResult::Yes)){
                                    this->Close();
```

MessageBoxIcon – Enum tipidagi ob‘yekt bo‘lib, u ham yuqoridagidek, nomlar fazosi va kutubxona bilan ishlaydi. Muloqot oynasida ko‘rsatish uchun muloqot oynalarining turlarini belgilovchi konstantalarni aniqlaydi.

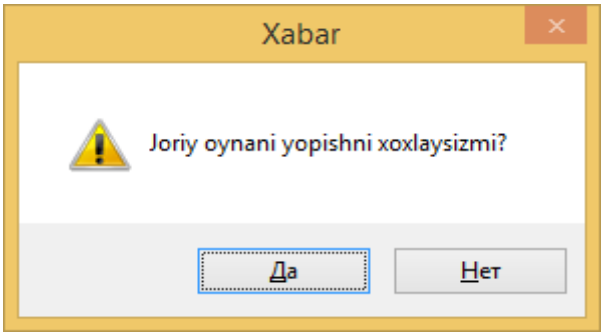
1.7-jadval. MessageBoxIcon qabul qiluvchi qiymatlar

№	Qiymat nomi	belgisi	vazifasi
1	Asterisk		Qandaydir hodisa haqida faqat xabar beruvchi muloqot oynasi uchun ikonka
2	Error		Qandaydir hodisa haqida faqat xato amal bajarilganligi haqida ma‘lumot beruvchi muloqot oynasi uchun ikonka
3	Exclamation		Qandaydir hodisa haqida faqat ogohlantirish haqida ma‘lumot beruvchi muloqot oynasi uchun ikonka.
4	Hand		Qandaydir hodisa haqida faqat xato amal bajarilganligi haqida ma‘lumot beruvchi muloqot oynasi uchun ikonka
5	Information		Qandaydir hodisa haqida faqat xabar beruvchi muloqot oynasi uchun ikonka
6	None		Ikonkasiz muloqot oynasi uchun
7	Question		Qandaydir hodisa haqida faqat savolga javob oluvchi muloqot oynasi uchun ikonka
8	Stop		Qandaydir hodisa haqida faqat xato amal bajarilganligi haqida ma‘lumot beruvchi muloqot oynasi uchun ikonka
9	Warning		Qandaydir hodisa haqida faqat ogohlantirish haqida ma‘lumot beruvchi muloqot oynasi uchun ikonka.

MessageBoxIcon ni ishlatish uchun joriy formani yopish uchun muloqot oynani ishlatish uchun dastur fragmentini keltiramiz. Buning uchun formaning FormClosing hodisasiga quyidagi dastur fragmentini yozamiz:

```
System::Void Form1_FormClosing(System::Object^ sender,
System::Windows::Forms::FormClosingEventArgs^ e) {

                        if ((MessageBox::Show(
                                "Joriy oynani yopishni xoxlaysizmi?",
                                "Xabar", MessageBoxButtons::YesNo,

MessageBoxIcon::Exclamation) == System::Windows::Forms::DialogResult::No)){
                                        e->Cancel = true;
                        }
```

Dastur fragmentida hodisaning to‘liq yozilishi keltirilgganligining sababi unda hodisaga ishlrv berilgan.



1.9-rasm. MessageBoxIcon bilan muloqot oynasini yaratish.

Mazkur oynada yo‘q tugmasi bosilsa, forma yopilmaydi va aksincha ha tugmasi bosilsa, forma yopiladi.

MessageBoxDefaultButton - Enum tipidagi ob‘yekt bo‘lib, u ham yuqoridagidek, nomlar fazosi va kutubxona bilan ishlaydi. Muloqot oynasida standart tugmalarni o‘rnatuvchi o‘zgarmaslarni belgilaydi.

1.7-jadval. MessageBoxDefaultButton qabul qiluvchi qiymatlar

№	Qiymat nomi	vazifasi
1	Button1	Muloqot oynasida starndart birinchi tugmani joriy tugma sifatida o‘rnatish
2	Button2	Muloqot oynasida starndart ikkinchi tugmani joriy tugma sifatida o‘rnatish

3	Button3	Muloqot oynasida starndart uchinchi tugmani joriy tugma sifatida o‘rnatish

Tugma bosilganda formani yopishni so‘rash muloqot oynasini yaratish dastur fragmentini keltiramiz.

```
String^ message = "Joriy oynani yopishni xoxlaysizmi?";
String^ caption = "Xabar";

MessageBoxButtons buttons = MessageBoxButtons::YesNo;
System::Windows::Forms::DialogResult result;

System::Windows::Forms::DialogResult mayli = System::Windows::Forms::DialogResult::Yes;



result = MessageBox::Show( this, message, caption, buttons, MessageBoxIcon::Question,
MessageBoxDefaultButton::Button2);

                            if ( result == mayli )
```

Odatda tugmalarning farqini aniqlash ancha murakkab, agar OT maska bo‘lsa, farqlash mumkin.

MessageBoxOptions - Enum tipidagi ob‘yekt bo‘lib, u ham yuqoridagidek, nomlar fazosi va kutubxona bilan ishlaydi. Muloqot oynasida standart xususiyatlarni o‘rnatish uchun foydalaniladi.

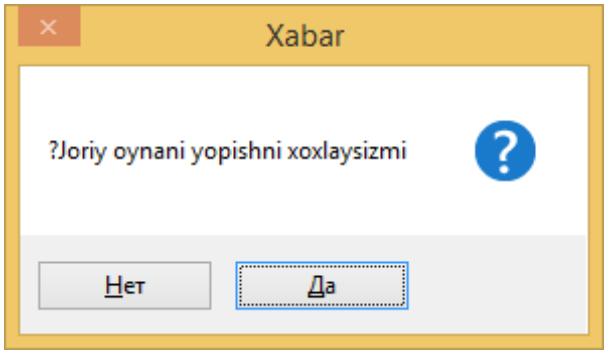
1.8-Jadval . MessageBoxOptions

№	Qiymat nomi	vazifasi
1	DefaultDesktopOnly	Oddiy muloqot oynalariga o‘xshaydi ammo faol ishchi maydonda ko‘rsatiladi.
2	RightAlign	Muloqot onynasining ma‘lumotini to‘g‘ri joylashtirish
3	RtlReading	Muloqot oynasida ma‘lumotni o‘ngdan chapga qarab o‘qish uchun moslab joylashtirish.
4	ServiceNotification	Oddiy muloqot oynalariga o‘xshaydi ammo faol ishchi maydonda ko‘rsatiladi va javob bermasdan o‘tib ketish mumkin xususiyati mavjud

Tugma bosilganda formani yopishni so‘rash muloqot oynasini yaratish dastur fragmentini keltiramiz.

```
String^ message = "Joriy oynani yopishni xoxlaysizmi?";
                                String^ caption = "Xabar";
                                MessageBoxButtons buttons = MessageBoxButtons::YesNo;
                                System::Windows::Forms::DialogResult result;
                                System::Windows::Forms::DialogResult mayli =
System::Windows::Forms::DialogResult::Yes;
                                 result = MessageBox::Show( this, message, caption, buttons,
MessageBoxIcon::Question,MessageBoxDefaultButton::Button1,MessageBoxOptions::RtlReading);
                                if ( result == mayli )
                                {
```

Dastur fragmentini ishlatib ko‘rib, natijasini tahlil qilish va nimalarni o‘ngdan chapga qarab o‘qish o‘zgarganini ko‘rish mumkin.



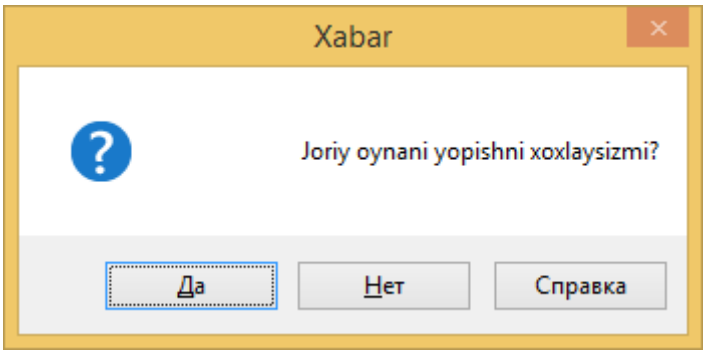
1.10-rasm. MessageBoxOptions bilan muloqot oynasini yaratish.

Barcha muloqot oynalari kabi yuqorida keltirilgan dastur fragmentlaridan ko‘rinib turibdiki, DialogResult tipini qaytaradi.

3. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, String) funksiyasi. Bu funksiyaga yordam fayli va ichki indeks fayllarga ko‘rsatkich yaratadi.

```
String^ message = "Joriy oynani yopishni xoxlaysizmi?";
String^ caption = "Xabar";

MessageBoxButtons buttons = MessageBoxButtons::YesNo;
System::Windows::Forms::DialogResult result;

System::Windows::Forms::DialogResult mayli = System::Windows::Forms::DialogResult::Yes;



result = MessageBox::Show( this, message, caption, buttons,
MessageBoxIcon::Question,MessageBoxDefaultButton::Button1,MessageBoxOptions::RightAlign,
"d:\tut.chm",HelpNavigator::KeywordIndex,"iv");

                            if ( result == mayli )
```

Dastur fragmentini ishlatib ko‘rib, natijasini tahlil qilish va nimalarni o‘zgarganini ko‘rish mumkin.



1.11-rasm. Yordam tugmali muloqot oynasini yaratish.

Dastur fragmentida e‘tibor bilan qarasangiz, MessageBoxButtons::YesNo tshlatilgan, ammo, muloqot oynasida 3 ta tugma chiqqan, demak yordam tugmasining faol holatga kelganining ko‘rish mumkin.

Show usulining boshqa yangi parametrlarni qabul qilmaganligi uchun ularni yozilishi va dastur fragmentlaridan keltiramiz.

4. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String)

```
result = MessageBox::Show( this, message, caption, buttons,
MessageBoxIcon::Question,MessageBoxDefaultButton::Button1,(MessageBoxOptions)0,
"d:\tut.chm",HelpNavigator::KeywordIndex,"iv");
```

5. Show(String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, HelpNavigator)

```
System::Windows::Forms::DialogResult r3 = MessageBox::Show( message, caption,
MessageBoxButtons::OK, MessageBoxIcon::Question, MessageBoxDefaultButton::Button1,
(MessageBoxOptions)0, "mspaint.chm", HelpNavigator::Index );
```

6. Show(String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, String)

```
System::Windows::Forms::DialogResult r7 = MessageBox::Show(message, caption,
MessageBoxButtons::OK, MessageBoxIcon::Question, MessageBoxDefaultButton::Button1,
(MessageBoxOptions)0, "mspaint.chm", "mspaint.chm::/paint_brush.htm" );
```

7. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions)

```
 result = MessageBox::Show( this, message, caption, buttons, MessageBoxIcon::Question,
MessageBoxDefaultButton::Button1, MessageBoxOptions::RightAlign );
```

8. Show(String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String)

```
System::Windows::Forms::DialogResult r1 = MessageBox::Show( " message, caption,
MessageBoxButtons::OK, MessageBoxIcon::Question, MessageBoxDefaultButton::Button1,
(MessageBoxOptions)0, "mspaint.chm" );
```

9. Show(String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, Boolean)

```
System::Windows::Forms::DialogResult r = MessageBox::Show(message, caption,
MessageBoxButtons::OK, MessageBoxIcon::Question, MessageBoxDefaultButton::Button1,
(MessageBoxOptions)0, true );
```

Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, HelpNavigator)

```
System::Windows::Forms::DialogResult r4 = MessageBox::Show( this, message, caption,
MessageBoxButtons::OK, MessageBoxIcon::Question, MessageBoxDefaultButton::Button1,
(MessageBoxOptions)0, "mspaint.chm", HelpNavigator::Index );
```

10. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton)

```
result = MessageBox::Show( this, message, caption, buttons, MessageBoxIcon::Question,
MessageBoxDefaultButton::Button1, MessageBoxOptions::RightAlign );
```

11. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon)

```
result = MessageBox::Show( this, message, caption, buttons, MessageBoxIcon::Question );
```

12. Show(String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton)

```
result = MessageBox::Show( this, message, caption, buttons, MessageBoxIcon::Question,
MessageBoxDefaultButton::Button1 );
```

13. Show(IWin32Window, String, String, MessageBoxButtons)

```
result = MessageBox::Show( this, message, caption, buttons);
```

14. Show(String, String, MessageBoxButtons, MessageBoxIcon)

```
  result = MessageBox::Show(message, caption, buttons, MessageBoxIcon::Question);
```

15. Show(IWin32Window, String, String)

```
result = MessageBox::Show( this, message, caption);
```

16. Show(IWin32Window, String)

```
result = MessageBox::Show( this, message);
```

17. Show(String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions)

```
result = MessageBox::Show( this, message, caption, buttons, MessageBoxIcon::Question,
MessageBoxDefaultButton::Button1, MessageBoxOptions::RightAlign );
```

18. Show(IWin32Window, String, String, MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton, MessageBoxOptions, String, HelpNavigator, Object)

```
System::Windows::Forms::DialogResult r6 = MessageBox::Show(this, message, caption,
MessageBoxButtons::OK, MessageBoxIcon::Question, MessageBoxDefaultButton::Button1,
(MessageBoxOptions)0, "mspaint.chm", HelpNavigator::KeywordIndex, "ovals" );
```

Bu usullarni .NET 5 Preview 1, .NET Core 3.1 3.0 va .NET Framework 4.8, 4.7.2, 4.7.1, 4.7, 4.6.2, 4.6.1, 4.6, 4.5.2, 4.5.1, 4.5, 4.0, 3.5, 3.0, 2.0, 1.1 kabi varintlarda qo‘llab quvatlanadi.

Muloqot oynalarini yaratish. Foydalanuvchi tomonidan muloqot oynalarini yaratish oyna formasiga ishlov berish asosida amalga oshiriladi. Forma oynasiga muloqot oynasini o‘rnatish uchun quyidagi qadamlari bajarish lozim.

1- qadam. Menyudan foydlanib, [menu] → [project] → [addClass] → [CLR] → [Windows Form] buyruqlar ketma ketligi asosida yangi forma qo‘shiladi. Yoki, [menu] → [project] → [add New item] → [UI] → [Windows Form] ham bajarsa bo‘ladi.

2- qadam. Yaratilgan yangi forma oynasigi o‘tib, formaga quyidagicha ishlov beriladi. [Autosize] xususiyatning qiymatini [true] ga, [StartPosition] xususiyatining qiymatini [CenterParent]ga, [FormBorderStyle] xususiyati qiymatiga [none], [FixedDialog], [FixedToolWindows], [SizableToolWindows]larning birini o‘rnatish mumkin.

3- qadam. Forma oynasini o‘zinigz xolagandek loyihalashingiz mumkin. Masalan, bir label, inputBox va bir button joylashtiramiz, ularni ham kerakli xususiyatlari o‘rnatamiz.

4- qadam. Asosiy formaga o‘tib, unga #include "MyForm.h" sarlavha faylni qo‘shamiz. Bu muloqot oynasi uchun yaratilgan forma oynasi bilan ishlash uchun kerak.

5- Qadam. Asosiy formaga bir label va bir button joylashtiramiz. Tugmaning klik hodisasida muloqot oynasini chaqirish dastur fragmentini yozamiz.

```
// MyForm myForm;
// myForm.Show();
// myForm.ShowDialog();
    MyForm^ myFormWith = gcnew MyForm();
 // myFormWith->Show();
    myFormWith->ShowDialog();
```

Dastur fragmentida izohga olib qo‘yilagan yordami ham muloqot oynalarni yaratish va chaqirish mumkin. Ammo ularni farqlari mavjud. Shuning uchun ularni hammasini shu dastur fragmentida keltirdik. Birinchi qatorda formani MyForm myForm oddiy ob‘yekt sifatida yaratilgan. Uning birinchi Show() usuli hisoblanib, bu usul bilan muloqo oynasini chaqirish mantiqan xato, chuniki tizim oynani yaratadi va ustunlikni asosiy formaga beradi. Bu holda yaratilgan oyna bir lahzaga ko‘rinadi xolos. Ikkinchi usuli bu ShowDialog() usul yaxshi yondashuvlardan bo‘lib, muloqot formasiga o‘rnatilgan barcha xususiyatlarga rioya qiladi hamda foydalanish mumkin. Ikkinchi MyForm() konstruktordan foydalanib, gcnew operatori asosida yaratilgan, uning birinchi fuknsiya Show() ham oldingisiga o‘xshash bo‘lib, ammo ustunlikni foydalanuvchining o‘ziga qo‘yib beradi, ya‘ni foydalanuvchi muloqot oynaga javob bermasdan turib, asosiy forma oynasiga o‘tishi mumkin. Ikkinchisi esa bu ShowDialog()usul yaxshi yondashuvlardan bo‘lib, muloqot formasiga o‘rnatilgan barcha xususiyatlarga rioya qiladi hamda foydalanish mumkin.

6-qadam. Muloqot oynasini tugmasida quyidagicha algoritmni yozamiz. Muloqot oynasiga kiritilgan ma‘lumotni asosiy formaga olib o‘tish uchun avval public: System::String^ email; kabi bir o‘zgaruvchi yaratib olamiz.

```
email = textBox1->Text;
this->Close();
```

7-qadam. Asosiy formaning tugmasining hodisasiga yozilgan dastur fragmenti davomidan quyidagini qo‘shib qshyamiz.

```
label1->Text = myFormWith->email;
```

8-qadam. Loyihani ishlatib yaratilgan, yaratilgan muloqot oynani ishlashini ko‘rish mumkin.

Muloqot oynani yuqori darajada yaratish uchun unga yangi konstruktor yozish ham mumkin. Uni quyidagicha amalga oshiriladi.

```
public:  MyForm(System::String^ title)
              {
                     InitializeComponent();
                     _title = title;
              }
public: System::String^ _title;
//….
private: System::Void MyForm_Load(System::Object^  sender, System::EventArgs^  e) {
                            this->Text = _title;
                     }
```

```
// Aspsiy formada esa
                    MyForm^ myFormWith = gcnew MyForm("Xabar");
                    myFormWith->ShowDialog();

                    label1->Text = myFormWith->email;
```

Bunday imkoniyat bilan foydalanuvchi uchun ixtiyoriy muloqot oynasini yaratish mumkin.

Muloqot oynalaridan foydalanish dastur foydalanuvchilariga ko‘plab qo‘layliklar yaratib beradi. Har bir dasturda bir asosiy oynaga kamida funksional imkoniyatiga qarab 4-5 ta muloqo oynalari bo‘ladi.

NAZORAT SAVOLLARI

1. Muloqot oynada nimalar aniq ko‘rsatilishi kerak?
2. [Dialogs] tab oynasida joylashgan muloqot komponentalarini sanab bering?
3. ColorDialog komponentasining vazifasini ayting?
4. Qaysi komponenta foydalanuvchilar uchun papkalar ko‘rish, yaratish va tanlash imkonini beradigan muloqot oynasini ko‘rsatadi.
5. Interaktiv xabarlarni berish va aniq javoblarni olish Visual C++ da qanday sinfi mavjud.
6. MessageBox sinfnining show funksiyasi paramertlarini sanab bering?
7. MessageBoxOptions qanday ob‘yekt tipi va nima vazifani bajaradi.
8. MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton ob‘yektlarini birini qabul qiluvchi kengaytirilgan ob‘yekt tipi nomini ayting.
9. Foydalanuvchi tomonidan yaratiladigan muloqot oynasi qanday yaratiladi.
10. Interaktiv muloqot oynalari kabi nimalarni yaratish va ularni boqarish, kerakli ma‘lumotlarni olish uchun ishlatiladi.
11. Qaysi sinfning nomlar fazosi System.Windows va kutubxonasi PresentationFramework.dll hisoblanadi.
12. ColorDialog muloqot oynasining asosiy vazifasi va ob‘yekti qanday yaratiladi.
13. ColorDialog muloqot oynasidan foydalanish uchun qanday konstruktorni ishga tushirish lozim.
14. CanRaiseEvents xususiyatining vazifasini ayting?
15. komponentaga ilova qilinadigan hodisalar ro‘yxatini oladi xususiyat nomini ayting?
16. Muloqot oynasi ochilganda maxsus ranglarni yaratish uchun ishlatiladigan boshqaruv elementlari ko‘rinib turishini ko‘rsatuvchi qiymatni oladigan yoki o‘rnatadigan xususiyat nomini ayting?

17. Dispose() bu qanday usul?
18. Komponenta yoki uning konteyneri tomonidan taqdim etilgan xizmatni ifodalovchi ob‘yektni qaytaradigan hodisani ayting?
19. Disposed hodisaning vazifasini ayting?
20. Foydalanuvchi umumiy muloqot oynasidagi yordam tugmasini bosganda sodir bo‘ladigan usul nomini ayting?
21. Foydalanuvchilarga kerakli komponenta uchun shrift va uning xususiyatlarini o‘rnatish imkonini beruvchi muloqot oynasini nomini ayting?
22. PrintPreviewDialog muloqot oynasi qaysi tabda joylashgan va vazifasini aniq aytib bering.
23. Dastur fragmentida bir StreamReader oqimi nima uchun yaratiladi.
24. Win32Window Interface nima uchun ishlatiladi.
25. Muloqot oynasida OK va Cancel tugmalarini o‘rnatish qayday amalga oshiriladi.
26. MessageBoxIcon muloqot oynalarida nima uchun ishlatiladi.
27. Qandaydir hodisa haqida faqat xato amal bajarilganligi haqida ma‘lumot beruvchi muloqot oynasi uchun ikonka nomi va yozlishini tushuntirib bering.
28. Muloqot oynasida standart tugmalarni o‘rnatuvchi o‘zgarmaslarni belgilaydigan parametrni ayting?
29. RtlReading qaysi parametr uchun ishlatiladi va muloqot oynasida nima hodisa sodir bo‘ladi.
30. System::Windows::Forms::DialogResult r6 = MessageBox::Show(this, message, caption, MessageBoxButtons::OK, MessageBoxIcon::Question, MessageBoxDefaultButton::Button1, (MessageBoxOptions)0, "mspaint.chm", HelpNavigator::KeywordIndex, "ovals") – dastur fragmentini tushuntirib bering
31. Asosiy formaga yang forma qanday qo‘shiladi
32. Nima uchun [StartPosition] xususiyatining qiymatini [CenterParent]ga tenglashtiriladi.
33. ShowDialog() va Show() usullarining farqini tushuntirib bering.
34. Oddiy ob‘yekt yaratish va konstruktor asosida ob‘yektni yaratish farqini tushuntirib bering.
35. Muloqot oynalarni yaratishda foydalanuvchi yangi konstruktor yaratishi mumkinmi?

14- MA'RUZA

VISUAL C++NING GRAFIK IMKONIYATLARI. (2 SOAT)

Reja:

1. Visual C++ning grafik imkoniyatlari
2. Graphics sinfi usulari asosida tasvirlarni qurish
3. Chart komponentalari.
4. Funksiyalarning grafiklarini qurish.
5. Visual Basic Power Packs komponentalari va ularni ishlatish.

Annotatsiya: *Ushbu ma'ruzada char tipidagi massivlarning elementlariga va string turidagi o'zgaruvchilarga qayta ishlov beruvchi maxsus funksiyalar haqida ma'lumotlar keltirilgan.*

Kalit so'zlar: *char turidagi massiv, string, strlen(), sizeof(), strcpy(), strcat(), strstr(), strchr(), assign(), append(), resize(), insert(), delete(), add().*

KIRISH

Visual C++ o'zi grafik tuzilmali mug'it bo'lib hisoblanadi. Grafik deganda ixtiyoriy narsani kompyuterda piksellarda hosil qilish tushuniladi. Pikselda ikkita argumentlari bor, ya'ni koorlinatalari A(x,u) ko'rinishida. Har qanday IDE muhitlarda grafika bilash ishlash imkoni bor. Ular turlicha nomlanishi mumkin. Ammo, ularning ruchkasi (qalami) va mo'yqalami bo'ladi. GUI ko'rinishda ishlash to'liq grafika bilan bog'liq bo'lib, texnikaning grafik rejimda ishlashini ta'minlaydi. Hozirda deyarli barcha foydalanuvchilar grafik imkoniyatlaridan foydalanib ishlaydilar.

Visual C++ning grafik imkoniyatlari. Grafik imkoniyatga ega bo'lgan tizimlarda asosan, nuqta, chiziq, to'rt burchak, aylana, ko'p burchak kabi shakllarni qamrab oladi. GUI asosidagi barcha elementlarga diqqat bilan qarasangiz shu grafik tuzilmalardan iborat bo'ladi.

Visual C++ning imkoniyatlarini ko'rish uchun Graphics sinfiga murojaat qilamiz. Bu sinfning nomlar fazosi System.Drawing va kutubxonasi System.Drawing.Common.dll bo'lib hisoblanadi.

Chizish uchun GDI+ modulni inkapsulyatsiya qiladi va bu sinfdan merosxo'r olish mumkin emas. Bu sinf MarshalByRefObject, IDisposable, System::Drawing::IdeviceContextga asoslangan bo'lib, Object → MarshalByRefObject

→ Graphics sinfining merosxo'ri hisoblanadi.

Grafika sinfi ekranga tasvirlarni chizish usullarini o'zi ichiga qamarab olgan.

Grafika muayyan qurilmani kontekst bilan bog'lash uchun ham xizmat qiladi.

Grafik ob‘yekt yordamida ko‘p turli shakl va chiziqlar chizish mumkin. Chiziqlar va shakllar chizish uchun maxsus DrawGraphicalElement usullarni o‘rganish lozim. Bu usullar DrawLine, DrawArc, DrawClosedCurve, DrawPolygon va DrawRectangle o‘z ichiga oladi. Chiziqlar va shakllar chizish uchun qalam yordamida va shakllarni to‘ldirish uchun mo‘yqalam yordamida amalga oshiriladi.

Grafika sinfi imkoniyatlarini uning xususiyatlari va usulari orqali ko‘rsatib o‘tamiz. Shuning yangi paraemet kelsa, unga ham to‘xtalib o‘tiladi.

1.1-jadval.Grafika sinfi xususiyatlari.

Xususiyait nomi	vazifasi
Clip	Grafikaning chizilgan chegarasini cheklaydigan chegarani oladi yoki o‘rnatadi.
ClipBounds	Grafikaning kesish chegarasini chegaralovchi RectangleF tuzilishini oladi.
CompositingMode	Kompozit tasvirlar chizish holatini oladi yoki o‘rnatadi.
CompositingQuality	Grafikaga chizilgan kompozitsion tasvirlarning ko‘rsatish sifatini o‘rnatadi.
DpiX	Ushbu grafikaning gorizontal o‘lchamini oladi.
DpiY	Ushbu grafikaning vertikal o‘lchamini oladi.
InterpolationMode	Ushbu grafikalar bilan bog‘liq interpolyatsiya rejimini oladi yoki o‘rnatadi.
IsClipEmpty	Bu grafika kesish sohasini bo‘sh yoki yo‘qligini ko‘rsatib, bir qiymat oladi.
IsVisibleClipEmpty	Bu grafika aniq kesish sohasini bo‘sh yoki yo‘qligini ko‘rsatib, bir qiymat oladi.
PageScale	Bu grafika uchun sahifa moduli va birlik moduli o‘rtasida chegarasini sozlash.
PageUnit	Bu grafika sahifa koordinatalarini uchun ishlatiladigan o‘lchov birligi sozlash.

PixelOffsetMode	Bu grafika ko‘rsatish paytida Piksel ofset qanday ko‘rsatilgan qiymat sozlash.
RenderingOrigin	Bu grafika ko‘rsatish rejimini o‘rnatadi.
SmoothingMode	Grafikalar uchun ko‘rsatish sifatini oladi yoki o‘rnatadi.
TextContrast	Matn ko‘rsatish uchun gamma qiymatini belgilash.
TextRenderingHint	Bu grafika bilan bog‘liq matn uchun ko‘rsatish rejimini o‘rnatadi.
Transform	Grafikalar uchun geometrik o‘zgarishining nusxasini oladi yoki o‘rnatadi.
VisibleClipBounds	Grafikning aniq kesish sohasini tekslash uchun to‘rtburchak oladi.

Grafika sinfi usulari quyidagilardan iborat:

1. AddMetafileComment(Byte[]) - Rasm metafayliga izoh qo‘shadi. Bunda Byte[] belgili massiv bo‘lib, <System::Byte> ^ data tipida aniqlanadi va masalan, array<Byte>^metaCom = {(Byte)'T',(Byte)'e',(Byte)'s',(Byte)'t'}; kabi aniqlanishi mumkin.

2. BeginContainer() - Grafikaning hozirgi holati bilan grafik konteynerni saqlaydi va yangi grafik konteynerni ochadi va ishlatadi.

BeginContainer(Rectangle, Rectangle, GraphicsUnit) - Grafik joriy holati bilan bir grafik konteyner saqlaydi, belgilangan parametrli o‘zgartirish bilan yangi grafik konteynerdan foydalanadi va ochadi. Bunda Rectangle to‘rt burchak bo‘lib, Rectangle(0,0,200,200) kabi aniqlanadi. Birinchi to‘rt burchak konteyner uchun shkalani va ikkinchisi konteyner uchun soha o‘zgarishiri aniqlaydi. GraphicsUnit – konteyner uchun o‘lchov birligini aniqlash uchun ishlatiladi. Uning enum bo‘lib, quyidagi 1.2–jadvalga keltirilgan qiymatlarni oladi va GraphicsUnit::Pixel ko‘rinishda foydalaniladi.

1.2-jadval. GraphicsUnit qiymatlari.

Qiymat nomi	**vazifasi**
Display	Ko‘rsatish qurilmasining o‘lchov birligini belgilaydi. Odatda video displeylar uchun piksellar va printerlar uchun 1/100 dyuym.
Document	O‘lchov birligi sifatida hujjat birligini (1/300 dyuym) belgilaydi.
Inch	O‘lchov birligi sifatida dyuymni bildiradi.
Millimeter	Millimetrni o‘lchov birligi sifatida belgilaydi.

Pixel	O‘lchov birligi sifatida qurilma pikselini belgilaydi.
Point	O‘lchov birligi sifatida printerning nuqtasini (1/72 dyuym) belgilaydi.
World	Koordinata tizimi birligini o‘lchov birligi sifatida belgilaydi.

BeginContainer(RectangleF, RectangleF, GraphicsUnit) - grafik joriy holati bilan bir grafik konteyner saqlaydi, belgilangan sohali o‘zgartirish bilan yangi grafik konteynerdan foydalanadi va ochadi. Bininchi ikkita parametlar tuzilma bo‘lib, ikki xil aniqlanishi mumkin:

RectangleF(PointF(30.0F,40.0F),SizeF(50.0F,100.0F)) va RectangleF(float x, float y, float width, float height); GraphicsUnit – esa konteyner uchun o‘lchov birligini aniqlash uchun ishlatiladi.

3. Clear(Color) - butun chizma yuzasini tozalaydi va belgilangan fon rangi bilan to‘ldiradi. Color tuzilma bo‘lib uning qiymatlariga kengaytirish amali [::] bilan murojaat qiladilar.

4.CopyFromScreen(Int32, Int32, Int32, Int32, Size) - Pikselli to‘rtburchakka mos rangli ma‘lumotlarni ekrandan grafikaning chizma yuzasiga bit-blokli uzatishni amalga oshiradi. CopyFromScreen(int sourceX, int sourceY, int destinationX, int destinationY, System::Drawing::Size blockRegionSize) – bunda x,u lar тщrtburchakning mos koordinatalari va Size sohaning o‘lchamidir. Ham tuzilma bo‘lib, Size(int width, int height) ko‘rinishda yaratiladi.

CopyFromScreen(Int32, Int32, Int32, Int32, Size, CopyPixelOperation) - Pikselli to‘rtburchakka mos rangli ma‘lumotlarni ekrandan grafikaning chizma yuzasiga bit-blokli uzatishni amalga oshiradi. Buning paremetrlari yuqoridagidek bo‘lib, CopyPixelOperation dagi manba rangi yakuniy rangga olib keladigan maqsad rangi bilan birlashtirilganligini aniqlaydi va Blackness, CaptureBlt, DestinationInvert, MergeCopy, MergePaint, NoMirrorBitmap, NotSourceCopy, NotSourceErase, PatCopy, PatInvert, PatPaint, SourceAnd, SourceCopy, SourceErase, SourceInvert, SourcePaint, Whiteness kabi qiymatlarni qabul qiladi.

CopyFromScreen(Point, Point, Size) - Ekrandan grafikaning chizma yuzasiga pikselli to‘rtburchakka mos rangli ma‘lumotlarni bit-blokli uzatishni amalga oshiradi. Point – bu nuqta yaratuvchi tuzilma bo‘lib, o‘zining 3ta Point(Int32), Point(Int32, Int32), Point(Size) konstruktorlari mavjud.

CopyFromScreen(Point, Point, Size, CopyPixelOperation) - ekrandan grafikaning chizma yuzasiga pikselli to‘rtburchakka mos rangli ma‘lumotlarni bit- blokli uzatishni amalga oshiradi.

5. CreateObjRef(Type) - ob‘yekt bilan muloqot qilish uchun ishlatiladigan proksi ishlab chiqarish uchun zarur bo‘lgan barcha tegishli ma‘lumotlarni o‘z ichiga olgan ob‘yekt yaratadi. Type - requestedType tipidagi abstrakt tip,

array<Type^> { int::typeid, int::typeid }); kabi aniqlanadi.

6. Dispose() - Bu grafika tomonidan ishlatiladigan barcha resurslarni null qiladi.

7. DrawArc. Bu funksiya uch o‘lchovli sohani ifodalovchi yoyni chizadi. Uning quyidagi variantlari bor:

DrawArc(Pen, Int32, Int32, Int32, Int32, Int32, Int32) - `void DrawArc(System::Drawing::Pen ^ pen, int x, int y, int width, int height, int startAngle, int sweepAngle);`

DrawArc(Pen, Rectangle, Single, Single) - `void DrawArc(System::Drawing::Pen ^ pen, System::Drawing::Rectangle rect, float startAngle, float sweepAngle);`

DrawArc(Pen, RectangleF, Single, Single) - `void DrawArc(System::Drawing::Pen ^ pen, System::Drawing::RectangleF rect, float startAngle, float sweepAngle);`

DrawArc(Pen, Single, Single, Single, Single, Single, Single) - `void DrawArc(System::Drawing::Pen ^ pen, float x, float y, float width, float height, float startAngle, float sweepAngle);`

DrawArc funksiyasida Pen bu qalam bo‘lib, uning uchun Pen sinfi ishlatiladi. Pen ning 4 ta konstruktori mavjud bo‘lib, Pen(Brush), Pen(Brush, Single), Pen(Color), Pen(Color, Single) konstruktorlardan foydalaniladi, xususiyat va usullarilarni o‘zgartirish orqali o‘rnatish mumkin. Brush bu mo‘yqalam bo‘lib, Brush sinfi ishlatiladi. Brush ning Brush() konstruktori mavjud bo‘lib, xususiyat va usullarilarni o‘zgartirish orqali o‘rnatish mumkin. Funksiyaning qolgan parametrlari barchasi yuqorida keltirilgan parametrlar kabi amalga oshiriladi.

8. DrawBezier. Bu funksiya berilgan 4 ta nuqtali soha uchun splayin chizadi. Uning 3 ta turi bor:

DrawBezier(Pen, Point, Point, Point, Point) - `void DrawBezier(System::Drawing::Pen ^ pen, System::Drawing::Point pt1, System::Drawing::Point pt2, System::Drawing::Point pt3, System::Drawing::Point pt4);`

DrawBezier(Pen, PointF, PointF, PointF, PointF) - `void DrawBezier(System::Drawing::Pen ^ pen, System::Drawing::PointF pt1, System::Drawing::PointF pt2, System::Drawing::PointF pt3, System::Drawing::PointF pt4);`

DrawBezier(Pen, Single, Single, Single, Single, Single, Single, Single, Single)

- Void DrawBezier(System::Drawing::Pen ^ pen, float x1, float y1, float x2, float y2, float x3, float y3, float x4, float y4);

9. DrawBeziers. Bu funksiya berilgan nuchtalar sohasi uchun splayin chizadi. Uning 2 ta turi bor:

DrawBeziers(Pen, Point[]) - void DrawBeziers(System::Drawing::Pen ^ pen, cli::array <System::Drawing::Point> ^ points);

DrawBeziers(Pen, PointF[]) - void DrawBeziers(System::Drawing::Pen ^ pen, cli::array <System::Drawing::PointF> ^ points);

Funksiyada Point[] - ikki o‘lchovli tekislikda nuqta belgilaydigan x va y koordinatalar uchun tartibli juftlikni ifodalaydi massivdir.

10. DrawClosedCurve. Bu funksiya yopiq egri chiziq chizishga mo‘ljallangan. Uning 4 ta turi bor:

DrawClosedCurve(Pen, Point[]) - void DrawClosedCurve(System::Drawing::Pen ^ pen, cli::array <System::Drawing::Point> ^ points);

DrawClosedCurve(Pen, Point[], Single, FillMode) - void DrawClosedCurve(System::Drawing::Pen ^ pen, cli::array <System::Drawing::Point> ^ points, float tension, System::Drawing::Drawing2D::FillMode fillmode);

DrawClosedCurve(Pen, PointF[]) - void DrawClosedCurve(System::Drawing::Pen ^ pen, cli::array <System::Drawing::PointF> ^ points);

DrawClosedCurve(Pen, PointF[], Single, FillMode) - void DrawClosedCurve(System::Drawing::Pen ^ pen, cli::array <System::Drawing::PointF> ^ points, float tension, System::Drawing::Drawing2D::FillMode fillmode);

Bu funksiyada FillMode tipi bor, bu sohani bo‘yaash turini belgilaydi. FillMode ning 2 xil usuli bor, Alternate - Muqobil to‘ldirish va Winding o‘rab to‘ldirish rejimlarini belgilaydi.

11. DrawCurve. Bu funksiya berilgan parametrlar asosida spalyn (egri chiziq) chizadi. Uning 7 ta turi mavjud.

DrawCurve(Pen, Point[]) - void DrawCurve(System::Drawing::Pen ^ pen, cli::array <System::Drawing::Point> ^ points);

DrawCurve(Pen, Point[], Int32, Int32, Single) - void DrawCurve(System::Drawing::Pen ^ pen, cli::array <System::Drawing::Point> ^ points, int offset, int numberOfSegments, float tension);

DrawCurve(Pen, Point[], Single) - void DrawCurve(System::Drawing::Pen ^

pen, cli::array <System::Drawing::Point> ^ points, float tension);

DrawCurve(Pen, PointF[]) - void DrawCurve(System::Drawing::Pen ^ pen, cli::array <System::Drawing::PointF> ^ points);

DrawCurve(Pen, PointF[], Int32, Int32) - void DrawCurve(System::Drawing::Pen ^ pen, cli::array <System::Drawing::PointF> ^ points, int offset, int numberOfSegments);

DrawCurve(Pen, PointF[], Int32, Int32, Single) - void DrawCurve(System::Drawing::Pen ^ pen, cli::array <System::Drawing::PointF> ^ points, int offset, int numberOfSegments, float tension);

DrawCurve(Pen, PointF[], Single) - void DrawCurve(System::Drawing::Pen ^ pen, cli::array <System::Drawing::PointF> ^ points, float tension);

Bu funksiyaning barcha parametrlari tasniflangan.

12. DrawEllipse. Bu funksiya berilgan parametrlar asosida ellipis chizadi. Uning 4 ta turi mavjud.

DrawEllipse(Pen, Int32, Int32, Int32, Int32) - void DrawEllipse(System::Drawing::Pen ^ pen, int x, int y, int width, int height);

DrawEllipse(Pen, Rectangle) - void DrawEllipse(System::Drawing::Pen ^ pen, System::Drawing::Rectangle rect);

DrawEllipse(Pen, RectangleF) - void DrawEllipse(System::Drawing::Pen ^ pen, System::Drawing::RectangleF rect);

DrawEllipse(Pen, Single, Single, Single, Single) - void DrawEllipse(System::Drawing::Pen ^ pen, float x, float y, float width, float height);

Bu funksiyaning barcha parametrlari tasniflangan.

13. DrawIcon. Bu funksiya berilgan parametrlar asosida belgilangan belgi bo‘yicha chizadi. Uning 3 ta turi mavjud.

DrawIcon(Icon, Int32, Int32) - void DrawIcon(System::Drawing::Icon ^ icon, int x, int y);

DrawIcon(Icon, Rectangle) - void DrawIcon(System::Drawing::Icon ^ icon, System::Drawing::Rectangle targetRect);

DrawIconUnstretched(Icon, Rectangle) - void DrawIconUnstretched(System::Drawing::Icon ^ icon, System::Drawing::Rectangle targetRect);

Bu funksiyaning barcha parametrlari tasniflangan.

14. DrawImage. Bu funksiya berilgan parametrlar asosida belgilangan tasvirni chizadi. Uning 30 ta turi mavjud.

DrawImage(Image, Int32, Int32)- `void DrawImage(System::Drawing::Image ^ image, int x, int y);`

DrawImage(Image, Int32, Int32, Int32, Int32) - `void DrawImage(System::Drawing::Image ^ image, int x, int y, int width, int height);`

DrawImage(Image, Int32, Int32, Rectangle, GraphicsUnit) - `void DrawImage(System::Drawing::Image ^ image, int x, int y, System::Drawing::Rectangle srcRect, System::Drawing::GraphicsUnit srcUnit);`

DrawImage(Image, Point) - `void DrawImage(System::Drawing::Image ^ image, System::Drawing::Point point);`

DrawImage(Image, Point[]) - `void DrawImage(System::Drawing::Image ^ image, cli::array <System::Drawing::Point> ^ destPoints);`

DrawImage(Image, Point[], Rectangle, GraphicsUnit) - `void DrawImage(System::Drawing::Image ^ image, cli::array <System::Drawing::Point> ^ destPoints, System::Drawing::Rectangle srcRect, System::Drawing::GraphicsUnit srcUnit);`

DrawImage(Image, Point[], Rectangle, GraphicsUnit, ImageAttributes) - `void DrawImage(System::Drawing::Image ^ image, cli::array <System::Drawing::Point> ^ destPoints, System::Drawing::Rectangle srcRect, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttr);`

DrawImage(Image, Point[], Rectangle, GraphicsUnit, ImageAttributes, Graphics+DrawImageAbort) - `void DrawImage(System::Drawing::Image ^ image, cli::array <System::Drawing::Point> ^ destPoints, System::Drawing::Rectangle srcRect, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttr, System::Drawing::Graphics::DrawImageAbort ^ callback);`

DrawImage(Image, Point[], Rectangle, GraphicsUnit, ImageAttributes, Graphics+DrawImageAbort, Int32) `void DrawImage(System::Drawing::Image ^ image, cli::array <System::Drawing::Point> ^ destPoints, System::Drawing::Rectangle srcRect, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttr,`

System::Drawing::Graphics::DrawImageAbort ^ callback, int callbackData);

DrawImage(Image, PointF) - void DrawImage(System::Drawing::Image ^ image, System::Drawing::PointF point);

DrawImage(Image, PointF[]) - void DrawImage(System::Drawing::Image ^ image, cli::array <System::Drawing::PointF> ^ destPoints);

DrawImage(Image, PointF[], RectangleF, GraphicsUnit) - void DrawImage(System::Drawing::Image ^ image, cli::array <System::Drawing::PointF> ^ destPoints, System::Drawing::RectangleF srcRect, System::Drawing::GraphicsUnit srcUnit);

DrawImage(Image, PointF[], RectangleF, GraphicsUnit, ImageAttributes) -

void DrawImage(System::Drawing::Image ^ image, cli::array <System::Drawing::PointF> ^ destPoints, System::Drawing::RectangleF srcRect, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttr);

DrawImage(Image, PointF[], RectangleF, GraphicsUnit, ImageAttributes, Graphics+DrawImageAbort) - void DrawImage(System::Drawing::Image ^ image, cli::array <System::Drawing::PointF> ^ destPoints, System::Drawing::RectangleF srcRect, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttr, System::Drawing::Graphics::DrawImageAbort ^ callback);

DrawImage(Image, PointF[], RectangleF, GraphicsUnit, ImageAttributes, Graphics+DrawImageAbort, Int32) - void DrawImage(System::Drawing::Image ^ image, cli::array <System::Drawing::PointF> ^ destPoints, System::Drawing::RectangleF srcRect, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttr, System::Drawing::Graphics::DrawImageAbort ^ callback, int callbackData);

DrawImage(Image, Rectangle) - void DrawImage(System::Drawing::Image ^ image, System::Drawing::Rectangle rect);

DrawImage(Image, Rectangle, Int32, Int32, Int32, Int32, GraphicsUnit) - void

DrawImage(System::Drawing::Image ^ image, System::Drawing::Rectangle destRect, int srcX, int srcY, int srcWidth, int srcHeight, System::Drawing::GraphicsUnit srcUnit);

DrawImage(Image, Rectangle, Int32, Int32, Int32, Int32, GraphicsUnit, ImageAttributes) - void DrawImage(System::Drawing::Image ^ image, System::Drawing::Rectangle destRect, int srcX, int srcY, int srcWidth, int srcHeight, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttr);

DrawImage(Image, Rectangle, Int32, Int32, Int32, Int32, GraphicsUnit, ImageAttributes, Graphics+DrawImageAbort) - void DrawImage(System::Drawing::Image ^ image, System::Drawing::Rectangle destRect, int srcX, int srcY, int srcWidth, int srcHeight, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttr, System::Drawing::Graphics::DrawImageAbort ^ callback);

DrawImage(Image, Rectangle, Int32, Int32, Int32, Int32, GraphicsUnit, ImageAttributes, Graphics+DrawImageAbort, IntPtr) - void

DrawImage(System::Drawing::Image ^ image, System::Drawing::Rectangle destRect, int srcX, int srcY, int srcWidth, int srcHeight, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttrs, System::Drawing::Graphics::DrawImageAbort ^ callback, IntPtr callbackData);

DrawImage(Image, Rectangle, Rectangle, GraphicsUnit) - void

DrawImage(System::Drawing::Image ^ image, System::Drawing::Rectangle destRect, System::Drawing::Rectangle srcRect, System::Drawing::GraphicsUnit srcUnit);

DrawImage(Image, Rectangle, Single, Single, Single, Single, GraphicsUnit) - void DrawImage(System::Drawing::Image ^ image, System::Drawing::Rectangle destRect, float srcX, float srcY, float srcWidth, float srcHeight, System::Drawing::GraphicsUnit srcUnit);

DrawImage(Image, Rectangle, Single, Single, Single, Single, GraphicsUnit, ImageAttributes) - void DrawImage(System::Drawing::Image ^ image, System::Drawing::Rectangle destRect, float srcX, float srcY, float srcWidth, float srcHeight, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttrs);

DrawImage(Image, Rectangle, Single, Single, Single, Single, GraphicsUnit, ImageAttributes, Graphics+DrawImageAbort) - void DrawImage(System::Drawing::Image ^ image, System::Drawing::Rectangle destRect, float srcX, float srcY, float srcWidth, float srcHeight, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttrs, System::Drawing::Graphics::DrawImageAbort ^ callback);

DrawImage(Image, Rectangle, Single, Single, Single, Single, GraphicsUnit, ImageAttributes, Graphics+DrawImageAbort, IntPtr) - void DrawImage(System::Drawing::Image ^ image, System::Drawing::Rectangle destRect, float srcX, float srcY, float srcWidth, float srcHeight, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Imaging::ImageAttributes ^ imageAttrs, System::Drawing::Graphics::DrawImageAbort ^ callback, IntPtr callbackData);

DrawImage(Image, RectangleF) - void DrawImage(System::Drawing::Image ^ image, System::Drawing::RectangleF rect);

DrawImage(Image, RectangleF, RectangleF, GraphicsUnit) - void DrawImage(System::Drawing::Image ^ image, System::Drawing::RectangleF destRect, System::Drawing::RectangleF srcRect, System::Drawing::GraphicsUnit srcUnit);

DrawImage(Image, Single, Single) - void DrawImage(System::Drawing::Image ^ image, float x, float y);

DrawImage(Image, Single, Single, RectangleF, GraphicsUnit) - void DrawImage(System::Drawing::Image ^ image, float x, float y, System::Drawing::RectangleF srcRect, System::Drawing::GraphicsUnit srcUnit);

DrawImage(Image, Single, Single, Single, Single) - void

DrawImage(System::Drawing::Image ^ image, float x, float y, float width, float height);

Bu funksiyaning parametrlari ko‘pchiligi tasvirlarga ishlov berish bilan bog‘liq. Shuning uchun o‘rnaish vaqtida zarurlar parametrlari keltirilgan. Shungdek, bir xulosa shundan iboratki, men o‘zim bularning hammasini ishlatib ko‘rmaganman.

15. DrawImageUnscaled. Bu funksiya berilgan parametrlar asosida tasvirni belgilangan joyda asl kattaligidan foydalanib chizadi. Uning 5 ta turi mavjud.

DrawImageUnscaled(Image, Int32, Int32) - void DrawImageUnscaled(System::Drawing::Image ^ image, int x, int y);

DrawImageUnscaled(Image, Int32, Int32, Int32, Int32) - void DrawImageUnscaled (System ::Drawing::Image ^ image, int x, int y, int width, int height);

DrawImageUnscaled(Image, Point) - void DrawImageUnscaled(System::Drawing::Image ^ image, System::Drawing::Point point);

DrawImageUnscaled(Image, Rectangle) - void DrawImageUnscaled(System::Drawing::Image ^ image, System::Drawing::Rectangle rect);

DrawImageUnscaledAndClipped(Image, Rect angle) - void DrawImageUnscaledAndClipped(System::Drawing::Image ^ image, System::Drawing::Rectangle rect);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

16. DrawLine. Bu funksiya berilgan parametrlar asosida chiziq chizadi. Uning 4 ta turi mavjud.

DrawLine(Pen, Int32, Int32, Int32, Int32) - void DrawLine(System::Drawing::Pen ^ pen, int x1, int y1, int x2, int y2);

DrawLine(Pen, Point, Point) - void DrawLine(System::Drawing::Pen ^ pen, System::Drawing::Point pt1, System::Drawing::Point pt2);

DrawLine(Pen, PointF, PointF) - void DrawLine(System::Drawing::Pen ^ pen, System::Drawing::PointF pt1, System::Drawing::PointF pt2);

DrawLine(Pen, Single, Single, Single, Single) - void DrawLine(System::Drawing::Pen ^ pen, float x1, float y1, float x2, float y2);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

22. DrawLines. Bu funksiya berilgan parametrlar asosida ulanadigan chiziq chizadi. Uning 2 ta turi mavjud.

DrawLines(Pen, Point[]) - `void DrawLines(System::Drawing::Pen ^ pen, cli::array <System::Drawing::Point> ^ points);`

DrawLines(Pen, PointF[]) - `void DrawLines(System::Drawing::Pen ^ pen, cli::array <System::Drawing::PointF> ^ points);`

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

17. DrawPath. Bu funksiya berilgan parametrlar asosida bog‘langan chiziqlar va egri chiziqlar qatorini chizadi. Bu sinf merosxo‘r bo‘lishi mumkin emas.

DrawPath(Pen, GraphicsPath) - `void DrawPath(System::Drawing::Pen ^ pen, System::Drawing::Drawing2D::GraphicsPath ^ path);`

18. DrawPie. Bu funksiya berilgan parametrlar asosida Koordinata jufti, kengligi, balandligi va ikkita radial chiziq bilan belgilangan ellips bilan belgilangan nok shaklini chizadi. Uning 4 ta turi mavjud.

DrawPie(Pen, Int32, Int32, Int32, Int32, Int32, Int32) - `void DrawPie(System::Drawing::Pen ^ pen, int x, int y, int width, int height, int startAngle, int sweepAngle);`

DrawPie(Pen, Rectangle, Single, Single) - `void DrawPie(System::Drawing::Pen ^ pen, System::Drawing::Rectangle rect, float startAngle, float sweepAngle);`

DrawPie(Pen, RectangleF, Single, Single) - `void DrawPie(System::Drawing::Pen ^ pen, System::Drawing::RectangleF rect, float startAngle, float sweepAngle);`

DrawPie(Pen, Single, Single, Single, Single, Single, Single) - `void DrawPie(System::Drawing::Pen ^ pen, float x, float y, float width, float height, float startAngle, float sweepAngle);`

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

19. DrawPolygon. Bu funksiya berilgan parametrlar asosida ko‘p shaklini chizadi. Uning 4 ta turi mavjud.

DrawPolygon(Pen, Point[]) - `void DrawPolygon(System::Drawing::Pen ^ pen, cli::array <System::Drawing::Point> ^ points);`

DrawPolygon(Pen, PointF[]) - `void DrawPolygon(System::Drawing::Pen ^ pen, cli::array <System::Drawing::PointF> ^ points);`

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

20. DrawRectangle. Bu funksiya berilgan parametrlar asosida to‘rtburchak shaklini chizadi. Uning 3 ta turi mavjud.

DrawRectangle(Pen, Int32, Int32, Int32, Int32) - void DrawRectangle(System::Drawing::Pen ^ pen, int x, int y, int width, int height);

DrawRectangle(Pen, Rectangle) - void DrawRectangle(System::Drawing::Pen ^ pen, System::Drawing::Rectangle rect);

DrawRectangle(Pen, Single, Single, Single, Single) - void DrawRectangle(System::Drawing::Pen ^ pen, float x, float y, float width, float height);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

21. DrawRectangles. Bu funksiya berilgan parametrlar asosida ulanadigan to‘rtburchak shaklini chizadi. Uning 3 ta turi mavjud.

DrawRectangles(Pen, Rectangle[]) - void DrawRectangles(System::Drawing::Pen ^ pen, cli::array <System::Drawing::Rectangle> ^ rects);

DrawRectangles(Pen, RectangleF[]) - void DrawRectangles(System::Drawing::Pen ^ pen, cli::array <System::Drawing::RectangleF> ^ rects);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

22. DrawString. Bu funksiya berilgan parametrlar asosida shrift ob‘yektlar bilan belgilangan joyda belgilangan matn chizadi (joylashtiradi). Uning 6 ta turi mavjud.

DrawString(String, Font, Brush, PointF) - void DrawString(System::String ^ s, System::Drawing::Font ^ font, System::Drawing::Brush ^ brush, System::Drawing::PointF point);

DrawString(String, Font, Brush, PointF, StringFormat) - void DrawString(System::String ^ s, System::Drawing::Font ^ font, System::Drawing::Brush ^ brush, System::Drawing::PointF point, System::Drawing::StringFormat ^ format);

DrawString(String, Font, Brush, RectangleF) - void DrawString(System::String ^ s, System::Drawing::Font ^ font, System::Drawing::Brush ^ brush, System::Drawing::RectangleF layoutRectangle);

void

DrawString(String, Font, Brush, RectangleF, StringFormat) - void DrawString(System::String ^ s, System::Drawing::Font ^ font, System::Drawing::Brush ^ brush, System::Drawing::RectangleF layoutRectangle, System::Drawing::StringFormat ^ format);

DrawString(String, Font, Brush, Single, Single) - void DrawString(System::String ^ s, System::Drawing::Font ^ font, System::Drawing::Brush ^ brush, float x, float y);

DrawString(String, Font, Brush, Single, Single, StringFormat) - void DrawString(System::String ^ s, System::Drawing::Font ^ font, System::Drawing::Brush ^ brush, float x, float y, System::Drawing::StringFormat ^ format);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

23. EndContainer. Bu funksiya berilgan parametrlar asosida joriy grafik konteynerni yopadi va BeginContainer() usuli uchun ko‘rsatkich orqali saqlangan bu grafik holatini chizadi. EndContainer(GraphicsContainer) - void EndContainer(System::Drawing::Drawing2D::GraphicsContainer ^ container);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

24. EnumerateMetafile. Bu funksiya berilgan parametrlar asosida matnni metafaylga uzatadi va belgilangan joyda chizadi (joylashtiradi). Uning 36 ta turi mavjud.

EnumerateMetafile(Metafile, Point, Graphics+EnumerateMetafileProc) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, System::Drawing::Point destPoint, System::Drawing::Graphics::EnumerateMetafileProc ^ callback);

EnumerateMetafile(Metafile, Point, Graphics+EnumerateMetafileProc, IntPtr) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, System::Drawing::Point destPoint, System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData);

EnumerateMetafile(Metafile, Point, Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, System::Drawing::Point destPoint,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData,
System::Drawing::Imaging::ImageAttributes ^ imageAttr);

EnumerateMetafile(Metafile, Point, Rectangle, GraphicsUnit, Graphics+EnumerateMetafileProc) - void

```
EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,
    System::Drawing::Point destPoint,
```

System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData,
System::Drawing::Imaging::ImageAttributes ^ imageAttr);

EnumerateMetafile(Metafile, Point, Rectangle, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr) - void

```
EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,
    System::Drawing::Point destPoint, System::Drawing::Rectangle srcRect,
    System::Drawing::GraphicsUnit srcUnit,
```

System::Drawing::Graphics::EnumerateMetafileProc ^ callback,
IntPtr callbackData);

EnumerateMetafile(Metafile, Point, Rectangle, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void

```
EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,
    System::Drawing::Point destPoint, System::Drawing::Rectangle srcRect,
    System::Drawing::GraphicsUnit unit,
```

System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData,
System::Drawing::Imaging::ImageAttributes ^ imageAttr);

EnumerateMetafile(Metafile, Point[], Graphics+EnumerateMetafileProc) -

void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array

```
<System::Drawing::Point> ^ destPoints,
```

System::Drawing::Graphics::EnumerateMetafileProc ^ callback);

EnumerateMetafile(Metafile, Point[], Graphics+EnumerateMetafileProc, IntPtr) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array <System::Drawing::Point> ^ destPoints, System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData);

EnumerateMetafile(Metafile, Point[], Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array <System::Drawing::Point> ^ destPoints, System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData, System::Drawing::Imaging::ImageAttributes ^ imageAttr);

EnumerateMetafile(Metafile, Point[], Rectangle, GraphicsUnit, Graphics+EnumerateMetafileProc) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array <System::Drawing::Point> ^ destPoints, System::Drawing::Rectangle srcRect, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Graphics::EnumerateMetafileProc ^ callback);

EnumerateMetafile(Metafile, Point[], Rectangle, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array <System::Drawing::Point> ^ destPoints, System::Drawing::Rectangle srcRect, System::Drawing::GraphicsUnit srcUnit, System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData);

EnumerateMetafile(Metafile, Point[], Rectangle, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void

EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array<System::Drawing::Point> ^ destPoints, System::Drawing::Rectangle srcRect, System::Drawing::GraphicsUnit unit, System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData, System::Drawing::Imaging::ImageAttributes ^ imageAttr);

EnumerateMetafile(Metafile, PointF, Graphics+EnumerateMetafileProc) - void

EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, System::Drawing::PointF destPoint, System::Drawing::Graphics::EnumerateMetafileProc ^ callback);

EnumerateMetafile(Metafile, PointF, Graphics+EnumerateMetafileProc, IntPtr) - void

EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, System::Drawing::PointF destPoint, System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData);

EnumerateMetafile(Metafile, PointF, Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void

EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, System::Drawing::PointF destPoint, System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData, System::Drawing::Imaging::ImageAttributes ^ imageAttr);

EnumerateMetafile(Metafile, PointF, RectangleF, GraphicsUnit, Graphics+EnumerateMetafileProc) - void

EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, System::Drawing::PointF destPoint, System::Drawing::RectangleF srcRect,

System::Drawing::GraphicsUnit srcUnit,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback);

EnumerateMetafile(Metafile, PointF, RectangleF, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,

System::Drawing::PointF destPoint, System::Drawing::RectangleF srcRect,

System::Drawing::GraphicsUnit srcUnit,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback,

IntPtr callbackData);

EnumerateMetafile(Metafile, PointF, RectangleF, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,

System::Drawing::PointF destPoint, System::Drawing::RectangleF srcRect,

System::Drawing::GraphicsUnit unit,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData,

System::Drawing::Imaging::ImageAttributes ^ imageAttr);

EnumerateMetafile(Metafile, PointF[], Graphics+EnumerateMetafileProc) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array

<System::Drawing::PointF> ^ destPoints,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback);

EnumerateMetafile(Metafile, PointF[], Graphics+EnumerateMetafileProc, IntPtr) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array <System::Drawing::PointF> ^ destPoints, System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData);

EnumerateMetafile(Metafile, PointF[], Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void

EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array

<System::Drawing::PointF> ^ destPoints,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData,

System::Drawing::Imaging::ImageAttributes ^ imageAttr);

EnumerateMetafile(Metafile, PointF[], RectangleF, GraphicsUnit, Graphics+EnumerateMetafileProc) - void

EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array <System::Drawing::PointF> ^ destPoints, System::Drawing::RectangleF srcRect,

System::Drawing::GraphicsUnit srcUnit,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback);

EnumerateMetafile(Metafile, PointF[], RectangleF, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr) - void

EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array <System::Drawing::PointF> ^ destPoints, System::Drawing::RectangleF srcRect,

System::Drawing::GraphicsUnit srcUnit,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData);

EnumerateMetafile(Metafile, PointF[], RectangleF, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void

EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, cli::array <System::Drawing::PointF> ^ destPoints, System::Drawing::RectangleF srcRect,

System::Drawing::GraphicsUnit unit,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData,

System::Drawing::Imaging::ImageAttributes ^ imageAttr);

EnumerateMetafile(Metafile, Rectangle, Graphics+EnumerateMetafileProc) - void

EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,

System::Drawing::Rectangle destRect,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback);

EnumerateMetafile(Metafile, Rectangle, Graphics+EnumerateMetafileProc, IntPtr) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, System::Drawing::Rectangle destRect, System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData);

EnumerateMetafile(Metafile, Rectangle, Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, System::Drawing::Rectangle destRect,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData, System::Drawing::Imaging::ImageAttributes ^ imageAttr);

EnumerateMetafile(Metafile, Rectangle, Rectangle, GraphicsUnit, Graphics+EnumerateMetafileProc) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, System::Drawing::Rectangle destRect, System::Drawing::Rectangle srcRect,

System::Drawing::GraphicsUnit srcUnit,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback);

EnumerateMetafile(Metafile, Rectangle, Rectangle, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr) - void EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile, System::Drawing::Rectangle destRect, System::Drawing::Rectangle srcRect,

System::Drawing::GraphicsUnit srcUnit,

System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData);

EnumerateMetafile(Metafile, Rectangle, Rectangle, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void

```
EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,
System::Drawing::Rectangle destRect, System::Drawing::Rectangle srcRect,
System::Drawing::GraphicsUnit unit,
System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData,
System::Drawing::Imaging::ImageAttributes ^ imageAttr);
```

EnumerateMetafile(Metafile, RectangleF, Graphics+EnumerateMetafileProc) - void

```
EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,
System::Drawing::RectangleFdestRect,
System::Drawing::Graphics::EnumerateMetafileProc ^ callback);
```

EnumerateMetafile(Metafile, RectangleF, Graphics+EnumerateMetafileProc, IntPtr) - void

```
EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,
System::Drawing::RectangleF destRect,
System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr
callbackData);
```

EnumerateMetafile(Metafile, RectangleF, Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void

```
EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,
System::Drawing::RectangleF destRect,
System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData,
System::Drawing::Imaging::ImageAttributes ^ imageAttr);
```

EnumerateMetafile(Metafile, RectangleF, RectangleF, GraphicsUnit, Graphics+EnumerateMetafileProc) - void

```
EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,
System::Drawing::RectangleF destRect, System::Drawing::RectangleF srcRect,
System::Drawing::GraphicsUnit srcUnit,
System::Drawing::Graphics::EnumerateMetafileProc ^ callback);
```

EnumerateMetafile(Metafile, RectangleF, RectangleF, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr) - void

```
EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,
System::Drawing::RectangleF destRect, System::Drawing::RectangleF srcRect,
System::Drawing::GraphicsUnit srcUnit,
System::Drawing::Graphics::EnumerateMetafileProc ^ callback,
IntPtr callbackData);
```

EnumerateMetafile(Metafile, RectangleF, RectangleF, GraphicsUnit, Graphics+EnumerateMetafileProc, IntPtr, ImageAttributes) - void

```
EnumerateMetafile(System::Drawing::Imaging::Metafile ^ metafile,
System::Drawing::RectangleF destRect, System::Drawing::RectangleF srcRect,
System::Drawing::GraphicsUnit unit,
System::Drawing::Graphics::EnumerateMetafileProc ^ callback, IntPtr callbackData,
System::Drawing::Imaging::ImageAttributes ^ imageAttr);
```

25. Equals. Belgilangan ob‘yekt joriy ob‘yektga teng yoki yo‘qligini aniqlaydi. Equals(Object) - Belgilangan ob‘yekt joriy ob‘yektga teng yoki yo‘qligini aniqlaydi.

26. ExcludeClip. Bu funksiya berilgan parametrlar asosida ko‘rsatilgan maydonni istisno qilish uchun ushbu grafikaning klip hududini yangilaydi. Uning 2 ta turi mavjud.

ExcludeClip(Rectangle) - `void ExcludeClip(System::Drawing::Rectangle rect);`

ExcludeClip(Region) - `void ExcludeClip(System::Drawing::Region ^ region);`

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

27. FillClosedCurve. Bu funksiya berilgan parametrlar asosida yopiq egri chiziqli splaynni bo‘yyalgan holda chizadi. Uning 6 ta turi mavjud.

FillClosedCurve(Brush, Point[]) - void

```
FillClosedCurve(System::Drawing::Brush ^ brush, cli::array
<System::Drawing::Point> ^ points);
```

FillClosedCurve(Brush, Point[], FillMode) - void

FillClosedCurve(System::Drawing::Brush ^ brush, cli::array <System::Drawing::Point> ^ points, System::Drawing::Drawing2D::FillMode fillmode);

FillClosedCurve(Brush, Point[], FillMode, Single) - void FillClosedCurve(System::Drawing::Brush ^ brush, cli::array <System::Drawing::Point> ^ points, System::Drawing::Drawing2D::FillMode fillmode, float tension);

FillClosedCurve(Brush, PointF[]) - void FillClosedCurve(System::Drawing::Brush ^ brush, cli::array <System::Drawing::PointF> ^ points);

FillClosedCurve(Brush, PointF[], FillMode) - void FillClosedCurve(System::Drawing::Brush ^ brush, cli::array <System::Drawing::PointF> ^ points, System::Drawing::Drawing2D::FillMode fillmode);

FillClosedCurve(Brush, PointF[], FillMode, Single) - void FillClosedCurve(System::Drawing::Brush ^ brush, cli::array <System::Drawing::PointF> ^ points, System::Drawing::Drawing2D::FillMode fillmode, float tension);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

28. FillEllipse. Bu funksiya berilgan parametrlar asosida ellipisni bo‘yyalgan holda chizadi. Uning 4 ta turi mavjud.

FillEllipse(Brush, Int32, Int32, Int32, Int32) - void FillEllipse(System::Drawing::Brush ^ brush, int x, int y, int width, int height);

FillEllipse(Brush, Rectangle) - void FillEllipse(System::Drawing::Brush ^ brush, System::Drawing::Rectangle rect);

FillEllipse(Brush, RectangleF) - void FillEllipse(System::Drawing::Brush ^

brush, System::Drawing::RectangleF rect);

FillEllipse(Brush, Single, Single, Single, Single) - void FillEllipse(System::Drawing::Brush ^ brush, float x, float y, float width, float height);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

29. FillPath. Bu funksiya berilgan parametrlar asosida bog‘langan va egri chiziqlar bo‘yalgan holda chizadi.

FillPath(Brush, GraphicsPath) - void FillPath(System::Drawing::Brush ^ brush, System::Drawing::Drawing2D::GraphicsPath ^ path);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

30. FillPie. Bu funksiya berilgan parametrlar asosida bir juft koordinatalar, kenglik, balandlik va ikkita radial chiziq bilan belgilangan ellipsni bo‘yalgan holda chizadi. Uning 3 ta turi mavjud.

FillPie(Brush, Int32, Int32, Int32, Int32, Int32, Int32) - void FillPie(System::Drawing::Brush ^ brush, int x, int y, int width, int height, int startAngle, int sweepAngle);

FillPie(Brush, Rectangle, Single, Single) - void FillPie(System::Drawing::Brush ^ brush, System::Drawing::Rectangle rect, float startAngle, float sweepAngle);

FillPie(Brush, Single, Single, Single, Single, Single, Single) - void FillPie(System::Drawing::Brush ^ brush, float x, float y, float width, float height, float startAngle, float sweepAngle);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

31. FillPolygon. Bu funksiya berilgan parametrlar asosida maydonni

32. bo‘yalgan holda chizadi. Uning 4 ta turi mavjud. void FillPolygon(System::Drawing::Brush ^ brush, cli::array <System::Drawing::Point> ^ points, System::Drawing::Drawing2D::FillMode fillMode);

FillPolygon(Brush, Point[]) -

FillPolygon(Brush, Point[], FillMode) - void FillPolygon(System::Drawing::Brush ^ brush, cli::array <System::Drawing::PointF>

^ points);

FillPolygon(Brush, PointF[]) - void FillPolygon(System::Drawing::Brush ^ brush, cli::array <System::Drawing::PointF> ^ points, System::Drawing::Drawing2D::FillMode fillMode);

FillPolygon(Brush, PointF[], FillMode) - void FillRectangle(System::Drawing::Brush ^ brush, int x, int y, int width, int height);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

33. FillRectangle. Bu funksiya berilgan parametrlar asosida to‘rtburchakni bo‘yalgan holda chizadi. Uning 4 ta turi mavjud.

FillRectangle(Brush, Int32, Int32, Int32, Int32) - void FillRectangle(System::Drawing::Brush ^ brush, System::Drawing::Rectangle rect);

FillRectangle(Brush, Rectangle) - void FillRectangle(System::Drawing::Brush ^ brush, System::Drawing::RectangleF rect);

FillRectangle(Brush, RectangleF) - void FillRectangle(System::Drawing::Brush ^ brush, float x, float y, float width, float height);

FillRectangle(Brush, Single, Single, Single, Single) - void FillRectangles(System::Drawing::Brush ^ brush, cli::array <System::Drawing::Rectangle> ^ rects);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

34. FillRectangles. Bu funksiya berilgan parametrlar asosida ulanadigan to‘rtburchakni bo‘yalgan holda chizadi. Uning 2 ta turi mavjud.

FillRectangles(Brush, Rectangle[]) - void FillRectangles(System::Drawing ::Brush ^ brush, cli::array <System::Drawing::RectangleF> ^ rects);

FillRectangles(Brush, RectangleF[]) - void FillRegion(System::Drawing::Brush ^ brush, System::Drawing::Region ^ region);

Bu funksiyaining barcha parametrlari ta‘lil qilingan.

35. FillRegion. Bu funksiya berilgan parametrlar asosida sohani bo‘yalgan holda chizadi.

FillRegion(Brush, Region) - `void FillRegion(System::Drawing::Brush ^ brush, System::Drawing::Region ^ region);`

Grafika sinfining 34 ta usullarini varinatlari asosida keltirdik. Bu usullarni va xususiyatlarni tasvirlarni qurish orqali ko‘rib chiqamiz.

Graphics sinfi usulari asosida tasvirlarni qurish. Tasvirlarni qurish uchun **PictureBox** komponentasidan foydalanamiz.

Chiziq chizish. Bunda komponentaning berilgan joyida, rangli chiziq chizishni ko‘ramiz. Buning uchun 1 ta komponenta, 1 ta tugma va 4 ta textBox komponentalarini oynaga qulay qilib joylashtiramiz.

1-qadam. Oynaning Form1_Load hodisasida quyidagi algoritmni joylashtiramiz.

```
this->Text = "DrawLine - chiziq chizish";
button1->Text = "CHIZISH";
```

2-qadam. Komponenta va textBox xususiyatlarini sozlash amallarni bajarish mumkin.

3-qadam. Chiziq chizish uchun nuqtalarga atab int myPoint[4]; o‘zgaruvchisi olamiz.

4-qadam. Komponentaning Paint degan hodisasiga quyidagi algoritmni joylashtiramiz.

```
e->Graphics->DrawLine(System::Drawing::Pens::Red,
                              myPoint[1], myPoint[2], myPoint[3], myPoint[4]);
```

5-qadam. Tugmaning button1_Click hodisasida quyidagi algoritmni joylashtiramiz.

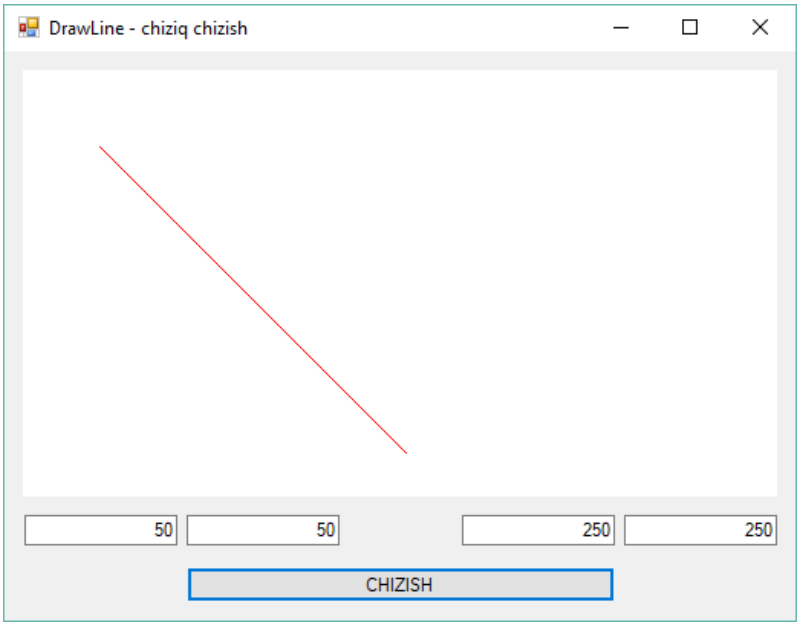
```
myPoint[1 = Convert::ToInt32(textBox1->Text);
    ]
myPoint[2 = Convert::ToInt32(textBox2->Text);
    ]
myPoint[3 = Convert::ToInt32(textBox3->Text);
    ]
myPoint[4 = Convert::ToInt32(textBox4->Text);
    ]
pictureBox1->Refresh();
```

Bunda matndan son tipiga o‘tish amalidan foydalanilgan va har bir tugma bosilganda komponentaning chizmasi o‘zgarib turadi.

Agar loyihani ishga tushursangiz, textBoxlarga kerakli sonlarni kiritib, chiziq chizish mumkin. Komponentaning o‘lchamidan katta chiziqlarni chizish mumkin ammo ko‘rinmaydi. Uni brshqarish

lozim, ya‘ni kerakli kattalikdan oshib ketganda foydalanuvchiga muloqot oynasi bilan xabar berish mumkin, buni textBoxlarning fokusi o‘zgarganda amalga oshirish maqsadga muvofiq.

Chiziq chizishning natijasi 1.1-rasmda keltirilgan.



1.1-rasm. Chiziq chizish natijasi.

Uchburchak chizish. Bunda komponentaning berilgan joyida, rangli chiziqlar orqali uchbursak chizishni ko‘ramiz. Buning uchun 1 ta komponenta, 1 ta tugma va 6 ta textBox komponentalarini oynaga qulay qilib joylashtiramiz. Chuniki 3 ta chiziqning tutashtirsak, uchburchak hosil bo‘ladi.

1-qadam. Oynaning Form1_Load hodisasida quyidagi algoritmni joylashtiramiz.

```
this->Text = "DrawLine - uchburchak chizish";
button1->Text = "CHIZISH";
```

2-qadam. Komponenta va textBox xususiyatlarini sozlash amallarni bajarish mumkin.

3- qadam. Chiziq chizish uchun nuqtalarga atab int myPoint[6]; o‘zgaruvchisi olamiz.

4- qadam. Komponentaning Paint degan hodisasiga quyidagi algoritmni joylashtiramiz.

```
e->Graphics->DrawLine(System::Drawing::Pens::Red,
                          myPoint[1], myPoint[2], myPoint[3], myPoint[4]);
e->Graphics->DrawLine(System::Drawing::Pens::Black,
                          myPoint[3], myPoint[4], myPoint[5], myPoint[6]);
e->Graphics->DrawLine(System::Drawing::Pens::Blue,
```

5-qadam. Tugmaning button1_Click hodisasida quyidagi algoritmni joylashtiramiz.

```
myPoint[1 = Convert::ToInt32(textBox1->Text);
    ]
myPoint[2 = Convert::ToInt32(textBox2->Text);
    ]
myPoint[3 = Convert::ToInt32(textBox3->Text);
    ]
myPoint[4 = Convert::ToInt32(textBox4->Text);
    ]
myPoint[5 = Convert::ToInt32(textBox3->Text);
    ]
myPoint[6 = Convert::ToInt32(textBox4->Text);
    ]
pictureBox1->Refresh();
```

Agar loyihani ishga tushursangiz, textBoxlarga kerakli sonlarni kiritib, uchburchakni chizish mumkin.

Ellipis chizish. Bunda komponentaning berilgan joyida, rangli ellipis chizishni ko‘ramiz. Buning uchun 1 ta komponenta, 1 ta tugma va 4 ta textBox komponentalarini oynaga qulay qilib joylashtiramiz.

1-qadam. Oynaning Form1_Load hodisasida quyidagi algoritmni joylashtiramiz.

```
this->Text = "Ellipse - chizish";
button1->Text = "CHIZISH";
```

2-qadam. Komponenta va textBox xususiyatlarini sozlash amallarni bajarish mumkin.
3-qadam. Chiziq chizish uchun nuqtalarga atab int myPoint[4]; o‘zgaruvchisi olamiz.

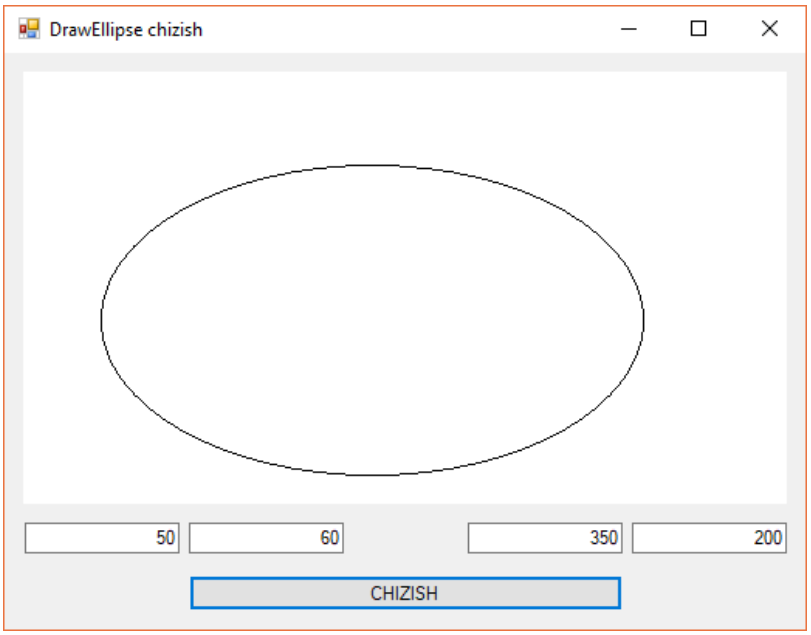
4-qadam. Komponentaning Paint degan hodisasiga quyidagi algoritmni joylashtiramiz.

```
Pen ^ pen = gcnew Pen(Color::Black);
                    e->Graphics->DrawEllipse(pen,
                            myPoint[1], myPoint[2], myPoint[3], myPoint[4]);
```

5-qadam. Tugmaning button1_Click hodisasida quyidagi algoritmni joylashtiramiz.

```
myPoint[1 = Convert::ToInt32(textBox1->Text);
    ]
myPoint[2 = Convert::ToInt32(textBox2->Text);
    ]
myPoint[3 = Convert::ToInt32(textBox3->Text);
    ]
myPoint[4 = Convert::ToInt32(textBox4->Text);
```

```
    ]
pictureBox1->Refresh();
```



1.2-rasm. DrawEllipse chizish natijasi.

Yuqorida keltirilgan chizishlar orqali boshqa ixtiyoriy shakllarni ham chizish mumkin. Turli shakllarni chizib bo‘yyashni ko‘ramiz.

Turli shakllarni bo‘yyash. Bunda komponentaning berilgan joyida, bo‘yyalgan shaklarni chizishni ko‘ramiz. Buning uchun **Brush –** mo‘yqalamdan foydalanib, 1 ta komponenta, 1 ta label va 1 ta Combox komponentalarini oynaga qulay qilib joylashtiramiz.

1-qadam. Oynaning Form1_Load hodisasida quyidagi algoritmni joylashtiramiz.

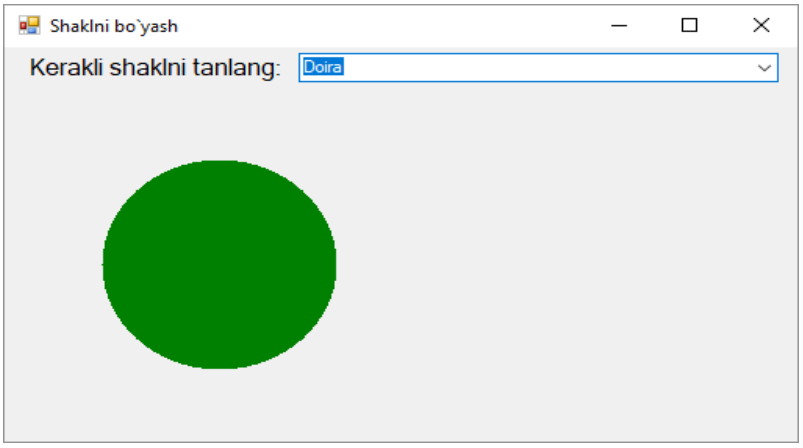
```
this->Text = "Shaklni bo‘yash";
label1->Text = "Kerakli shaklni tanlang:";
comboBox1->Text = "Shakllar";comboBox1->Items->Add("To‘rtburchak");
                              comboBox1->Items->Add("Ellipis");              comboBox1->Items-
                              >Add("Qirqilgan doira ");
```

```
                              comboBox1->Items->Add("Qirqilgan doira ");
```

2-qadam. Komponenta va ComboBox xususiyatlarini sozlash amallarni bajarish mumkin.

3-qadam. ComboBox komponentasining comboBox1_SelectedIndexChanged hodisasida quyidagi algoritmni joylashtiramiz.

```
Graphics ^grp = pictureBox1->CreateGraphics();
Brush ^ brsh = gcnew SolidBrush(Color::Green);
grp->Clear(SystemColors::Control);


switch (comboBox1->SelectedIndex){
        case 0:
              grp->FillRectangle(brsh,50,50,150,150); break;
       case 1:
              grp->FillEllipse(brsh,50,50,300,150); break;
       case 2:
              grp->FillEllipse(brsh,50,50,150,150); break;
       case 3:
              grp->FillPie(brsh,50,50,150,150,150,100); break;
       default:
```



1.3-rasm. Turli shakllarni bo‘yyash natijasi.

Ko‘p burchaklarni chizish. Bunda komponentaning berilgan joyida, bo‘yyalgan foydalanuvchi shaklarni chizishni ko‘ramiz. Buning uchun **Brush –** mo‘yqalamdan foydalanib, 1 ta komponenta, 1 ta button komponentasini oynaga qulay qilib joylashtiramiz.

1-qadam. Oynaning Form1_Load hodisasida quyidagi algoritmni joylashtiramiz.

```
this->Text = "Ko‘p burchakni chizish";
button1->Text = "Chizish";
```

2-qadam. Komponenta va button xususiyatlarini sozlash amallarni bajarish mumkin.

3- qadam. button komponentasining button1_Click hodisasida quyidagi algoritmni joylashtiramiz.

```
Graphics^ graf = pictureBox1->CreateGraphics();
Pen^ gPen = gcnew Pen( Color::Black,1);
HatchBrush^ HBrush = gcnew HatchBrush( Hatchtyle::Sphere, Color::Green, Color::Black );
S


Point  point1  =  Point(50,50);
Point  point2  =  Point(50,200);
Point  point3  =  Point(100,250);
Point  point4  =  Point(150,150);
Point  point5  =  Point(240,250);
Point  point6  =  Point(280,25);
Point  point7  =  Point(300,50);
 array<Point> ^Points = {point1,point2,point3,point4,point5,point6,point7};


graf->DrawPolygon( gPen, Points );
graf->FillClosedCurve( HBrush, Points);
```

Bu algoritmdagi HatchBrush sinfi ishlashi uchun using namespace System::Drawing::Drawing2D; nomlar fazosini qo‘shib qo‘yish kerak. Algoritmda 7 ta nuqtalarni birlashtirish orqali shakd chiziladi.

1.4-rasm. Turli shakllarni bo‘yyash natijasi.

Hodisalar orqali shakl chizish. Hamma "Paint" kabi dasturlarni biladi. Uning eng ajoyib xususiyatlaridan biri sichqoncha bilan chiziqlar chizishdir. Buni amalga oshirish uchun komponentning sichqoncha bilash ishlash **"MousDown"**, **""MausUp"** i **"MouseMove"** hodisalarijan foydalanish mumkin. Dasturning algoritmi g‘oyasi quyidagicha: foydalanuvchi sichqonchaning chap tugmasini bosganda kursor orqasiga juda ko‘p kichik kvadratlar chizila boshlaydi. Bu kvadratlar hajmi kodda ko‘rsatilgan. Bundan tashqari, "tugma" tugmachasini rasmga tushiradigan shaklga ko‘chirishingiz kerak. Bunda komponentaning berilgan joyida, sichqoncha

hodisalari orqali shaklarni chizishni ko‘ramiz. Buning uchun **Brush –** mo‘yqalamdan foydalanib, 1 ta komponenta, 1 ta button komponentasini oynaga qulay qilib joylashtiramiz.

1-qadam. Oynaning Form1_Load hodisasida quyidagi algoritmni joylashtiramiz.

2-qadam. Komponenta va button xususiyatlarini sozlash amallarni bajarish mumkin.

```
button1->Text = "Tozalash";
this->Text = "Shakl chizish";
```

3-qadam. button komponentasining button1_Click hodisasida quyidagi algoritmni joylashtiramiz. Bu algoritm komponentaning tozalash uchun ishlatiladi.

```
Graphics ^ grp = pictureBox1->CreateGraphics();
grp->Clear(SystemColors::Window);
```

4- qadam. Komponentaning sichqoncha hodisalarini qayta ishlash uchun mantiqiy bir o‘zgaruvchi olinadi, uning qiymati [0] bo‘lsin. Bool Drow = false;

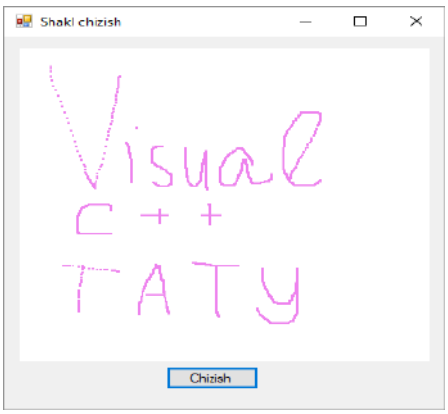
5- qadam. Sichqoncha komponentaning ustiga kelganda mantiqiy o‘zgaruvchining qiymati 1 ga o‘zgaradi. Chunki chizishni boshlash uchun. Buni amalga oshirishsh uchun sichqonchaning pictureBox1_MouseDown hodisasiga Drow = true; ni yozib qo‘yamiz.

6- qadam. Sichqoncha komponentaning ustidan ketganda mantiqiy o‘zgaruvchining qiymati 0 ga o‘zgaradi. Chunki chizishni tugatish uchun. Buni amalga oshirishsh uchun sichqonchaning pictureBox1_MouseUp hodisasiga Drow= false; ni yozib qo‘yamiz.

7-qadam. Sichqoncha komponentaning ustida kelganda chizishi uchun quyidagi algoritmni kiritamiz.

```
Graphics^ graf = pictureBox1->CreateGraphics();
      if (Drow == true){
      graf->FillEllipse(Brushes::Violet, e->X, e->Y, 3,3); // mo‘y qalam qalinligi
}
```

Algoritmda shakllarni hosil qilish uchun juda kichik ellipischalardan foydalanamiz.



1.5-rasm. Sichqoncha bilan turli shakllarni chizish.

Rasmlarni o‘zgartirish. Buning uchun fayllarni yuklash **openFiledialog** va fayllarni saqlash **saveFiledialog** komponentalaridan foydalanamiz. Buning uchun **Brush –** mo‘yqalamdan foydalanib, 1 ta komponenta, 3 ta button komponentasini oynaga qulay qilib joylashtiramiz. Button larning birinchisi, rasmni tahrirlashni tozalashni amalga oshiradi, ikkinchisi, tahrirlangan rasmni saqlash uchun, uchinchisi, rasmni tahrirlash uchun yuklashni amalga oshiradi. using namespace System::Drawing::Drawing2D; ni qo‘shishb qo‘yish kerak.

1-qalam. Rasmlarni o‘zgartirish uchun ishlatiladigan o‘zgaruvchilarni e‘lon qilamiz.

```
        private: Bitmap ^ bmp_for_draw;
```

```
        private: Point start_point;
        private:  bool  Draw;
        public: Pen^ pen_for_draw;

        private: String ^ full_name_of_image;
```

2-qadam. Dastur oynasining Form1_Load hodisasiga dastlabki sozlamalarni o‘rnatamiz.

```
this->Text = "Rasmni tahrirlash";
  button1->Text = "Tozalash";
  button2->Text= "Rasmni saqlash";
  button3->Text = "Rasmni yuklash";

  pen_for_draw = gcnew Pen(Color::Black, 4);
  pen_for_draw->StartCap = System::Drawing::Drawing2D::LineCap::Round;
  pen_for_draw->EndCap = System::Drawing::Drawing2D::LineCap::Round;
```

Bunda komponentalarni sozlash va bir qalamni yaratib olish, uning StartCap, EndCap xususiyatlarini o‘rnatish amalga oshiriladi.

3- qadam. Komponentdagi rasmni tozalash uchun button1_Click hodisaga algoritmni yozamiz.

```
Graphics^ grp = pictureBox1->CreateGraphics();
grp->Clear(SystemColors::Window);
```

4-qadam. Sichqoncha komponentaning ustida kelganda chizishi uchun pictureBox1_MouseDown hodisasiga quyidagi algoritmni kiritamiz

```
if (e->Button == System::Windows::Forms::MouseButtons::Left){
                                    Draw = true;
                                    start_point = e->Location;
                                }
```

5-qadam. Sichqoncha komponentaning ustida ketganda chizishni to‘xtatish uchun pictureBox1_MouseUp hodisasiga quyidagi algoritmni kiritamiz

```
if (e->Button == System::Windows::Forms::MouseButtons::Left){
                                    Draw = false;
                                    }
```

6-qadam. Komponentaga kerakli rasmni yuklash uchun button3_Click hodisasiga quyidagi algoritmni yoziladi

```
OpenFileDialog ^ open_dialog = gcnew OpenFileDialog();
open_dialog->Filter = "Image Files(*.BMP;*.JPG;*.GIF;*.PNG)|*.BMP;*.JPG;*.GIF;*.PNG|All files (*.*)|*.*";
if (open_dialog->ShowDialog() == System::Windows::Forms::DialogResult::OK){
        try{
```

```
            full_name_of_image = open_dialog->FileName;
            bmp_for_draw = gcnew Bitmap(open_dialog->FileName);
            //pictureBox1->Size = bmp_for_draw->Size;
            pictureBox1->SizeMode = PictureBoxSizeMode::StretchImage;
            pictureBox1->Image = bmp_for_draw;
            pictureBox1->Invalidate();
        } catch(Exception^ e) {
        System::Windows::Forms::DialogResult rezult = MessageBox::Show("Tanlangan faylni ochib bo'lmaydi "+ e->ToString(),"Diqqat", MessageBoxButtons::OK, MessageBoxIcon::Error);
```

Bundagi asosi nuqson shundan iboratki, agar rasmning o‘lchamlari komponentaning o‘lchamlari bilan bir xil bo‘lsa, yaxshi chizadi, bir xil bo‘lmasa, foydalanuvchi ekranining proporsiyasini olib chizadi.

7-qadam. Komponentadagi rasmni saqlash uchun button2_Click hodisasiga quyidagi algoritmni yoziladi

```
if (pictureBox1->Image != nullptr){
String ^format = full_name_of_image->Substring(full_name_of_image->Length - 4, 4); SaveFileDialog ^savedialog = gcnew SaveFileDialog();
savedialog->Title = "Rasmni saqlash ..."; savedialog->OverwritePrompt = true;
savedialog->CheckPathExists = true;
```

```
savedialog->Filter = "Image Files(*.BMP)|*.BMP|Image Files(*.JPG)|*.JPG|Image
Files(*.GIF)|*.GIF|Image Files(*.PNG)|*.PNG|All files (*.*)|*.*";

savedialog->ShowHelp = true;

// If selected, save

if (savedialog->ShowDialog() == System::Windows::Forms::DialogResult::OK){
       try{

bmp_for_draw->Save(savedialog->FileName,  System::Drawing::Imaging::ImageFormat::Jpeg);

} catch(Exception ^ e) {

System::Windows::Forms::MessageBox::Show("Impossible to save image"+e->ToString(), "FATAL
ERROR", MessageBoxButtons::OK, MessageBoxIcon::Error);

} } }
```

8-qadam. Sichqoncha komponentaning ustida kelganda chizishi uchun quyidagi algoritmni kiritamiz.

```
if (e->Button ==  System::Windows::Forms::MouseButtons::Left){
Graphics ^graf = Graphics::FromImage(pictureBox1->Image);
graf->DrawLine(pen_for_draw, start_point, e->Location);
start_point = e->Location;
pictureBox1->Invalidate();

}
```

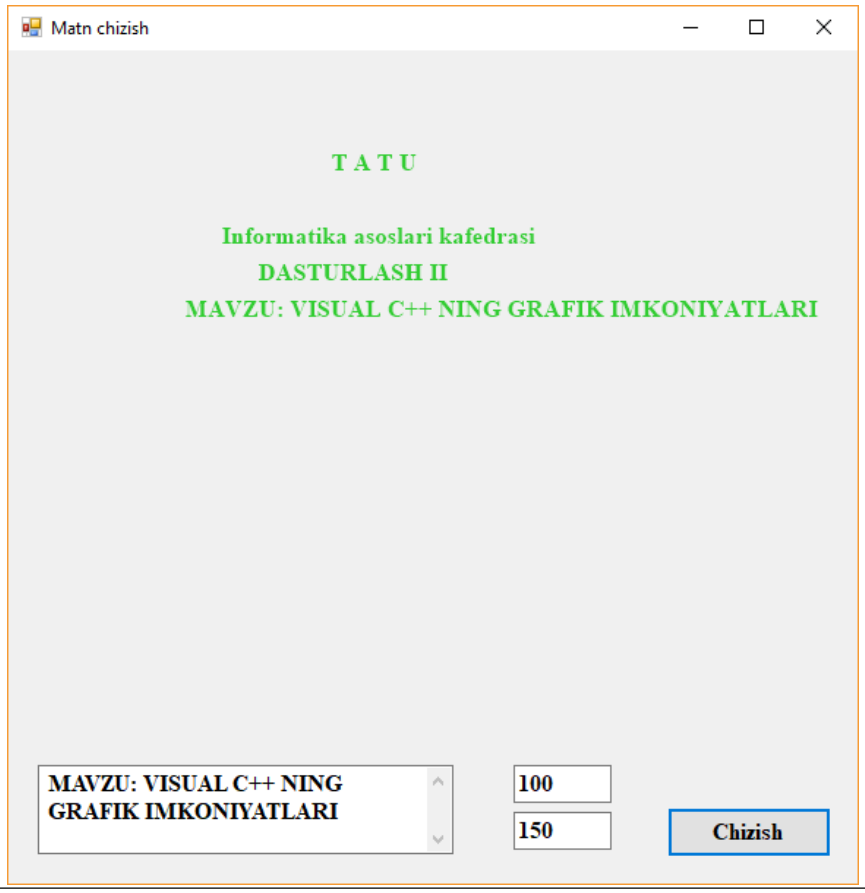
Dasturni ishlatganda quyidagi oyna chiqadi.

1.6-rasm. Rasmlarni tahrirlash.

Matnlarni tasvir kabi joylashtirish. Bunda komponentaning belgilangan joyida, aniq o‘lchamli va rangli matnni joylashtirishni ko‘ramiz. Buning uchun komponentaga 3 ta textbox va 1ta botton komponentlarini qulay qilib joylashtiramiz. Birinchisi matn yozish uchun, qolgan ikkitasi matnni joylashtirish nuqtasi uchun.

1-qadam. Textbox ni rejimini multiline rejimiga va scrolBars xususityatiga Vertical qiymati o‘rnatiladi.

2-qadam. Oynaning Form1_Load hodisasida quyidagi algoritmni joylashtiramiz.




```
Font = gcnew System::Drawing::Font("Times New Roman", 12, FontStyle::Bold);
button1->Text = "Chizish";

this->Text = "Matn chizish";
```

3-qadam. Komponenta va button xususiyatlarini sozlash amallarni bajarish mumkin.

4-qadam. button komponentasining button1_Click hodisasida quyidagi algoritmni joylashtiramiz. Bu algoritm komponentaning tozalash uchun ishlatiladi.

```
String^ Text = String::Format("{0}", textBox1->Text);
Brush^ brsh = gcnew SolidBrush(Color::LimeGreen);
Graphics^ G = pictureBox1->CreateGraphics();

G->TextRenderingHint = System::Drawing::Text::TextRenderingHint::AntiAlias;
int xP = Convert::ToInt16(textBox2->Text);

int yP = Convert::ToInt32(textBox3->Text);
```

Dasturni ishga tushiramiz va quyidagi natijani olamiz

1.7-rasm. Matnlarni joylashtirish.

Matnli rasmlarni yaratish. Matnli rasmlarni yaratish uchun 1 ta komponenta, 4 ta label, 4 ta button, 4 ta textbox larni qulay qilib joylashtiriladi.

1-qadam. 2 ta Textbox ni rejimini multiline rejimiga va scrolBars xususityatiga Vertical qiymati o‘rnatiladi.

2-qadam. Oynaning Form1_Load hodisasida quyidagi algoritmni joylashtiramiz.

```
                = "|";        this->label1->Text = "| Yuqoridagi matn";
label2->Text
label3->Text  = "| Pastdagi matn";
label4->Text  = "|";
button1->Text = "Tozalsh";
button2->Text = "Chizish";
button3->Text = "Saqlash";
button4->Text = "Rasm yuklash";
```

3- qadam. Komponenta va button xususiyatlarini sozlash amallarni bajarish mumkin.

4- qadam. TextBox3 va TextBox4 ni mos ravishda TextBox1 va TextBox2 lar orqasiga tashlaymiz.

5- qadam. Formaning formBorderStyle xususiyatiga SizableToolWindow ni o‘rnatamiz. Formani o‘lchamlarini o‘zgartirmaslik uchun, buni boshqacha ham amalga oshirish mumkin.

6- qadam. TextBox3 va TextBox4 ni shrift o‘lchamlarini 12, TextBox1 va TextBox2 lar boshqa kattaroq o‘lchamni o‘rnatamiz. Chunki dastur ishga tushgada asosiy ko‘rinadigann matnlar ko‘rinmaydigan matnlarga ko‘chiriladi va undan komponentaga ko‘chiriladi.

7-qadam. Zarurriy o‘zgaruvchilarni aniqlashmiz.

```
            private: Bitmap^ bmp_for_draw;
            private: String^ full_name_of_image;
```

8-qadam. Oynaning Form1_Load hodisasi davomidan quyidagi algoritmni joylashtiramiz.

```
            Text = "Matnni rasmga joylashtirish";
            Font = gcnew System::Drawing::Font("Times New Roman", 32, FontStyle::Bold);
            this->pictureBox1->SizeMode = PictureBoxSizeMode::StretchImage;
```

9-qadam. Komponentaga kerakli rasmni yuklash uchun button4_Click hodisasiga quyidagi algoritmni yoziladi

```
OpenFileDialog^ open_dialog = gcnew OpenFileDialog();
open_dialog->Filter = "Image Files(*.BMP;*.JPG;*.GIF;*.PNG)|*.BMP;*.JPG;*.GIF;*.PNG|All files (*.*)|*.*";

if (open_dialog->ShowDialog() == System::Windows::Forms::DialogResult::OK)        {
       try{

             full_name_of_image = open_dialog->FileName;
             bmp_for_draw = gcnew Bitmap(open_dialog->FileName);
             pictureBox1->Image = bmp_for_draw;

             pictureBox1->Invalidate();
             } catch(Exception^ e) {
             System::Windows::Forms::DialogResult rezult = MessageBox::Show("Fayl xato
```

10-qadam. Komponentadagi rasmni saqlash uchun button3_Click hodisasiga quyidagi algoritmni yoziladi

```
if (pictureBox1->Image != nullptr){
  String^ format = full_name_of_image->Substring(full_name_of_image->Length - 4, 4);
  SaveFileDialog^ savedialog = gcnew SaveFileDialog();

  savedialog->OverwritePrompt = true; // Agar shu nomli fayl bo‘lsa
  savedialog->CheckPathExists = true; // Agar noto‘g‘ri nom kiritilsa
  savedialog->ShowHelp = true;

  savedialog->Filter = "Image Files(*.BMP)|*.BMP|Image Files(*.JPG)|*.JPG|Image
Files(*.GIF)|*.GIF|Image Files(*.PNG)|*.PNG|All files (*.*)|*.*";

  if (savedialog->ShowDialog() == System::Windows::Forms::DialogResult::OK){
   try {

          Bitmap^ MBB = gcnew Bitmap(pictureBox1->Image);
          MBB->Save(savedialog->FileName,  System::Drawing::Imaging::ImageFormat::Jpeg);
         } catch(Exception^ e){
         MessageBox::Show("Rasmni saqlashda xatolik", "FATAL ERROR",
```

11-qadam. Komponentadagi rasmni tozalash uchun button1_Click hodisasiga quyidagi algoritmni yoziladi

```
Bitmap^ BM = gcnew Bitmap(pictureBox1->Image);
Graphics^ C = System::Drawing::Graphics::FromImage(BM);

//Graphics C = CreateGraphics();
C->Clear(pictureBox1->BackColor);
pictureBox1->Image = BM;
```

12- qadam. Komponentaga matnlarni joylashtirish uchun uchun button2_Click hodisasiga quyidagi algoritmni yoziladi

```
System::Drawing::Font^ FontTemp  = gcnew System::Drawing::Font("Times New Roman", 32,
```

```
FontStyle::Bold);
 textBox4->Text = textBox1->Text;
textBox3->Text = textBox2->Text;

SaveFileDialog^ savedialog = gcnew SaveFileDialog();
String^ Text = String::Format("{0}", textBox4->Text);
String^ Txt = String::Format("{0}", textBox3->Text);
Brush^ brsh = gcnew SolidBrush(Color::White);

Bitmap^ MBB = gcnew Bitmap(pictureBox1->Image);
Graphics^ G = System::Drawing::Graphics::FromImage(MBB);
Graphics^ Q = G;

G->TextRenderingHint = System::Drawing::Text::TextRenderingHint::AntiAlias;
G->DrawString(Text, FontTemp, brsh, 75, 2);

Q->TextRenderingHint = System::Drawing::Text::TextRenderingHint::AntiAlias;
Q->DrawString(Txt, FontTemp, brsh, 5, 225);

pictureBox1->Image = MBB;
```

Dasturni ishga tushiramiz va quyidagi natijani olamiz

1.8-rasm. Matnlarni rasmga joylashtirish.

Graphics sinfi usulari asosida tasvirlarni qurishning turli usullari qarab chiqdik, bu amallar yordamida rasmlani tahrirlash, ishlov berish, matnlarni joylashtirish kabi ixtiyoriy amallarni bajarish mumkin.

Chart komponenta xususiyati va hodisalari. Bu komponenta Data tab bo‘limiga joylashgan bo‘lib, asosan ma‘lumotlarni infografiklarni yaratish uchun ishlatiladi. Infografik uchun ma‘lumotlar to‘plami kerak. Bu komponentaning xusussiyati va hodisalari boshqa komponentalarniki kabi bo‘lib, xuddi o‘shalar kabi bo‘limlarga bo‘lingan. Ularning maxsuslarini keltirib o‘tamiz.

1. BorderSkin xususiyatlar gruppasi bo‘lib, unda komponentaning yangi niqobga solish mumkin. Niqob deganda, uning yangi ko‘rinishi inobatga olingan. Rang (color), rasm (image), stil (style), kengligi (width) kabi xususiyatlari mavjud. Bu xususiyatlarni o‘rnatish muammo keltirib chiqarmaydi. Oldingi o‘rganganlaringizda bunday xususiyatlardan foydalangansiz. Shuningdek asosiy niqob bu SkinStyle bo‘lib, komponentaning asosiy ko‘rinishini o‘zgartrish uchun xizmat qiladi. Uning mos qiymatlar ro‘yxati mavjud, shundan keraklisini tanlabolish mumkin.
2. Palette (palitra) – xususiyati yordamida komponentaning infografikani ko‘rsatadigan shaklini tanlash mumkin. Uning mos qiymatlar ro‘yxati mavjud.
3. PaletteCustomColors – bunda ham komponentaning infografikani ko‘rsatadigan shaklini tanlash mumkin. Ammo foydalanuvchi o‘zining rangi tanlashimi mumkin. Bunda maxsus muloqot oynasi aosida palitraga turli ranglarni qo‘shish mumkin.
4. Annotations – bunda komponentaga izohlarni yozish mumkin. Buning o‘zining xususiyatlari maxsus muloqot oynasi yordamida o‘rnatiladi.
5. ChartAreas – bu xususiyat orqali komponentaga bir nechata chart infografika

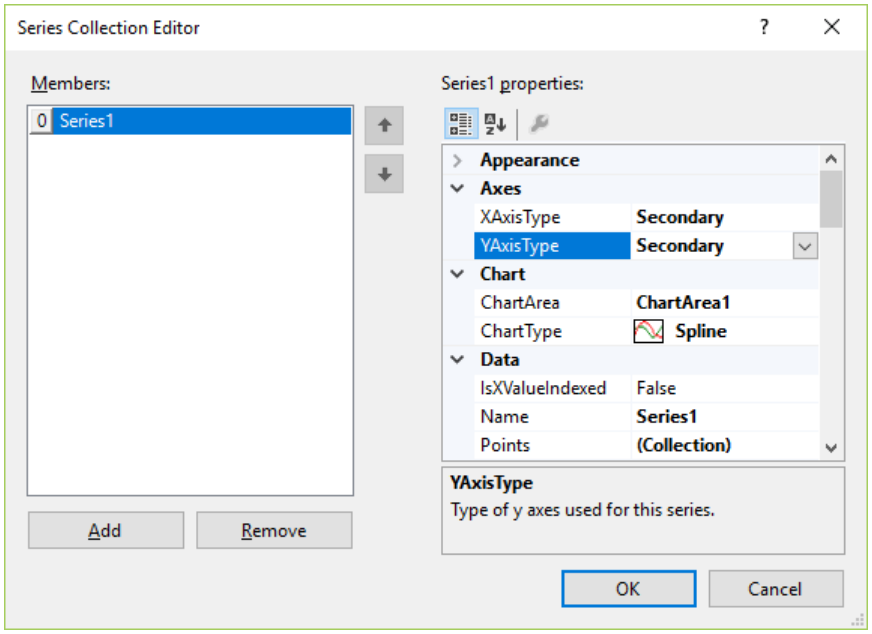
joylashtirish mumkin. Buning ham o‘zi mos xususiyatlarini mos muloqot oynasi bilan o‘rnatish lozim.

6. Legends – infografikaga keltirilgan qiymatlarining joylashish maydoni. Buni ham maxsus muloqot oynasi asosida tahrir qilish mumkin

7. Series – bu xususiyat asosiy bo‘lib, infografikaning qiymatlarini belgilovchi, har bir qiymat tegishliligini bildiradi. Buning uchun maxsus mulovot oynasi mavjud. Bunga to‘liqroq to‘xtalib o‘tamiz.

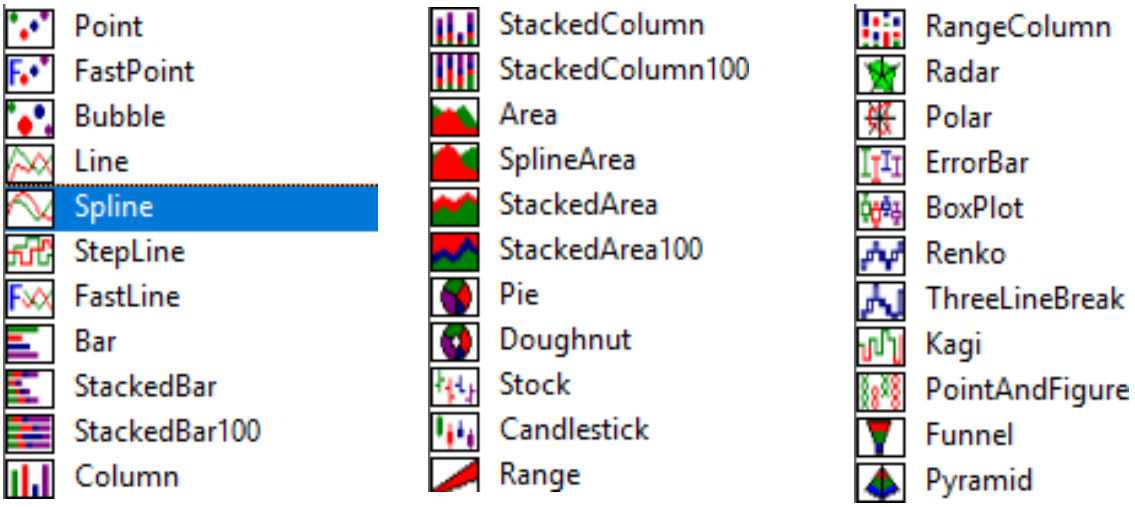
8. Titles – komponentalarga joylashtirilgan infografikalarga sarlavha qo‘yish uchun ishlatiladi. Uning maxsus muloqot oynasi orqali ishlov berish orqali shrnatish mumkin.

Series – bu komponentaga qiymatlarni qo‘shish va uni tasvirlash uchun xizmat qiladi. Komponentaning ichiga joylashtirilgan sohani tahrirlash uchun ishlatiladi. Unga bosganda quyidagi muloqot oynasi chiqadi.



1.9-rasm. Tahrirlash oynasi.

Bu oynda yangi qiymatlar maydoni qo‘shish uchun [Add] tugmasi va uni o‘chirish uchun [Remove] tugmasi ishlatiladi. Infografikaning asosiy xususiyatlarini boshqarish uchun o‘ng tomondagi xususiyatlar panelidan foydalanish mumkin. Unda infografikani taxhrirlash uchun zarur bo‘lgan barcha xususiyatlar bor. Ulardan biri bu ChartType bo‘lib, infografikaning turlarini belgilash uchun xizmat qiladi. Uning turlari quyidagi 1.10-rasmda keltirilgan.



1.10-rasm. Infografikaning turlari

Shuningdek, infografikaning ma‘lumotlari, yozuvlari, qiymatlari, maydoni, qiymat turlari, chegaralari bilan ishlash xususiyatlari xam muvjud. Odatda bu xususiyatlarni dasturlash orqali dastur fragmentlarida o‘rnatish fa foydalanish dasturchiga qulay hisoblanadi. Ammo vizual dusturlashning imkoniyatidan foydalanish uchun buni ham ishlatishni o‘rganish lozim.

Funksiyalarni grafiklarini qurish. Infografika komponentasiga mos ravishda funksiyalarni grafikgini chizish usullari ko‘rib chiqamiz.

Matematik funksiyalarni grafiklarini chizish uchun avval shu funksiyalarni bir sinfga yaratib olamiz.

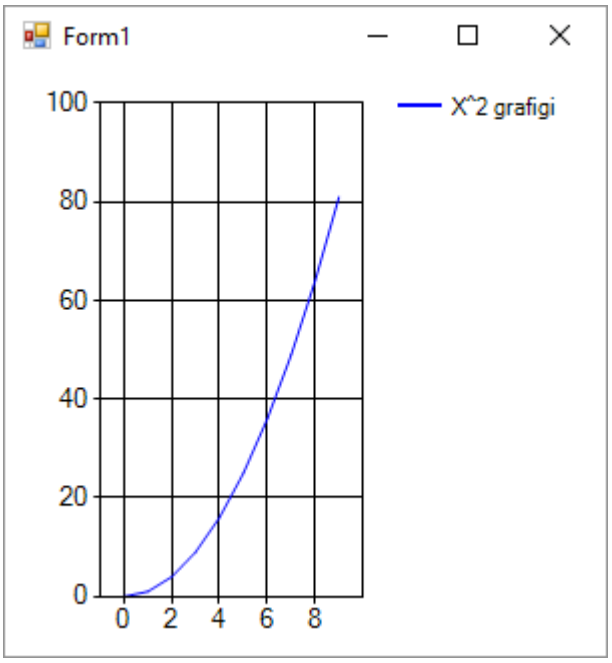
```
private value class MyFunction
	{
	private:
		double _value;
	public:
		double getValuePow(double x) {
			return Math::Pow(x, 2);

		}
		double  getValueX4(double x){
			return x*x*x*x;

		}
		double  getValuekxa(double x, int k, int a){
			return k*x+a;
```

Bu ko‘rinishda matematikaning barcha funksiyalarini yaratib olish yoki to‘g‘ridan to‘g‘ri foydalanish mumkin.

Chart komponentasiga grafikni chizish uchun Form1_Load hodisasiga quyidagicha algoritmni yozamiz.

```
chart1->Series->Clear();
    Series^ series1 = gcnew Series(L"X^2 grafigi");
    // rangni tanlash
    series1->Color = Color::Blue;
    series1->IsVisibleInLegend = true;
    series1->IsXValueIndexed = true;
    // infografikani turini tanlash
    series1->ChartType = SeriesChartType::Line;
    // qiymatlar qatlamini qo‘shish
    chart1->Series->Add(series1);
    // qiymatlarni
    MyFunction^ func = gcnew MyFunction();
    for (double i = 0; i < 10; i++) {
              series1->Points->AddXY(i,  func->getValuePow(i));
    }
```

Agar dasturni ishlatsak, bir x^2 funksiyaning grafigini chizish imkoniyatini beradi.

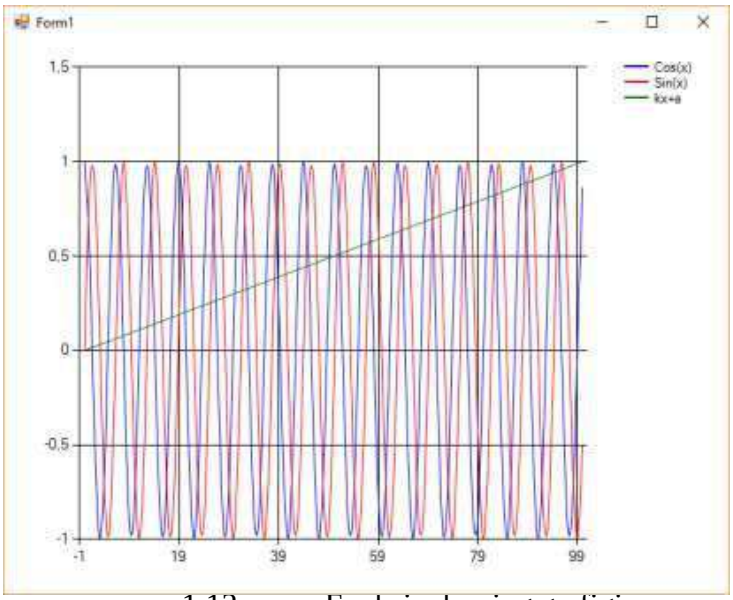


1.11-rasm. x^2 funksiyaning grafigi.

Bir vaqtning o‘zida bir nechta funksiyaning grafiklarini chizish uchun yuqorida aniqlangan sinfdan foydalanib, Form1_Load hodisasiga quyidagicha algoritmni yozamiz.

```
chart1->Series->Clear();
    Series^ series1 = gcnew Series(L"Cos(x)");
       Series^ series2 = gcnew Series(L"Sin(x)");
       Series^ series3 = gcnew Series(L"kx+a");

    // rangni tanlash
    series1->Color = Color::Blue;
       series2->Color = Color::Red;
       series3->Color = Color::Green;

       // infografikani turini tanlash
       series1->ChartType = SeriesChartType::Spline;
       series2->ChartType = SeriesChartType::Spline;
       series3->ChartType = SeriesChartType::Spline;

       // qiymatlar qatlamini qo‘shish
    chart1->Series->Add(series1);
    chart1->Series->Add(series2);
    chart1->Series->Add(series3);
    // qiymatlarni
    MyFunction^ func = gcnew MyFunction();
    for (double i = 0; i <= 100; i++) {
              series3->Points->AddXY(i, func->getValuekxa(i*0.01,1,0));
              series1->Points->AddXY(i, Math::Cos(i));
              series2->Points->AddXY(i, Math::Sin(i));
    }
```

Agar dasturni ishlatsak, bir funksiyalarning grafigini chizish imkoniyatini beradi.

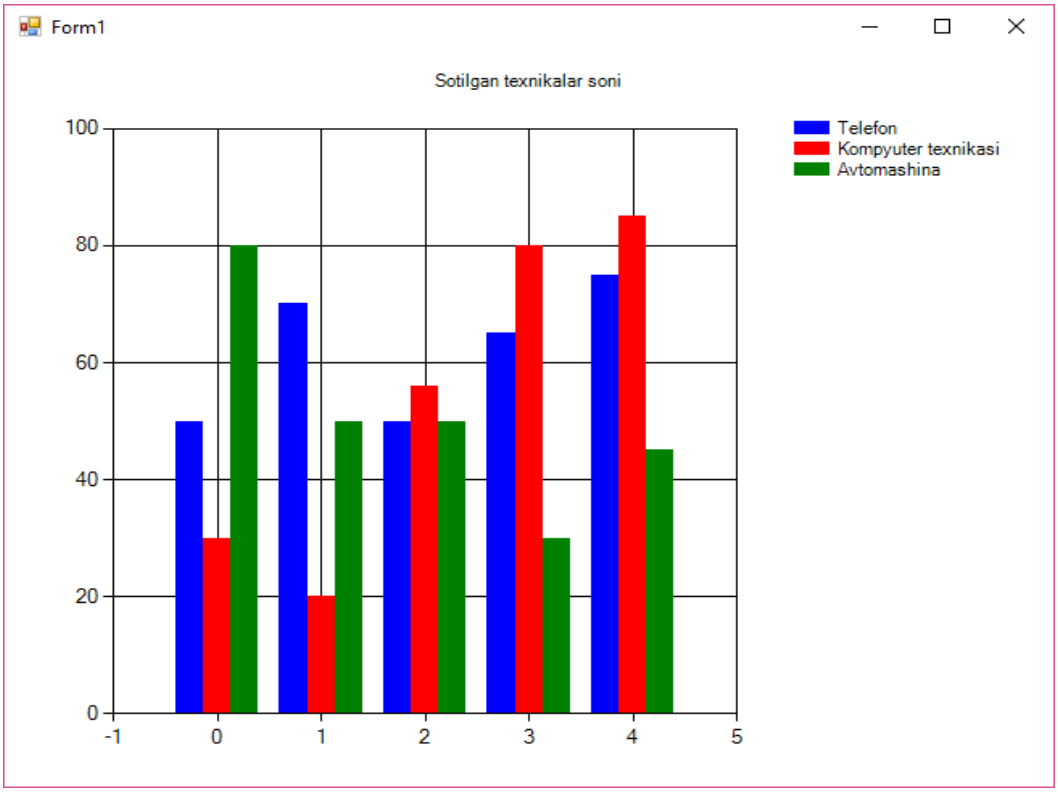


1.12-rasm. Funksiyalarning grafigi.

Gistagramma grafiklarini chizish uchun massivlardan yoki ixtiyoriy to‘plamlardan foydalanish mumkin. Buning uchun 3 ta massiv olamiz, Form1_Load hodisasiga quyidagicha algoritmni yozamiz

```
chart1->Series->Clear();
       Title^ title = gcnew Title("Sotilgan texnikalar soni ");
       chart1->Titles->Add(title);
    Series^ series1 = gcnew Series(L"Telefon");
       Series^ series2 = gcnew Series(L"Kompyuter texnikasi");
       Series^ series3 = gcnew Series(L"Avtomashina");
    // rangni tanlash
    series1->Color = Color::Blue;
       series2->Color = Color::Red;
       series3->Color = Color::Green;
       // infografikani turini tanlash
       series1->ChartType = SeriesChartType::Column;
       series2->ChartType = SeriesChartType::Column;
       series3->ChartType = SeriesChartType::Column;
       // qiymatlar qatlamini qo‘shish
    chart1->Series->Add(series1);
    chart1->Series->Add(series2);
    chart1->Series->Add(series3);
    // qiymatlarni
       array<int>^ arr1 = {30,20,56,80,85};
       array<int>^ arr2 = {50,70,50,65,75};
        array<int>^ arr3 = {80,50,50,30,45};
for (int i = 0; i < arr1->Length; i++) {
             series3->Points->AddXY(i,  arr3[i]);
             series1->Points->AddXY(i, arr2->GetValue(i));
             series2->Points->AddXY(i,  arr1->GetValue(i));
    }
```

Agar dasturni ishlatsak, Gistagramma grafigini chizish imkoniyatini beradi.



1.13-rasm. Gistagramma grafiklarini chizish

Grafiklarini alohida sohasha chizish uchun massivlardan yoki ixtiyoriy to‘plamlardan foydalanish mumkin. Buning uchun 3 ta massiv olamiz ,

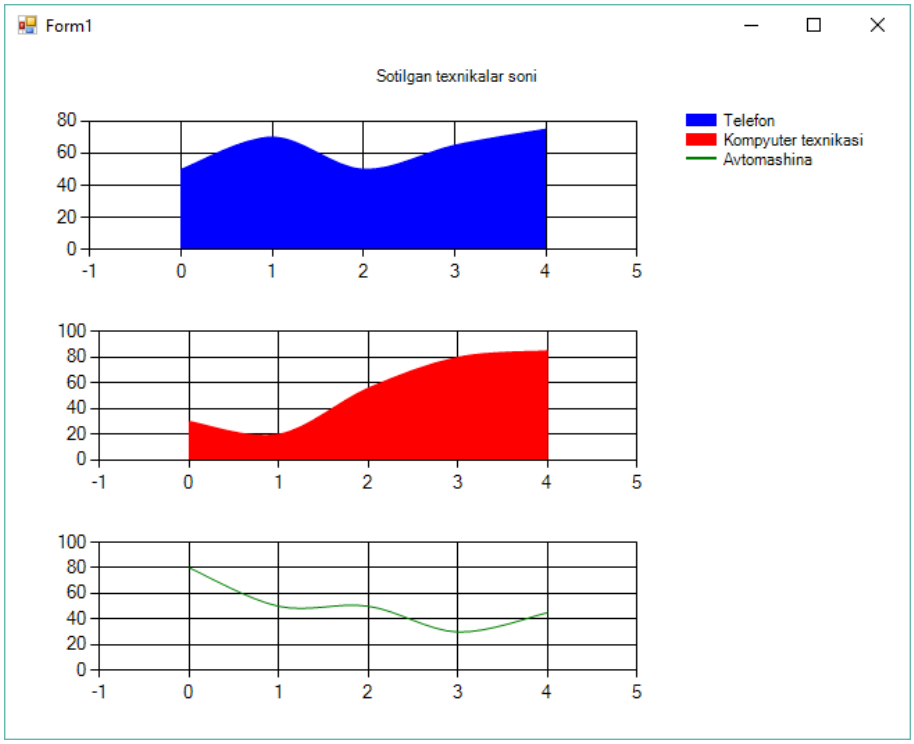
Form1_Load hodisasiga quyidagicha algoritmni yozamiz

```
chart1->Series->Clear();
ChartArea^ chartArea1 = gcnew
    ChartArea(); ChartArea^
    chartArea2 = gcnew ChartArea();
    ChartArea^ chartArea3 = gcnew
    ChartArea(); chartArea1->Name =
    "1";
    chartArea2->Name = "2";
    chartArea3->Name = "3";
    this->chart1->ChartAreas-
>Add(chartArea1); this->chart1-
>ChartAreas->Add(chartArea2);
```

```
    this->chart1->ChartAreas->Add(chartArea3);
    Title^ title = gcnew Title("Sotilgan texnikalar soni
    "); chart1->Titles->Add(title);
    Series^ series1 = gcnew Series(L"Telefon");
    Series^ series2 = gcnew Series(L"Kompyuter
    texnikasi"); Series^ series3 = gcnew
    Series(L"Avtomashina");
// rangni tanlash
series1->Color =
    Color::Blue; series2-
    >Color = Color::Red;
    series3->Color =
    Color::Green;
    // infografikani turini tanlash
    series1->ChartType =
    SeriesChartType::SplineArea; series2-
    >ChartType =
    SeriesChartType::SplineRange; series3-
    >ChartType = SeriesChartType::Spline;
    series1->ChartArea = chartArea1->Name;
    series2->ChartArea =
    chartArea2->Name; series3-
    >ChartArea = chartArea3-
    >Name;
    // qiymatlar qatlamini
    qo‘shish chart1-
    >Series->Add(series1);
chart1->Series-
>Add(series2);
chart1->Series-
>Add(series3);
// qiymatlarni
```

```
array<int>^ arr1 =
{30,20,56,80,85};
array<int>^ arr2 =
{50,70,50,65,75};
array<int>^ arr3 =
{80,50,50,30,45};

for (int i = 0; i < arr1->Length; i++)
        {  series3->Points->AddXY(
        i, arr3[i]);

        series1->Points->AddXY( i, arr2-
        >GetValue(i)); series2->Points-
        >AddXY( i, arr1->GetValue(i));
}
```

Dastur fragmentini tahlil qilish orqali yangi infografika sohalarni qo‘shish va ularni qiymatlar o‘plami bilan bog‘lashni ko‘rishinig mumkin. Agar dasturni ishlatsak, Gistagramma grafigini chizish imkoniyatini beradi.



1.14-rasm. Bitta Chart ga bir nechta sohani joylashtirish.

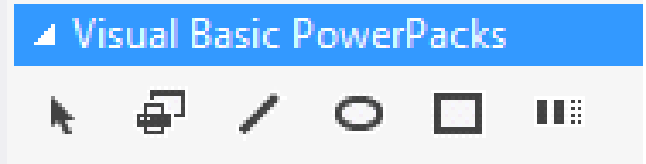
Chart komponentasi yordamida foydlanuvchining ixtiyoriy ma‘lumotlarini infografikasini yaratish mumkin. Bu juda katta imkoniyatli komponenta hisoblanadi. Buni xammasini o‘rganish uchun ko‘proq amaliyot qilish lozim. Yuqorida uning ba‘zi

imkoniyatlarini keltirgan holda komponentaning xususiyatlari va hodisalari bo‘yicha nazariy va amaliy ma‘lumotlarni keltirdik.

Visual Basic Power Packs komponentalari va ularni ishlatish. Bu komponentalar guruhi alohida bo‘lib, ularni Microsoft Visual Basic Power Packs deb atashadi. Buni yangi varintlarini ham internetdan olish va o‘rnatish mumkin.

Asosiy vazifasi formani bezash uchun ishlatiladi, shuning uchun barcha komponentalarga bor bo‘lgan hodisalar mavjud.

Bu Microsoft.VisualBasic PowerPacks nomlar fazosida joylashgan bo‘lib, Visual Basic Power Packs elementlar uchun sinflari mavjud. Visual Basic Power to‘plamlar elementlari qo‘shimcha Windows formalari elementlari uchun mo‘ljallangan. Ular dastlab bepu l plagin-smaylik sifatida ishlatilgan va endi Visual Studio tarkibiga kiritilgan.



1.15-rasm. Visual Basic Power Packs komponentalari.

1.3-jadval. Visual Basic Power Packs komponentalari

Rasmi	Komponenta nomi	vazifasi
	PrintForm	Formani chop qilishga ruxsat berishni boshqaradi
	LineShape	Gorozantal, vertikal, diognal chiziqlarni boshqarishni ta‘minlaydi
	OvalShape	Oval ko‘rinishdagi ob‘yektlarni boshqarishni ta‘minlaydi
	RectangleShape	To‘rtburchak ko‘rinishdagi ob‘yektlarni boshqarishni ta‘minlaydi
	DataRepeater	Talab asosida formatlangan ma‘lumotlarni ko‘rsatadi

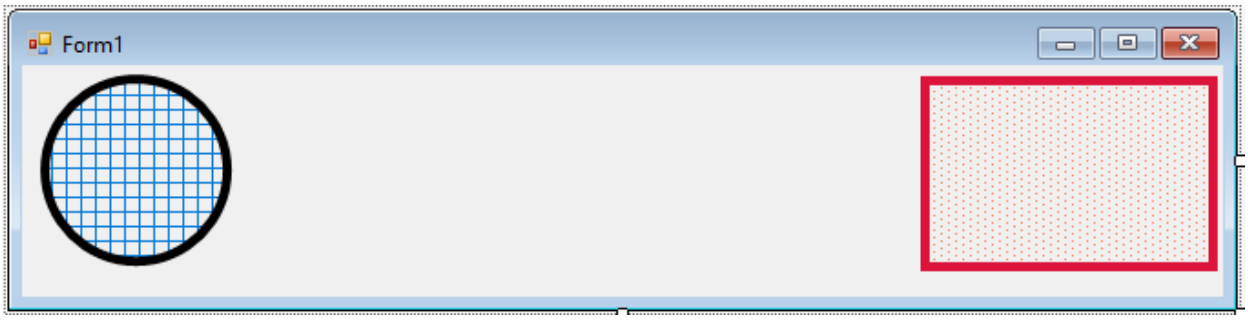
OvalShape komponentasining ba‘zi xususiyatlari va hodisalari

1. BackColor – komponentaning orqa rangini o‘rnatadi va qiymatlari ranglar to‘plami.
2. BackStyle - komponentaning orqa stilini o‘rnatadi va qiymatlari 2 ta statik berilgan.
3. BorderColor - komponentaning chegara rangini o‘rnatadi va qiymatlari ranglar

to‘plami.

4. BorderStyle - komponentaning chegara stilini o‘rnatadi va qiymatlari 6 ta statik berilgan.
5. BorderWight - komponentaning chegara rang qalinligini o‘rnatadi va qiymatlari sonlar.
6. FillColor - komponentaning aktiv bo‘lgandagi rangni o‘rnatadi va qiymatlari ranglar.
7. Location – komponentaning joylashuv o‘rnini o‘rnatish..
8. SelectionColor - komponentaning tanlangan bo‘lgandagi rangni o‘rnatadi va qiymatlari ranglar.

Bir loyiha yarating va unda 1.16-rasm kabi formani tayyorlang.



1.16-rasm. Loyiha ko‘rinishi.

Yaratilgan ovalShape1 ni forma bo‘yicha xarakatlanishni masalasini ko‘ramiz. Sizning formadagi 2 ob‘yektni yaratishda ularning xususiyatlari bilan tanishib olgansiz.

Buning uchun loyihaga bir Timer1 ob‘yektidan joylashtiramiz. Tez bajarilishi uchun uning interval xususiyatiga 1 qiymatni kiritamiz. Timer1 ob‘yektining ustiga sichqonchani ikki marta bosib, timer1_Tick hodisasiga quyidagicha algoritm kiritamiz.

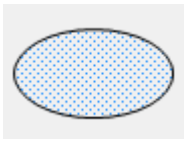
```
if(ovalpos <= (this->Size.Width - ovalShape1->Size.Width)){
  ovalpos += 5;

  ovalShape1->Location = Point(ovalpos,ovalShape1->Location.Y);
        }
```

Bunda ovalpos qiymati 0ga teng bo‘lgan butun son tipidagi o‘zgaruvchi. Location – bu ob‘yektning joylashinini belgilaydi va juft qiymatga ega Point qiymatni qabul qiladi. Shuningdek, Form1_Load hodisasida quyidagi algoritmni yozamiz.

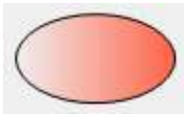
```
ovalpos = 0;
timer1->Start();
```

Agar dasturni ishga tushirsangiz ovalShape1 ob‘yektining harakatini ko‘rasiz.

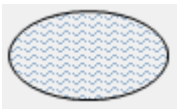
Shakllarning FillColor, FillGradientColor, FillGradientStyle, FillStyle xususiyatlaridan foydalanib, turli xil tugmalarni ham yaratish mumkin.



FillColor	MenuHighlight
FillGradientColor	Maroon
FillGradientStyle	ForwardDiagonal
FillStyle	Percent20



FillColor	Control
FillGradientColor	Tomato
FillGradientStyle	Horizontal
FillStyle	Solid



FillColor	ActiveCaption
FillGradientColor	White
FillGradientStyle	Horizontal
FillStyle	Wave

FillColor	ActiveCaption
FillGradientColor	OrangeRed
FillGradientStyle	Horizontal
FillStyle	SolidDiamond

Berilgan button tugmani bezash masalasini qaraymiz. Buning uchun formaga bitta tugma joylashtiramiz va uning xususiyatlarini quyidagicha aniqlaymiz.

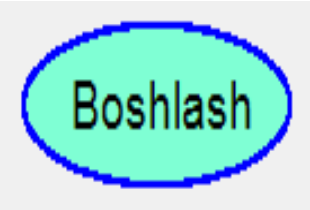
```
using namespace System::Drawing::Drawing2D;
using namespace System::Drawing::Text;

// ...
this->button1->Location = System::Drawing::Point(115, 438);
this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(121, 49);
this->button1->TabIndex = 0;

this->button1->Text = L"button1";
```

Tugmaning button1_Paint hodisasiga o‘tib, quyidagi algoritimni yozamiz.

```
Pen^ pen = gcnew Pen(Color::Blue,10);

Brush^ brush = gcnew SolidBrush(Color::FromKnownColor (KnownColor::Control));
SolidBrush ^ brushinside = gcnew SolidBrush(Color:: Aquamarine);

Graphics ^gr = e->Graphics;
```

```
gr->FillRectangle(brush, 0, 0, button1->Width,button1->Height);

gr->FillEllipse(brushinside, 0, 0, button1->Width, button1->Height);

gr->DrawEllipse(pen, 0, 0, button1->Width, button1->Height);
GraphicsPath^ path = gcnew GraphicsPath();

button1->Region = gcnew System::Drawing::Region(path);
String^ text = "Boshlash";

System::Drawing::Font^ drawfont = gcnew System::Drawing::Font("Arial", 14);
SolidBrush^ exbrush = gcnew SolidBrush(Color::Black);

RectangleF rect = RectangleF(button1->Width/2-40,button1->Height/2-10,button1->Width,
button1->Height);

gr->DrawString(text, drawfont, exbrush,rect);
```

Bunda chizish uchun qalam - Pen , mo‘yqalam - Brush, to‘liq bo‘yash uchun mo‘yqalam - SolidBrush yaratib olamiz. Kerali shakllarni chizish ob‘yektinini tugmaning o‘lchamiga moslab tayyorlaymiz. Tugmaning ustiga matn joylagtirish uchun yana bir ob‘yekt yaratamiz, unga matn, matn rangi, formati va egallash sohasini ko‘rsatib, DrawString bilan tugmaning ustiga chizamiz.



1.17-rasm. Tugmani bezashga misol.

Yangi yaratilgan tugmachalar orqali shakllarni oqim orqali boshqarish masalasini qaraymiz. Buning uchun Loyiha formasiga, 1 ta PictureBox, 2 ta button tugmani yuqoridagidek qilib joylashtiramiz. PictureBox ning BackColor xususiyatiga oq rangni o‘rnatimiz. Shaklning xarakatlantirish uchun uning tayanch nuchtasini ko‘chirib, oldingisi oq bilan bo‘yaladi va keyingisi hosil qilinadi, shunday qilib iteratsiya davom etaveradi.

Ikkita berilgan tugmalarni yuqoridagidek qilib, moslab olamiz.

Oqimni boshqarish uchun avval bir funksiya yaratib olamiz. Bu funksiya chizilgan shaklni PictureBox bo‘ylab harakatlanish va oldin keltirilgan iteratsiyani amalga oshirish uchun kerak. Funksiyaning

```
public: void Go(){
	int R = 30;

//	 button2->BeginInvoke(gcnew  InvokeDelegateSetEnabled(this,
&Form1::setEnabledToButton),  false);

	Bitmap ^ image = gcnew Bitmap(pictureBox1->Width, pictureBox1->Height);
	Graphics^ g = Graphics::FromImage(image);
```

```
        for (int x = 0; x <= pictureBox1->Width - 2 * R; x += 2){
                                g->Clear(Color::White);
                                g->FillEllipse(Brushes::Aqua, x, 0, 2 * R, 2 * R);
                                pictureBox1->BeginInvoke(gcnew InvokeDelegateSetImage(this,
&Form1::setImageToPictureBox),  image);
                                Thread::Sleep(10);
                        }
        for (int y = 0; y <= pictureBox1->Height - 2 * R; y += 2){
                                g->Clear(Color::White);
                                g->FillEllipse(Brushes::Aqua, pictureBox1->Width - 2 * R, y, 2
* R, 2 * R);
                                pictureBox1->BeginInvoke(gcnew InvokeDelegateSetImage(this,
&Form1::setImageToPictureBox), image);
                                Thread::Sleep(10);
                        }
        for (int x = pictureBox1->Width - 2 * R; x >= 0; x -= 2){
                                g->Clear(Color::White);
                                g->FillEllipse(Brushes::Aqua, x, pictureBox1->Height - 2 * R,
2 * R, 2 * R);
                                pictureBox1->BeginInvoke(gcnew InvokeDelegateSetImage(this,
&Form1::setImageToPictureBox), image);
                                Thread::Sleep(10);
                        }
        for (int y = pictureBox1->Height - 2 * R; y >= 0; y -= 2)                {
                                g->Clear(Color::White);
                                g->FillEllipse(Brushes::Aqua, 0, y, 2 * R, 2 * R);
                                pictureBox1->BeginInvoke(gcnew InvokeDelegateSetImage(this,
&Form1::setImageToPictureBox),  image);
```

algoritmi quyidagicha:

Bu funksiyani ishlatish uchun qiymatni to‘g‘ridan to‘g‘ri berib bo‘lmaydi shuning uchun bir deligant (funksiya ko‘rsatkich saqlovchi) va qiymatni funksiya yaratamiz.

```
delegate void InvokeDelegateSetImage(Bitmap^ image);
public: void setImageToPictureBox(Bitmap^ image){
                                        pictureBox1->Image = image;
                                }
```

Formaning ochiq joyiga oqim yaratib olamiz.

```
        using namespace System::Threading;
// ...
        private: Thread^ myThread;
        private: Thread^ myThreadOne;
```

Birichi tugma bosilganda mazkur oqimlarni ishga tushurish uchun button1_Click hodisasiga quyidagicha algoritm yoziladi.

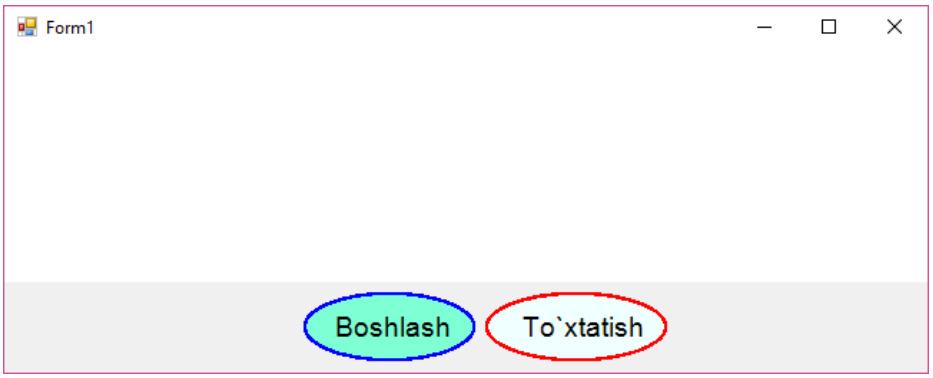
```
  myThread = gcnew Thread(gcnew ThreadStart(this,&Form1::Go));
  myThread->Start();

  myThreadOne = gcnew Thread(gcnew ThreadStart(this,&Form1::GoOne));
  myThreadOne->Start();
```

Oqimni saqlab tuurish uchun yaratilgan funksiyada Thread::Sleep(10); dan foydalanilgan.

Oqim ishga tushgan paytdan ishlaydi, ammo forma yopilganda uning ishini to‘xtatish lozim.Shuning uchun oqimni to‘xtashish uchun button2_Click va Form1_FormClosing hodisalariga quyidagi algoritmni kiritamiz.

```
try{
  myThread->Abort();
  myThreadOne->Abort();

     }
```

Loyihani ishga tushirsangiz, quyidagi 1.18- rasm hosil bo‘ladi.



1.18- rasm. Oqimlar asosida xarakatni boshqarish

Dasturlash boshlash tugmasini bosganda ikkita paralel bo‘lgan shakllarining harakatini ko‘rasiz. To‘xtash bosilganda ularni to‘xtashini ko‘rasiz. Xarakat davom etatgan vaqtda ham formani yopiq ishni tugallash mumkin.

Umuman olgada grafika bilan ishlash uchun geometriyani yaxshi bilash talab qilinadi. Visual C++ da grafika sinfi, uning usullari va turli shakllarni chizish usullari, Chart va Shape sinflarining xususiyatlari va usullarini кщrib chiqdik. Bularni щziga xos ma‘lumotlarda ishlatishni tavsiya qilamiz.

Nazorat savollari

1. grafika sahifa koordinatalarini uchun ishlatiladigan o‘lchov birligi sozlash xususiyatini ayting.
2. GraphicsUnit qiymatlari sanab bering.
3. - butun chizma yuzasini tozalaydi va belgilangan fon rangi bilan to‘ldiradigan usulning nomini ayting.
4. Bu funksiya berilgan parametrlar asosida tasvirni belgilangan joyda asl kattaligidan foydalanib chizadi. Funksiya nomini ayting.
5. Chiziq chizishni qanday amalga oshiriladi.
6. Uchburchak chizishni qanday amalga oshiriladi
7. Ellipis chizishni qanday amalga oshiriladi
8. Turli shakllarni bo‘yyashni qanday amalga oshiriladi
9. Ko‘p burchaklarni chizishni qanday amalga oshiriladi
10. Hodisalar orqali shakl chizishni qanday amalga oshiriladi
11. Rasmlarni o‘zgartirishni qanday amalga oshiriladi
12. Matnlarni tasvir kabi joylashtirishni qanday amalga oshiriladi
13. Matnli rasmlarni yaratishni qanday amalga oshiriladi
14. Chart komponenta xususiyati va hodisalari haqida gapirib bering.
15. BorderSkin nima.
16. Qaysi xususiyat yordamida komponentaning infografikani ko‘rsatadigan shaklini tanlash mumkin
17. Qaysi xususiyat asosiy bo‘lib, infografikaning qiymatlarini belgilovchi, har bir qiymat tegishliligini bildiradi
18. ChartType infografikaning turlarini belgilash uchun xizmat qiladi. Uning qiymatlarini bilasizmi.
19. Bir infografikada nechtagacha gistogramma joylashtirish mumkin.
20. Chart komponentasi yordamida foydlanuvchining ixtiyoriy ma‘lumotlarini infografikasini yaratish mumkinmi
21. Nimalar dastlab bepul plagin-smaylik sifatida ishlatilgan va endi Visual

Studio tarkibiga kiritilgan

22. Visual Basic Power Packs komponentalarini sanab bering.
23. Visual Basic Power Packs komponentalarining asosiy vazifalari nimadan iborat.
24. Komponentaning chegara rang qalinligini o‘rnatadi va qiymatlari sonlardan iborat bo‘lgan xususiyat nomini ayting.
25. Shakllarning FillColor, FillGradientColor, FillGradientStyle, FillStyle xususiyatlaridan foydalanib, niam qilish mumkin.
26. Tugmning ko‘rinishini o‘zgartirish uchun nima amallar bajario‘ kerak.
27. Rasmlarni harakatlantirishda oqimlardan qanday foydalanamiz.

15- MA’RUZA
MAVZU: KICHIK LOYIHALARNI YARATISH (2 SOAT)

Reja:

1. **Visual C++ muhitida turli sohaga oid masalalarni yechish uchun kichik loyihalarni loyihalash usullari.**
2. **Kichik loyihalar algorimlarini yozish usullari.**
3. **Visual C++ muhitida hisobot shaklini tayyorlash va chop etish elementlari.**
4. **Foydalanuvchi interfeysini loyihalash va testlash usullari.**

Annotatsiya: *Visual C++ muhitida turli sohaga oid masalalarni yechish uchun kichik loyihalarni loyihalash usullari asosan axborot tizimlarni loyihalashtirish usullari nazarda tutiladi.*

Shuning uchun axborot tizimining ob‘yektlari va usullari, algoritmlariga, qo‘llanish sohasi va chegaralariga qarab loyihani katta, kichik loyihalar deyish mumkin. Shuning uchun kichik loyiha deganda funksional imkoniyatlari sanoqli bo‘lgan axborot tizimlari tushuniladi.

Kalit so‘zlar: Operatsion tizim, amaliy dastur, kichik loyiha, axborot tizimi, IDEF,

DFD, Hayot sikli, algoritm, EPS metodologiyasi, BRMI metodologiyaisi, hisobot, hisobot paneli.

KIRISH

Visual C++ muhitida turli sohaga oid masalalarni yechish uchun kichik loyihalarni loyihalash usullari asosan axborot tizimlarni loyihalashtirish usullari nazarda tutiladi. Shuning uchun axborot tizimining ob‘yektlari va usullari, algoritmlariga, qo‘llanish sohasi va chegaralariga qarab loyihani katta, kichik loyihalar deyish mumkin. Shuning uchun kichik loyiha deganda funksional imkoniyatlari sanoqli bo‘lgan axborot tizimlari tushuniladi.

Axborot tizimlarining loyihalashtirish uchun axborot tizimining hayot davrini yaxshi bilish va tushunish kerak. Chunki axborot tizimini loyhalashda uchun hayot davri muhim ahamiyatga ega.

Axborot tizimining hayot davri. Hayot davri (HD) tushunchasi axborot tizimlarini loyihalash metodologiyasining asosiy tushunchalaridan biridir. Axborot tizimining hayotiy dari - bu axborot tizimini yaratish fikridan boshlanib, u butunlay foydalnishdan chiqqan paytda tugaydigan uzluksiz jarayondir. Hayot davrining tuzilishini belgilovchi asosiy standart GOST ISO/IEC 12207-02 hisoblanadi. Standartga ko‘ra, hayot davri strukturasi uchta jarayon guruhiga asoslangan:

1. **Asosiy** (buyurtma, ishlab chiqish, etkazib berish, sinov, texnik xizmat ko‘rsatish);
2. **Yordamchi** (asosiy jarayonlarning bajarilishini ta‘minlash uchun):

- Hujjatlar uchun - barcha manfaatdor shaxslar tomonidan zarur bo‘lgan hujjatlarni ishlab chiqish, tahrirlash, tarqatish va saqlash bo‘yicha ishlar;
- konfiguratsiyani boshqarish (configuration management) quyidagilarni o‘z ichiga oladi: tizimdagi dasturiy ob‘yektlarning holatini aniqlash va o‘rnatish; ob‘yektlarning o‘zgarishi va chiqarilishini boshqarish; ob‘yektlarning to‘liqligi, mosligi va to‘g‘riligini ta‘minlash; ob‘yektlarni saqlash, tashish va etkazib berishni boshqarish;
- sifatni ta‘minlash – yaratilayotgan tizim va amalga oshirilayotgan hayot davri jarayonlarining belgilangan talablar va tasdiqlangan rejalarga javob berishini ta‘minlash ishlari;
- tekshirish - loyiha amalga oshirilayotganda yaratilgan muvaqqat natijalarning belgilangan talablarga javob berishini tekshirish uchun tegishli shaxs (buyurtmachi, etkazib beruvchi yoki mustaqil tomon) ning ishi. Shartnoma, jarayon, talablar, loyiha tizimi, qurilish tizimi va hujjatlarni tekshirishni ajrata;
- sertifikatlash - alablarning va yakuniy mahsulotning tizimning funksional maqsadiga to‘liq mosligini tekshirish uchun tegishli shaxsning ishi;
- qo‘shma tahlil – har qanday ish (tizim) holatini yoki natijalarini baholash uchun ish);
- audit - mustaqil (loyihaga nisbatan) ekspertlarning korxona faoliyatining qabul qilingan

talablar, rejalar va shartnoma shartlariga muvofiqligini aniqlash ishlari;

- muammoni hal qilish - loyihani amalga oshirish vaqtida topilgan muammolarni tahlil qilish va tuzatish ishlari;

3. Tashkiliy**:**

- loyiha boshqaruvi - rejalashtirish va jarayonlarni boshqarish, shu jumladan monitoring, tekshirish va hisobot bilan tugallangan ishlarni baholash;

-loyiha infratuzilmasini yaratish - boshqa har qanday jarayon uchun zarur bo‘lgan infratuzilmani tashkil etish va ta‘minlash uchun ishlaydi. Infrastrukturada texnik va dasturiy vositalar, vositalar, usullar, standartlar va tizimni ishlab chiqish, ishlatish yoki xizmat ko‘rsatish shartlari bo‘lishi mumkin;

- yaxshilash - hayot davri jarayonlarini baholash, kuzatish va yaxshilash uchun ishlash;
- o‘quv-kadrlar tayyorlashni rejalashtirish va o‘tkazish, shu jumladan o‘quv materiallarini ishlab chiqish. Shu bilan birga, xodimlar nafaqat tizimni boshqaradigan oxirgi foydalanuvchilarga, balki tizim ishlab chiquvchilariga ham tegishlidir. Masalan, ishlab chiquvchilar tashkilotda qabul qilingan texnologiyalar va dasturiy vositalar bo‘yicha o‘qitilishi, hatto oxirgi foydalanuvchilarni tizim bilan ishlashni to‘g‘ri amalga oshirish va o‘qitishga o‘rgatilishi kerak. Qanchalik paradoksal bo‘lmasin, lekin o‘qitishning to‘g‘ri usul va metodlarini o‘rgatish ham kerak.

AT yaratish uchun asosiy mezon quyidagilarni o‘z ichiga oladi:

- tizimning funksional va tashkiliy tuzilmalari;
- texnik va dasturiy vositalar majmuasining tarkibi va tuzilishi;
- ishlatiladigan uskunalar;
- axborotni qayta ishlash texnologiyalari;
- axborot bazasini yuritish tarkibi, tuzilishi va texnologiyasi;
- KIRISH va chiqish formalari;
- ma‘lumotlarni qayta ishlash algoritmlari.

Axborot tizimni ishlab chiqishdan oldin mijoz va ishlab chiquvchi tizimda qanday funksionallik o‘rnatilishini va tizim ichida qanday funksional o‘zaro munosabatlar tashkil etilishini aniq tushunishi kerak.

Funksional modelni ishlab chiqishda (funksional talablarni belgilashda) ko‘pgina muammolar paydo bo‘lishi mumkin:

- mijoz axborot tizimi oldiga qanday vazifalar qo‘yilganini aniq ifoda eta olmaydi.

Ko‘pincha mijoz ham talab nima va uni shakllantirish qanday bilmaydi;

- buyurtmachi vakillari (turli darajadagi rahbarlar, texnologiya mutaxassislari, oddiy foydalanuvchilar) kelajak tizimi haqida o‘z qarashlariga ega va ko‘pincha ularning tizimga bo‘lgan talablari o‘zaro farq qiladi. Ushbu holat, ayniqsa, ishlab chiqilayotgan tizim bir nechta avtomatlashtirish ob‘yektlarida amalga oshirilganda odatiy holdir;
- mijoz ko‘pincha zamonaviy hisoblash tizimlarining imkoniyatlarini bilmaydi va avtomatlashtirish jarayonini qo‘lda bajariladigan elementar faoliyatni kompyuterlarga oddiy uzatish deb hisoblashga intiladi. Shu bilan birga, ular yangi texnologiyalar kelishi bilan tashkilot ichidagi biznes-jarayonlarni optimallashtirish haqida o‘ylamaydilar;
- mijoz ba‘zi funksiyalarni virtual mashinalar bajarishi mumkinligiga ishonmaydi.

Funksional model qurish, bu muammolarni eng avval hal qilish kerak. Bu jarayon batafsil ko‘rib chiqamiz.

Uni ishlab chiqishda ishni mavjud tashkil etishning modeli avvalo ish tavsiflari, farmoyishlari, hisobotlari, me‘yoriy hujjatlari va boshqalar asosida quriladi. Modelni tahlil qilish zaif tomonlar qaerda, yangi jarayonlarning afzalliklari qanday bo‘lishini va korxona (kompaniya, bo‘lim) ning mavjud tashkilotiga qanday chuqur o‘zgarishlar kiritilishini tushunishga imkon berishi lozim. Faoliyatni samarasiz tashkil etish belgilari bo‘lishi mumkin:

- foydasiz, nazoratdan tashqari va duplikativ ishlar;
- natijasiz ishlar;
- samarasiz hujjat oqimi (to‘g‘ri hujjat o‘z vaqtida to‘g‘ri joyda emas) va boshqalar.

Modelda topilgan kamchiliklar modelini yaratishda tuzatilib, korxonaning yangi tashkiloti modeli taklif qilinishi ham mumkin. Bu model muammoni hal qilishning muqobil yo‘llarini tahlil qilish va eng yaxshisini tanlash uchun ishlatiladi.

SADT metodologiyasi (Structured Analysis and Design Technique - strukturali tahlil va loyihalash texnikasi) - tizimning funksional modelini qurish uchun mo‘ljallangan usullar, qoidalar va protseduralar majmui.

Bu metodikani ishlab chiqishni XX asrning 60-yillari o‘rtalarida Duglas Ross (Aqsh) boshlab bergan. Keyin, Softech, Inc tahlilchilar tomonidan SADT yaxshilandi va muammolar keng ko‘lamli hal qilish uchun ishlatiladi. Telefon tarmog‘i dasturiy ta‘minot, diagnostika, uzoq muddatli va strategik rejalashtirish, kompyuter yordam ishlab chiqarish va loyihalash, kompyuter tizimi konfiguratsiya, kadrlar tayyorlash, va moliyaviy va logistika boshqarish SADT samarali ilovalar yaratila boshlandi. Keng sohalar SADT metodikasining ko‘p qirraliligi va qudratini ko‘rsatadi.

Amerika qo‘shma Shtatlari Mudofaa vazirligining integratsiyalashgan kompyuter quvvat ishlab chiqarish (ICAM) dasturi SADTning foydaliligini tan oldi. Bu 1981-yilda uning bir qismini "IDEF0" (Integrated Computer Aided Manufacturing, ICAM) deb nomlab, dasturiy ta‘minotni ishlab chiqish bo‘yicha Federal standart sifatida chop etishga olib keldi. Bu nom ostida SADTdan harbiy va sanoat tashkilotlarida minglab mutaxassislar foydalanishgan. Milliy standartlar va texnologiyalar institutida IDEF0 standartining so‘nggi versiyasi 1993-yilning dekabrida chiqarildi.

Shuni ta‘kidlash kerakki, IDEF0 Rossiya va Respublikamiz Gosstandard tomonidan foydalanish uchun tavsiya etildi va milliy davlat idoralarida faol qo‘llaniladi.

Ushbu metodologiya axborot tizimining funksional jihatini tavsiflashda ma‘lumotlar oqimiga yo‘naltirilgan usullar (DFD) bilan raqobatlashadi. Aksincha, IDEF0 imkoniyatlari ko‘yidagilardan beradi:

har qanday tizimni tavsiflansh, nafaqat axborot tizimlari (dfd dasturiy ta‘minotni tasvirlash uchun mo‘ljallangan);

- unga qo‘yiladigan yakuniy talablarni belgilashdan oldin tizim va uning tashqi muhit tavsifini yaratish. Boshqa so‘zlar bilan aytganda, bu metodologiyasi yordamida, siz asta-sekin qurish va uning bajarilishini tasavvur qilish qiyin bo‘lsa ham, tizim tahlil qilish mumkin.

Shunday qilib, IDEF0 keng doiradagi tizimlarni yaratishning dastlabki bosqichlarida foydalanish mumkin. Shu bilan birga mavjud tizimlarning vazifalarini tahlil qilish va ularni takomillashtirish echimlarini ishlab chiqish uchun foydalanish mumkin.

IDEF0 metodikasi grafik jarayonni tasvirlash tiliga asoslangan. IDEF0 notatsiyadagi model iyerarxik tartibli va o‘zaro bog‘langan diagrammalar to‘plamidir. Har bir diagramma tizim tavsifining birligi bo‘lib, alohida varaqda joylashgan.

Model (AS-IS, TO-BE yoki SHOULD-BE) diagrammalar 4 turni o‘z ichiga olishi mumkin

- kontekst diagrammasi;
- parchalanganda diagrams;
- tugun daraxt diagrammalari;
- faqat EHM uchun grafiklar (FEO).

Diagrammalar daraxt tuzilishining yuqorisi bo‘lmish kontekst diagramma (top-level diagramma) tizimning maqsadi (asosiy vazifasi) va uning tashqi muhit bilan o‘zaro ta‘sirini ko‘rsatadi. Har bir model faqat bitta kontekst grafikka ega bo‘lishi mumkin. Asosiy funksiyani

tavsiflagandan so‘ng funksional dekompozitsiya amalga oshiriladi, ya‘ni asosiy funksiyani tashkil etuvchi funksiyalar aniqlanadi.

Bundan tashkari, o‘rganilayotgan tizimning kerakli darajada detallashtirilgunga qadar funksiyalar subfunksiyalarga bo‘linadi va hokazo. Sistemaning har bir shunday fragmentini tasvirlovchi diagrammalar dekompozitsion diagrammalar deyiladi. Har bir dekompozitsiya seansidan so‘ng ekspert tekshiruv seanslari o‘tkaziladi – domen ekspertlari real jarayonlar yaratilgan diagrammlarga mos kelishini ko‘rsatadilar. Topilgan har qanday nomuvofiqliklar bartaraf etiladi, so‘ngra jarayonlarni yanada batafsil davom etiradi.

Tugun daraxti diagrammasida funksiyalar (ish o‘rinlari) ning iyerarxik munosabatlari ko‘rsatilgan, lekin ular o‘rtasidagi munosabatlar emas. Ulardan bir nechtasi bo‘lishi mumkin, chunki daraxt har qanday chuqurlikka va har qanday tugundan qurilishi mumkin.

EHM diagrammalari tizimda yuz beradigan jarayonlarning muqobil ko‘rinishini ko‘rsatish uchun modelning alohida qismlarini ko‘rsatish uchun mo‘ljallangan (masalan, tashkilot boshqaruvi nuqtai nazaridan).

Kichik loyihalar algorimlarini yozish usullari. Tizim vazifalarini belgilab va axborot asoslarini ishlab chiqqandan so‘ng, keyingi qadam tizimida axborot oqimi harakati loyihalashtirish iborat. Tizimning modeli kerakli funksiyalarga qanday erishish va uni ta‘minlash uchun qanday ma‘lumotlar ishlatilishini ko‘rsatadi. Shunday qilib, model bevosita tizimning funksional va axborot modellariga asoslanadi.

Bu modellar odatda IDEF0 va DFD dekompozitsiya diagrammalarining oxirgi darajalarida ko‘rsatilgan funksiyalar (jarayonlar) uchun quriladi. DFD uchun diagramma yoki diagramma ko‘rinishidagi modeli elementar jarayonlarni minispektlar ko‘rinishida tasvirlashdan ko‘ra yaxshiroq echimdir.

Bunday vaziyatlarda funksional imkoniyatni tasvirlashning eng eng taniqli va mashhur usullari va uslubiyati bor:

- algoritmlarning blok diagrammalari;
- Jadvalning EPC ;
- BPMN metodologiyasi.

Algoritmlar va BPMN diagrammalarining oqimlari an‘anaviy tizimlarni tizimli yondashuv tamoyillariga asoslangan holda modellashtirish uchun eng mos keladi va funksiyalarni elementar bosqichlarga (protseduralar, operatorlar, harakatlar va hokazolarga) aniq bo‘linishi bilan

tavsiflanadi.) ma‘lum ketma- ketlikda (algoritm bo‘yicha) bajariladi. EPC diagrammalar va BPMN diagrammalar bir harakat yoki ularning majmui ijrosi tizimida sodir bo‘lgan voqealar bog‘liq bo‘lgan voqeaga yo‘naltirilgan tizimlarini modellashtirish uchun yaxshi vosita hisoblanadi. Bundan tashqari, bu usul va uslubiyatlardan nafaqat xulq-atvorli modellashtirish uchun, balki funksional modellashtirish uchun ham foydalanish mumkin. Ularda ma‘lum amallarni (mantiqiy simvollarni) bajarish shartlarini ko‘rsatishga imkon beruvchi elementlarning mavjudligi tizim funksiyalarining mantiqi va ketma-ketligini yaxshiroq tushunishga imkon beradi.

Algoritmlarni tasvirlashda uzoq vaqt davomida flowcharts (Basic Flowchart) muvaffaqiyatli ishlatilgan. Algoritmlarning blok diagrammalarini qurish GOST 19.701-90 (ISO 5807-85) dasturiy hujjatlarning yagona tizimi bilan tartibga solinadi. Dasturlar, ma‘lumotlar va tizimlar uchun algoritmlarning sxemalari mavjud. Ushbu davlat standarti "ISO 5807-85" xalqaro standarti asosida tashkil etilgan. Axborotni qayta ishlash – ma‘lumotlar, dastur va tizim oquvchilari, dastur tarmoq jadvallari va tizim resurslari jadvallari uchun hujjat simvollari va konvensiyalari moslashtirilgan.

GOST 19.701-90 ga ko‘ra, diagramma-bu muammoni hal qilishning ta‘rifi, tahlili yoki usulining grafik tasviri. Tizimning statik va dinamik jihatlarini ko‘rsatish uchun diagrammalardan foydalanishingiz mumkin. Davlat standartida berilgan simvollardan quyidagi turdagi sxemalarda foydalanish mumkin:

- ma‘lumotlar sxemalari - ma‘lumotlarni qayta ishlash ketma-ketligini va ularning tashuvchilarini aniqlaydi;
- dastur diagrammalari - dasturdagi amallar ketma-ketligini ko‘rsatish (aslida bu an‘anaviy ma‘noda algoritmlarning asosiy xususiyati);
- tizimning ishlash sxemasida tizimdagi amallar va ma‘lumotlar oqimlari ko‘rsatiladi;
- dasturning o‘zaro ishlash diagrammalari – dasturlar (modullar) ning aktivlashtirish yo‘lini ko‘rsatish va ularning tegishli ma‘lumotlar bilan o‘zaro hamkorligi uchun;
- tizim resurs diagrammalari - ma‘lumotlar bloklari va qayta ishlash bloklari konfiguratsiyasini ko‘rsatish.

Yuqorida keltirilgan turdagi sxemalardan ko‘rinib turibdiki, ulardan nafaqat xulq-atvor ob‘yektini modellashtirish uchun, balki funksional, axborot va komponentli loyihalash vazifalari uchun ham foydalanish mumkin.

Tizimning funksional modelini qurishda strukturaviy yondashuvning asosiy tamoyillari - dekompozitsiya va iyerarxik algoritm tamoyillaridan foydalaniladi. Funksional modeli - bu turli

darajadagi detallarga ega bo‘lgan o‘zaro bog‘langan diagrammalar majmui bo‘lib, har bir yangi detal darajasi bilan tizim yanada to‘liqlashadi.

Diagrammalar grafik belgilashning quyidagi elementlarini o‘z ichiga olishi mumkin:

- data symbollar - ma‘lumotlarning mavjudligini, axborotlarning turini yoki ma‘lumotlarni kiritish va chiqarish usulini ko‘rsating;
- jarayon simvollari - ma‘lumotlar ustida bajariladigan amallar;
- chiziq symbols - jarayonlar va/yoki saqlash vositalari orasidagi ma‘lumotlar oqimlarini, shuningdek jarayonlar orasidagi nazorat oqimlarini ko‘rsatadi;
-maxsus belgilar - diagrammalarni yozish va o‘qishni osonlashtirish uchun ishlatiladi.

Semantik mazmunga ko‘ra bo‘linishdan tashqari har bir turkum simvollari (maxsus kategoriyalardan tashqari) asosiy va o‘ziga xos simvollarga bo‘linadi. Asosiy belgi jarayon yoki ma‘lumotlar tashuvchining aniq turi noma‘lum bo‘lganda yoki haqiqiy ma‘lumotlar tashuvchisi (jarayoni) ni ta‘riflashga hojat bo‘lmaganda qo‘llaniladi. Jarayon yoki ma‘lumotlar tashuvchining aniq turi ma‘lum bo‘lganda va diagrammada ko‘rsatilishi kerak bo‘lganda o‘ziga xos simvoldan foydalaniladi.

Hodisali jarayon zanjiri (EPC) - biznes-jarayonlarni modellashtirish, tahlil qilish va qayta tashkil etish (funksional modellashtirish) uchun ishlatiladigan diagramma turi. Shu bilan birga EPC diagrammalaridan funksiyalarni amalga oshirishda tizimning alohida qismlari funksionallikni modellashtirishda foydalanish va an‘anaviy oqimlarni almashtirish uchun xizmat qilish mumkin.

EPS metodi Avgust-Vilgelm Sheer tomonidan 1990-yillar boshida integrallashgan axborot tizimlari arxitekturasi metodikasiga oid ishining bir qismi sifatida ishlab chiqilgan.

EPC notatsiyasida jarayon (funksiya) diagrammasi hodisa va funksiyalarning tartibli birikmasidir. Har bir funksiya uchun unga hamroh bo‘lgan dastlabki va oxirgi hodisalar, ishtirokchilar, ijrochilar, moddiy va axborot oqimlari aniqlanishi hamda quyi darajalarga bo‘linishi mumkin.

DFD kabi, EPC metodologiyasi turli notatsiyalarni (elementlarning sintaksisi va semantikasini) qo‘llab talqin qilish bilan rivojlanmoqda. EPC (ARIS, Microsoft Visio, Business Studio, Bflow) yordamida biznes-jarayonlarni modellashtirish qobiliyatini amalga oshirgan metodika muallifi ham, dasturiy ta‘minot ishlab chiqaruvchilari ham bunga qo‘l urishgan. Blok diagrammasiga o‘xshashlik bo‘yicha simvollar (elementlar) grafik yozuvlarini maqsadga muvofiq guruhlash

mumkin. Quyidagi jadvalda adabiyot va dasturiy ta‘minotda eng ko‘p uchraydigan EPC lari va ularning muqobil tasvirlari mavjud.

EPC yordamida dekompozitsiya jarayonlarni modellashtirish va iyerarxik algoritmlar jarayonini modellashtirishning klassik tamoyillariga amal qiladiyu. Dekompozitsiya, alohida diagrammalarda ko‘rsatish bilan, IDEF0 diagrammalarda oldindan belgilangan jarayonlarda ishlashga o‘xshash funksiyalar uchun bajariladi.

1. EPC jarayonining diagrammasi kamida bir boshlash hodisasi bilan boshlash kerak va kamida bir tugash hodisasi bilan yakunlandi.

2. Jarayon rivojlanib borgan sari hodisalar va funksiyalar almashinib turishi kerak.

3. Diagramma vazifalari tavsiya soni ko‘pi bilan 20 ta bo‘lishi mumkin. Agar grafik funksiyalari soni 20 dan sezilarli darajada oshsa, yuqori darajadagi jarayonlar noto‘g‘ri ta‘kidlangan va modelni sozlash kerak bo‘ladi.

4. Hodisalar va funksiyalar jarayonning borishini aks ettiruvchi qat‘iy bitta kiruvchi va bitta chiquvchi aloqani (nazorat oqimini) o‘z ichiga olishi kerak.

5. Grafik bir tegishsiz elementlar bo‘lmasligi kerak.

6. Yuqori modul diagrammasida funksiya atrofidagi hodisalar va mantiqiy operatorlar funksisiya dekompozitsiya diagrammasida boshlang‘ich / natijaviy hodisalar va operatorlar bo‘lishi kerak.

7. Har bir birlashma kamida ikkita kiruvchi ulanishga va faqat bitta chiquvchi, ayirmalar - faqat bitta kiruvchi aloqa va kamida ikkita chiquvchi bo‘lishi kerak. Operatorlar bir vaqtning o‘zida bir nechta kiruvchi va chiquvchi bog‘lanishlarga ega bo‘la olmaydi.

8. Mantiqiy operatorlar faqat funksiyalarni yoki faqat hodisalarni birlashtirishi yoki birlashtirmasligi mumkin. Bir vaqtning o‘zida funksiya va hodisani birlashtira olmaydi.

Visual C++ muhitida hisobot shaklini tayyorlash va chop etish elementlari. Buning uchun ma‘lumotlar bazasini bilish talab qilinadi. Quyidagicha masala berilgan bo‘lsin.

Masala. Universitet ma‘lumotlar bazasi berilgan bo‘lsin. Unda talaba va semestr nomli jadvallar bo‘lsin.

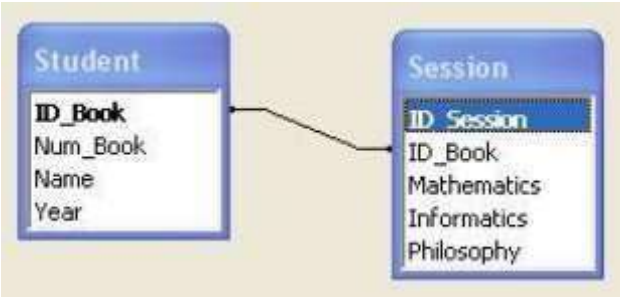
1.1-jadval.Talaba jadvalining tuzilishi

Maydon nomi	tipi	Izoh
ID_Book	Butun son (int)	Birlamchi kalit maydon
Num_Book	matnli (10 belgi)	Talabaning reyting daftari raqami
Name	matnli (30 belgi)	Talabaning ismi sharifi
Year	Butun son (int)	Tug‘ilgan yili

1.2-jadval. Semestr jadvalining tuzilishi

Maydon nomi	tipi	Izoh
ID_Session	Butun son (int)	Birlamchi kalit maydon
ID_Book	Butun son (int)	reyting daftari raqami
Mathematics	Butun son (int)	Matematika bahosi
Informatics	Butun son (int)	Informatika bahosi
Philosophy	Butun son (int)	Falsafa bahosi

Ushbu jadvallarning o‘zaro munosabati quyidagi bog‘lanish orqali ifodalangan bo‘lsin.



1.1-rasm. Jadvallarning relyatsion bog‘lanishi.

Berilgan ma‘lumotlar asosida hisobotlardan foydalanishni tashkil qilib beruvchi dasturni ishlab chiqishni ko‘ramiz. Quyidagi ma‘lumotlarni o‘z ichiga olgan hisobot yaratish lozim:

- Reyting daftarini
- Talabaning ismi – sharifi
- Matematika bahosi
- Informatika bahosi
- Falsafa bahosi
- Talabaning o‘rtacha bahosi

Joriy hisobotni yaratishda ma‘lumotlar Microsoft Access DBMS tomonidan hosil qilingan so‘rov orqali olinadi. So‘rov Query1 deb ataladi. Hisobot alohida shaklda ko‘rsatilishi kerak.

Buning uchun quyidagi amallar bajariladi:

1. Microsoft Visual Studio tizimi orqali Windows Forms dastur shabloni yordamida loyiha yaratiladi. Microsoft Visual Studio ishga tushirilgandan so‘ng, siz yangi loyiha yaratish kerak. Windows Forms dastur shabloni yordamida yangi loyihani yaratish va saqlashning batafsil olroq ytib o‘tilgan.

2. Ma‘lumotlar bazasini yaratish yoki tayyor ma‘lumotlar bazasi faylini yuklash olish lozim. Ma‘lumotlar bazasi uchun Microsoft Access ma‘lumotlar bazasini boshqarish tizimi yordamida yaratiladi. Ma‘lumotlar bazasida ikkita

tegishli jadval talaba va sesmestr, shuningdek, hisobotni yaratish uchun ishlatiladigan Query1 so‘rovi yaratib olish kerak.

3. Ma‘lumotlar bazasi faylining loyihaga ulanishi. Ma‘lumotlar bazasi faylini loyihaga ulashdan oldin ushbu faylni loyiha fayllari saqlanadigan papkaga saqlashni tavsiya etamiz. Foydalanish uchun “Education.mdb” ma‘lumotlar bazasi, uni standart usulda loyihaga ulash kerak. Loyihaga ma‘lumotlar bazasi faylini ulash uchun "Add Connection" dan foydalaniladi. Bu amal kontekst menyudan buyruq ro‘yxatida chaqiriladi.

4. Natijada “ma‘lumotlar manbasini tanlash” oynasida ma‘lumotlar manbai (Data Source) ni tanlash uchun - Microsoft Access ma‘lumotlar bazasi fayli ulash kerak. Keyingi oynada "ma‘lumotlar bazasi fayl nomi" dagi “ulanish Qo‘shish” ma‘lumotlar bazasi fayliga yo‘lni ko‘rsatiladi (“Browse” tugmasi orqali).

5. Asosiy loyiha oynasini ishlab chiqish. Topshiriq shartiga ko‘ra hisobot alohida shaklda ko‘rsatilishi lozim. Shuning uchun loyihaning asosiy shakli 12.2- rasmda ko‘rsatilgandek shaklga ega bo‘ladi.

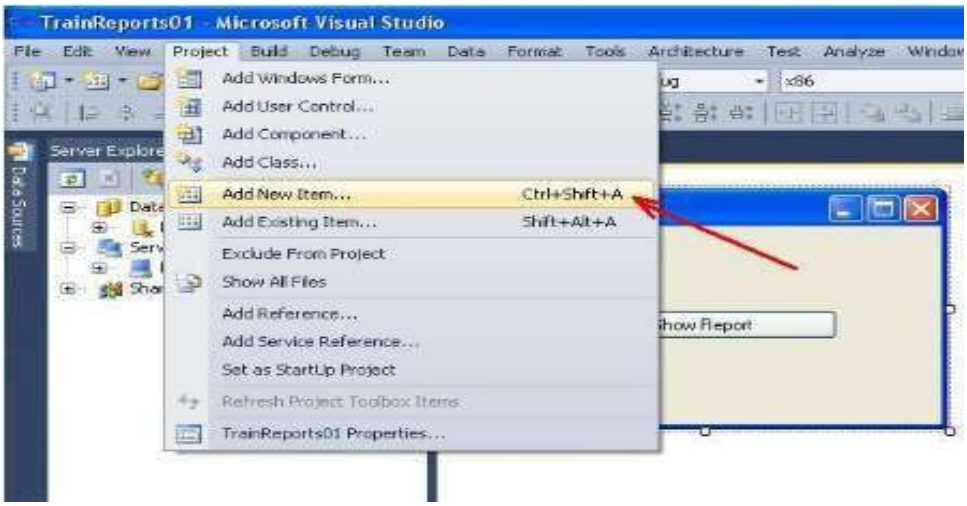
1.2-rasm.

Agar rasmda ko‘rib turganingizdek, shakl button1 nomidagi faqat bitta tugma turini o‘z ichiga oladi. "Show Report" tugmasini bosgandan so‘ng hisobot boshqacha shaklda ko‘rsatiladi. Quyidagi xususiyatlarini sozlash kerak:

Boshqarish elementi button1 ning Text = «Show Button»

Boshqarish elementi Form1 ning Text = «Report Viewer»

Hisobot faylining loyihaga ulanishi. Microsoft Visual Studio dasturida har bir hisobotda «*.rdlc» kengaytmasiga ega fayl mavjud. Bu fayl hisobotda hosil qilingan



1.3-rasm. Loyiha menyusidan Add New Item amali

ma‘lumotlarni o‘z ichiga oladi. Microsoft Visual Studio dasturida fayl yaratish uchun loyiha menyusidan Add New Item buyrug‘ini chaqiriladi (1.3- rasm).

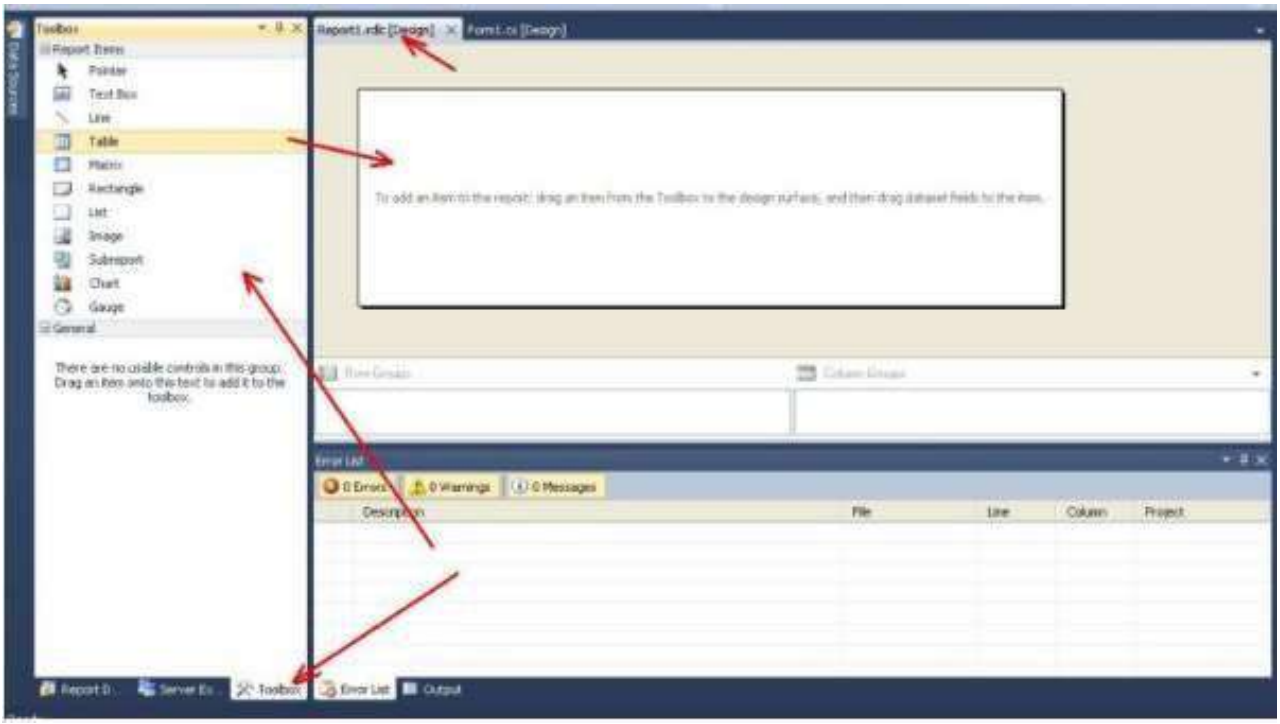
6. Natijada, «Add New Item…» hisobot shablonini tanlash kerak bo‘lgan oyna ochiladi (1.4-rasm) va hisobot fayli uchun nom belgilanadi.

1.4-rasm. Hisobot shablonini tanlash va hisobotni o‘rnatish

Tanlovni tasdiqlagandan so‘ng Microsoft Visual Studio oynasida hisobotning asosiy oynasi ko‘rsatiladi.

Hisobot oynasida hisobotni loyihalashtirish lozim. Buning uchun quyidagi lozim.

1. ToolBox asboblar paneli. ToolBox paneli - hisobot elementlarini loyihalash uchun ishlatiladi.



1.5-rasm. ToolBox panel va hisobot loyihalashtirish vositalari.

2. Hisobot yaratish uchun - Microsoft Visual Studio hisobotini ishlab chiqish uchun bir

qator taklif etiladigan standart shablonlar bor. Bu boshqaruvlardan turli hisobotlarning ko‘rinishini (taqdimotini) loyihalashtirish uchun foydalanishingiz mumkin.

Quyidagi elementlar taklif etiladi:

TextBox - Hisoblashlar natijasida olingan teglar, maydonlar yoki qiymatlarni ko‘rsatadi;

Line - nuqta yoki undan ko‘p qalinlikka ega bo‘lgan chiziq chizadi;

Table (jadval) - nazorat ustunlar va qatorlar o‘zgarmaydigan ma‘lumotlarni ko‘rsatadi;

Matrix - o‘zgaruvchan raqamli, ustunlar va qatorlar o‘zgaruvchan raqami bor ma‘lumotlarni jamlash ko‘rsatiladi; Rectangle - hisobot elementlari uchun shablon sifatida to‘rtburchak chizadi;

List - har bir guruh yoki ma‘lumotlar qatori uchun takrorlanadigan hisobot bandlari to‘plamini ko‘rsatadi;

Image - tasvirni bitmap sifatida ko‘rsatadi (masalan, logotip yoki fotosurat); Subreport - hisobot chegaralari doirasida bog‘liq sub-hisobot ko‘rsatadi;

Chart - Ma‘lumotlarni har xil turdagi diagrammalar ko‘rinishida ko‘rsatadi;

Gauge - Maydon yoki ifodaning qiymatini chiziqli yoki radial qarama-qarshilik ko‘rinishida ko‘rsatadi.

Har bir nazoratdan foydalanishning o‘ziga xos xususiyatlariga kirmasdan, muammoni hal qilish uchun jadval nazoratidan foydalanamiz. Bu nazorat ustunlar son va qatorlar bir o‘zgarmaydigan qator bor ma‘lumotlarni ko‘rsatadi.Bu elementlarning foydalanish xususiyatlari tizimda bir xil amalga oshiriladi.

Inson-mashina interfeysi (IMI) keng tushunchadir. Inson-operator va ular boshqaradigan mashinalar bilan o‘zaro aloqasi ta‘minlashning muhandislik echimlarni o‘z ichiga oladi. Inson - mashina interfeysi tizimlarini yaratishda yaqindan ergonomika tushunchalari bilan bog‘liq. IMI loyihalashtirish quyidagilarni o‘z ichiga oladi:

- ish joyini yaratish: ishchi maydon va boshqaruv paneli, qurilmalar va boshqaruvlarni joylashtirish, ish joyi va ehtimol ranglar uyg‘unligi.
- operatorning barcha boshqaruv bilan o‘zaro aloqasini ta‘minlash hisoblanadi: ularning mavjudligi va zarur harakatlari, KIRISH samaradorligi va tezligi, mustahkamligi, nazorat harakatlarining (izchilligi), joylashgan o‘rni , displeylar va ulardagi teglar hajmi (bularning barchasi mavjudligi).

Inson - kompyuter o‘zaro hamkorligi ilmiy ko‘p yo‘nalishli konteksti kompyuter grafika, muhandislik psixologiyasi, ergonomika, tashkilot nazariyasi, kognitiv fan, informatika va boshqalarni o‘z ichiga oladi.

Foydalanuvchi interfeysi-bu foydalanuvchi kompyuter bilan o‘zaro aloqa qilish imkonini beruvchi dasturiy va apparat vositalar majmuasidir. Bu o‘zaro muloqotlar dialoglarga asoslanadi. Bu holda dialog deganda inson va kompyuter o‘rtasida Real vaqtda amalga oshiriladigan va muayyan vazifani birgalikda hal etishga qaratilgan axborot almashuvi tushuniladi. Axborot almashish va harakatlarni muvofiqlashtirish har bir dialog foydalanuvchi va kompyuterni fizik ulaydigan alohida i/o jarayonlardan iborat. Xabarlarni uzatish orqali axborot almashiladi va nazorat signallari. Xabar-dialog almashinuvida ishtirok etuvchi axborot qismidir. Ular quyidagilardir:

- odam tomonidan kiritish vositalari yordamida hosil qilinadigan KIRISH xabarlari: klaviatura, manipulyatorlar, sichqoncha kabi va boshqalar;
- kompyuter tomonidan matnlar, audio signallar va/yoki tasvirlar ko‘rinishida hosil qilinadigan va foydalanuvchiga monitor ekranida yoki boshqa chiqish qurilmalarida ko‘rsatiladigan chiqish xabarlari.

Asosan foydalanuvchi quyidagi turdagi xabarlarni hosil qiladi: axborot so‘rovi, yordamchi so‘rovi, operatsiya yoki funksiya so‘rovi, ma‘lumot kiritish yoki o‘zgartirish, shablon maydonini tanlash va boshqalar. Bunga javoban, u qabul qiladi: maslahatlar yoki yordam, javob talab qilmaydigan axborot xabarlari, harakatni talab qiladigan so‘rovlar, javob kerak bo‘lgan xato xabarlari va shablon formatidagi o‘zgarishlar.

Dasturlashga protsessual va ob‘yektga yo‘naltirilgan yondashuvlar bilan taqqoslaganda, interfeysni ishlab chiqishga protsessual yo‘naltirilgan va ob‘yektga yo‘naltirilgan yondashuvlar mavjud. Protseduraga yo‘naltirilgan interfeyslar "protsedura" va "operatsiya" tushunchalariga asoslangan foydalanuvchi o‘zaro aloqasining an‘anaviy modelidan foydalanadi. Bu model doirasida dasturiy ta‘minot foydalanuvchiga tegishli ma‘lumotlarni va uning natijasini foydalanuvchi aniqlaydigan ma‘lum amallarni bajarish imkoniyatini beradi.

Ob‘yekt yo‘naltirilgan interfeyslarni foydalanuvchi o‘zaro bir oz boshqacha modelini foydalanib, manipulyatsiya domen ob‘yektlarini qaratishi mumkin.

Zamonaviy realliklardan farqli o‘laroq, birinchi kompyuterlar grafik foydalanuvchi interfeyslari juda zaif bo‘lgan. Shuning uchun, eng boshida, odamlar faqat buyruq qatordan foydalanishingan (CLI yoki buyruq qatorni interfeysi), unda buyruqlar so‘rovlar yordamida belgilangan edi.

Keyinchalik, bu TUI interfeyslarni ishlab, endi operatsion tizimlari o‘rnatish jarayonida ishlatiladi. Kompyuterlarning mavjudligi foydalanuvchilarga qulay interfeysni ishlab chiqish zarurligiga olib keladi.

Grafik foydalanuvchi interfeysi-bu kompyuterning tobora ortib borayotgan ishlashi bilan birga mustahkam o‘rnatilgan interfeys turi. Yaqin kelajakda, odamlar nutq yordamida kompyuter bilan o‘zaro imkonini beradi foydalanuvchi audio interfeyslarni (VUI yoki ovozli foydalanuvchi interfeysi) bo‘lishi mumkin.

Turli kompyuter o‘yinlari tabiiy foydalanuvchi interfeysi (NUI yoki tabiiy foydalanuvchi interfeysi) yaratib beradi. Uning tizimi insonning harakatlarini tahlil qiladi va ularni o‘yindagi harakatlarga aylantiradi. Perseptual foydalanuvchi interfeysi (PUI) va aqlli-kompyuter interfeysi (BCI yoki brain-kompyuter interfeysi) hozirda ishlab chiqilmoqda. eng so‘nggi rivojlanish odamlarga kompyuterlarni fikri kuchi bilan boshqarish qobiliyatini ta‘minlashga qaratilgan.

Web-ilovalarning foydalanuvchi interfeysi ham katta auditoriyadan kelib chiqqan holda o‘ziga xos xususiyatlarga ega. Notiqlik haqida, notiqlik-ritorika haqida shunday fan mavjud. Fikrlash insoniyat bor ekan, bu mavzu bo‘yicha birinchi ishlar Empedokllarga (taxminan miloddan avvalgi v asr) taalluqlidir. Ritorikadan bilamizki, hozirgi odamlar sonining ortishi bilan tomoshabinlarning intellektual darajasini kamaytirish qonuni mavjud. Auditoriya qanchalik katta bo‘lsa, nutq shunchalik sodda bo‘lishi, iboralar shunchalik qisqa bo‘lishi va nutqdagi pauzalar shunchalik muhim ahamiyat kasb etadi. Boshqa tomondan, ijtimoiy tarmoqlarning rivojlanishi bizga muloqot mutlaqo og‘zaki emasligini ko‘rsatadi va foydalanuvchi interfeysi orqali ham muloqot qilishingiz mumkin. Yuqoridagilardan kelib chiqqan holda, veb-dastur interfeysi sodda bo‘lishi kerakligi mantiqan kelib chiqadi. Iloji boricha oddiy. Aytaylik, ma‘lum bir korxona ichida ishlash uchun mo‘ljallangan dastur uchun interfeys ishlab chiqyapsiz. Shu bilan birga, siz har doim tizimning kelajakdagi foydalanuvchilari uchun o‘quv kurslarini tashkil qilishingiz, ekranda nima borligini va ishlab chiquvchi sifatida nima demoqchi ekanligingizni aytishingiz mumkin.

FI dizayni ko‘plab yondashuvlarni o‘z ichiga oladi: kaskadli, ichki, tashqi, iterativ va boshqalar. Yondashuvlarning har biri ichida bir qator usullar ham mavjud. Yuqori darajadagi yondashuvdan qat‘iy nazar, yuqori darajadagi va past darajadagi o‘zaro aloqalarni loyihalash o‘ziga xos usullarni talab qiladi. Ob‘yektga yo‘naltirilgan yondashuv tasvirlangan, garchi ko‘plab dizayn bosqichlari uchun an‘anaviy rivojlanish usullari bilan almashtirilishi mumkin. Ushbu yondashuv yordamida

olingan dizayn natijalari FIga yo‘naltirilgan ilovalarni hujjatlashtirish, modellashtirish, prototiplash, baholash va amalga oshirish uchun foydalidir.

Nostandart yondashuv. Yuqori darajadagi loyiha tizimi bilan FI va foydalanuvchi o‘zaro standart xatti haqida o‘ylash mumkin, lekin tafsilotlar qo‘shiladi, deb, u standarti erishish uchun yanada qiyin bo‘ladi — va bir vaqtning o‘zida izchil-qo‘llash turli qismlariga ajratiladi. Tafsilotlar odatda kompromisslar va kutilmagan hodisalar bilan bohliqdir .

Foydali qoida. Bu tasodifan yoki loyihaga muvofiq sodir bo‘lsa, nostandart xatti harakatlari qo‘llash kerak.

Yangilangan grafik. Dastlabki grafik imkoniyatlar va vizuallashtirish masalalari dastlabki loyihalash bosqichlarida ishlab chiqiladi, shuning uchun batafsil loyihalash bu grafik imkoniyatlarni aniqlashni talab qiladi (ba‘zan grafikadan foydalanish dastlabki loyihalash tushunchalaridan tashqariga chiqadi). Bundan tashqari, bu bosqichda ekranlar va xabarlarning yakuniy batafsil tartibi talab qilinadi.

Javob vaqti. Og‘ir tahlil yoki simulyatsiya orqali, muhim va eng tez-tez ishlatiladigan dasturiy ta‘minot joylari uchun javob vaqt smetasini olish kerak. Oxirgi foydalanuvchi tomonidan amalga oshirilgan har bir funksiya uchun ehtimoliy javob vaqtiga asoslanib, dasturiy ta‘minot uchun mos teskari aloqa mexanizmi ishlab chiqiladi.

Interfeys yaratishning asosiy tamoyillari

1. Tabiiylik (intuitiv). Tizim bilan ishlash vazifasini hal qilish jarayonini boshqarish uchun kerakli direktivalarni (interfeys elementlarini) topishda foydalanuvchiga qiyinchilik tug‘dirmasligi kerak.

2. Mustahkamlik. Agar foydalanuvchi tizim bilan ishlayotgan paytda tizimning biror qismi bilan ishlash uchun ayrim metodlardan foydalangan bo‘lsa, u holda metodlar tizimning boshqa qismida bir xil bo‘lishi kerak. Shuningdek, interfeys orqali tizim bilan ishlash o‘rnatilgan, doimiy, qulay me‘yorlarga (masalan, Enter klavishidan foydalanish) mos kelishi kerak.

3. Ortiqchalilik. Bu foydalanuvchi faoliyat yoki tizimini boshqarish uchun faqat minimal ma‘lumot kiritish kerak degan ma‘noni anglatadi. Masalan, foydalanuvchi ahamiyatsiz sonlarni kiritmasligi kerak (00010 o‘rniga 10). Xuddi shunday, foydalanuvchi oldindan kiritilgan yoki avtomatik ravishda tizimdan olinishi mumkin bo‘lgan ma‘lumotlarni kiritishni talab qila olmaydi. Axborot kiritish jarayonini kamaytirish uchun iloji boricha standart qiymatlardan foydalanish tavsiya etiladi.

4. Yordam tizimiga to‘g‘ridan-to‘g‘ri KIRISH. Ish vaqtida tizim foydalanuvchiga kerakli ko‘rsatmalar berishi kerak bo‘ladi. Yordam tizimi uchta asosiy jihatga javob beradi: taqdim etilgan buyruqlar sifati va miqdori; xato xabarlarining tabiati va tizimning nima qilayotganini tasdiqlash. Xato xabarlari foydalanuvchi uchun foydali va tushunarli bo‘lishi kerak. Shu bilan birga, bu savollarga foydalanuvchilar uchun imkon qadar muloyim va tushunarli bo‘lgan tilda javob berishi kerak.

5. Moslashuvchanlik. Tizim interfeysi turli darajadagi tayyorgarlikka ega foydalanuvchilarga xizmat qilishi kerak. Tajribasiz foydalanuvchilar uchun interfeys iyerarxik menyu strukturasi va tajribali foydalanuvchilar uchun buyruqlar, kalit kombinatsiyalar va parametrlar sifatida tashkil etilishi mumkin.

Sayt moslashuvchanligi - foydalanuvchilar osonlik bilan saytida manipulyatsiyadan ortiqcha raqami bilan bezovta holda zarur ma‘lumotlarni topish imkonini beradi va sayt materiallar tashkil etish hisoblanadi. Maxsus tadqiqotlarga ko‘ra, o‘rtacha bir foydalanuvchi saytda o‘ttiz etti soniya sarflaydi va bu vaqt ichida ular kamdan-kam sahifaning oxiriga o‘tadi. Shu munosabat bilan, sayt samimiy ma‘lumotlar chiqishi tashkil etish sohasida zamonaviy standartlarga javob bermasa, agar, bunday sayt targ‘ib qilish, barcha sa‘yharakatlari kerakli maqsadga erishish mumkin emas, chunki foydalanuvchi saytida zarur ma‘lumotlarni topib bo‘lmasa, ular shunchaki ko‘proq samimiy boshqa resursga o‘tadi. Saytning eng samarali moslashuvchanligi ta‘minlash Internetda axborot taqdim etish xususiyatlarini bilishni talab qiladi. Foydalanuvchilar, birinchi navbatda, ularning ko‘z oldida moddiy ko‘rish muhimdir. Shuning uchun interfeysning maksimal soddaligi va qulayligi internetda ma‘lumotlarni tashkil qilishning asosiy tamoyilidir. Saytda oddiy va samimiy interfeysni yaratish uchun mavjudlikning asosiy tamoyillarini amalga oshirishga imkon beruvchi texnikalardan foydalanishin kerak. Butun sayt uchun navigatsiya umumiy bo‘lishi kerak, sayt logo yuqori chap burchagida joylashtirilgan bo‘lishi kerak va logo bosish bosh sahifa olib kelishi kerak va veb-saytida header va altbilgide aloqa ma‘lumotlarni joylashtirilgan kerak. Saytda malakali qidiruvni tashkil etish saytning mavjudligini ta‘minlashning muhim elementi hisoblanadi. Qidiruv tugmasi, logotipdan tashqari, sayt uchun ikkinchi eng muhim element hisoblanadi va u har qanday saytda foydalanuvchi uchun birinchi yordamchidir. Foydalanuvchi birinchi 2 soniya ichida bu tugmani topsa kamida, ular xavfsiz his va sayt tomon keskinlik ularning darajasi kamayadi. Moslashuchanlik sinov o‘tkazish paytida ta‘qib qilinishi kerak va uning ma‘lum bosqichlari bor. Sinov o‘tkazishning mohiyati quyidagicha:

1. Muammolarni aniqlash;

2. Gipotezani hosil qilish;
3. Test uchun o‘lchov aniqlash;
4. Belgilar va test ssenariysini aniqlash;
5. Tanlang respondents;
6. Anketalarni to‘ldirish;
7. KIRISH brifingini o‘tkazish:
8. Mavjudligi test o‘tkazish;
9. Intervyu respondents;
10. Natijalarni tahlil qilish;
11. Sayt dizayni uchun talablarni aniqlash.

Umuman olganda FIni ishlab chiqish soha va foydalanuvchilarning imkoniyatlari va xoxishlarini inobatga olishi kerak.

Ma‘lumotlarni eksport qilish vositalari va komponentalari. Zamonaviy dasturlar foydalanuvchini qiziqtiradigan ma‘lumotlarga KIRISHning ko‘p usullarini ta‘minlaydi. Ma‘lumotlar bilan ishlash uchun ixtisoslashgan dastur ilovalari ishlab chiqilgan. Microsoft Office kabi umumiy ofis vositalari bilan integrallashgan ma‘lumotlarni chiqarish imkoniyatlarini o‘z ichiga oladi. Foydalanuvchi hozirda ma‘lumotlar bazasidan kerak ma‘lumotlarni chiqarib eksport qilishi mumkin. Biroq, turli xil ma‘lumotlar bilan ishlashning eng keng tarqalgan va tanish usuli an‘anaviy hisobot shaklidir. Bir tomondan, ko‘p foydalanuvchilar, ayniqsa, hali hisobot standart qog‘oz shaklidan voz kechishga kuch topa olmaydi. Boshqa tomondan, hisobot, asosan, turli xil xususiyatlarga ega bo‘lgan ma‘lumotlarni birlashtiruvchi vositadir.

Hisobot berish funksiyalarini ta‘minlash talabi ko‘pincha zamonaviy dasturiy mahsulotlarni ishlab chiqishda paydo bo‘ladi. Databeacon Toolkit sifatida dastur plagin-smaylik va hisobot qo‘llab-quvvatlash vositalari va komponentalari mavjud.

Shunga o‘xshash vositalar Microsoft Visual Studio. net (VS.NET)dir. Microsoft Visual Studio. net (CRVS) uchun Crystal Reports deb nomlangan, biznes ob‘yektlarini ishlab chiqish va litsenziyalash sxemasi bo‘yicha etkazib berishga yщnaltirilgan. Yana bir vositasi, Microsoft hisobot xizmatlar (MSRS), Microsoft SQL Server DBMS asoslangan platforma hisoblanadi.

Crystal Reports for Microsoft Visual Studio .NET (Hisobotlarni qurish instrumenti). Crystal Decisions mahsulotlarini Microsoft development tools bilan bog‘lash tarixi 1993-yilga borib taqaladi. O‘shanda Crystal Decisions hisobotlarning tarkibiy qismlari avval Visual Basic 3.0 da taqdim etilgan edi. 1996 Crystal hisobotlar dasturi Visual Studio development tools paketining bir qismiga kiritildi va 2002 da uning bir qismi bo‘ldi VS.NET va. net Framework, rasmiy nomi CRVS qabul qilindi.

Crystal Reports komponentlarining paydo bo‘lishi va ularni o‘zaro rivojlangan varinatlari mavjud. Masalan, VS.NET 2003VS.NET 2005.

CRVS komponentalarining asosiy maqsadi dasturning hisobot funksiyalarini ishlab chiqishni qo‘llab-quvvatlashdir. Bu maxsus hisobot shablon talablariga muvofiq ishlab chiqilgan bo‘ladi. Hisobot sahifasida ma‘lumotlarni joylashtirishni belgilovchi rpt kengaytmasidan foydalanib, shablonda qaysi ma‘lumotlar tuzilmalari, hisobot ma‘lumotlari uchun manba bo‘lib xizmat qilishi ko‘rsatilgan bo‘lishi kerak. Keyinchalik shablon manbadan ma‘lumotlarni chiqarib, yakuniy hisobotning tuzilishini shakllantirish uchun foydalanadigan CRVS komponentlarini kiritishga o‘tiladi. Hisobotni ko‘rsatish uchun ishlab chiqilayotgan dasturga ko‘shilgan maxsus funksiyalardan foydalanib web va Windows ilovalarini ham qo‘llab-quvvatlanadi.

Aslida, VS.NETda ekspert jarayoni kodi bir necha satrlarni chaqirib kamayadi, lekin " hisobot dizayner " yordamida amalga oshiriladi. Shunday qilib, hisobot shablonini yaratish uchun Microsoft. net platformasida dasturlashingiz shart emas (MS.NET). xohlasangiz, uni o‘zgartirish imkonini beruvchi, dastur hisobot shablonni o‘z ichiga olishi mumkin. Bu juda qulay, chunki bu holda, har doim ishlab chiqilgan dasturni qo‘llab-quvvatlash uchun amalga oshirish mumkin.

Yuqorida aytib o‘tilganidek, CRVS komponentlari vebga yo‘naltirilgan va muntazam Windows ilovalarini ishlab chiqish uchun ishlatilishi mumkin. Ma‘lumotlar bilan ishlash nuqtai nazaridan Crystal hisobotlar komponentlari ikki rejimda ishlash - surish modeli (Push model) va tortish modelida (Pull model) ishlatilishi mumkin.

Surish modeli - oraliq ma‘lumotlar tuzilmasi bilan ishlash modeli. Shu bilan birga hisobotlar uchun ma‘lumotlar odatiy takrorlanish ko‘rinishida hosil qilinadi

.NET Framework - ma‘lumotlar bazasi deb ataladigan ma‘lumotlar tuzilmasidir. Dastur hisobotlarni qurish uchun kiritish ma‘lumotlarni tayyorlash barcha mantiq amallarga bog‘liq. Bu qulay, chunki ma‘lumotlar faqat ma‘lumotlar manbaiga bevosita KIRISH yo‘li bilan emas, balki mutlaqo biron-bir tarzda hosil qilinishi mumkin. Biroq, agar hisobot yetarlicha katta bo‘lsa, juda noqulay tuzilmani to‘ldirishingiz kerak.

Tortish modeli - talab bo‘yicha ma‘lumotlarni so‘rash, ya‘ni ma‘lumotlar bazasidan olish. Bu holda, Crystal Reports komponentlariga o‘tgan hisobot shablon manbadan ma‘lumotlarni olish bilan bog‘liq ishlarni amalga oshiradi. Ushbu yondashuv ko‘p sahifali hisobotlarni namoyish qilish uchun juda samarali, chunki ko‘rsatilgan sahifalar uchun faqat ma‘lumotlar manbadan olinadi.

Hisobotni ishlab chiqarish va aks ettirishdan tashqari, CRVS komponentlari olingan ma‘lumotlarni PDF, HTML, XLS, RTF va DOC formatlariga eksport qilishni qo‘llab-quvvatlaydi. Bu holda, dastur fayl tizimi katalogda bevosita zarur hisobot fayllarni yaratish mumkin. Bu hisobot fayllarini yaratishning ommaviy rejimi uchun juda qulay, foydalanuvchi keyinchalik natijalarni manfaatdor tomonlarga uzatish uchun ularni oldindan ko‘rmasdan hisobotlar guruhini yaratish kerak.

CRVS ning afzalliklari va kamchiliklarini qisqacha eslatib o‘tamiz. Shubhasiz afzalligi. net Framework va ham yaqin integratsiya bo‘ladi VS.NET, qaysi ilova tuzuvchi rivojlanish vositalari bilan hisobot yaratish imkonini beradi. Crvs komponentlariga asoslangan echimlar, shuningdek, bu xususiyat hisobotlar va Microsoft .NET. arxitekturasida joylashganligi sababli, murakkab va interaktiv hisobotlarni amalga oshirish qiyin emas, chunki ularni interaktiv nazorat elementlari kabi "kublar" CRVSda mavjud.

Ushbu mahsulot Microsoft texnologiyalari oldingi avlod (u Win32 va C++yordamida ishlab chiqilgan) bo‘yicha butlovchi yadroda yashirin amalga oshirishga mo‘ljallangan, uning ishlab chiqaruvchi hisoblanadi. Dinamik kutubxonalar ko‘rsatkichlarni o‘rnatish va dasturiy juda oddiy eksport hisobotlarni yordamida amalga oshiriladi. Shu bilan birga, yadro ishlaydigan bunday platform

.net va an‘anaviy xotira algoritm dinamik yig‘ish, raqobat kabi nostandart infratuzilma muammolari olib kelishi mumkin. Bu muammoni hal qilish yo‘lida harakat qilib, ishlab chiqaruvchi 1.1 versiyasi qismi sifatida erkin foydalanishga berilgan va .net platforma bosqichma-bosqich o‘tishni boshlagan VS.NET 2003). Bu o‘zgartirish Crystal hisobotlar komponentlarini foydalanadi va dastur bilan faqat. net ma‘lumotlarni ekspert qilib beruvchi tomonidan tarqatish jarayonini soddalashtirish kerak. Crvsning platformaga to‘liq o‘tishi kutilmoqda.

CRVS komponentlarining arxitekturasi yakuniy hisobot ishlab chiqarish jarayonlarini ko‘p darajali taqsimlashning ssenariysini tashkil etishga imkon berishiga qaramasdan, ushbu mahsulot butun hisobotni boshqarish siklini tashkil qilish uchun mos emas. MSRS platformasi bu muammoni hal qilish uchun mo‘ljallangan va tayyor ekspert fayllarni hosil qilishga qaratilgan.

Testlash va instruksiya yozish usullari. Dasturiy ta‘minot testi-nuqsonlarni aniqlash va mahsulot sifatini yaxshilash maqsadida dasturiy ta‘minot va tegishli hujjatlarni tahlil qilish jarayonidir.

Dasturiy ta‘minot ishlab chiqish o‘n yillar davomida, turli xil yo‘llar bilan sinov va sifat kafolati orqali juda yaqinlashib qilindi. Uning bir necha asosiy "sinov davrlari" bor.

O‘tgan asrning 50-60-yillarida sinov jarayoni keng rasmiylashtirilgan, bevosita dasturiy ta‘minot ishlab chiqish jarayonidan ajralgan va "matematiklashtirilgan" edi. Aslida, test ko‘proq disk raskadrovka dasturlari kabi bo‘lgan. "Mukammal testing4"-deb atalmish bir tushuncha bor ediki, barcha mumkin bo‘lgan kiritish ma‘lumotlar bilan kodni ijrosi barcha mumkin bo‘lgan yo‘llarini tekshirish amalga oshirilardi. Kiritish ma‘lumotlar soni juda katta, chunki, mukammal sinov mumkin emas, va bu yondashuv bilan u hujjatlardagi muammolarni topish qiyin bo‘lgan.

70 yillar sinov ikki asosiy g‘oyalar asosida bo‘lgan: birinchi muayyan belgilangan sharoitlarda (ijobiy testing) dastur faoliyatini isbotlash jarayoni sifatida qabul qilindi va keyin-qat‘iy qarama-qarshi: dastur muayyan belgilangan sharoitlarda (salbiy testing) inoperabl ekanligini isbotlash jarayoni sifatida qaraladi. Nafaqat bu ichki ziddiyat vaqt davomida g‘oyib bo‘lmaydi, lekin bugungi kunda ko‘p mualliflar juda to‘g‘ri sinov ikki to‘ldiruvchi maqsad bo‘lgan deb hisoblaydi. U aniqlangan muammolarga ko‘z yumishga ruxsat bermaydi.

80-yillarda, dasturiy ta‘minot ishlab chiqish sinov joyda asosiy o‘zgarish yuz berdi: loyiha yaratish yakuniy bosqichlarida biri, test butun rivojlanish sikli davomida tatbiq etildi (dasturiy lifecycle). Bundan tashqari, oldindan va ularning yuzaga keladigan vaziyatlarni oldini olish uchun emas, balki faqat ko‘p hollarda ruxsat berilgan "dasturiy ta‘minot ishlab chiqish modellari" iterativ ortib dasturiy ta‘minot ishlab chiqish modeli tavsifini ko‘rish maqsadga muvofiq.

90-yillarda, testlash jarayoni butun dasturiy ta‘minot ishlab chiqish sikli o‘z ichiga oladi va mavjud sinov hollarda va sinov muhitlar qo‘llab-quvvatlash, rejalashtirish, loyihalash, yaratish va sinov ishlarni amalga oshirish jarayonlarini ta‘sir "sifat kafolati" deb nomlangan yanada keng qamrovli jarayoni kabi sinov o‘tish bor edi. Sinov sifat jihatidan yangi darajaga etdi, bu tabiiy ravishda uslubiyotning yanada rivojlanishiga, sinov jarayonini boshqarish uchun juda kuchli vositalar va hozirgi avlodlariga juda o‘xshash bo‘lgan sinovlarni avtomatlashtirish vositalarini paydo bo‘lishiga olib keldi.

Testlashning boshida har qanday mutaxassis (tester ham bundan mustasno emas) ijrochi va talaba. Sinov holatlari va nuqson hisobotlari qanday ekanligini yaxshi tushunish, talablarni o‘qiy olish, bir nechta vositalardan foydalanish va jamoada yaxshi ishtirok etish kifoya. Asta-sekin sinaluvchi loyihani ishlab chiqishning barcha bosqichlariga KIRISH boshlanadi, bu esa o‘z navbatida ularni yanada to‘laroq anglatadi, ulardan nafaqat faol foydalana boshlaydi, balki loyiha hujjatlarini ishlab chiqib, yanada mas‘uliyatli qarorlar qabul qila boshlaydi.

Testlashning asosiy maqsadini majoziy ma‘noda ifodalasangiz, shunday bo‘ladi: loyiha hozirgi vaqtda nimaga muhtojligini, loyiha bu zarurni o‘z o‘lchovida qabul qiladimi, yo‘qmi, qanday qilib vaziyatni yaxshiroq o‘zgartirish kerakligini tushuntirib berish. Loyiha menejerining maqsadiga o‘xshaydi, to‘g‘rimi? Sogi-hidori. hu Muayyan rivojlanish darajasidan boshlab, IT mutaxassislari, faqat texnik ko‘nikmalar to‘plamlarida va bu ko‘nikmalarni qo‘llashning asosiy sohasida farq qiladi.

Sinovchi sifatida ishlashni muvaffaqiyatli boshlash uchun qanday texnik ko‘nikmalar kerak?

1. Chet tillar ko‘nikmalari. Ha, bu texnik bo‘lmagan mahorat. Buni aksioma deb hisoblashingiz mumkin.

2.. Haqiqiy rivojlangan foydalanuvchi darajasida ishonchli kompyuter qobiliyatlari va bu sohada doimo rivojlanish kuzatib borish.

3. Dasturlashtirish. Bu tartibda birinchi navbatda har qanday IT shaxs va dasturiy ta‘minot tester hayotini osonlashtiradi. Dasturlash bilimisiz test qilish mumkinmi? Ha, mumkin. Bu, albatta, yaxshi amalga oshirilishi mumkin.

4. Ma‘lumotlar bazalari va SQL tili. Bu yerda testlovchi ham tor mutaxassislar darajasida malakaga ega bo‘lishi talab etilmaydi, lekin eng ko‘p tarqalgan DBMlar bilan ishlashning minimal ko‘nikmalari va oddiy so‘rovlarni yozish qobiliyati majburiy deb hisoblanishi mumkin.

5. Tarmoqlar va operatsion tizimlar qanday ishlashi haqida tushuncha. Hech bo‘lmasa, muammoni tashxislash va iloji bo‘lsa, uni o‘zingiz hal qilish imkonini beruvchi minimal darajada.

6. Veb-ilovalar va mobil ilovalar tamoyillari haqida tushuncha. Shu kunlarda deyarli hamma narsa bunday ilovalar shaklida yoziladi va tegishli texnologiyalarni tushunish samarali sinov uchun majburiy bo‘ladi.

Xulosa sifatida testlovchiga tezda ajoyib mutaxassis bo‘lish imkonini beruvchi shaxsiy sifatlarni ham qayd etamiz:

1) mas‘uliyat va ishlashning ortishi;
2) yaxshi muloqot qobiliyatlari, o‘z fikrlarini aniq, tez, aniq ifodalash qobiliyati;
3) sabr-toqat, qat‘iyat, tafsilotga e‘tibor, kuzatish;
4) yaxshi mavhum va analitik fikrlash;
5) nostandart tajribalar o‘rnatish qobiliyati, tadqiqot uchun imkoniyat.

Bu fazilatlarga teng egalik qiladigan odamni topish qiyin, ammo o‘z-o‘zini rivojlantirish uchun ma‘lum bir mos yozuvlar nuqtasiga ega bo‘lish har doim foydalidir.

NAZORAT SAVOLLARI

1. Hayot davrining tuzilishini belgilovchi asosiy standart nomini ayting.
2. Hayot davri strukturasi nechta jarayon guruhiga asoslangan.
3. Mustaqil (loyihaga nisbatan) ekspertlarning korxona faoliyatining qabul qilingan talablar, rejalar va shartnoma shartlariga muvofiqligini aniqlash ishlari nima deb yuritiladi.
4. Strukturali tahlil va loyihalash texnikasiga asoslangan metodologiya nomini ayting.
5. Kichik loyihalar algorimlarini yozish usullari nechta.
6. BPMN diagrammalarning vazifasi nimadan iborat.
7. Algoritmlarning blok diagrammalarini qurish qanday GOST dasturiy hujjatlarning yagona tizimi bilan tartibga solinadi.
8. Ma‘lumotlar sxemalari nima uchun kerak.
9. Hisobot shaklini tayyorlash va chop etish uchun nimani bilash kerak va nima uchun.
10. Jadvallarning relyatsion bog‘lanishi deganda nimani tushunasiz.
11. Yangi hisobot loyihasini yaratish uchun qanday amallar bajariladi.
12. Hisobot konchtruktorining elementlarini sanab bering.
13. Hisobotda jadvalga asoslangan ma‘lumotlarni chiqarish uchun qaysi elementdag foydalanish maqsadga muvofiq.
14. Kiritish vositalari yordamida hosil qilinadigan KIRISH xabarlarining texnik vositalarini sanib bering.
15. Interfeys yaratishning asosiy tamoyillari sanab bering.
16. Ma‘lumotlarni eksport qilish vositalari va komponentalari deganda nima nazarda tutiladi?
17. Turli xil xususiyatlarga ega bo‘lgan ma‘lumotlarni birlashtiruvchi vosita – bu nima?

2-SEMESTR UCHUN
AMALIYOT MASHG‘ULOTLARI MATERIALLARI

№1-AMALIY MASHG’ULOT

NOMA’LUM TURLAR VA NOMA’LUM NOMLAR FAZOSI

Ishning maqsadi: C++ dasturlash tilida obyektga yo‘naltirilgan dasturlash tamoyilaridan foydalanib noma’lum nomlar fazosini tashkil qilish ko‘nikmalarini shakllantirish.

Masalaning qo‘yilishi:

Variantda berilgan class, konstruktor va class maydonlari yordamida noma’lum nomlar fazosini yarating. Class da obyektlar massivi ustida quyidagi funksiyalarni bajaruvchi 3 ta static metod yozing:

- Qidirish metodi

- Saralash metodi

- O‘zgartirish metodi

Yuqoridagi metodlar uchun class maydonlaridan ixtiyoriysi tanlansin.

Chelak va tog‘ora class larini yarating va ularni bog‘lovchi funksiyalar yarating

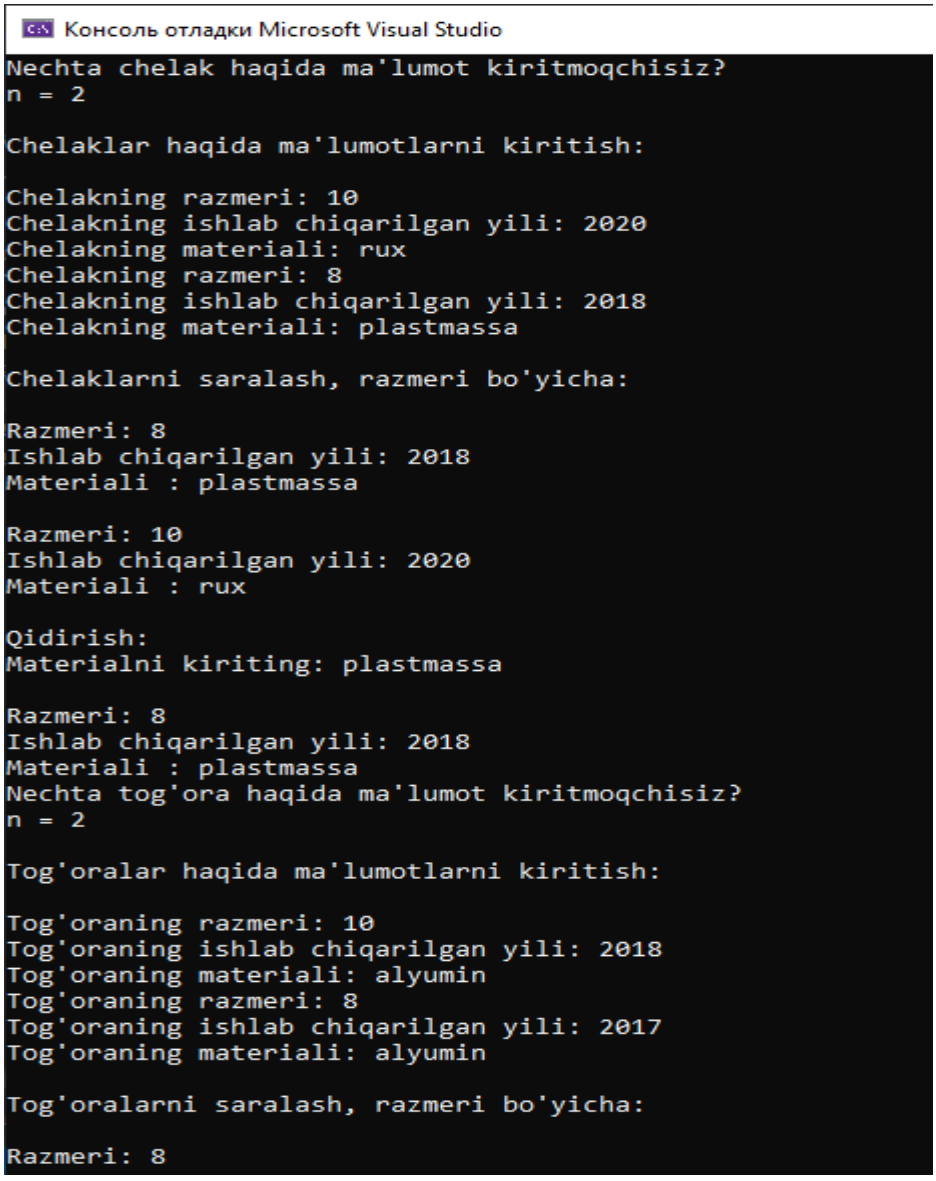
Dastur:

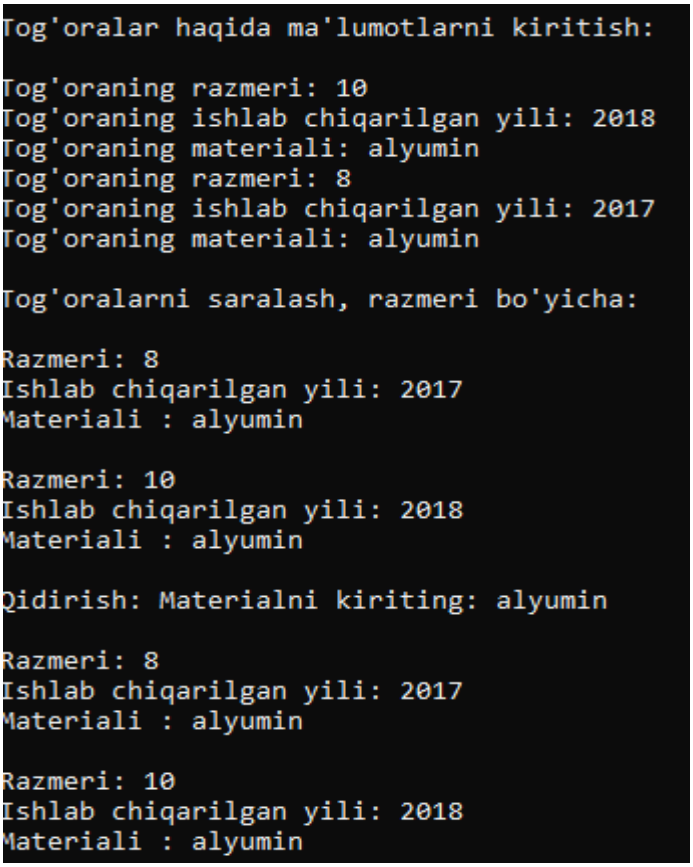
```
#include<iostream>
usingnamespace std;

namespace yangi_nomlar_fazosi { //Yangi nomlar fazosi
	classChelak {
	protected:
		int razmer, ishlab_chiqarilgan_yili;
		string materiali;

	public:
		Chelak() {}
		Chelak(intrazmer, intyil, stringmateriali) {
			this->razmer = razmer;
			this->ishlab_chiqarilgan_yili = yil;
			this->materiali =materiali;
		}
		staticvoid qidirish(Chelak* massiv, intn, stringmat) { //qidirish materiali bo'yicha
			for (int i = 0; i <n; i++) {
				if (massiv[i].materiali ==mat) massiv[i].print();
			}
		}
		staticvoid saralash(Chelak* massiv, intn) { //saralash, razmeri bo'yicha
			for (int i = 0; i <n; i++) {
				for (int j = i + 1; j <n; j++) {
```

```
                                        if (massiv[i].razmer >massiv[j].razmer)
                                                swap(massiv[i], massiv[j]);    } } }
        void ozgartirish(intrazmer, intyil, stringmateriali) {//o'zgartirish metodi
                this->razmer = razmer;
                this->ishlab_chiqarilgan_yili = yil;
                this->materiali =materiali;
        }
        void print() {
                cout <<"\nRazmeri: "<< razmer
                        <<"\nIshlab chiqarilgan yili: "<< ishlab_chiqarilgan_yili
                        <<"\nMateriali : "<< materiali << endl;
        }       };
classTogora :publicChelak {
public:         Togora(){}
        Togora(intrazmer, intyil, stringmateriali) {
                this->razmer = razmer;
                this->ishlab_chiqarilgan_yili = yil;
                this->materiali =materiali;
        }       };}
int main() {
        usingnamespace yangi_nomlar_fazosi; // Yangi hosil qilingan nomlar fazosidan foydalanish
        int n, m;       cout <<"Nechta chelak haqida ma'lumot kiritmoqchisiz?\nn = "; cin >> n;
        Chelak* chelaklar;
        chelaklar = newChelak[n];
        if (n) cout <<"\nChelaklar haqida ma'lumotlarni kiritish:\n\n";
        for (int i = 0; i < n; i++) {
                int razmer, yili;
                string materiali;
                cout <<"Chelakning razmeri: "; cin >> razmer;
                cout <<"Chelakning ishlab chiqarilgan yili: "; cin >> yili;
                cout <<"Chelakning materiali: "; cin >> materiali;
                chelaklar[i] =Chelak(razmer, yili, materiali);
        }
        cout <<"\nChelaklarni saralash, razmeri bo'yicha:\n";
        chelaklar[0].saralash(chelaklar, n);
        //Ekranga chiqarish:
        for (int i = 0; i < n; i++) {
                chelaklar[i].print();
        }
        cout <<"\nQidirish: \nMaterialni kiriting: ";
        string man; cin >> man;
        chelaklar[0].qidirish(chelaklar, n, man);
        cout <<"Nechta tog‘ora haqida ma'lumot kiritmoqchisiz?\nn = "; cin >> m;
        Togora* togoralar;
        togoralar = newTogora[m];
        if (m) cout <<"\nTog‘oralar haqida ma'lumotlarni kiritish:\n\n";
        for (int i = 0; i < m; i++) {
                int razmer, yili;
                string material;
```

```
            cout <<"Tog‘oraning razmeri: "; cin >> razmer;
            cout <<"Tog‘oraning ishlab chiqarilgan yili: "; cin >> yili;
            cout <<"Tog‘oraning materiali: "; cin >> material;
            togoralar[i] =Togora(razmer, yili, material);
        }       cout <<"\nTog‘oralarni saralash, razmeri bo'yicha:\n";
        togoralar[0].saralash(togoralar, m);
        //Ekranga chiqarish:
        for (int i = 0; i < m; i++) {
            togoralar[i].print();
        }
        cout <<"\nQidirish: Materialni kiriting: ";
        cin >> man;
        togoralar[0].qidirish(togoralar, m, man);

}
```

Natija:





0

1.2 Classlar o‘rtasida vorislikni tashkil qilish. Overriding va overloading bo‘ladigan funksiyalarni yaratish

Ishning maqsadi: Merosxo‘rlik imkoniyatlaridan foydalanib dasturlash ko‘nikmalarini shakllantirish.

Topshiriq:1 - jadvaldagi berilgan vazifalar uchun ajdod class va voris class larni yarating. Voris class da quyidagi funksiyalarni yozing:

- Konstruktor.
- Uchinchi ustundagi qo‘yilgan vazifani bajaruvchi funksiya.
- Avlod obyektlari haqida ma’lumotlarini chop qiluvchi funksiya.

Ajdod va voris class larning obyektlari haqida ma’lumotlarni kiritish va chiqarishni bajaruvchi friend funksiyalar va ularning shablonini yarating.

12.	Tovar: nomi, narxi,	Tovar: *nomi, narxi, ishlab*	Ishlab chiqarilganiga ikki yildan oshgan tovarlar uchun chegirma
	ishlab chiqaruvchi	*chiqaruvchi*, ishlab chiqarilgan yil, foizli chegirmasi	narxlarni chiqaring.

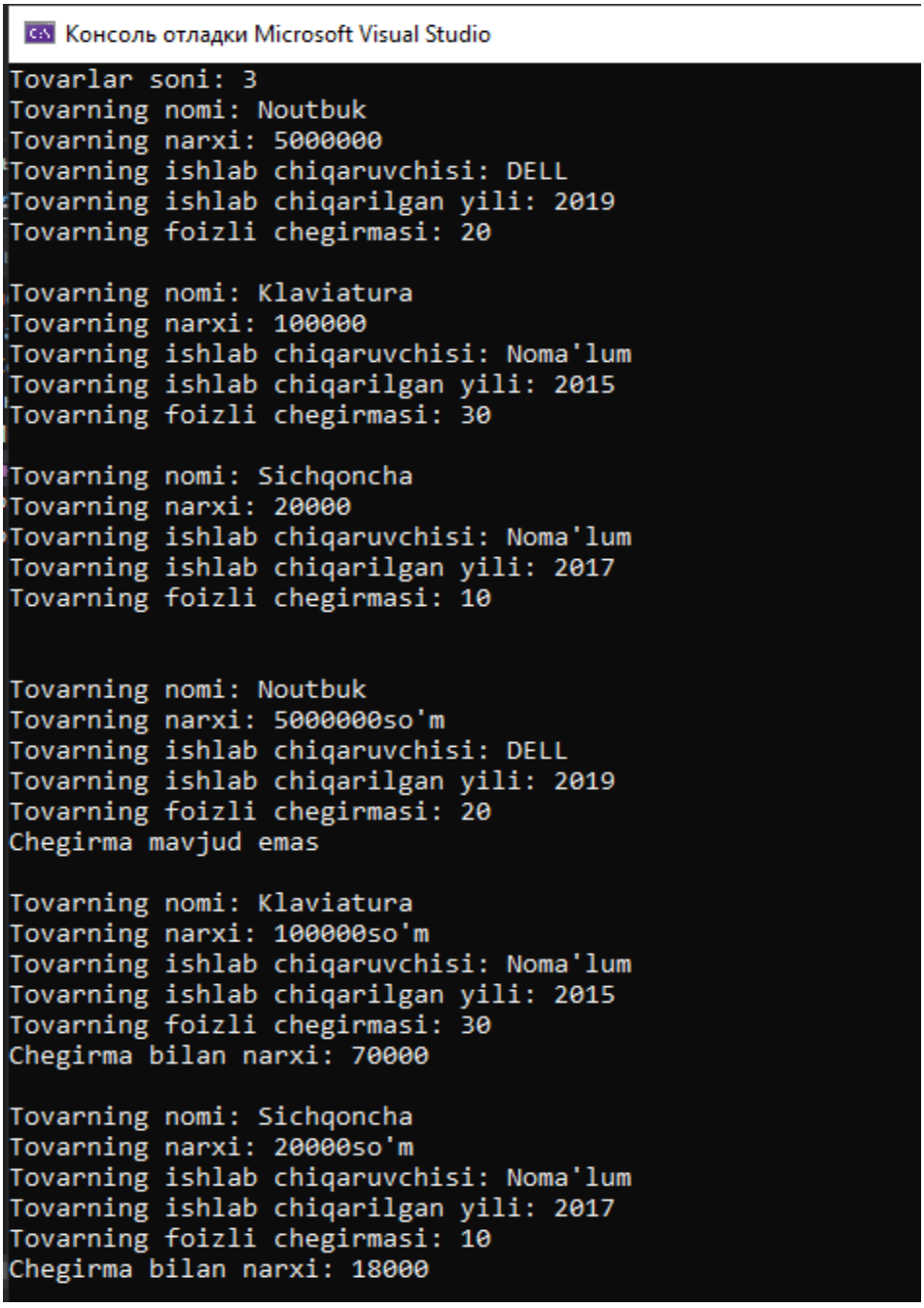
Dastur:

```
#include<iostream>
#include<string>
constint HOZIRGI_YIL = 2020;
usingnamespace std;
classTovar { // Ajdod sinf
protected:
	string nomi;
	int narxi;
	string ishlab_chiqaruvchi;
	//friend funksiyalar shablonlari
	template<classT>
	friendvoid kirit(T&);
	template<classT>
	friendvoid chiqar(T);
};
classTovar2 :publicTovar {
	int yili;
	int chegirma;

public:
	//Konstruktorlar:
	Tovar2() {}
	Tovar2(stringnomi, intnarxi, stringish, intyil, intchegirma) {
		this->nomi =nomi;
		this->narxi = narxi;
		ishlab_chiqaruvchi =ish;
		this->yili = yil;
		this->chegirma = chegirma;
	}
	//Uchinchi ustundagi vazifani bajaruvchi metod:
	void uchinchi() {
		if (HOZIRGI_YIL - yili >= 2) {
			int narx = narxi * (100 - chegirma) / 100.0;
```

```
                cout <<"Chegirma bilan narxi: "<< narx<<endl;
            }
            else {
                cout <<"Chegirma mavjud emas\n";
            }
        }
        //chiqarish:
        void print() {
            cout << endl;
            chiqar(*this);
        }
        //friend funksiyalar shablonlari
        template<classT>
        friendvoid kirit(T&);
        template<classT>
        friendvoid chiqar(T);
};
//kiritish funksiyasi shabloni
template<classT>
void kirit(T&tovar) {
        cout <<"Tovarning nomi: ";
        cin.ignore();
        getline(cin, tovar.nomi);
        cout <<"Tovarning narxi: "; cin >>tovar.narxi;
        cout <<"Tovarning ishlab chiqaruvchisi: "; cin.ignore(); getline(cin,
tovar.ishlab_chiqaruvchi);
        if (typeid(&tovar) ==typeid(&Tovar2())) {
            cout <<"Tovarning ishlab chiqarilgan yili: "; cin >>tovar.yili;
            cout <<"Tovarning foizli chegirmasi: "; cin >>tovar.chegirma;
        }
        cout << endl;
}
//chiqarish funksiyasi shabloni
template<classT>
void chiqar(Ttovar) {
        cout <<"Tovarning nomi: "<<tovar.nomi << endl
            <<"Tovarning narxi: "<<tovar.narxi <<"so'm"<< endl
            <<"Tovarning ishlab chiqaruvchisi: "<<tovar.ishlab_chiqaruvchi << endl;
        if (typeid(&tovar) ==typeid(&Tovar2())) {
            cout <<"Tovarning ishlab chiqarilgan yili: "<<tovar.yili << endl;
            cout <<"Tovarning foizli chegirmasi: "<<tovar.chegirma<<endl;
        }
}
int main() {
        int n;
        cout <<"Tovarlar soni: "; cin >> n;
        Tovar2* tovarlar;
        tovarlar = newTovar2[n];
        //Kiritish
```

```
for (int i = 0; i < n; i++) {
        kirit(tovarlar[i]);
}
//Uchinchi ustundagi shart + chiqarish:
for (int i = 0; i < n; i++) {
        tovarlar[i].print();
        tovarlar[i].uchinchi();
}
}
```

Natija:



1.3 OYD ning merosxo‘rlik va polimorfizm xususiyatlaridan foydalangan holda vazifaga muvofiq dasturlar yaratish

Ishning maqsadi: Polimorfizm(overload va override) metodlarni qayta e’lon qilish va qayta yuklash orqali dasturlash ko‘nikmalarini shakllantirish.

Topshiriq: OYD ning merosxo‘rlik va polimorfizm xususiyatlaridan foydalangan holda vazifaga muvofiq dasturni yozing.

- Barcha classlarda class obyektlari yaratilishi kerak bo'lgan konstruktorlarni yozing.
- Yaratiladigan obyektlarning parametrlari klaviaturadan (talaba xohlaganidek) kiritiladi va parametr sifatida konstruktorga uzatiladi.
- Obyektlar haqida barcha ma'lumotlar chiqarilsin.
- Virtual va do‘st funksiyalar hamda do‘st class lar yaratilsin.

“Kinofilm” classini yarating. Maydonlari: Nomlanishi, Rejessor, davomiyligi (minut da), aktyorlar soni. “Narx” nomli virtual metod yozing, kinofilm narxi quyidagi formula orqali hisoblanadi: davomiyligi * 20 + aktyorlarning soni * 30, lekin agar rejissori = “Stiven Spilberg” yoki “Djeyms Kemeron” bo’lsa narxi ikki barobarga qimmat bo’ladi. Shuningdek, film haqidagi ma'lumotlarni ekranga chiqaruvchi “Axborot” metodini yozing: Nomlanishi, rejissori, davomiyligi, aktyorlar soni va narxi.

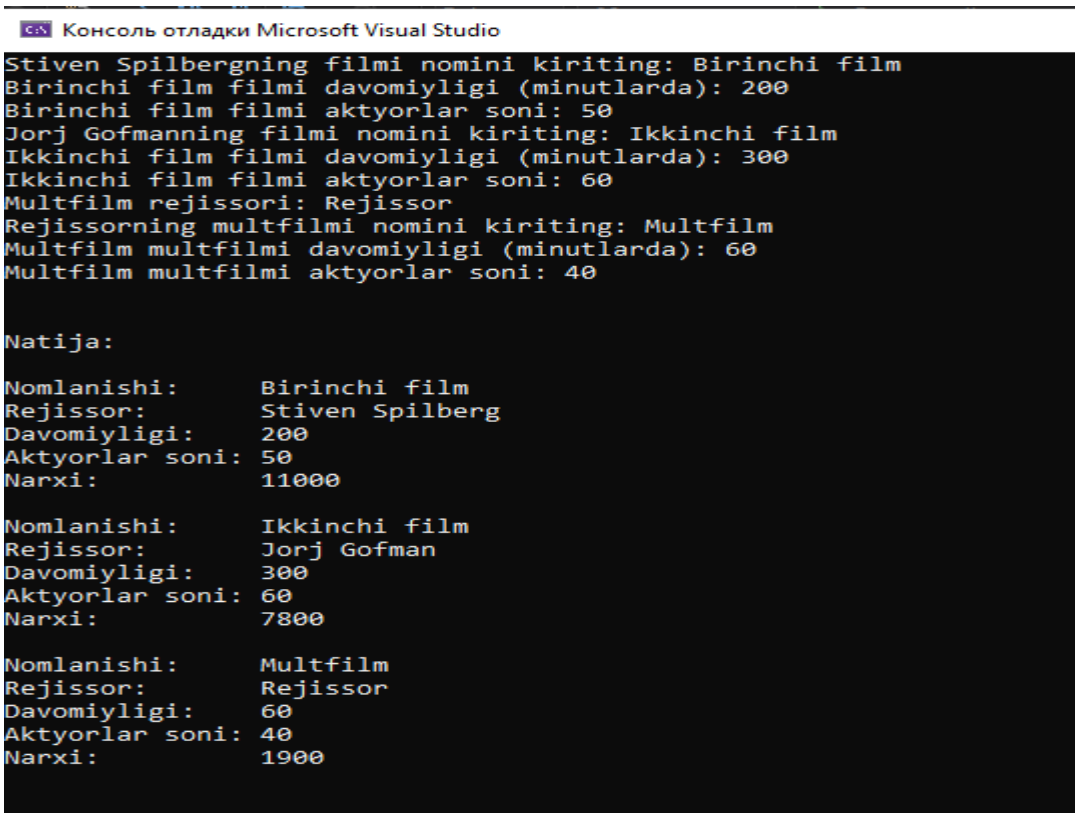
Bundan tashqari, **“Multifilm”** avlod classini yarating. Qayta yuklangan “Narx” metodi formulasi quyidagicha: davomiyligi * 25 + aktyorlarning soni * 10 (rejissyordan qat'i nazar).

Asosiy dasturda 2 ta **film** yarating rejissori “Stiven Spilberg” va “Jorj Gofman” filmlari, **multifilm** yarating va ular haqida ma’lumotni chiqaring.

Dastur:

```
#include<iostream>
#include<string>
usingnamespace std;
conststring QIMMAT_REJISSOR_1 = "Stiven Spilberg", QIMMAT_REJISSOR_2="Djeyms Kemeron";
classKinofilm {
protected:
	string nomlanishi, rejissor;
	int davomiyligi; //minutlarda
	int aktyorlar_soni;
public:
	Kinofilm(stringnom, stringrej, intdav, intakt) {
		nomlanishi =nom;
		rejissor =rej;
		davomiyligi = dav;
		aktyorlar_soni = akt;
```

```
        }
        virtualint narx() {
                int narxi = davomiyligi * 20 + aktyorlar_soni * 30;
                if (rejissor == QIMMAT_REJISSOR_1 || rejissor == QIMMAT_REJISSOR_2) {
                        narxi *= 2;
                }
                return narxi;
        }
        void axborot() {
                cout <<"Nomlanishi:     "<< nomlanishi << endl
                        <<"Rejissor:      "<< rejissor << endl
                        <<"Davomiyligi:    "<< davomiyligi << endl
                        <<"Aktyorlar soni: "<< aktyorlar_soni << endl
                        <<"Narxi:         "<< narx() <<"\n\n";}};
classMultfilm :publicKinofilm {
public: Multfilm(stringnom, stringrej, intdav, intakt) : Kinofilm(nom, rej, dav, akt){}
        int narx() {                return davomiyligi * 25 + aktyorlar_soni * 10;        }};
int main() {
        //Ikkita film kiritish:
        string nom, rej = QIMMAT_REJISSOR_1;
        int dav, akt;
        cout << rej <<"ning filmi nomini kiriting: "; getline(cin, nom);
        cout << nom <<" filmi davomiyligi (minutlarda): "; cin >> dav;
        cout << nom <<" filmi aktyorlar soni: "; cin >> akt; cin.ignore();
        Kinofilm birinchi = Kinofilm(nom, rej, dav, akt);
        rej ="Jorj Gofman";
        cout << rej <<"ning filmi nomini kiriting: "; getline(cin, nom);
        cout << nom <<" filmi davomiyligi (minutlarda): "; cin >> dav;
        cout << nom <<" filmi aktyorlar soni: "; cin >> akt; cin.ignore();
        Kinofilm ikkinchi = Kinofilm(nom, rej, dav, akt);
        cout <<"Multfilm rejissori: "; getline(cin, rej);
        cout << rej <<"ning multfilmi nomini kiriting: "; getline(cin, nom);
        cout << nom <<" multfilmi davomiyligi (minutlarda): "; cin >> dav;
        cout << nom <<" multfilmi aktyorlar soni: "; cin >> akt; cin.ignore();
        Multfilm mulfilm = Multfilm(nom, rej, dav, akt);
        cout <<"\n\nNatija:\n\n";
        birinchi.axborot();
        ikkinchi.axborot();
        mulfilm.axborot();
        }
```
Natija:



Nazorat savollari:

1. Namespace nima va u qanday afzalliklarga ega?
2. Turdan-turga o‘tish operatorlaridan qaysilarini bilasiz?
3. static_cast nima va u qanday vazifani bajaradi?
4. dynamic_cast nima va u qanday vazifani bajaradi?
5. const_cast nima va u qanday vazifani bajaradi?
6. reinterpret_cast nima va u qanday vazifani bajaradi?
7. static_cast va dynamic_cast operatorlarining farqini tushuntirib bering.
8. bad_cast nima va u qanday vazifani bajaradi?
9. Noma’lum turlar bilan ishlash uchun qaysi ilovadan(kutubxonadan) foydalaniladi.
10. Noma’lum turlar bilan ishlovchi funksiyalarni sanab bering.
1. Konteyner(kolleksiya) deganda nima tushuniladi?
2. Massiv va konteyner orasidagi farqni tushuntirib bering.
3. STL(Standart Template Library) qanday kutubxona hisoblanadi?
4. Ketma-ket konteynerlarga qaysilar kiradi?
5. Array konteyneri sinfi va uning metodlari haqida nimalarni bilasiz?
6. Vector konteyneri sinfi va uning metodlari haqida nimalarni bilasiz?
7. Deque konteyneri sinfi va uning metodlari haqida nimalarni bilasiz?
8. List konteyneri sinfi va uning metodlari haqida nimalarni bilasiz?
9. Forward_list konteyneri sinfi va uning metodlari haqida nimalarni bilasiz?
10. Ketma-ket konteynerlardan foydalanish uchun qaysi kutubxonalardan foydalaniladi?

Topshiriqlar.

Topshiriq №1

1.Klavyatura va sichqoncha class larini yarating va ularni bog‘lovchi metodlar yarating.

2.Kompyuter monitor va video kartasini class larini yarating va ularni bog‘lovchi funksiyalar yarating.

3.Telefon va notebook class larini yarating va ularni bog‘lovchi funksiyalar yarating.

4.Universitet va kollej class larini yarating va ularni bog‘lovchi funksiyalar yarating.

5.Kollej va maktab class larini yarating va ularni bog‘lovchi funksiyalar yarating.

6.Piyola va choynak class larini yarating va ularni bog‘lovchi funksiyalar yarating.

7.Ruchka va qalam class larini yarating va ularni bog‘lovchi funksiyalar yarating.

8.Kastyum va shim class larini yarating va ularni bog‘lovchi funksiyalar yarating.

9.Stol va stul class larini yarating va ularni bog‘lovchi funksiyalar yarating.

10.Kitob va daftar class larini yarating va ularni bog‘lovchi funksiyalar yarating.

11.Eshik va deraza class larini yarating va ularni bog‘lovchi funksiyalar yarating.

12.Chelak va tog‘ara class larini yarating va ularni bog‘lovchi funksiyalar yarating.

13.Kalit va qulf class larini yarating va ularni bog‘lovchi funksiyalar yarating.

14.Scet va sham class larini yarating va ularni bog‘lovchi funksiyalar yarating.

15.Dom va hovli class larini yarating va ularni bog‘lovchi funksiyalar yarating.

16.Muzlatgich va mikrovalovka class larini yarating va ularni bog‘lovchi funksiyalar yarating.

17.Printer va skanner class larini yarating va ularni bog‘lovchi funksiyalar yarating.

18.Mashina va motosikl class larini yarating va ularni bog‘lovchi funksiyalar yarating.

19.Ovqat va ichimlik class larini yarating va ularni bog‘lovchi funksiyalar yarating.

20.Fleshka va disk class larini yarating va ularni bog‘lovchi funksiyalar yarating.

21.Tufli va tapichka class larini yarating va ularni bog‘lovchi funksiyalar yarating.

22.Inson va hayvon class larini yarating va ularni bog‘lovchi funksiyalar yarating.

№2-AMALIY MASHG'ULOT

KONTEYNERLAR (KOLLEKSIYALAR).

Ishning maqsadi: C++ dasturlash tilida konteynerlar (Kolleksiyalar), STL kutubxonasini ishlatilishi va unga tegishli bo'lgan konteyner turlarini o'rganins hamda undan foydalanish ko'nikmalarini shakllantirish.

Masalaning qo'yilishi:

Variantda berilgan konteynerlar turini aniqlash va masalada berilgan shart asosida konteyner yarating. Konteynerda tuplam ustida quyidagi funksiyalarni bajaruvchi dastur tuzing. – **Navbat elementlarini shakllantiring;**

Navbat elementlari yig'indisini hisoblang.

Yuqoridagi amallarni bajarish uchun qiymatlar intejer tipda tanlansin.

Chelak va tog'ora class larini yarating va ularni bog'lovchi funksiyalar yarating

Dastur:

Namuna:

1-masala: Navbat berilgan elementlari yig'indisini topadigan dastur yarating.

```
#include<iostream>
#include<deque>
using namespace std;
int main()
{
int sum=0;
deque<int> deq;
deq.push_back(11);
deq.push_back(7);
deq.push_back(-8);
deq.push_back(100);

while(!deq.empty())
{
sum+=deq.front();
```

```
deq.pop_front();


}
cout<<sum;


return 0;


}
```

Natija:110

Masalani yechish uchun oldin deque sinfi uchun kutubxona e’lon qilinadi. push_back() funksiyasi orali navbatga elementlarni qo’shib boriladi, qachonki navbat bo’sh bo’lmasa sum=0 o’zgaruvchisiga pop_front() funksiyasi orqali qo’shib boriladi va yig’indi hisoblanadi.

2-masala: Ro’yhat berilgan elementlarini chop qiling va oxirgi elemetini aniqlang;

```
#include<iostream>
#include<list>
using namespace std;
void show(list<int>newlist)
{
list<int>::iterator i;
for(i=newlist.begin(); i!=newlist.end();i++)
cout<<*t;
cout<<endl;
}

int main()
{
list<int> newlist;
newlist.push_back(-100);
newlist.push_back(26);
newlist.push_back(-17);
```

newlist.push_back(-11);

newlist.push_back(133);

show(list);

list.pop_back();

}

Dastur natijasi:

-100 26 -17 -11 133

133

Masalani yechish uchun oldin list sinfi uchun kutubxona e'lon qilinadi. push_back() funksiyasi orali ro'yhatga elementlarni qo'shib boriladi,

show funksiyasi yaratiladi ushbu funksiya ro'yhatning elementlarini iterator orqali chop qilinadi. Ro'yhatning oxirgi elementini chop qilish uchun esa pop_back() funsiyasidan foydalaniladi.

3-masala: Vektorning juft elementlarini va juft indeksidagi elementlarini yig'indisini toping.

```
#include <iostream>
#include <vector>
using namespace std;
void showVectorElements(vector<int>vek)
{
    cout<<"Vektorning elementlari:"<<endl;
    for (int i = 0; i < vek.size(); i++)
    cout<<"vek["<<i<<"]="<< vek.at(i) << endl;
}
int showVectorSum(vector<int>vek)
{    int s=0;
        for (int i = 0; i < vek.size(); i++)
    {
        if(vek.at(i)%2==0)
        s+=vek.at(i);
    }
    return s;
}
int showVectorIndexSum(vector<int>vek)
{
    int s=0;
    for (int i = 0; i < vek.size(); i+=2)
    s+=vek.at(i);
    return s;
}
```

```
int main(int argc, char** argv) {

  vector<int> vek;
  for (int i = 1; i <=40; i+=7)
  vek.push_back(i);
  showVectorElements(vek);
  cout<<"Vektorning juft  elementlari yig'indisi s="<<showVectorSum(vek)<<endl;
  cout<<endl;
  cout<<"Vektorning juft indeksidagi elementlari yig'indisi"
  "\n(indeksi 0 dan boshlangan deb hisobga olingan): s="<<showVectorIndexSum(vek);

    return 0;
}
```

Dastur natijasi:

```
Vektorning elementlari:
vek[0]=1
vek[1]=8
vek[2]=15
vek[3]=22
vek[4]=29
vek[5]=36
Vektorning juft  elementlari yig'indisi s=66

Vektorning juft indeksidagi elementlari yig'indisi
(indeksi 0 dan boshlangan deb hisobga olingan): s=45
```

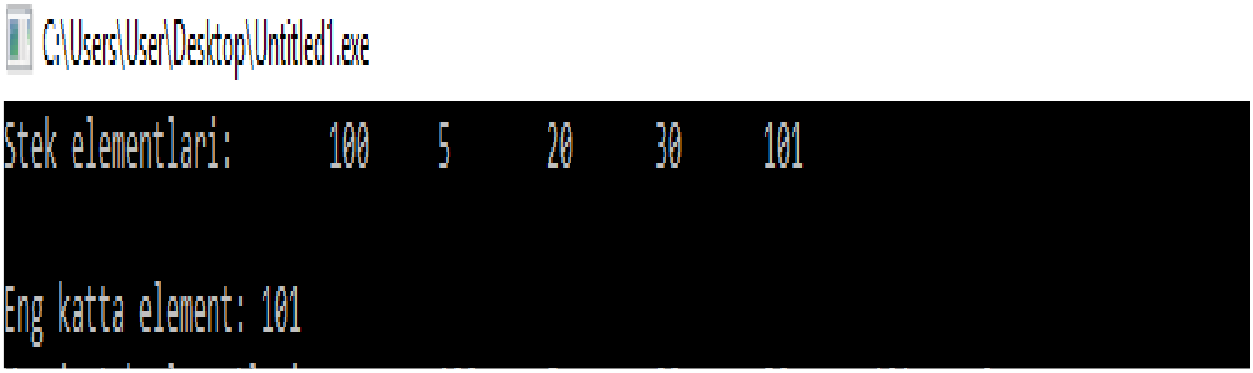
Masalani yechish uchun oldin vector sinfi uchun kutubxona e'lon qilinadi. push_back() funksiyasi orali vectorga elementlarni qo'shib boriladi,

showVectorElements(), showVectorSum(), showVectorIndexSum() funksiyalari yaratiladi ushbu funksiya mos ravishda vectorning elementlarini ,juft elementalarini yig'indisini, juft indexsida turgan elementlarini yig'indisini chop qilinadi. at() funsiyasidan foydaaniladi.

4-masala: Stackning maximum elementi topilsin.

```
#include <iostream>
#include <stack>
using namespace std;

//Stek elementlarini ko'rsatuvchi funksiya
```

```
void stackElementlari(stack <int> s)
{
while (!s.empty())
{
cout << '\t' << s.top();
s.pop();
}
cout << '\n';
}
//Stekdan max elementni topuvchi funksiya
int maxElement(stack <int> s){
int max = s.top();
while (!s.empty())
{
if(s.top() > max)
{
max = s.top();
}
s.pop();
}
return max;
}


//Asosiy funksiya
int main ()
{
stack <int> s;
s.push(101);
s.push(30);
s.push(20);
s.push(5);
s.push(100);
cout << "Stek elementlari: ";
stackElementlari( s);
int max = maxElement (s);
cout << "\nEng katta element: " << max;
return 0;
}
```



Masalani yechish uchun oldin stack sinfi uchun kutubxona e'lon qilinadi. push() funksiyasi orali stekga elementlarni qo'shib boriladi,

stackElementlari(), maxElement() funksiyalari yaratiladi ushbu funksiyalar mos ravishda stekning elementlarini va maximum elementini chop qiladi.

Qachonki stek bo'sh bo'lmasa ya'ni empty() bo'lmasa top() funksiyasi orqali elementlari chop qiladi.maximum topish uchun esa oxirgi joylashgan elementini max qilib olinadi va boshqalari bilan solishtirib chiqiladi.

Topshiriq №2

1. Ro'yxatdagi eng kichik elementini toping.
2. Ro'yxat berilgan ekrandan kirtilgan sondan qiymati katta elementlarni toping.
3. Ro'yxatdagi eng katta elementini aniqlang.
4. Ro'yxat berilgan ekrandan kirtilgan sondan qiymati kichik elementlarni toping.
5. Ro'yxat berilgan ekrandan kirtilgan sonning toq qiymatdagi elementlarini ekranga chiqaring.
6. Ro'yxatning ikkinchi elementiga ko'rsatkich qaytaruvchi funksiya yozing.
7. Ro'yxat berilgan uning juft qiymatdagi elementlarini ekranga chiqaring.
8. Ro'yxatdagi oxirgi elementga ko'rsatkich qaytaruvchi funksiya yozing.
9. Ro'yxat berilgan ekrandan kiritilgan so'zga teng bo'lgan elementi bor yo'qligini aniqlang.
10. Birinchi elementni har qanday turdagi elementlari uchun ro'yxatdan olib tashlaydigan funksiyani yozing
11. Barcha ro'yxat elementlarini ekranga chop etuvchi funksiyani yozing.

12. Ro‘yxat berilgan ekrandan kiritilgan songa teng bo‘lgan elementlarni toping.
13. Ro‘yxatning nusxasini qurish, lekin teskari tartibda.
14. VECTOR hosil qiling elementlarini kamayish tartibida saralang
15. VECTOR hosil qiling elementlarini o‘sish tartibida saralang
16. VECTOR elementlaridan birinchi uchragan manfiy elementini chop qiling.
17. VECTOR hosil qiling va uning tub elementlarini yig‘indisini toping
18. VECTOR hosil qiling va uning toq qiymatdagi elementlarini yig‘indisini toping
19. VECTOR hosil qiling va uning mukammal son bo‘lgan i elementlarini ko‘paytmasini toping

(mukammal son deb bo‘luvchilari yig‘indisi(o‘zidan tashqari) o‘ziga teng bo‘lgan songa aytiladi. Masalan 6=1+2+3 ;

20. VECTOR hosil qiling va 7 ga karrali elementlar sonini uning yigindisiga nisbati topilsin.
21. Navbat o‘rtasidagi element o‘chirib tashlansin. Agar navbat elementlari soni toq bo‘lsa, bitta element, aks holda ikkita element o‘chirilsin.
22. Navbatni juft o‘rinda turgan elementlari o‘chirilsin.
23. Navbat eng kichik elementi topilsin va undan keyin 0 joylashtirilsin.
24. Navbat eng kichik elementi o‘chirilsin.
25. Navbat o‘rtasiga '+' belgi joylashtirilsin.
26. Stek birinchi va oxirgi elementlari o‘rni almashtirilsin.
27. Stek elementlari teskari tartibda joylashtirib chiqilsin.
28. Stek o‘rtasidagi element o‘chirib tashlansin. Agar stek elementi toq bo‘lsa, bitta element, aks holda ikkita element o‘chirilsin.
29. Stekning juft o‘rinda turgan elementlari o‘chirilsin.
30. Stek eng katta elementi topilsin va undan keyin 0 joylashtirilsin
31. Dekning har 2 ta elementidan keyin ularning yig‘indisini joylang.
32. Dekning o‘rtasidagi element yoki elementlarni o‘chiring.

№3-AMALIY MASHG'ULOT

MAVZU: KONTEYNER SINFLAR. CHIZIQLI KONTEYNERLAR

Ishdan maqsad: : C++ dasturlash tilida chiziqli konteynerlar array, vector va deque to'plamlar va ularning ustida bajarilishi mumkin bo'lgan amallarni o'rganish hamda ko'nikmalarini egallsh.

Masalaning array, vector va deque dan qay bir to'plamga tegishli ekanligini aniqlash va yaratilgan to'plam elementlarini **array, vector va deque** ning maxsus funksiyalari yordamida qayta ishlash.

Dastur

1.1. To'plamning yarim elementlarini o'chirish.

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

void printInt (int number);

int main (){

    vector<int> myVec;
    vector<int>::iterator first, last;
    for (long i=0; i<10; i++){
        myVec.push_back(i);
    }
    first = myVec.begin();
       last = myVec.begin() + 5;
    if (last >= myVec.end()){
        return - 1;
    }
    myVec.erase(first, last);
    for_each (myVec.begin(), myVec.end(), printInt);
       system("pause");
    return 0;
}
void printInt (int number)
{
cout << number << endl;
```

```
}
```

1.1-dastur natijasi. Output

```
5
6
7
8
9
```

Dastur

1.2-dastur. To‘plamning sig‘imini tekshirish.

```
#include "stdafx.h"
#include <iostream>
#include <vector>

using namespace std;
int main(){
vector<int> vec;
cout << "Real size of array in vector: " << vec.capacity() << endl;
for (int j = 0; j < 25; j++){
    vec.push_back (10);
}
cout << "Real size of array in vector: " << vec.capacity() << endl;
system("pause");
return 0;
}
```

1.2-dastur natijasi. Output

```
Real size of array in vector: 0
Real size of array in vector: 28
```

Dastur

Navbat berilgan elementlari yig’indisini topadigan dastur yarating.

#include<iostream>

#include<deque>

```
using namespace std;
int main()
{
int sum=0;
deque<int> deq;
deq.push_back(11);
deq.push_back(7);
deq.push_back(-8);
deq.push_back(100);

while(!deq.empty())
{
sum+=deq.front();
deq.pop_front();

}
cout<<sum;

return 0;

}
```

Natija:110

Masalani yechish uchun oldin deque sinfi uchun kutubxona e'lon qilinadi. push_back() funksiyasi orali navbatga elementlarni qo'shib boriladi, qachonki navbat bo'sh bo'lmasa sum=0 o'zgaruvchisiga pop_front() funksiyasi orqali qo'shib boriladi va yig'indi hisoblanadi.

Topshiriqlar:

№	Topshiriq №1
1	Arraydagi eng kichik elementini toping.

2	Array berilgan ekrandan kirtilgan sondan qiymati katta elementlarni toping.
3	Arraydagi eng katta elementini aniqlang.
4	Array berilgan ekrandan kirtilgan sondan qiymati kichik elementlarni toping.
5	Array berilgan ekrandan kirtilgan sonning toq qiymatdagi elementlarini ekranga chiqaring.
6	Arrayning ikkinchi elementiga ko‘rsatkich qaytaruvchi funksiya yozing.
7	Array berilgan uning juft qiymatdagi elementlarini ekranga chiqaring.
8	Arraydagi oxirgi elementga ko‘rsatkich qaytaruvchi funksiya yozing.
9	Array berilgan ekrandan kiritilgan so’zga teng bo’lgan elementi bor yo’qligini aniqlang.
10	Birinchi elementni har qanday turdagi elementlari uchun Arraydan olib tashlaydigan funksiyani yozing
11	Barcha Array elementlarini ekranga chop etuvchi funksiyani yozing.
12	Array berilgan ekrandan kiritilgan songa teng bo’lgan elementlarni toping.
13	Arrayning nusxasini qurish, lekin teskari tartibda.
14	VECTOR hosil qiling elementlarini kamayish tartibida saralang
15	VECTOR hosil qiling elementlarini o’sish tartibida saralang
16	VECTOR elementlaridan birinchi uchragan manfiy elementini chop qiling.
17	VECTOR hosil qiling va uning tub elementlarini yig’indisini toping
18	VECTOR hosil qiling va uning toq qiymatdagi elementlarini yig’indisini toping
19	VECTOR hosil qiling va uning mukammal son bo’lgan i elementlarini ko’paytmasini toping (mukammal son deb bo’luvchilari yig’indisi(o’zidan tashqari) o’ziga teng bo’lgan songa aytiladi. Masalan 6=1+2+3 ;
20	VECTOR hosil qiling va 7 ga karrali elementlar sonini uning yigindisiga nisbati topilsin.

№	**Topshiriq №2**
1	Navbat o‘rtasidagi element o‘chirib tashlansin. Agar navbat elementlari soni toq bo‘lsa, bitta element, aks holda ikkita element o‘chirilsin.
2	Navbatni juft o‘rinda turgan elementlari o‘chirilsin.
3	Navbat eng kichik elementi topilsin va undan keyin 0 joylashtirilsin.

4	Navbat eng kichik elementi o‘chirilsin.
5	Navbat o‘rtasiga '+' belgi joylashtirilsin.
6	Stek birinchi va oxirgi elementlari o‘rni almashtirilsin.
7	Stek elementlari teskari tartibda joylashtirib chiqilsin.
8	Stek o‘rtasidagi element o‘chirib tashlansin. Agar stek elementi toq bo‘lsa, bitta element, aks holda ikkita element o‘chirilsin.
9	Stekning juft o‘rinda turgan elementlari o‘chirilsin.
10	Stek eng katta elementi topilsin va undan keyin 0 joylashtirilsin
11	Dekning har 2 ta elementidan keyin ularning yig‘indisini joylang.
12	Dekning o‘rtasidagi element yoki elementlarni o‘chiring.
13	Dekning juft elementlari yig‘indisini hisoblang.
14	Berilgan so‘zning unli harflarini dekning chap tomonidan, undoshlarini o‘ng tomondan kiriting.
15	Dekning toq elementlaridan navbat, juft elementlaridan stek hosil qiling.
16	Dekdagi manfiy sonlarni o‘chiring.

Nazorat savollari

1. Konteyner(kolleksiya) deganda nima tushuniladi?
2. Massiv va konteyner orasidagi farqni tushuntirib bering.
3. STL(Standart Template Library) qanday kutubxona hisoblanadi?
4. Ketma-ket konteynerlarga qaysilar kiradi?
5. Array konteyneri sinfi va uning metodlari haqida nimalarni bilasiz?
6. Vector konteyneri sinfi va uning metodlari haqida nimalarni bilasiz?
7. Deque konteyneri sinfi va uning metodlari haqida nimalarni bilasiz?
8. Ketma-ket konteynerlardan foydalanish uchun qaysi kutubxonalardan foydalaniladi?

№4-AMALIY MASHG’ULOT

MAVZU: LIST VA FORWARD_LIST KONTEYNERLAR

Ishdan maqsad: : C++ dasturlash tilida **list va forward_list konteynerlar** va uning metodlaridan foydalanish ko‘nikmalarini egallash.

Masalaning qo‘yilishi: list va forward_list konteynerlar asosida yaratilgan to‘plam elementlarini **list va forward_list konteynerlar** ning maxsus funksiyalari yordamida qayta ishlash.

Dastur

1.1.Listga elementlarni qo’shish

```
#include <iostream>
#include <list>
int main()
{
   std::list<char> charList;
   int x = 0;
   for (int i=0; i<10; ++i)
      charList.push_front(65 + i);
   std::list<char>::iterator iter;
   for (iter = charList.begin();
         iter != charList.end(); iter++)
   {
      std::cout << "Element #" << x++ << ": "<< *iter << std::endl;
   }
   return 0;   }
```

Natija:

Element #0: J

Element #1: I

Element #2: H

Element #3: G

Element #4: F

Element #5: E

Element #6: D

Element #7: C

Element #8: B

Element #9: A

Dastur

1.2. Ro'yxatning istalgan joyidagi elementlarni o’chirish

```
#include <iostream>
#include <list>
int main()
{
   std::list<char> charList;
   for (int x=0; x<10; ++x)
      charList.push_front(65 + x);
      std::cout<<"Asosiy list: "<<std::endl;
```

```
std::list<char>::iterator iter;
   for (iter = charList.begin();
         iter != charList.end(); iter++)
   {
      std::cout << *iter;
   }
   std::list<char>::iterator start =
      charList.begin();
   charList.erase(++start);
   std::cout<<std::endl;
//list elementi o'chirilgandan keyingi holat
     std::cout<<"Natija list: "<<std::endl;
   for (iter = charList.begin();
         iter != charList.end(); iter++)
   {
      std::cout << *iter;
   }
   return 0;
}
Natija:
Asosiy list:
JIHGFEDCBA
Natija list:
JHGFEDCBA
```

Topshiriqlar:

№	Topshiriq №1
1	Ro’yxatdagi eng kichik elementini toping.
2	Ro’yxat berilgan ekrandan kirtilgan sondan qiymati katta elementlarni toping.
3	Ro’yxatdagi eng katta elementini aniqlang.
4	Ro’yxat berilgan ekrandan kirtilgan sondan qiymati kichik elementlarni toping.
5	Ro’yxat berilgan ekrandan kirtilgan sonning toq qiymatdagi elementlarini ekranga chiqaring.
6	Ro‘yxatning ikkinchi elementiga ko‘rsatkich qaytaruvchi funksiya yozing.
7	Ro’yxat berilgan uning juft qiymatdagi elementlarini ekranga chiqaring.

8	Ro‘yxatdagi oxirgi elementga ko‘rsatkich qaytaruvchi funksiya yozing.
9	Ro’yxat berilgan ekrandan kiritilgan so’zga teng bo’lgan elementi bor yo’qligini aniqlang.
10	Birinchi elementni har qanday turdagi elementlari uchun ro‘yxatdan olib tashlaydigan funksiyani yozing
11	Barcha ro‘yxat elementlarini ekranga chop etuvchi funksiyani yozing.
12	Ro’yxat berilgan ekrandan kiritilgan songa teng bo’lgan elementlarni toping.
13	Ro‘yxatning nusxasini qurish, lekin teskari tartibda.
14	IKKI TOMONLAMA RO’YHAT hosil qiling elementlarini kamayish tartibida saralang
15	IKKI TOMONLAMA RO’YHAT hosil qiling elementlarini o’sish tartibida saralang
16	IKKI TOMONLAMA RO’YHAT elementlaridan birinchi uchragan manfiy elementini chop qiling.
17	IKKI TOMONLAMA RO’YHAT hosil qiling va uning tub elementlarini yig’indisini toping
18	IKKI TOMONLAMA RO’YHAT hosil qiling va uning toq qiymatdagi elementlarini yig’indisini toping
19	IKKI TOMONLAMA RO’YHAT hosil qiling va uning mukammal son bo’lgan i elementlarini ko’paytmasini toping (mukammal son deb bo’luvchilari yig’indisi(o’zidan tashqari) o’ziga teng bo’lgan songa aytiladi. Masalan 6=1+2+3 ;
20	IKKI TOMONLAMA RO’YHAT hosil qiling va 7 ga karrali elementlar sonini uning yigindisiga nisbati topilsin.

Nazorat savollar

1. List konteyneriga ta’rif bering?
2. List konteynerining bir o‘lchovli massivdan ustun tomonini aytib bering?
3. forward_list konteyneriga ta’rif bering?
4. forward_list konteynerining bir o‘lchovli massivdan ustun tomonini aytib bering?
5. forward_list konteynerining list konteyneri bilan farqini aytib bering?
6. List konteyneri qanday ro’yxat?
7. forward_list konteyneri qanday ro’yxat?

№5-AMALIY MASHG'ULOT

MAVZU: ASSOTSIATIV VA TARTIBLANMAGAN ASSOTSIATIV KONTEYNERLAR

Ishdan maqsad: : C++ dasturlash tilida **set va multiset** sinflari va uning metodlaridan foydalanish ko'nikmalarini egallash.

Masalaning qo'yilishi: set va **multiset** asosida yaratilgan to'plam elementlarini **set va mutiset** ning maxsus funksiyalari yordamida qayta ishlash.

1- Masala: Berilgan **int** turidagi to'plam qiymatlari 2- to'plamda nechta marta qatnashganligini aniqlovchi va ularni ekranga chiqaruvchi dastur tuzing. Ikkala to'plamdan ham bir xil qiymatli elementlar o'chirilib Saralangan to'plam hosil qilinsin hamda uning qiymatlari ekranga chiqarilsin.

Masalani yechish g'oyasi: set konteyneri to'plami yaratiladi. **list** ning **insert()** iteratoridan foydalanib, A to'plamga qiymatlar o'zlashtiriladi. Z to'plamga esa A to'plam qiymatlarini juftlari yoziladi.

Dastur matni:

```
1.  #include "stdafx.h"
2.  #include <set>
3.  #include<iostream>
4.  #include<stdlib.h>
5.  #include<time.h>
6.  using namespace System;
7.  using namespace std;
8.  int main()
9.  {  srand(time(NULL));
10. set<int> s; int n;
11. cout<<"Elementlar sonini: "; cin>>n;
12. multiset<int> M, M2;
13. for (int i = 0; i < n; i++)
14. { int j = rand()%n+n;        M.insert(j);
15. j = rand()%n+n;     M2.insert(j);
16. }
17. auto k2 = M.begin();
18. set<int> s2;
19. for (int i = 0; i < n; i++)
20. {
21. s2.insert(*k2); k2++;
22. }
23. cout<<endl<<"M to'plam elementlari:"<<endl;
24. for (auto i=M.begin(); i!=M.end(); i++)
25. {
26. cout<<*i<<" ";
27. }
28. cout<<endl;
```

```
cout<<endl<<"M2 to'plam elementlari:"<<endl;
for (auto i=M2.begin(); i!=M2.end(); i++)
{
cout<<*i<<" ";
}
cout<<endl<<endl<<endl<<"M1 ning elementlari M2 to'plamda qatnashganlari soni:"<<endl;
int soni=0;
for (auto i=s2.begin(); i!=s2.end(); i++)
{
for (auto j=M2.begin(); j!=M2.end(); j++)
{
if(M2.count(*i)){
if(*i==*j) {soni++; }
}
}
if(soni>0)cout<<*i<<" -"<<soni<<" marta "<<endl;
else {cout<<*i<<" - qatnashmagan "<<endl;}
soni = 0;
}
//cout<<endl<<"Soni:"<<soni<<endl;
auto k = M.begin();
auto l = M2.begin();
for (int i = 0; i < n; i++)
{
s.insert(*k);k++;
s.insert(*l); l++;
}
cout<<endl;
cout<<endl<<"Saralangan to'plam elementlari:"<<endl;
for (auto i=s.begin(); i!=s.end(); i++)
{
cout<<*i<<" ";
}
getchar();
return 0;
}
```

Dastur natijasi:

Elementlar sonini: 25
M to'plam elementlari:
25 27 27 27 27 28 29 30 31 31 31 32 33 33 35 35 37 37 40 42 42 43 43 47 48
M2 to'plam elementlari:
26 27 30 31 31 31 31 34 35 37 38 39 42 43 45 45 45 46 46 46 46 47 47 49 49
M1 ning elementlari M2 to'plamda qatnashganlari soni:
25 - qatnashmagan
27 -1 marta
28 - qatnashmagan
29 - qatnashmagan

30 -1 marta
31 -4 marta
32 - qatnashmagan
33 - qatnashmagan
35 -1 marta
37 -1 marta
40 - qatnashmagan
42 -1 marta
43 -1 marta
47 -2 marta
48 - qatnashmagan

Saralangan to'plam elementlari:
25 26 27 28 29 30 31 32 33 34 35 37 38 39 40 42 43 45 46 47 48 49

2- Masala: Telefon classi yaratilsin va obyektlarni set konteyneriga joylashtiring. Tartiblash narxlarining o‘sish tartibida bo‘lsin. Telefonlar haqida to‘liq ma’lumot chop etilsin.

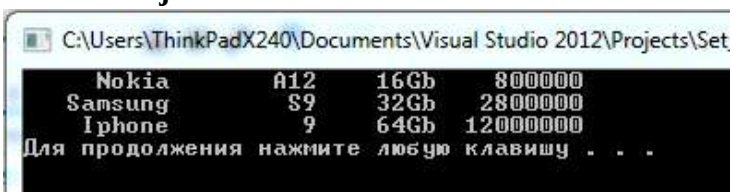
Masalani yechish g‘oyasi: Telefon classi yaratiladi. Set konteyneri qoidasiga ko‘ra uning elementlari ma’lum qoida asosida tartiblanishi kerak. Shu sababli telefon classi da so‘ralgan maydon bo‘yicha tartiblash uchun munosabat operatorini qayta yuklashga to‘g‘ri keladi. So‘ngra asosiy funksiyada set konteyneri yaratilib unga telefon obyektlarimizni joylashtiramiz.

Dastur matni:

```
#include <iostream>
#include <set>
#include <iterator>
#include <algorithm>
#include <iomanip>
#include <string>
using namespace std;
class Telefon{
public: string nomi, versiyasi, xotirasi;
                int yili, narxi;
                Telefon(string nomi, string versiyasi, string xotirasi, int yili, int narxi){
                        this->nomi=nomi; this->versiyasi=versiyasi; this->xotirasi=xotirasi;
                        this->yili=yili; this->narxi=narxi;
                }
                void chop_qil(){

        cout<<setw(10)<<nomi<<setw(10)<<versiyasi<<setw(8)<<xotirasi<<setw(5)<<setw(10)
<<narxi<<endl;
                }

                bool operator<(const Telefon& right) const
                {
```

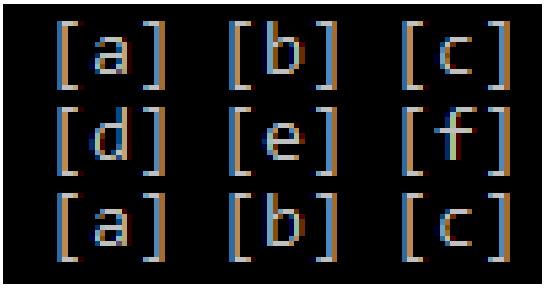
```
                    if(narxi<=right.narxi) return true;
                    else return false;
                }
};
void show(Telefon i){i.chop_qil();}

int main(){
        Telefon t("Nokia","A12","16Gb",2019,800000);
        Telefon t1("Iphone","9","64Gb",2020,12000000);
        Telefon t2("Samsung","S9","32Gb",2017,2800000);
        set<Telefon>st;
        st.insert(t);
        st.insert(t1);
        st.insert(t2);
        for_each(st.begin(),st.end(), show);

   system("pause");
}
```

Dastur natijasi:



unordered_set::swap ga misol:

```cpp
#include "stdafx.h"
#include <unordered_set>
#include <iostream>

typedef std::unordered_set<char> Myset;
int main()
{
Myset c1;

c1.insert('a');
c1.insert('b');
c1.insert('c');

// display contents " [c] [b] [a]"
for (Myset::const_iterator it = c1.begin();
it != c1.end(); ++it)
std::cout << " [" << *it << "]";
std::cout << std::endl;

Myset c2;

c2.insert('d');
c2.insert('e');
c2.insert('f');

c1.swap(c2);

// display contents " [f] [e] [d]"
for (Myset::const_iterator it = c1.begin();
it != c1.end(); ++it)
std::cout << " [" << *it << "]";
std::cout << std::endl;

swap(c1, c2);

// display contents " [c] [b] [a]"
for (Myset::const_iterator it = c1.begin();
it != c1.end(); ++it)
std::cout << " [" << *it << "]";
std::cout << std::endl;
getchar();
return (0);
}
```



Nazorat savollari

1. Assotsiativ konteyner(kolleksiya) deganda nima tushuniladi?
2. Massiv va assotsiativ konteyner orasidagi farqni tushuntirib bering.
3. Assotsiativ tartiblangan konteynerlarni sanab bering.
4. Tartiblanmagan assotsiativ konteynerlarni sanab bering.
5. Set va multiset konteynerlari sinfi va uning metodlari haqida nimalarni bilasiz?
6. Unordered_set va unordered_multiset konteynerlari sinfi va uning metodlari haqida nimalarni bilasiz?
7. Assotsiativ konteynerlarni boshqa turdagi konteynerlardan farqi nimada?
8. Shaxsiy class obyektlarini assotsiativ konteynerlarga joylashtirish uchun qanday amallarni bajarish kerak bo‘ladi?

5-topshiriq. set va multiset sinflaridan foydalanib, amaliy dasturlar yaratish

Ishning maqsadi: C++ dasturlash tilida set va multiset sinflari va uning metodlaridan foydalanish ko‘nikmalarini egallsh.

Masalaning qo‘yilishi:

set va multiset asosida yaratilgan to‘plam elementlarini set va mutiset ning maxsus funksiyalari yordamida qayta ishlash.

Berilgan vazifalarni bajarishda kiruvchi qiymatlar matinli fayldan o‘qib olinishi kerak.

1	Berilgan **int** turidagi to‘plam qiymatlarining juftlari 2- to‘plamda nechta marta qatnashganligini aniqlovchi va ularni ekranga chiqaruvchi dastur tuzing. Ikkala to‘plamdan ham bir xil qiymatli elemetlar o‘chirilib Saralangan to‘plam hosil qilinsin hamda uning qiymatlari ekranga chiqarilsin.
2	Berilgan **int** turidagi to‘plam qiymatlarining toqlari 2- va 3- to‘plamda nechta marta qatnashganligini aniqlovchi va ularni ekranga chiqaruvchi dastur tuzing. Ikkala to‘plamdan ham bir xil qiymatli elemetlar o‘chirilib Saralangan to‘plam hosil qilinsin hamda uning qiymatlari ekranga chiqarilsin.
3	Berilgan **int** turidagi to‘plam qiymatlarining tublari 2- to‘plamda nechta marta qatnashganligini aniqlovchi va ularni ekranga chiqaruvchi dastur tuzing. Ikkala

	to‘plamdan ham bir xil qiymatli elemetlar o‘chirilib Saralangan to‘plam hosil qilinsin hamda uning qiymatlari ekranga chiqarilsin.
4	Berilgan **float** turidagi to‘plam qiymatlarining butun(nuqtadan oldin) qismi 2- to‘plamga, kasr(nuqtadan keyin) qismi 3- to‘plamga joylashtiruvchi va ularni ekranga chiqaruvchi dastur tuzing. Ikkala to‘plamdan ham bir xil qiymatli elemetlar o‘chirilib Saralangan to‘plam hosil qilinsin hamda uning qiymatlari ekranga chiqarilsin.
5	Berilgan **float** turidagi to‘plamning 2 ta elementi yig‘indisidan 2- to‘plamni, ayirmasidan 3- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing. Ikkala to‘plamdan ham bir xil qiymatli elemetlar o‘chirilib Saralangan to‘plam hosil qilinsin hamda uning qiymatlari ekranga chiqarilsin.
6	Berilgan **float** turidagi to‘plamning 2 ta juft elementi yig‘indisidan 2- to‘plamni, ayirmasidan 3- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing. Ikkala to‘plamdan ham bir xil qiymatli elemetlar o‘chirilib Saralangan to‘plam hosil qilinsin hamda uning qiymatlari ekranga chiqarilsin.
7	Berilgan **string** turidagi to‘plamning 2 ta elementidan 2- to‘plamni, 3 ta elementidan 3- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing. Ikkala to‘plamdan ham bir xil qiymatli elemetlar o‘chirilib Saralangan to‘plam hosil qilinsin hamda uning qiymatlari ekranga chiqarilsin.
8	Berilgan **string** turidagi to‘plamning juft uzinlikka ega bo‘lgan elementidan 2- to‘plamni, toq uzinlikdagi elementidan 3- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing. Ikkala to‘plamdan ham bir xil qiymatli elemetlar o‘chirilib Saralangan to‘plam hosil qilinsin hamda uning qiymatlari ekranga chiqarilsin.
9	Berilgan string turidagi to‘plamning 2 ta so‘zdan iborat elementidan 2- to‘plamni, 1 ta so‘zdan iborat elementidan 3- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing. Ikkala to‘plamdan ham bir xil qiymatli elemetlar o‘chirilib Saralangan to‘plam hosil qilinsin hamda uning qiymatlari ekranga chiqarilsin.
10	Berilgan string turidagi to‘plamning 1 ta so‘zdan iborat elementlarini o‘chiruvchi va 1 ta so‘zdan iborat elementidan 2- to‘plamni hosil qiluvchi hamda ularni ekranga chiqaruvchi dastur tuzing.
11	Berilgan string turidagi to‘plamning kata harfdan iborat elementlarini o‘chiruvchi va kichik harfdan iborat elementidan 2- to‘plamni hosil qiluvchi hamda ularni ekranga chiqaruvchi dastur tuzing.
12	Berilgan string turidagi to‘plamda kata harfdan va kichik harfdan boshlanuvchi elementlari soni aniqlovchi hamda ularni ekranga chiqaruvchi dastur tuzing.
13	Berilgan string turidagi to‘plamning kata harfdan iborat elementlarini kichik harfga o‘girib, 2- to‘plamni va kichik harfdan iborat elementlarini katta harfga o‘girib, 3- to‘plamni hosil qiluvchi hamda ularni ekranga chiqaruvchi dastur tuzing.
14	Berilgan char turidagi to‘plamning 2 ta elementidan 2- to‘plamni, 3 ta elementidan 3- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing.
15	Berilgan char turidagi to‘plamning juft uzinlikka ega bo‘lgan elementidan 2- to‘plamni, toq uzinlikdagi elementidan 3- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing.
16	Berilgan char turidagi to‘plamning 2 ta so‘zdan iborat elementidan 2- to‘plamni, 1 ta so‘zdan iborat elementidan 3- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing.
17	Berilgan char turidagi to‘plamning 1 ta so‘zdan iborat elementlarini o‘chiruvchi va 1 ta so‘zdan iborat elementidan 2- to‘plamni hosil qiluvchi hamda ularni ekranga chiqaruvchi dastur tuzing.

18	Berilgan char turidagi to‘plamning kata harfdan iborat elementlarini o‘chiruvchi va kichik harfdan iborat elementidan 2- to‘plamni hosil qiluvchi hamda ularni ekranga chiqaruvchi dastur tuzing.
19	Berilgan char turidagi to‘plamda kata harfdan va kichik harfdan boshlanuvchi elementlari soni aniqlovchi hamda ularni ekranga chiqaruvchi dastur tuzing.
20	Berilgan char turidagi to‘plamning kata harfdan iborat elementlarini kichik harfga o‘girib, 2- to‘plamni va kichik harfdan iborat elementlarini katta harfga o‘girib, 3- to‘plamni hosil qiluvchi hamda ularni ekranga chiqaruvchi dastur tuzing.

№6-AMALIY MASHG’ULOT

MAVZU: MAP VA MULTIMAP SINFLARIDAN FOYDALANIB, AMALIY DASTURLAR YARATISH

Ishning maqsadi: C++ dasturlash tilida **map va mulmap** sinflari va uning metodlaridan foydalanish ko‘nikmalarini egallsh.

Masalaning qo‘yilishi: map va **multimap** asosida yaratilgan to‘plam elementlarini **map** va **mutimap** ning maxsus funksiyalari yordamida qayta ishlash.

1-Masala: Talabalarning haqida (familiya, ismi va stipendiyasi) **string** va **float** turidagi to‘plam berilgan. Familiyasi yoki ismi to‘liq yozilganlar va stipendiyasi N ga teng bo‘lganlaridan 2- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing.

Masalani yechish g‘oyasi: map konteyneri to‘plami yaratiladi. **map** ning **insert(pair<char,int>('a',10))** iteratoridan foydalanib, A to‘plamga qiymatlar o‘zlashtiriladi.

Dastur matni:

```
1.  #include "stdafx.h"
2.  #include <map>
3.  #include<iostream>
4.  #include<string>
5.  using namespace System;
6.  using namespace std;
7.  int main()
8.  {       int n; bool t=false;
9.  cout<<"Talabalar sonini kiriting: "; cin>>n; cin.ignore();
10. multimap<string,int> M, M2;
11. for (int i = 0; i < n; i++)
12. { string St;
13. int step;
14. cout<<i+1<<" - talaba ma'lumotlarini kiriting:"<<endl;
15. cout<<"Fam, ismi: "; getline(cin,St);
16. cout<<"Stipendiyasi: "; cin>>step; cin.ignore();
17. M.insert(pair<string, int>(St,step));
18. }
19. string S;
```

```
int N;
cout<<"Qanday stipendiya oladigan talaba haqida ma'lumot kerak?:";
cin>>N;
int k=0;
for (auto it = M.begin(); it != M.end(); ++it){
S = it->first;
if(S.find(" ")<S.length() && it->second == N){
M2.insert(pair<string, int>(S,N));t=true;}
}
cout<<"So'rov Natijasi:"<<endl;
if(t){ cout<<"bor"<<endl;
for (auto it = M2.begin(); it != M2.end(); ++it){
  cout<<it->first<<" "<<it->second<<endl;
}
}
system("pause");
return 0;
}
```

Dastur natijasi:

Talabalar sonini kiriting: 3

1 - talaba ma'lumotlarini kiriting:

Fam, ismi: Sobirov Sardor

Stipendiyasi: 400000

2 - talaba ma'lumotlarini kiriting:

Fam, ismi: Botirov Xusan

Stipendiyasi: 500000

3 - talaba ma'lumotlarini kiriting:

Fam, ismi: Asadullayev

Stipendiyasi: 400000

Qanday stipendiya oladigan talaba haqida ma'lumot kerak?: 400000

bor

So'rov Natijasi:

Sobirov Sardor 400000

Nazorat savollari

1. Assotsiativ konteyner(kolleksiya) deganda nima tushuniladi?
2. Massiv va assotsiativ konteyner orasidagi farqni tushuntirib bering.
3. Assotsiativ tartiblangan konteynerlarni sanab bering.
4. Tartiblanmagan assotsiativ konteynerlarni sanab bering.
5. Map va multimap konteynerlari sinfi va uning metodlari haqida nimalarni bilasiz?

6. Unordered_map va unordered_multimap konteynerlari sinfi va uning metodlari haqida nimalarni bilasiz?
7. Assotsiativ konteynerlarni boshqa turdagi konteynerlardan farqi nimada?
8. Shaxsiy class obyektlarini assotsiativ konteynerlarga joylashtirish uchun qanday amallarni bajarish kerak bo‘ladi?

6-topshiriq

map va multimap sinflaridan foydalanib, amaliy dasturlar yaratish

Ishning maqsadi: C++ dasturlash tilida **map va mulmap** sinflari va uning metodlaridan foydalanish ko‘nikmalarini egallsh.

Masalaning qo‘yilishi:

map va **multimap** asosida yaratilgan to‘plam elementlarini **map** va **mutimap** ning maxsus funksiyalari yordamida qayta ishlash.

Amaliyot mashg‘ulotini bajarish bo‘yicha variantlar

Berilgan vazifalarni bajarishda kiruvchi qiymatlar matinli fayldan o‘qib olinishi kerak.

1	Talabalar haqida (familiya, ismi, sharfi va telefon) to‘plam berilgan. FIO si to‘liq yozilganlar va telefoni T ga teng bo‘lganlaridan 2- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing.
2	Talabalar haqida (familiya, ismi, sharfi, gruhi va bahosi) to‘plam berilgan. Bahosi 5 bo‘lganlaridan 2- to‘plamni, bahosi 4 bo‘lganlaridan 3- to‘plamni, bahosi 3 bo‘lganlaridan 4- to‘plamni va fanni o‘zlashtira olmaganlaridan 5- to‘plamni hosil qiluvchi va 5 ta to‘plamni ekranga chiqaruvchi dastur tuzing.
3	Talabalar haqida (familiya, ismi, sharfi, viloyat, tuman va qishloq) to‘plam berilgan. Har bir viloyatda nechta talaba borligini aniqlovchi va viloya, tuman bo‘yich talabalarni qidiruvchi dastur tuzing.
4	Talabalar haqida (familiya, ismi, sharfi, fani, guruhi va bahosi) to‘plam berilgan. Ixtiyoriy qidirilayotgan guruhda har bir fandan nechta talaba 5 bahoga, nechtasi 4 bahoga, nechtasi 3 bahoga va nechtasi o‘zlashtira olamayotganligi aniqlovchi va bahosi bo‘yich talabalarni qidiruvchi dastur tuzing.
5	Kompyuterlar haqida (nomi, firmasi, xotirasi, yadrosi, yili va yadrosi) to‘plam berilgan. Kompyuterlarni yadrosi va firmasi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
6	Kompyuter viruslari haqida (nomi, zarar darajasi, tarqalish tezligi, turi, va internet_id) to‘plam berilgan. Kompyuter viruslarini nomi va tarqalish tezligi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
7	Kafedralar haqida (nomi, mudir_FIO, xodimlar soni, fanlar soni va professorlar soni) to‘plam berilgan. Kafedralarni nomi, mudir_FIO, fanlari soni va professorlar soni bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
8	Universitetlar haqida (nomi, rektor_FIO, xodimlar soni, fanlar soni, talabalar soni va professorlar soni) to‘plam berilgan. Universitetlarni nomi, rektor_FIO, xodimlar soni,

	fanlar soni, talabalar soni va professorlar soni bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
9	TTJ lar haqida (nomi, xokim_FIO, xonalar soni, talabalar soni, qizlar va bollar soni hamda nogironlar soni, nomeri) to‘plam berilgan. TTJ ni nomi, xokim_FIO, xonalar soni, talabalar soni, qizlar va bollar soni hamda nogironlar soni, nomeri bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
10	Maktablar lar haqida (nomi, direktor_FIO, xodimlar va o‘qituvchilar soni, qizlar va bollar soni, olinpiada g‘oliblari soni hamda maktab nomeri) to‘plam berilgan. Maktablarni nomi, direktor_FIO, xodimlar va o‘qituvchilar soni, qizlar va bollar soni, olinpiada g‘oliblari soni hamda maktab nomeri bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
11	Fanalar haqida (nomi, o‘qituvchi_FIO, 5 chilar soni, 4 chilar soni, 3 chilar soni va o‘zlashtira olmaganlar soni) to‘plam berilgan. Fanlarni nomi, o‘qituvchi_FIO, 5 chilar soni, 4 chilar soni, 3 chilar soni va o‘zlashtira olmaganlar soni bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
12	Operatsion tizimlar haqida (nomi, yili, razryadlar soni, turi va imkoniyatlarining soni) to‘plam berilgan. Opetarsion tizimlarni nomi, yili, razryadlar soni, turi va imkoniyatlarining soni bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
13	Avtomobillar haqida (nomi, yili, versiyasi va narxi) to‘plam berilgan. Avtomobillarni nomi, yili, versiyasi va narxi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
14	Ho‘l mavalar haqida (nomi, mazasi, rangi, turi va narxi) to‘plam berilgan. Ho‘l mevalarni nomi, mazasi, rangi, turi va narxi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
15	Korxona xodimlari haqida (FIO, yili, staji, lavozimi va oyligi) to‘plam berilgan. Xodimlarning familiyasi bir xillardan 2- to‘plamni, ismi bir xillardan 3- to‘plamni, staji bir xillardan 4- to‘plamni va oyligi bir xillardan 5- to‘plamni tashkil qiluvchi hamda xodimni FIO, yili, staji, lavozimi va oyligi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
16	Mashxur futbolchilar haqida (FIO, kuboklar soni, staji, mashxurlik darajasi va oyligi) to‘plam berilgan. Futbolchilarning familiyasi bir xillardan 2- to‘plamni, ismi bir xillardan 3- to‘plamni, staji bir xillardan 4- to‘plamni va oyligi bir xillardan 5- to‘plamni tashkil qiluvchi hamda futbolchini FIO, kuboklar soni, staji, mashxurlik darajasi va oyligi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
17	Kompyuter o‘yinlari haqida (nomi, turi, versiyasi, hajmi, zararli oqibatlari, foydalilik darajasi va aktuallik darajasi) to‘plam berilgan. Kompyuter o‘yinlarining nomi bir xillardan 2- to‘plamni, zararli oqibatlari bir xillaridan 3- to‘plamni, aktuallaik darajasi bir xillaridan 4- to‘plamni tashkil qiluvchi hamda Kompyuter o‘yinini nomi, turi, versiyasi, hajmi, zararli oqibatlari, foydalilik darajasi va aktuallik darajasi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
18	Mobil telefonlar haqida (nomi, turi, versiyasi, hajmi, foydalilik darajasi, yadrosi, kamera o‘lchami) to‘plam berilgan. Mobil telefonlarning nomi bir xillardan 2- to‘plamni, turi bir xillaridan 3- to‘plamni, yadrosi bir xillaridan 4- to‘plamni tashkil qiluvchi hamda Mobil telefonni nomi, turi, versiyasi, hajmi, foydalilik darajasi, yadrosi, kamera o‘lchami bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
19	Sinf xonalar haqida (nomeri, qavati, Javobgar shaxs_FIO, stollar soni, stellar soni, dars bo‘lish vaqtlari va kompyuterlar soni) to‘plam berilgan. Sinf xonalarning nomeri va qavati

	bir xillaridan 2- to‘plamni, Javobgar shaxs_FIO si bir xillaridan 3- to‘plamni, stollar soni bir xillaridan 4- to‘plamni, stullar soni bir xillaridan 5- to‘plamni, kompyuter soni bir xillaridan 6- to‘plamni tashkil qiluvchi hamda ixtiyoriy sana va vaqtda bo‘sh bo‘lgan xonalarni aniqlovchi dastur tuzing. Topilgan natijani faylga saqlashni tashkil qilsin.
20	O‘qituvchilar haqida (FIO, qabul qilish vaqti, fan nomi, lavozimi va fanlar soni) to‘plam berilgan. O‘qituvchilarning familiyasi va ismi bir xillaridan 2- to‘plamni, qabul qilish vaqti bir xillaridan 3- to‘plamni, lavozimi bir xillaridan 4- to‘plamni, fanlar soni bir xillaridan 5- to‘plamni tashkil qiluvchi hamda ixtiyoriy sana va vaqtda nechta va qaysi o‘qituvchilar talabalarni qabul qilishi mumkinligini aniqlovchi dastur tuzing. Topilgan natijani faylga saqlashni tashkil qilsin.

№7-AMALIY MASHG’ULOT
MAVZU:UNORDRED_MAP VA UNORDRED_MULTIMAP SINFLARIDAN FOYDALANIB, AMALIY DASTURLAR YARATISH

Ishning maqsadi: C++ dasturlash tilida **unordred_map va unordred_mulmap** sinflari va uning metodlaridan foydalanish ko‘nikmalarini egallsh.

Masalaning qo‘yilishi: unordred_map va **unordred_multimap** asosida yaratilgan to‘plam elementlarini **unordred_map** va **unordred_mutimap** ning maxsus funksiyalari yordamida qayta ishlash.

1-Masala: Talabalarning haqida (familiya, ismi va stipendiyasi) **string** va **float** turidagi to‘plam berilgan. Familiyasi yoki ismi to‘liq yozilganlar va stipendiyasi N ga teng bo‘lganlaridan 2- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing.

Masalani yechish g‘oyasi: unordred_map konteyneri to‘plami yaratiladi. **unordred_map** ning **insert(pair<char,int>('a',10))** iteratoridan foydalanib, A to‘plamga qiymatlar o‘zlashtiriladi.

Dastur matni:

```
#include <iostream>
#include <string>
#include <unordered_map>

int main()
{
   // Create an unordered_map of three strings (that map to strings)
   std::unordered_map<std::string, std::string> u = {
      {"RED","#FF0000"},
      {"GREEN","#00FF00"},
      {"BLUE","#0000FF"}
   };

   // Helper lambda function to print key-value pairs
   auto print_key_value = [](const auto& key, const auto& value) {
```

```
    std::cout << "Key:[" << key << "] Value:[" << value << "]\n";
  };

  std::cout << "Iterate and print key-value pairs of unordered_map, being\n"
            "explicit with their types:\n";
  for( const std::pair<const std::string, std::string>& n : u ) {
    print_key_value(n.first, n.second);
  }

  std::cout << "\nIterate and print key-value pairs using C++17 structured binding:\n";
  for( const auto& [key, value] : u ) {
    print_key_value(key, value);
  }

  // Add two new entries to the unordered_map
  u["BLACK"] = "#000000";
  u["WHITE"] = "#FFFFFF";

  std::cout << "\nOutput values by key:\n"
            "The HEX of color RED is:[" << u["RED"] << "]\n"
            "The HEX of color BLACK is:[" << u["BLACK"] << "]\n\n";

  std::cout << "Use operator[] with non-existent key to insert a new key-value pair:\n";
  print_key_value("new_key", u["new_key"]);

  std::cout << "\nIterate and print key-value pairs, using `auto`;\n"
            "new_key is now one of the keys in the map:\n";
  for( const auto& n : u ) {
    print_key_value(n.first, n.second);
  }
}
```

Possible output:

```
Iterate and print key-value pairs of unordered_map, being
explicit with their types:
Key:[BLUE] Value:[#0000FF]
Key:[GREEN] Value:[#00FF00]
Key:[RED] Value:[#FF0000]

Iterate and print key-value pairs using C++17 structured binding:
Key:[BLUE] Value:[#0000FF]
Key:[GREEN] Value:[#00FF00]
Key:[RED] Value:[#FF0000]

Output values by key:
The HEX of color RED is:[#FF0000]
The HEX of color BLACK is:[#000000]
```

```
Use operator[] with non-existent key to insert a new key-value pair:
Key:[new_key] Value:[]

Iterate and print key-value pairs, using `auto`;
new_key is now one of the keys in the map:
Key:[new_key] Value:[]
Key:[WHITE] Value:[#FFFFFF]
Key:[BLACK] Value:[#000000]
Key:[BLUE] Value:[#0000FF]
Key:[GREEN] Value:[#00FF00]
Key:[RED] Value:[#FF0000]
```

C++ dasturlash tilida unordered_multimap ning turli funksiyalarni namoyish qilish uchun dastur.

Dastur

```
#include <bits/stdc++.h>
using namespace std;
 // making typedef for short declaration
typedef unordered_multimap<string, int>::iterator unit;
// Utility function to print unordered_multimap
void printUmm(unordered_multimap<string, int> umm)
{
    // begin() returns iterator to first element of map
    unit it = umm.begin();
     for (; it != umm.end(); it++){
        cout << "<" << it->first << ", " << it->second
             << ">  ";
        cout << endl;
    }
}
// Driver code
int main()
{
    // empty initialization
    unordered_multimap<string, int> umm1;
```

```
 // Initialization by initializer list
unordered_multimap<string, int> umm2(
    { { "apple", 1 },
      { "ball", 2 },
      { "apple", 10 },
      { "cat", 7 },
      { "dog", 9 },
      { "cat", 6 },
      { "apple", 1 } });
 // Initialization by assignment operation
umm1 = umm2;
printUmm(umm1);
 // empty returns true, if container is empty else it
// returns false
if (umm2.empty())
    cout << "unordered multimap 2 is empty\n";
else
    cout << "unordered multimap 2 is not empty\n";
 // size returns total number of elements in container
cout << "Size of unordered multimap 1 is "
     << umm1.size() << endl;
 string key = "apple";
 // find and return any pair, associated with key
unit it = umm1.find(key);
if (it != umm1.end()) {
    cout << "\nkey " << key << " is there in unordered "
         << " multimap 1\n";
    cout << "\none of the value associated with " << key
         << " is " << it->second << endl;
}
else
```

```
        cout << "\nkey " << key
             << " is not there in unordered"
             << " multimap 1\n";
     // count returns count of total number of pair
    // associated with key
    int cnt = umm1.count(key);
    cout << "\ntotal values associated with " << key
        << " are " << cnt << "\n\n";
     printUmm(umm2);
     // one insertion by making pair explicitly
    umm2.insert(make_pair("dog", 11));
     // insertion by initializer list
    umm2.insert({ { "alpha", 12 }, { "beta", 33 } });
    cout << "\nAfter insertion of <alpha, 12> and <beta, "
         "33>\n";
    printUmm(umm2);
     // erase deletes all pairs corresponding to key
    umm2.erase("apple");
    cout << "\nAfter deletion of apple\n";
    printUmm(umm2);
     // clear deletes all pairs from container
    umm1.clear();
    umm2.clear();
     if (umm2.empty())
        cout << "\nunordered multimap 2 is empty\n";
    else
        cout << "\nunordered multimap 2 is not empty\n";
  }
```

Output

```
<dog, 9>
<cat, 6>
<cat, 7>
<ball, 2>
```

<apple, 1>
<apple, 10>
<apple, 1>
unordered multimap 2 is not empty
Size of unordered multimap 1 is 7

key apple is there in unordered multimap 1

one of the value associated with apple is 1

total values associated with apple are 3

<dog, 9>
<cat, 6>
<cat, 7>
<ball, 2>
<apple, 1>
<apple, 10>
<apple, 1>

After insertion of <alpha, 12> and <beta, 33>
<alpha, 12>
<dog, 11>
<dog, 9>
<beta, 33>
<cat, 6>
<cat, 7>
<ball, 2>
<apple, 1>
<apple, 10>
<apple, 1>

After deletion of apple
<alpha, 12>
<dog, 11>
<dog, 9>
<beta, 33>
<cat, 6>
<cat, 7>
<ball, 2>

unordered multimap 2 is empty

7-topshiriq

unordered_map va unordered_multimap sinflaridan foydalanib, amaliy dasturlar yaratish

Ishning maqsadi: C++ dasturlash tilida **unordered_map va unordered_mulmap** sinflari va uning metodlaridan foydalanish ko‘nikmalarini egallsh.

Masalaning qo‘yilishi:

unordered_map va **unordered_multimap** asosida yaratilgan to‘plam elementlarini **unordered_map** va **unordered_mutimap** ning maxsus funksiyalari yordamida qayta ishlash.

Amaliyot mashg‘ulotini bajarish bo‘yicha variantlar

Berilgan vazifalarni bajarishda kiruvchi qiymatlar matinli fayldan o‘qib olinishi kerak.

21	Talabalar haqida (familiya, ismi, sharfi va telefon) to‘plam berilgan. FIO si to‘liq yozilganlar va telefoni T ga teng bo‘lganlaridan 2- to‘plamni hosil qiluvchi va ularni ekranga chiqaruvchi dastur tuzing.
22	Talabalar haqida (familiya, ismi, sharfi, gruhi va bahosi) to‘plam berilgan. Bahosi 5 bo‘lganlaridan 2- to‘plamni, bahosi 4 bo‘lganlaridan 3- to‘plamni, bahosi 3 bo‘lganlaridan 4- to‘plamni va fanni o‘zlashtira olmaganlaridan 5- to‘plamni hosil qiluvchi va 5 ta to‘plamni ekranga chiqaruvchi dastur tuzing.
23	Talabalar haqida (familiya, ismi, sharfi, viloyat, tuman va qishloq) to‘plam berilgan. Har bir viloyatda nechta talaba borligini aniqlovchi va viloya, tuman bo‘yich talabalarni qidiruvchi dastur tuzing.
24	Talabalar haqida (familiya, ismi, sharfi, fani, guruhi va bahosi) to‘plam berilgan. Ixtiyoriy qidirilayotgan guruhda har bir fandan nechta talaba 5 bahoga, nechtasi 4 bahoga, nechtasi 3 bahoga va nechtasi o‘zlashtira olamayotganligi aniqlovchi va bahosi bo‘yich talabalarni qidiruvchi dastur tuzing.
25	Kompyuterlar haqida (nomi, firmasi, xotirasi, yadrosi, yili va yadrosi) to‘plam berilgan. Kompyuterlarni yadrosi va firmasi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
26	Kompyuter viruslari haqida (nomi, zarar darajasi, tarqalish tezligi, turi, va internet_id) to‘plam berilgan. Kompyuter viruslarini nomi va tarqalish tezligi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
27	Kafedralar haqida (nomi, mudir_FIO, xodimlar soni, fanlar soni va professorlar soni) to‘plam berilgan. Kafedralarni nomi, mudir_FIO, fanlari soni va professorlar soni bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
28	Universitetlar haqida (nomi, rektor_FIO, xodimlar soni, fanlar soni, talabalar soni va professorlar soni) to‘plam berilgan. Universitetlarni nomi, rektor_FIO, xodimlar soni, fanlar soni, talabalar soni va professorlar soni bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
29	TTJ lar haqida (nomi, xokim_FIO, xonalar soni, talabalar soni, qizlar va bollar soni hamda nogironlar soni, nomeri) to‘plam berilgan. TTJ ni nomi, xokim_FIO, xonalar soni, talabalar soni, qizlar va bollar soni hamda nogironlar soni, nomeri bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.

30	Maktablar lar haqida (nomi, direktor_FIO, xodimlar va o‘qituvchilar soni, qizlar va bollar soni, olinpiada g‘oliblari soni hamda maktab nomeri) to‘plam berilgan. Maktablarni nomi, direktor_FIO, xodimlar va o‘qituvchilar soni, qizlar va bollar soni, olinpiada g‘oliblari soni hamda maktab nomeri bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
31	Fanalar haqida (nomi, o‘qituvchi_FIO, 5 chilar soni, 4 chilar soni, 3 chilar soni va o‘zlashtira olmaganlar soni) to‘plam berilgan. Fanlarni nomi, o‘qituvchi_FIO, 5 chilar soni, 4 chilar soni, 3 chilar soni va o‘zlashtira olmaganlar soni bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
32	Operatsion tizimlar haqida (nomi, yili, razryadlar soni, turi va imkoniyatlarining soni) to‘plam berilgan. Opetarsion tizimlarni nomi, yili, razryadlar soni, turi va imkoniyatlarining soni bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
33	Avtomobillar haqida (nomi, yili, versiyasi va narxi) to‘plam berilgan. Avtomobillarni nomi, yili, versiyasi va narxi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
34	Ho‘l mavalar haqida (nomi, mazasi, rangi, turi va narxi) to‘plam berilgan. Ho‘l mevalarni nomi, mazasi, rangi, turi va narxi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
35	Korxona xodimlari haqida (FIO, yili, staji, lavozimi va oyligi) to‘plam berilgan. Xodimlarning familiyasi bir xillardan 2- to‘plamni, ismi bir xillardan 3- to‘plamni, staji bir xillardan 4- to‘plamni va oyligi bir xillardan 5- to‘plamni tashkil qiluvchi hamda xodimni FIO, yili, staji, lavozimi va oyligi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
36	Mashxur futbolchilar haqida (FIO, kuboklar soni, staji, mashxurlik darajasi va oyligi) to‘plam berilgan. Futbolchilarning familiyasi bir xillardan 2- to‘plamni, ismi bir xillardan 3- to‘plamni, staji bir xillardan 4- to‘plamni va oyligi bir xillardan 5- to‘plamni tashkil qiluvchi hamda futbolchini FIO, kuboklar soni, staji, mashxurlik darajasi va oyligi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
37	Kompyuter o‘yinlari haqida (nomi, turi, versiyasi, hajmi, zararli oqibatlari, foydalilik darajasi va aktuallik darajasi) to‘plam berilgan. Kompyuter o‘yinlarining nomi bir xillardan 2- to‘plamni, zararli oqibatlari bir xillaridan 3- to‘plamni, aktuallaik darajasi bir xillaridan 4- to‘plamni tashkil qiluvchi hamda Kompyuter o‘yinini nomi, turi, versiyasi, hajmi, zararli oqibatlari, foydalilik darajasi va aktuallik darajasi bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
38	Mobil telefonlar haqida (nomi, turi, versiyasi, hajmi, foydalilik darajasi, yadrosi, kamera o‘lchami) to‘plam berilgan. Mobil telefonlarning nomi bir xillardan 2- to‘plamni, turi bir xillaridan 3- to‘plamni, yadrosi bir xillaridan 4- to‘plamni tashkil qiluvchi hamda Mobil telefonni nomi, turi, versiyasi, hajmi, foydalilik darajasi, yadrosi, kamera o‘lchami bo‘yicha qidiruvchi hamda topilgan natijani faylga saqlashni taklif qiladigan dastur tuzing.
39	Sinf xonalar haqida (nomeri, qavati, Javobgar shaxs_FIO, stollar soni, stellar soni, dars bo‘lish vaqtlari va kompyuterlar soni) to‘plam berilgan. Sinf xonalarning nomeri va qavati bir xillaridan 2- to‘plamni, Javobgar shaxs_FIO si bir xillaridan 3- to‘plamni, stollar soni bir xillaridan 4- to‘plamni, stullar soni bir xillaridan 5- to‘plamni, kompyuter soni bir xillaridan 6- to‘plamni tashkil qiluvchi hamda ixtiyoriy sana va vaqtda bo‘sh bo‘lgan xonalarni aniqlovchi dastur tuzing. Topilgan natijani faylga saqlashni tashkil qilsin.
40	O‘qituvchilar haqida (FIO, qabul qilish vaqti, fan nomi, lavozimi va fanlar soni) to‘plam berilgan. O‘qituvchilarning familiyasi va ismi bir xillaridan 2- to‘plamni, qabul qilish vaqti

	bir xillaridan 3- to‘plamni, lavozimi bir xillaridan 4- to‘plamni, fanlar soni bir xillaridan 5- to‘plamni tashkil qiluvchi hamda ixtiyoriy sana va vaqtda nechta va qaysi o‘qituvchilar talabalarni qabul qilishi mumkinligini aniqlovchi dastur tuzing. Topilgan natijani faylga saqlashni tashkil qilsin.

№8-AMALIY MASHG’ULOT

MAVZU: KONTEYNERLAR. STACK VA NAVBAT, USTUVOR NAVBAT QO’LLANILISHI

Ishning maqsadi: C++ dasturlash tilida **stack va navbat, ustuvor navbat** hosil qilish va uning dasturini yaratish ko‘nikmalarini egallsh.

Masalaning qo‘yilishi:

Stekning eng kichik elementini toping?

Masala:

Masalani yechish g‘oyasi:

Stackga elementlar kiritiladi. Dastlab **min** ga stackning dastabki elementini qiymatini beramiz. So‘ngra stackning qolgan elemenlarini shu min element bilan solishtiramiz.

Stack va navbat, ustuvor navbat haqida

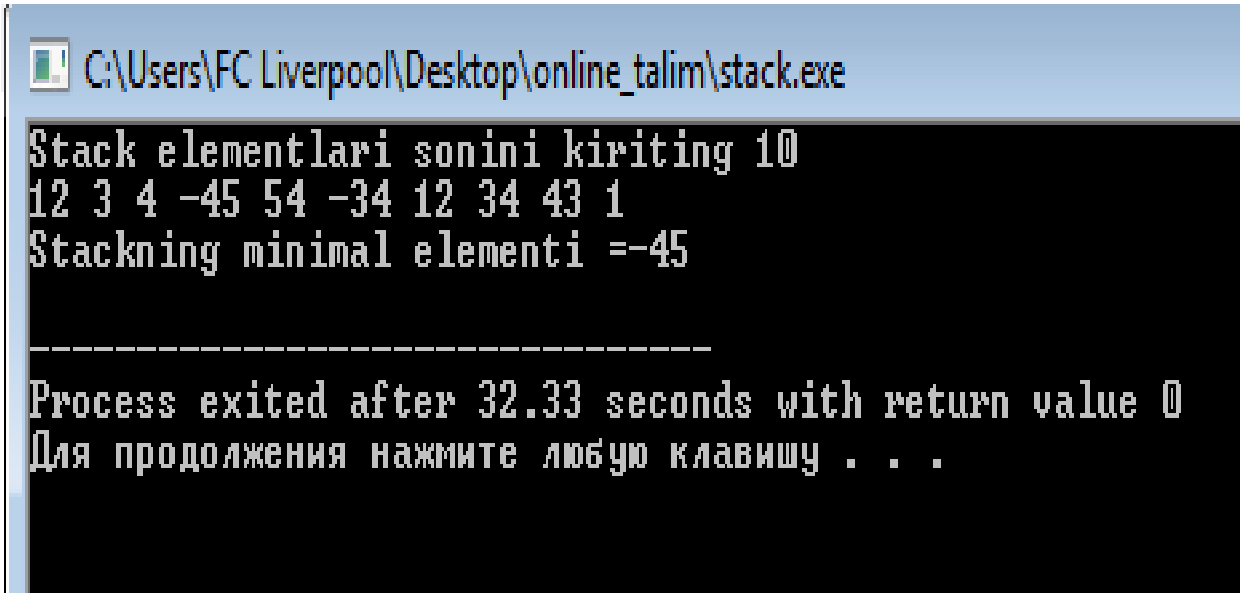
Stek deb shunday strukturaga aytiladiki, stekka kelib tushgan oxirgi elementga birinchi bo‘lib xizmat ko‘rsatiladi va stekdan chiqariladi. Mazkur ko‘rinishdagi xizmat ko‘rsatishni LIFO (Last input-First output, ya’ni oxirgi kelgan – birinchi ketadi) nomlash qabul qilingan. Stek bir tomondan ochiq bo‘ladi. Stack = g‘aram, to‘p, bo‘glam (angl.)

Biz navbat bilan ko‘p joylarda duch kelamiz: magazinda, o‘qishda, ishda va hokazo. Odatda biz unga e’tibor bermaymiz. Dasturiy tizimlarda ham bu navbat tushunchasi ishlatiladi. Masalan, hujjatni chop etish uchun printerga jo‘natsak, u navbatga turadi.

Dastur kodi:

```
#include "iostream"
#include "string.h"
#include "stdlib.h"
#include "stdio.h"
#include "iomanip"
#include "math.h"
```

```
#include "stack"
using namespace std;
int main()
{ stack <int> stek;
int imin,min,n,son;
cout<<"Stack elementlari sonini kiriting ";
cin>>n;
for(int i=0;i<n;i++)
{
        cin>>son;
        stek.push(son);
}
min=stek.top();
for(int i=n-1;i>=0;i--)
{    if(min>stek.top())
        {
                min=stek.top();
                imin=i;
        }
        stek.pop();
}
cout<<"Stackning minimal elementi ="<<min<<endl;
return 0;
}
```



Nazorat savollar

1. Stack konteyneriga ta’rif bering?
2. Stack konteynerining bir o‘lchovli massivdan ustun tomonini aytib bering?
3. LIFO atamasini tushuntirib bering? Misollar keltiring?
4. FIFO atamasini tushuntirib bering? Misollar keltiring?
5. Deque(DEK) konteyneriga ta’rif bering? Misollar keltiring?
6. Navbat deb nimaga aytiladi? Navbatga misollar keltiring?
7. Ustuvor navbat nima va uning navbatdan farqi nimada?
9. Konteynerlar bilan ishlaydigan qanday funksiyalarni bilasiz?
10. Konteynerlar bilan ishladigan algoritmlarga misollar keltiring?
11. Pop() va Push() funksiyalari nima uchun ishlatiladi?

Amaliyot mashg‘ulotini bajarish bo‘yicha variantlar

1) Topshiriqqa muvofiq navbat va stekga doir bo‘lgan masalalarni yechish lozim.

1	Navbatda birinchi va oxirgi elementlar o‘rni almashtirilsin.
2	Navbat o‘rtasidagi element o‘chirib tashlansin. Agar navbat elementlari soni toq bo‘lsa, bitta element, aks holda ikkita element o‘chirilsin.
3	Navbatni juft o‘rinda turgan elementlari o‘chirilsin.
4	Navbat o‘rtasiga '+' belgi joylashtirilsin.
5	Navbat eng kichik elementi topilsin va undan keyin 0 joylashtirilsin.

6	Navbat eng katta elementi topilsin va undan keyin 0 joylashtirilsin.
7	Navbat eng kichik elementi o‘chirilsin.
8	Navbatda birinchi elementga teng barcha elementlar o‘chirilsin.
9	Navbatda oxirgi elementga teng barcha elementlar o‘chirilsin.
10	Navbat eng katta elementi o‘chirilsin.
11	Navbat eng kichik elementi topilsin va uning o‘rniga 0 joylashtirilsin.
12	Stek birinchi va oxirgi elementlari o‘rni almashtirilsin.
13	Stek elementlari teskari tartibda joylashtirib chiqilsin.
14	Stek o‘rtasidagi element o‘chirib tashlansin. Agar stek elementi toq bo‘lsa, bitta element, aks holda ikkita element o‘chirilsin.
15	Stekning juft o‘rinda turgan elementlari o‘chirilsin.
16	Stek o‘rtasiga '*' belgi joylashtirilsin.
17	Stek eng kichik elementi topilsin va undan keyin 0 joylashtirilsin.
18	Stek eng katta elementi topilsin va undan keyin 0 joylashtirilsin.
19	Stek eng kichik elementi o‘chirilsin.
20	Stekda birinchi elementga teng barcha elementlar o‘chirilsin.

№9-AMALIY MASHG’ULOT

MAVZU: KONTEYNERLAR. KONTEYNERLAR BILAN ISHLASH ALGORITMLARI. FUNKTORLARNING QO’LLANILISHI

Ishning maqsadi: C++ dasturlash tilida **Funktorlarning qo’llanilishi** hosil qilish va uning dasturini yaratish ko‘nikmalarini egallsh.

Masalaning qo‘yilishi:

Masala:

Masalani yechish g‘oyasi:

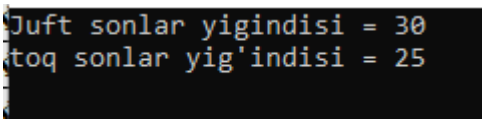
Funktorlardan foydalanib berilgan to’plam elementlari juft va toq elementlari yig’indisini hisoblash dasturini tuzing.

Dastur

```
#include <iostream>
```

```
#include <vector>
using namespace std;
 class summator : private vector<int> {
public:
 summator(const vector<int>& ini)
  {
 for (auto x : ini)
 this->push_back(x);
 }
 int operator()(bool hodisa) {
 int sum = 0;
 auto i = begin();
 if (hodisa) i++;
 while (i < end()) {
 sum += *i++;
 if (i == end()) break;
 i++;
 }
 return sum;
 }
};

int main() {
  summator sums(vector<int>({ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }));

 cout << "Juft sonlar yigindisi = " << sums(true) << endl
 << "toq sonlar yig'indisi = " << sums(false) << endl;
}
```

Dastur natijasi:



Dastur

Berilgan amal va ifodalarni kiritib natijani hosil qiling.

```
#include <iostream>
using namespace std;
 class calculate {
   char op;
public:
   calculate( char op ) : op( op ) {}
   int operator()( int op1, int op2 ) {
     switch( op ) {
       case '+': return op1 + op2;
       case '-': return op1 - op2;
       case '*': return op1 * op2;
       case '/': return op1 / op2;
       case '%': return op1 % op2;
       case '^': {
         int ret = op1;
         while( op2-- > 1 ) ret *= op1;
         return ret;
       }
       default:
         cout << "bunday amal mavjud emas" << endl;
         return 0;
     }
   }
};
```

```
int main(  ) {


  char oper;
  int op1, op2;
  do {
    cout << "Ifodani kiriting kiriting  (birinchi ifoda amal ikkinchi ifoda)=: " ;
    cin >> op1 >> oper >> op2;
    cout << op1 << " " << oper << " " << op2 << " = "
        << calculate( oper )( op1, op2 ) << endl;
  } while( true );
  return 0;
}
```

Dastur natijasi:

```
Ifodani kiriting kiriting  (birinchi ifoda amal ikkinchi ifoda)=: 45+25
45 + 25 = 70
Ifodani kiriting kiriting  (birinchi ifoda amal ikkinchi ifoda)=: 96-32
96 - 32 = 64
Ifodani kiriting kiriting  (birinchi ifoda amal ikkinchi ifoda)=:
```

Amaliy mashg’ulotni bajarish bo’yicha variantlar

1) Topshiriqqa muvofiq navbat va stekga doir bo’lgan masalalarni yechish lozim.

1- jadval

1	Navbatda birinchi va oxirgi elementlar o‘rni almashtirilsin.
2	Navbat o‘rtasidagi element o‘chirib tashlansin. Agar navbat elementlari soni toq bo‘lsa, bitta element, aks holda ikkita element o‘chirilsin.
3	Navbatni juft o‘rinda turgan elementlari o‘chirilsin.
4	Navbat o‘rtasiga '+' belgi joylashtirilsin.
5	Navbat eng kichik elementi topilsin va undan keyin 0 joylashtirilsin.

6	Navbat eng katta elementi topilsin va undan keyin 0 joylashtirilsin.
7	Navbat eng kichik elementi o‘chirilsin.
8	Navbatda birinchi elementga teng barcha elementlar o‘chirilsin.
9	Navbatda oxirgi elementga teng barcha elementlar o‘chirilsin.
10	Navbat eng katta elementi o‘chirilsin.
11	Navbat eng kichik elementi topilsin va uning o‘rniga 0 joylashtirilsin.
12	Stek birinchi va oxirgi elementlari o‘rni almashtirilsin.
13	Stek elementlari teskari tartibda joylashtirib chiqilsin.
14	Stek o‘rtasidagi element o‘chirib tashlansin. Agar stek elementi toq bo‘lsa, bitta element, aks holda ikkita element o‘chirilsin.
15	Stekning juft o‘rinda turgan elementlari o‘chirilsin.
16	Stek o‘rtasiga '*' belgi joylashtirilsin.
17	Stek eng kichik elementi topilsin va undan keyin 0 joylashtirilsin.
18	Stek eng katta elementi topilsin va undan keyin 0 joylashtirilsin.
19	Stek eng kichik elementi o‘chirilsin.
20	Stekda birinchi elementga teng barcha elementlar o‘chirilsin.

№10-AMALIY MASHG’ULOT

STANDART ALGORITMLAR VA ITERATORLAR. STANDART ALGORITMLAR. ITERATORLAR VA ULARNING QO`LLANILISHI.

Ishning maqsadi: C++ dasturlash tilida standart algoritmlar, iteratorlar bilan ishlash ko‘nikmalarini shakllantirish.

Masalaning qo‘yilishi:

1) Variantda berilgan topshiriq bo‘yicha konteyner hosil qilish va berilgan standart funksiya yordamida konteyner ustida amallar bajarish;
2) Variantda berilgan konteyner ustida iteratorlardan foydalanib amallar bajarish;
3) Variantda berilgan funksiya bo‘yicha xotirani taqsimlashni amalga oshiring.

Masala:

multiset konteyneri yordamida mst nomli konteyner e'lon qilib uni *rand()* funksiyasi yordamida tasodifiy sonlardan tarkib topgan elementlar bilan to‘latib ularni chop qilish. Konteyner asosida *it* nomli iterator hosil qilib *begin()* funksiyasi yordamida iteratorga boshlang‘ich qiymat berish, itertaordan foydalanga holda konteyner elementlarini chop qilish. Avvaldan hosil qilingan *isEven()* funksiyasidan foydalanib *find_if()* standart funksiyasini ishlatish.

Masalani yechilishi:

```
#include "stdafx.h"
#include <iostream>
#include<algorithm>
#include <vector>
#include <ctime>
#include <set>
using namespace std;

bool isEven(int i)
{
return((i % 2) == 0);
}

int main() {
    //setlocale(LC_ALL, "Russian");
    srand(time_t(NULL));

    multiset <int> mst;//mst nomli multiset konteynerini e'lon qilish

    cout << "Tasodifiy qiymatlarni qo'shish: " << endl;
    for (int i = 0; i < 10; i++) {
        int random = rand() % 100 + 1;//random ga 100 gacha bo'lgan tasodifiy son o'zlashtirish
        mst.insert(random);//mst konteynerga random sonni qo'shish
        cout << i + 1 << ") " << random << endl;// har safar random sonni chop qilish
    }

    multiset <int> ::iterator it = mst.begin();//multisetdan it nomli iterator e'lon qilinib
    //unga mst ning boshlang'ich qiymatini o'zlashtirish


    cout << "Saralangan variant: " << endl;
    for (it = mst.begin(); it != mst.end(); it++) {//iterator yordamida konteyner elementlarini chop qilish
        cout << *it << " ";     }

    it = find_if(mst.begin(), mst.end(), isEven);//find_if() funksiyasining ishlashi
    cout << "shart bajariluvchi birinchi navbatda uchragan element bu: " << *it << "\n";

    system("pause");
    return 0; }
```

Natija:

```
D:\ab\akmal\progs visual c++\VCproject2020\Iteratorlar\Debu...
1) 39
2) 20
3) 39
4) 38
5) 56
6) 98
7) 66
8) 86
9) 51
10) 13
Saralangan variant:
13 20 38 39 39 51 56 66 86 98 Berilgan elementlarning ichida
shart bajariluvchi birinchi navbatda uchragan element bu: 20
Для продолжения нажмите любую клавишу . . .
```

Masala:

Yangi v nomli vektorni e’lon qilish, unga *srand()* funksiyasi yordamida hosil qilingan tasodifiy sonlarni element sifatida *push_back()* funksiyasidan foydalanib qo‘shish, qo‘shilgan elementlarni iterator yordamida 2 martaga oshirib chop qilish

Masalani yechilishi:

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <ctime>

using namespace std;

int _tmain(int argc, _TCHAR* argv[]) {

    srand(time_t(NULL));
    std::vector<int> v;
    int n;
    cout << "elementlar soni="; cin >> n;

    cout << "Vektorni tasodifiy qiymatlar bilan to'latish: " << endl;
    for (int i = 0; i < n; i++) {
        int random = rand() % 100 + 1;//random ga 100 gacha bo'lgan tasodifiy son o'zlashtirish
        v.push_back(random);// v konteynerga random sonni qo'shish
        cout << i + 1 << ") " << random << endl; } // har safar random sonni chop qilish

    auto iter = v.begin();
    while (iter != v.end())    // oxirigacha borish
    {
        //elementlar qiymatini 2 marta oshirish
        *iter = 2 * (*iter); // kvadrat son
        ++iter;     }
    // qiymati oshgan elementlarni chop qilish
    for (iter = v.begin(); iter != v.end(); ++iter)
    {
        cout << *iter << "\t";
    }
    system("pause");
    return 0;   }
```

Natija:

```
D:\ab\akmal\progs visual c++\VCproject2020\Iteratorlar\Debu...
elementlar soni=8
Vektorni tasodifiy qiymatlar bilan to'latish:
1) 39
2) 20
3) 39
4) 38
5) 56
6) 98
7) 66
8) 86
78      40      78      76      112     196     132     172
Для продолжения нажмите любую клавишу . . .
```

NAZAROT SAVOLLARI

1. Standart algoritmlar kutubxonasi nima uchun yaratilgan.
2. Algoritm kutubxonasi tomonidan taqdim etiladigan funksiyalar odatda nechta toifaga bo‘linadi va qaysilar?
3. Mutatorlar nima va qanday vazifani amalga oshiradi.
4. Berilgan qiymat bo‘yicha elementlarni qidirish algoritmi qaysi va nechta parametr qabul qiladi?
5. Konteynerda mavjud elementlar orasida ma’lum bir qiymat saqlaydigan element qidirish uchun qaysi funksiya ishlatiladi?
6. Konteynerdan berilgan elementlarini sanash vazifasini bajaruvchi algoritmlarni sanab bering?
7. Funksiya taqqoslash uchun paramert sifatida ikkita parametr oladi. Bu qaysi funksiya.
8. Ikki ob’ektning qiymatlarini almashtirish algoritmni ayting.
9. Tartiblangan to‘plamlarni boshqa tartiblangan to‘plamni o‘z ichiga olishi yoki olmasligini tekshirish algoritmini ayting.
10. Iteratorlar nima uchun foydalaniladi.
11. **begin()** va **end()** funksiyalaridan nima maqsadda foydalaniladi.
12. Agar **begin()** va **end()** funksiyalari o‘zaro teng bo‘lmasa, ularning orasidan kamida nechta element bor.
13. Keyingi elementga murojjat qilish uchun iteratorni harakatlantirish amalni ayting.
14. Agar konteynerda o‘zgarmas qiymatli elementlar bo‘lsa, bu holda konteyner elementlariga murojaat qilish uchun nimadan foydalanishsh kerak.
15. cbegin() va cend()funksiyalaridan nima saqsadda foydalaniladi.
16. Agar konteynerda o‘zgaruvchi qiymatli elementlar bo‘lsa, bu holda konteyner elementlariga murojaat qilish o‘zgarmas iteratorlardan foydalanishsh mumkinmi.
17. Iteratorlari qo‘shima amallar qaysi konteynerlar uchun ishlatilmaydi.
18. Qaysi amal iter1 va iter2 lar orasidagi pozitsiyalar sonini qaytaradi.
19. (move_ita != move_itb) va !(move_ita == move_itb) munosabatlar bir xilmi, bir xil bo‘lsa nima uchun, farqi bo‘lsa, nima uchun.

№11-AMALIY MASHG’ULOT

STANDART ALGORITMLAR VA ITERATORLAR. XOTIRANI TAQSIMLOVCHILAR VA ULARGA QO`YILGAN TALABLAR. STANDART BO`YICHA TAQSIMLOVCHI.

Ishning maqsadi: C++ dasturlash tilida standart algoritmlar, iteratorlar va xotirani taqsimlovchilar bilan ishlash ko‘nikmalarini shakllantirish.

Masalaning qo‘yilishi:

1)Variantda berilgan topshiriq bo‘yicha konteyner hosil qilish va berilgan standart funksiya yordamida konteyner ustida amallar bajarish;

2)Variantda berilgan konteyner ustida iteratorlardan foydalanib amallar bajarish;

3)Variantda berilgan funksiya bo‘yicha xotirani taqsimlashni amalga oshiring.

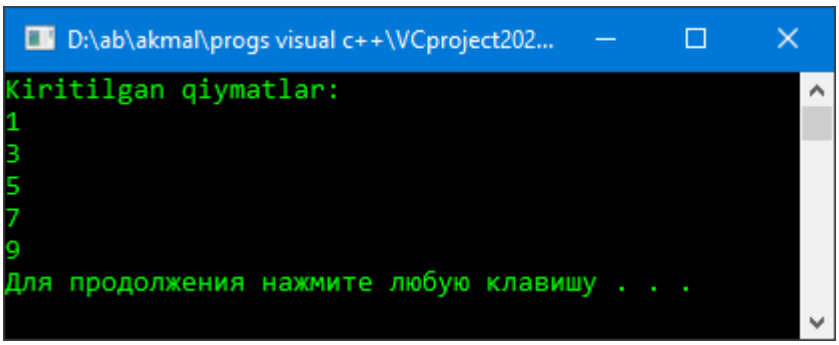
Masala:

Xotirani dinamik ajratish uchun *malloc()* funksiyasiga *sizeof*() funksiyasi yordamida xotira o‘lchamini berilgan va berilmagan hollarda ishlashini ko‘rib chiqish.

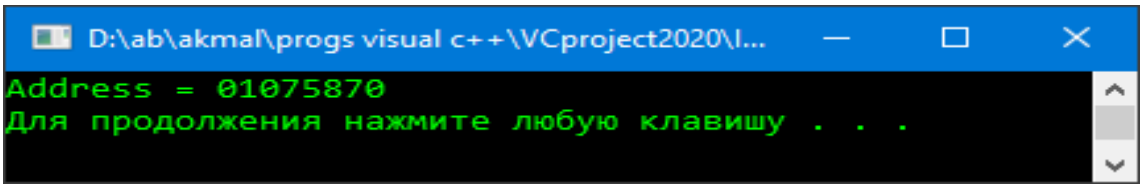
Masalani yechilishi:

```
#include "stdafx.h"
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
    int *ptr;
    ptr = (int*)malloc(5 * sizeof(int));//5*4 bayt = 20 bayt
    if (!ptr) {
        cout << "Xotirani ajratish amalga oshmadi";
        exit(1);
    }
    // Qiymatlarni kiritish
    for (int i = 0; i<5; i++)
    {
        ptr[i] = i * 2 + 1;
    }
    cout << "Kiritilgan qiymatlar:" << endl;
    for (int i = 0; i<5; i++)
    {
        /* ptr[i] va *(ptr+i) bir-birining o'rnida ishlatilishi mumkin */
        cout << *(ptr + i) << endl;
    }
    free(ptr);

    system("pause");
    return 0;
}
```

Natija:



```
#include "stdafx.h"
#include <iostream>
#include <cstdlib>
using namespace std;

int main()
{
    int *ptr = (int*)malloc(0);
    if (ptr == NULL) {
        cout << "Null ko'rsatkichi";
    }
    else {
        cout << "Address = " << ptr << endl;
    }
    free(ptr);

    system("pause");
    return 0;
}
```



NAZAROT SAVOLLARI

1. Xotirani taqsimlovchilar va ularga qo‘yilgan talablar nima uchun kerak.
2. Merosxo‘rlar odatda oddiy new/delete operatorlari bilan ishlatiladi, shuningdek BlockAlloc ni ham nima uchun joylatirish kerak.
3. Agar, new operatoridan so‘ng zudlik bilan delete operatori ishlatilsa, xotira kamroq ishlatiladi va qaerdagi ma’lumotlar ham kamayadi.
4. Kaysi operatordan new operatori yoki delete operatori foydalanish qiyinroq va muammoliroq, sababini tushintirib bering.
5. New operatori qaysi bo‘limiga joylashtirilsa, uni allokatorsiz ishlashaga ruxsat bermaydi.
6. Tuzilmali o‘zgaruvchisi uchun xotira ajratish va bo‘shatish uchun qanday amallar bajarish kerak.
7. Sinf ob’ekti uchun dinamik xotira ajratish uchun qanday amallar bajarish kerak.
8. new operatori yordamida massivlarga xotiradan joy ajratish va foydalanish uchun qanday amallar bajarish kerak.
9. Bazaviy tipli ko‘rsatkich massiviga dinamik xotira ajratish va foydalanish uchun qanday amallar bajarish kerak.
10. Ma’lum bir sinf massiv ob’ektlari uchun xotira ajratish kerak bo‘lsa, bu sinf qanday parametrli konstruktorga ega bo‘lishi kerak.
11. Massiv o‘lchamini dinamik kattalashtirish (kichiklashtirish) uchun xotira ajratish va foydalanish uchun qanday amallar bajarish kerak.

11- AMALIY MASHG'ULOT

STANDART ALGORITMLAR VA ITERATORLAR. STANDART ALGORITMLAR. ITERATORLAR VA ULARNING QO`LLANILISHI. XOTIRANI TAQSIMLOVCHILAR VA ULARGA QO`YILGAN TALABLAR. STANDART BO`YICHA TAQSIMLOVCHI.

Ishning maqsadi: C++ dasturlash tilida standart algoritmlar, iteratorlar va xotirani taqsimlovchilar bilan ishlash ko‘nikmalarini shakllantirish.

Masalaning qo‘yilishi:

4) Variantda berilgan topshiriq bo‘yicha konteyner hosil qilish va berilgan standart funksiya yordamida konteyner ustida amallar bajarish;
5) Variantda berilgan konteyner ustida iteratorlardan foydalanib amallar bajarish;
6) Variantda berilgan funksiya bo‘yicha xotirani taqsimlashni amalga oshiring.

1)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating va for_each funksiyasi yordamida uni chop qiling;
b) Iteratorlardan foydalanib *list* konteyneri ustida amallar bajaring
c) calloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
2)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating find funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *deque* konteyneri ustida amallar bajaring;
c) malloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring.
3)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating count funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *map* konteyneri ustida amallar bajaring;
c) fre () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
4)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating equal funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *vector* konteyneri ustida amallar bajaring;
c) realloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
5)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating search funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *multimap* konteyneri ustida amallar bajaring;

c) calloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
6)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating copy funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *set* konteyneri ustida amallar bajaring;
c) malloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
7)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating move funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *vector* konteyneri ustida amallar bajaring;
c) free () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
8)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating swap funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *multiset* konteyneri ustida amallar bajaring;
c) realloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
9)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating transform funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *list* konteyneri ustida amallar bajaring;
c) calloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
10)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating replace funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *deque* konteyneri ustida amallar bajaring;
c) malloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
11)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating fill funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *multimap* konteyneri ustida amallar bajaring;
c) free () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
12)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating remove funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *set* konteyneri ustida amallar bajaring;
c) realloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
13)

a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating unique funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *multiset* konteyneri ustida amallar bajaring;
c) calloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
14)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating reverse funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *list* konteyneri ustida amallar bajaring;
c) malloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
15)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating rotate funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *vector* konteyneri ustida amallar bajaring;
c) free () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
16)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating shuffle funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *deque* konteyneri ustida amallar bajaring;
c) realloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
17)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating sort funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *set* konteyneri ustida amallar bajaring;
c) calloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
18)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating merge funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *multimap* konteyneri ustida amallar bajaring;
c) malloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
19)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating set_union funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *vector* konteyneri ustida amallar bajaring;
c) free () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .
20)
a) Yangi konteynerlar hosil qiling, uni standart funksiyalar yordamida elementlar bilan to‘lating is_sorted_until funksiyasi yordamida amallar bajaring;
b) Iteratorlardan foydalanib *list* konteyneri ustida amallar bajaring;

c) realloc () funksiyasidan foydalanib xotirani dinamik taqsimlashni amalga oshiring .

№12-AMALIY MASHG'ULOT

SONLI SINFLAR BILAN ISHLASH.

Ishning maqsadi: Sonli sinflarni o‘rganish va quyidagi ro‘yxat asosida masalani ishlash algoritmlarni ishlab chiqish

Vazifa: Visual C ++ tilida raqamli sinflarning elementlarini yaratish, unda turli xil amallarni bajarish uchun dastur ishlab chiqish.

Ish tartibi:

- Amaliyot ishlarining nazariy ma'lumotlarini o'rganish;
- Berilgan topshiriqning algoritmini ishlab chiqish;
- Visual C ++ dasturlash muhitida dastur yaratish;
- Natijalarni tekshirish;
- Amaliyot ishi bo'yicha hisobot tayyorlash va taqdim etish.

Nazariy qism

Complex sonlar haqida

Complex son deb a+bi ifodaga aytiladi, bu yerda a va b haqiqiy sonlar, i – mavhum birlik bo‘lib, u yoki i^2= -1 tengliklar bilan aniqlanadi; a – kompleks sonning haqiqiy qismi, bi – mavhum qismi deyiladi. Faqat mavhum qismining ishorasi bilan farq qiladigan ikki kompleks son: a+bi va a-bi o‘zaro qo‘shma deyiladi. Ko‘pincha a+bi kompleks son bitta α harfi bilan belgilanadi: α=a+bi. a+bi kompleks sonning haqiqiy qismi a=Reα bilan, mavhum qismining koeffitsientini b=Lmα bilan belgilaydilar. α kompleks sonning a+bi ko‘rinishidagi yozuviga uning algebraik shakli deyiladi.

Agar ikkita $\alpha_1=a_1+b_1i$ va $\alpha_2=a_2+b_2i$ kompleks sonda $a_1= \alpha_2$, $b_1= b_2$ bu ikki son teng deyiladi ($\alpha_1= \alpha_2$). Agar α=a+bi kompleks sonda a=0, b=0 bo‘lsa, bu kompleks son 0 ga (α=0) teng bo‘ladi. Agar α=a+bi kompleks sonda b=0 bo‘lsa, haqiqiy son hosil bo‘ladi; agar a=0 bo‘lsa, 0+bi=bi sof mavhum son deyiladi.

Complex sonlar sinfi quyidagi funksiyalardan tashkil topgan:

abs, acos, acosh, arg, asin, asinh, atan, atanh, conj, cos, cosh, exp,

imag - Murakkab sonning noma’lum tarkibiy qismini ajratib oladi.

log, log10, norm, polar - Belgilangan modul va argumentga mos keladigan murakkab sonni Cartesian shaklida qaytaradi.

pow- Murakkab son bo'lgan bazani boshqa murakkab raqamning kuchiga ko'tarish orqali olingan murakkab sonni baholaydi.

proj, real- Murakkab sonning haqiqiy tarkibiy qismini ajratib oladi.

sin, sinh- Kompleks sonning giperbolik sinusini qaytaradi.

sqrt, tan, tanh - Returns the hyperbolic tangent of a complex number.

Complex sonlar uchun quyidagi operatorlar qayta yuklangan:

1- jadval

operator!=	Haqiqiy va xayoliy qismlar uchun turlarning pastki qismiga tegishli bo'lishi mumkin bo'lgan ikkita yoki murakkab ikkala murakkab sonlar o'rtasidagi tengsizlik uchun testlar.
operator*	Haqiqiy va xayoliy qismlar uchun turlarning pastki qismiga tegishli bo'lishi mumkin bo'lgan ikkita murakkab raqamni ko'paytiradi.
operator+	Haqiqiy va xayoliy qismlar uchun tipning pastki qismiga tegishli bo'lishi mumkin bo'lgan ikkita murakkab sonni qo'shadi.
operator-	Haqiqiy va xayoliy qismlar uchun tipning pastki qismiga tegishli bo'lishi mumkin bo'lgan ikkita murakkab raqamni ayiradi.
operator/	Haqiqiy va xayoliy qismlar uchun turlarning pastki qismiga tegishli bo'lishi mumkin bo'lgan ikkita murakkab raqamni bo‘ladi.
operator<<	Chiqish oqimiga murakkab raqamni kiritadigan shablon funktsiyasi.
operator==	Haqiqiy va xayoliy qismlar uchun turlarning pastki qismiga tegishli bo'lishi mumkin bo'lgan ikkita yoki murakkab ikkala murakkab sonlar o'rtasidagi tenglik uchun testlar.
operator>>	KIRISH oqimidan murakkab qiymatni chiqaradigan shablon funktsiyasi.

Complex sonlar uchun quyidagi sinflar yaratilgan:

2- jadval

complex<double>	Aniq ixtisoslashtirilgan sinf shablonlari ikkitomonlama tipdagi buyurtma qilingan juftlikni saqlaydigan ob'ektni tasvirlaydi, bu erda birinchisi murakkab sonning haqiqiy qismini, ikkinchisi esa xayoliy qismni anglatadi.
complex<float>	Shubhasiz ixtisoslashtirilgan sinf shablonlari buyurtma qilingan juftlikni saqlaydigan ob'ektni tavsiflaydi, har ikkala turdagi float ham, birinchisi murakkab sonning haqiqiy qismini, ikkinchisi esa xayoliy qismni anglatadi.
complex<long double>	Aniq ixtisoslashtirilgan shablon shablonni buyurtma qilingan juftlikni saqlaydigan ob'ektni tavsiflaydi, har ikkala turdagi uzun juft juftlik, bu erda murakkab raqamning haqiqiy qismi, ikkinchisi xayoliy qism tasvirlangan.
complex	Sinf shablonlari murakkab sonlar tizimini ifodalash va murakkab arifmetik amallarni bajarish uchun ishlatiladigan ob'ektni tavsiflaydi.

Komplex sonlarning yuqori keltirilgan tarkibiy qismlarini bazi birlariga misollar keltiramiz.

1. **abs()** complex funksiyasiga misol:

Funksiya shabloni:

```
template <class Type>
Type abs(const complex<Type>& complexNum);
```

Dastur matni:

```
1. #include "stdafx.h"
2. #include <complex>
3. #include <iostream>
4. using namespace System;
5. using namespace std;

6. int main( )
7. {
8. using namespace std;
9. double pi = 3.14159265359;

10. complex <double> c1 ( polar ( 5.0 ) );   // Default argument = 0
11. complex <double> c2 ( polar ( 5.0 , pi / 6 ) );
12. complex <double> c3 ( polar ( 5.0 , 13 * pi / 6 ) );
13. cout << "c1 = polar ( 5.0 ) = " << c1 << endl;
14. cout<< "c2 = polar ( 5.0 , pi / 6 ) = " << c2 << endl;
15. cout<<"c3 = polar (5.0, 13*pi / 6 ) = " << c3 << endl;

16. double absc1 = abs ( c1 );
17. double argc1 = arg ( c1 );
18. cout << "C1 moduli c1 dan quyidagilar yordamida tiklanadi: abs ( c1 ) = "
19. << absc1 << endl;
20. cout << "C1 argumenti c1 dan quyidagilar yordamida tiklanadi:\n arg ( c1 ) = "
21. << argc1 << " radians, qaysiki " << argc1 * 180 / pi
22. << " darajasi." << endl;

23. double absc2 = abs ( c2 );
24. double argc2 = arg ( c2 );
25. cout << "C2 moduli c2 dan s2 yordamida tiklanadi: abs ( c2 ) = "
26. << absc2 << endl;
27. cout << "C2 argumenti c2 dan quyidagilar yordamida tiklanadi:\n arg ( c2 ) = "
28. << argc2 << " radianda, qaysiki " << argc2 * 180 / pi
29. << " darajalar." << endl;
30. if ( (arg ( c2 ) <= ( arg ( c3 ) + .00000001) ) ||
31. (arg ( c2 ) >= ( arg ( c3 ) - .00000001) ) )
32. cout << " c2 & c3 komplex sonlar bir xil argumentlardan "
33. << "tashkil topgan."<< endl;
34. else
```

```
35. cout << " c2 & c3 komplex sonlar bir xil argumentlardan "
36. << "tashkil topmagan."<< endl;
37. getchar();
38. }
```

Dastur natijasi:

c1 = polar (5.0) = (5,0)

c2 = polar (5.0 , pi / 6) = (4.33013,2.5)

c3 = polar (5.0 , 13 * pi / 6) = (4.33013,2.5)

C1 moduli c1 dan quyidagilar yordamida tiklanadi: abs (c1) = 5

C1 argumenti c1 dan quyidagilar yordamida tiklanadi:

arg (c1) = 0 radians, qaysiki 0 darajasi.

C2 moduli c2 dan s2 yordamida tiklanadi: abs (c2) = 5

C2 argumenti c2 dan quyidagilar yordamida tiklanadi:

arg (c2) = 0.523599 radians, qaysiki 30 darajalar.

c2 & c3 komplex sonlar bir xil argumentlardan tashkil topgan.

2. **comj()** complex funksiyasiga misol:

Funksiya shabloni:

```
template <class Type>
complex<Type> conj(const complex<Type>& complexNum);
```

Dastur matni:

```
1. #include "stdafx.h"
2. #include <complex>
3. #include <iostream>
4. int main( )
5. {
6. using namespace std;

7. complex <double> c1 ( 4.0 , 3.0 );
8. cout << "Komplex raqam c1 = " << c1 << endl;

9. double dr1 = real ( c1 );
10. cout << " c1 ning haqiqiy qismi bu haqiqiy ( c1 ) = "
11. << dr1 << "." << endl;

12. double di1 = imag ( c1 );
13. cout << " c1 ning no'malum qismi bu imag ( c1 ) = "
```

```
<< di1 << "." << endl;

complex <double> c2 = conj ( c1 );
cout<<"c1 ning komplex kasr qismi bu c2 = conj(c1)= "
<< c2 << endl;

double dr2 = real ( c2 );
cout << " c2 ning haqiqiy qismi bu real ( c2 ) = "
<< dr2 << "." << endl;

double di2 = imag ( c2 );
cout << " c2 ning no'malum qismi bu imag ( c2 ) = "
<< di2 << "." << endl;

complex <double> c3 = c1 * c2;
cout<<"(c1 * conj (c1) ) ning haqiqiysi bu c1 * c2 = "
<< real( c3 ) << endl;
getwchar();
}
```

Dastur natijasi:

Komplex raqam c1 = (4,3)

c1 ning haqiqiy qismi bu haqiqiy (c1) = 4.

c1 ning no'malum qismi bu imag (c1) = 3.

c1 ning komplex kasr qismi bu c2 = conj (c1)= (4,-3)

c2 ning haqiqiy qismi bu real (c2) = 4.

c2 ning no'malum qismi bu imag (c2) = -3.

(c1 * conj (c1)) ning haqiqiysi bu c1 * c2 = 25

3. **imag()** complex funksiyasiga misol:

Funksiya shabloni:

```
template <class Type>
Type imag(const complex<Type>& complexNum);
```

Dastur matni:

```
#include "stdafx.h"
#include <complex>
#include <iostream>

int main( )
{
```

```
using namespace std;
complex <double> c1 ( 4.0 , 3.0 );
cout << " c1 komplex son = " << c1 << endl;
double dr1 = real ( c1 );
cout << " c1 ning haqiqiy qismi bu real ( c1 ) = "
<< dr1 << "." << endl;

double di1 = imag ( c1 );
cout << " c1 ning noma'lum qismi bu imag ( c1 ) = "
<< di1 << "." << endl;
getchar();
}
```

Dastur natijasi:

c1 komplex son = (4,3)

c1 ning haqiqiy qismi bu real (c1) = 4.

c1 ning noma'lum qismi bu imag (c1) = 3.

4. **polar()** complex funksiyasiga misol:

Funksiya shabloni:

```
template <class Type>
complex<Type> polar(const Type& _Modulus, const Type& _Argument = 0);
```

Dastur matni:

```
#include "stdafx.h"
#include <complex>
#include <iostream>

int main( )
{
using namespace std;
double pi = 3.14159265359;

complex <double> c1 ( polar ( 5.0 ) );
// Default argument = 0
complex <double> c2 ( polar ( 5.0 , pi / 6 ) );
complex <double> c3 ( polar ( 5.0 , 13 * pi / 6 ) );
cout << "c1 = polar ( 5.0 ) = " << c1 << endl;
cout << "c2= polar ( 5.0 , pi / 6 ) = " << c2 << endl;
cout << "c3= polar(5.0, 13*pi / 6 ) = " << c3 << endl;

if ( (arg ( c2 ) <= ( arg ( c3 ) + .00000001) ) ||
(arg ( c2 ) >= ( arg ( c3 ) - .00000001) ) )
```

```
17. cout << " c2 & c3 komplex sonlar bir xil argumentlardan "<< "tashkil topgan."<<
    endl;
18. else
19. cout << " c2 & c3 komplex sonlar bir xil argumentlardan  "<< "tashkil topmagan." <<
    endl;
20. double absc2 = abs ( c2 );
21. double argc2 = arg ( c2 );
22. cout << " c2 ning moduli bu : abs ( c2 ) = "
23. << absc2 << endl;
24. cout << " c2 argumenti bu :\n arg ( c2 ) = "
25. << argc2 << " radianda, qaysiki " << argc2 * 180 / pi
26. << " daraja." << endl;
27. getchar();
28. }
```

Dastur natijasi:

c1 = polar (5.0) = (5,0)

c2 = polar (5.0 , pi / 6) = (4.33013,2.5)

c3 = polar (5.0 , 13 * pi / 6) = (4.33013,2.5)

c2 & c3 komplex sonlar bir xil argumentlardan tashkil topgan.

c2 ning moduli bu : abs (c2) = 5

c2 argumenti bu :

arg (c2) = 0.523599 radianda, qaysiki 30 daraja.

Komplex sonlar uchun qayta yuklangan teng emas operatori - **operator!=** ga misol:

Operator shabloni:

```
template <class Type>
bool operator!=(
   const complex<Type>& left,
   const complex<Type>& right);

template <class Type>
bool operator!=(
   const complex<Type>& left,
   const Type& right);

template <class Type>
bool operator!=(
   const Type& left,
   const complex<Type>& right);
```

Dastur matni:

```
1. #include "stdafx.h"
```

```
#include <complex>
#include <iostream>

int main( )
{
using namespace std;
double pi = 3.14159265359;
complex <double> cl1 ( polar (3.0, pi / 6 ) );
complex <double> cr1a ( polar (3.0, pi /6 ) );
complex <double> cr1b ( polar (2.0, pi / 3 ) );

cout << " cl1 komplex sonning chap tarafi= " << cl1 << endl;
cout<<"cr1a komplex sonning birinchi o'ng tarafi = " <<cr1a<<endl;
cout<<"cr1b komplex sonning ikkinchi o'ng tarafi = " <<cr1b<<endl;
if ( cl1 != cr1a )
cout << " cl1 & cr1a komplex sonlar teng emas." << endl;
else
cout << " cl1 & cr1a komplex sonlar teng." <<endl;
if ( cl1 != cr1b )
cout << " cl1 & cr1b komplex sonlar teng emas."<< endl;
else
cout << " cl1 & cr1b komplex sonlar teng." << endl;
cout << endl;
complex <int> cl2a ( 3, 4 );
complex <int> cl2b ( 5,0 );
int cr2a =3;
int cr2b =5;

cout<<"cl2a komplex sonning birinchi chap tarafi ="<<cl2a << endl;
cout<<"cr2a komplex sonning birinchi o'ng tarafi = " << cr2a << endl;
if ( cl2a != cr2a )
cout << " cl2a & cr2a komplex sonlar teng emas." << endl;
else
cout << " cl2a & cr2a komplex sonlar teng." << endl;
cout<<"cl2b komplex sonning ikkinchi chap tarafi="<<cl2b<< endl;
cout<<"cr2b komplex sonning ikkinchi o'ng tarafi="<<cr2b<< endl;
if ( cl2b != cr2b )
cout << " cl2b & cr2b komplex sonlar teng emas." << endl;
else
cout << " cl2b & cr2b komplex sonlar teng." << endl;
cout << endl;
double cl3a =3;
double cl3b =5;
complex <double> cr3a ( 3, 4 );
complex <double> cr3b ( 5,0 );

cout<<"cl3a komplex sonning birinchi chap tarafi="<<cl3a<< endl;
cout<<"cr3a komplex sonning birinchi o'ng tarafi="<<cr3a<< endl;
if ( cl3a != cr3a )
```

```
47. cout << " cl3a & cr3a komplex sonlar teng emas." << endl;
48. else
49. cout << " cl3a & cr3a komplex sonlar teng." << endl;
50. cout<<"cl3b komplex sonning ikkinchi chap tarafi="<<cl3b<<endl;
51. cout<<"cr3b komplex sonning ikkinchi o'ng tarafi="<<cr3b<<endl;
52. if ( cl3b != cr3b )
53. cout << " cl3b & cr3b komplex sonlar teng emas." << endl;
54. else
55. cout << " cl3b & cr3b komplex sonlar teng." << endl;
56. cout << endl;
57. getchar();
58. }
```

Dastur natijasi:

cl1 komplex sonning chap tarafi= (2.59808,1.5)

cr1a komplex sonning birinchi o'ng tarafi = (2.59808,1.5)

cr1b komplex sonning ikkinchi o'ng tarafi = (1,1.73205)

cl1 & cr1a komplex sonlar teng.

cl1 & cr1b komplex sonlar teng emas.

cl2a komplex sonning birinchi chap tarafi = (3,4)

cr2a komplex sonning birinchi o'ng tarafi = 3

cl2a & cr2a komplex sonlar teng emas.

cl2b komplex sonning ikkinchi chap tarafi = (5,0)

cr2b komplex sonning ikkinchi o'ng tarafi = 5

cl2b & cr2b komplex sonlar teng.

cl3a komplex sonning birinchi chap tarafi = 3

cr3a komplex sonning birinchi o'ng tarafi= (3,4)

cl3a & cr3a komplex sonlar teng emas.

cl3b komplex sonning ikkinchi chap tarafi = 5

cr3b komplex sonning ikkinchi o'ng tarafi = (5,0)

cl3b & cr3b komplex sonlar teng.

NAZAROT SAVOLLARI

1. Nima uchun sonli sinflar yaratiladi?
2. Qanday sonli sinflarni bilasiz?
3. Sonli sinflarda amallarni bajarish uchun funksiyalarni qaysi usul yordamida aniqlanadi?
4. Kompleks va arifmatik amalar o‘zaro hamkorlikda ishlaydimi?
5. Kopmleks tipni arifmetik tipka tenglashtirish mumkinmi?
6. Haqiqiy (real) yoki mavhum qismini oddiy arifmetik tipga qiymat qilib berish mumkinmi?
7. Kopmleks sonning real qismi qaysi funksiya bilan olinadi?
8. Kopmleks sonning mavhum qismi qaysi funksiya bilan olinadi?

9. double re = complex_obj.real(); va double re = real(complex_obj); sintaktiklarning farqi bormi?

10. Kompleks soning mavhum qismini qanday kiritiladi?

11. Kompleks sonni to‘liq kiritish uchun qavs ichida real va mavhum qismlari qanday belgi bilan ajratiladi?

12. Kompleks sinfi qanday operandlarni qo‘llab quvvtlamaydi?

13. Valarray konteyneri nima uchun kerak?

14. Massiv element indekslari bo‘yicha birlashtrish operatorlari va elementlararo matematik amallarni hisoblash sinfi ayting?

15. **apply()** – bu massiv elementlari bo‘yicha bir vaqtda o‘zgarishlarni bajaradi va nimani qaytaradi?

16. **cshift() -** berilgan son qiymat bo‘yicha indeks asosida massivni qanday shaklida surib yangi massiv qaytaradi?

17. Valarray tiplari bilan bajariladigan amallarni qayta aniqlash mumkinmi?

18. Kaysi sinf valarray sinfning merosxo‘ridir va bir o‘lchovli to‘plam ostilarni yaratishda ishlatiladi?

19. slice sinf qanday tipdagi ob’ektni tavsiflovchi parametrlarni saqlaydi.

20. slice sinfining konstruktorida nechta paramert bor va qaysilar, sanab bering?

21. Agar qism to‘plam yordamida aniqlangan massiv doimiy valarrayning kichik bo‘lagi bo‘lsa, bu massiv yangi nima?

22. **stride() -** valarray to‘plam ostining elementlari orasidagi nimani topadi.

23. Qaysi sinf valarray ning ko‘p o‘lchamli to‘plam ositilari bilan ishlashga mo‘ljallangan.

24. Sinf gslice - Valarray bazaviy sinfdan olingan va nechta paramertni qabul qiladi?

25. Agar to‘plam gslice bilan aniqlangan va valarray ning to‘plam ostisi bo‘lsa, gslice yangi nima hisoblanadi.

26. gslice – bir nechta qismdan tashkil topgan valarray uchun to‘plam osti valarrayni aniqlaydi va hammasi belgilangan bir element indeksidan boshlanadimi yoki xar xil element indeksidan boshlanadimi?

27. gslice i slice ning farqlari nimadan iborat.

28. Qaysi sinf Valarray qismlari bilan aniqlangan to‘plam osti massivlar uchun amallarni qo‘llab quvvatlaydi.

29. Shaboln sinf muayyan valarray amallari tomonidan bilvosita yaratilgan va dasturda bevosita foydalanish mumkinmi yoki mumkin emasmi?

30. Bazaviy valarray sinf elementlari asosidagi to‘plam osti bo‘lib, belgilangan to‘plam osti o‘rtasida amallarni ta’minlash orqali valarray ob’ektlarining to‘plam osti ob’ektlar bo‘lgan ob’ektlarni qo‘llab-quvvatlaydigan yordamchi sinfni ayting?

31. Tanlangan elementlar ketma ketligi asosida Indirect_array sinf funksiyalari valarray uchun belgilangan tegishli funksiyalar qanday ishlaydi.

32. Bazaviy valarray sinf elementlari asosidagi to‘plam osti bo‘lib, belgilangan to‘plam osti uchun mantiqiy amallarni ta’minlash orqali valarray ob’ektlarining to‘plam osti ob’ektlar bo‘lgan ob’ektlarni qo‘llab-quvvatlaydigan yordamchi sinf ayting?

33. Interval matematikaning amallarini bajarishga yo‘naltirilgan sinf nomini ayting?

34. In sinfda nechta konchtruktor bor va ular qanday ishlaydi?

35. In(double a, double b) intervalga qiymat berish funksiyasida agar _a> a_ bo‘lsa, nima vazifa bajariladi.

12-AMALIYOT ISHI
SONLI KLASSLAR

Ishning maqsadi: Sonli sinflarni o‘rganish va quyidagi ro‘yxat asosida masalani ishlash algoritmlarni ishlab chiqish

Vazifa: Visual C ++ tilida raqamli sinflarning elementlarini yaratish, unda turli xil amallarni bajarish uchun dastur ishlab chiqish.

Ish tartibi:

- Amaliyot ishlarining nazariy ma'lumotlarini o'rganish;
- Berilgan topshiriqning algoritmini ishlab chiqish;
- Visual C ++ dasturlash muhitida dastur yaratish;
- Natijalarni tekshirish;
- Amaliyot ishi bo'yicha hisobot tayyorlash va taqdim etish.

Nazariy qism

Complex sonli sinfi

Kompleks sonlar sinf complex standart kutubxonaning yana bir sinfidir. Odatdagidek, siz foydalanish uchun sarlavha faylini kiritishingiz kerak:

```
#include complex
```

Kompleks son ikki qismdan iborat - haqiqiy va mavhum. Mavhum qism manfiy sonning kvadrat ildizidir. Kompleks son odatda quyidagicha yoziladi

```
2 + 3i
```

Bu yerda 2 – haqiqiy qism, 3i esa mavhum. Complex tipida obyektlarni aniqlash quyida keltirilgan:

```
// sof mavhum raqam: 0 + 7-i
complex double purei( 0, 7 );
// mavhum qism nolga teng: 3 + Oi
complex float rea1_num( 3 );
// haqiqiy va mavhum qism nolga teng: 0 + 0-i
complex long double zero;
// bir kompleks sonni boshqasi bilan ishga tushurish
complex double purei2( purei );
```

12-Amaliyot ishi topshiriqlari

1. Complex sinfidan foydalanib, ikkita kompleks sonlarni qo‘shing.
2. Complex sinfidan foydalanib, ikkita kompleks sonlarni ayiring
3. Complex sinfidan foydalanib, ikkita kompleks sonlarni ko‘paytiring.
4. Complex sinfidan foydalanib, ikkita kompleks sonlarni bo‘ling.
5. Complex sinfidan foydalanib, kompleks sonlar inkrementini (prefiks (++i) va postfiks (i++) lar uchun) amalga oshiring.
6. Complex sinfidan foydalanib, kompleks sonlar dekrementini (prefiks (--i) va postfiks (i--) lar uchun) amalga oshiring.
7. Complex sinfidan foydalanib, ikkita kompleks sonlarni qo‘shing.
8. Complex sinfidan foydalanib, ikkita kompleks sonlarni ayiring
9. Complex sinfidan foydalanib, ikkita kompleks sonlarni ko‘paytiring.

10. Complex sinfidan foydalanib, ikkita kompleks sonlarni bo‘ling.
11. Complex sinfidan foydalanib, kompleks sonlar inkrementini (prefiks (++i) va postfiks (i++) lar uchun) amalga oshiring.
12. Complex sinfidan foydalanib, kompleks sonlar dekrementini (prefiks (--i) va postfiks (i--) lar uchun) amalga oshiring.
13. Complex sinfidan foydalanib, kompleks sonlarni ildizini toping.
14. Complex sinfidan foydalanib, kompleks sonlarni darajaga oshiring.
15. Complex sinfidan foydalanib, kompleks sonlarni yig‘indisi va bo‘linmasini toping.

№13-AMALIY MASHG’ULOT

SONLI SINFLAR BILAN ISHLASH. VALARRAY, SLICE, GSLICE SINFLARI

Ishning maqsadi: Sonli sinflarni o‘rganish va quyidagi ro‘yxat asosida masalani ishlash algoritmlarni ishlab chiqish

Vazifa: Visual C ++ tilida raqamli sinflarning elementlarini yaratish, unda turli xil amallarni bajarish uchun dastur ishlab chiqish.

Ish tartibi:

- Amaliyot ishlarining nazariy ma'lumotlarini o'rganish;
- Berilgan topshiriqning algoritmini ishlab chiqish;
- Visual C ++ dasturlash muhitida dastur yaratish;
- Natijalarni tekshirish;
- Amaliyot ishi bo'yicha hisobot tayyorlash va taqdim etish.

Valarray son sinfi va uning funksiyalari. Yaqin orada S++ da massiv elementlari bilan ishlash va samarali saqlash uchun Valarray konteyneri paydo bo‘ldi. Bu hamma S++ standartlarida bo‘lmasligi ham mumkin.

- Bu sinf massiv element indekslari bo‘yicha birlashtrish operatorlari va elementlararo matematik amallarni bajaradi.

- Vektor bilan solishtirganda Valarray sinf matematik amallarni bajarishda samaraliroq hisoblanadi.

Valarray sinfning ochiq funksiyalari:

apply() – bu massiv elementlari bo‘yicha bir vaqtda o‘zgarishlarni bajaradi va yangi massiv qaytaradi.

sum() – massivdagi elementlar yig‘indisini qaytaradi.

1.1-dastur. apply () va sum () funksiyalaridan foydalanish.

```
#include "stdafx.h"
#include<iostream>
#include<valarray>
using namespace std;
int main(){
        int myints[] = { 5, 25, 55, 85, 115 };
        valarray<int> varr (myints, sizeof(myints)/sizeof(myints[0]));
  valarray<int> varr_one ;

  varr_one = varr.apply([](int x){return x=x+5;});

  cout << "Massivning yangi qiymatllar: ";
```

```
        for (int &x: varr_one) cout << '\t' << x;
    cout << endl;

    cout << "Oldingi massiv sum() => ";
    cout << varr.sum() << endl;
    cout << "Yangi massiv sum() => ";
        cout << varr_one.sum() << endl;
        system("pause");
    return 0;
}
```

1.1-dastur.output
Massivning yangi qiymatllar: 10 30 60 90 120 Oldingi massiv sum() => 285 Yangi massiv sum() => 310

Min() va max() fnuktsiyalari mos ravishda massiv eng kichik va eng katta elementlari qimatlarini qaytaradi.

```
cout << "min() => ";
        cout << varr.min() << endl;
    cout << "max() => ";
        cout << varr.max() << endl;
```

shift() – berilgan son qiymat bo‘yicha indeks asosida massivni surib yangi massiv qaytaradi. Agar son musbat bo‘lsa chapga suriladi, manfiy bo‘lsa, o‘nga suriladi.

cshift() - berilgan son qiymat bo‘yicha indeks asosida massivni aylana shaklida surib yangi massiv qaytaradi. Agar son musbat bo‘lsa chapga suriladi, manfiy bo‘lsa, o‘nga suriladi.

1.2-dastur. shift() va cshift()dan foydalanish.

```
#include "stdafx.h"
#include<iostream>
#include<valarray>
using namespace std;
int main(){
        int myints[] = { 5, 25, 55, 85, 115 };
        valarray<int> varr (myints, sizeof(myints)/sizeof(myints[0]));
    valarray<int> varr_one, varr_two;
    cout << "Massivning eski qiymatllar: ";
        for (int &x: varr_one) cout << '\t' << x;
    cout << endl;

        varr_one = varr.shift(-2);
    cout << "Massivning shift(-2) qiymatllar: ";
        for (int &x: varr_one) cout << '\t' << x;
    cout << endl;

        varr_one = varr.shift(2);
    cout << "Massivning shift(2) qiymatllar: ";
        for (int &x: varr_one) cout << '\t' << x;
    cout << endl;
```

```
        varr_one = varr.cshift(2);
    cout << "Massivning cshift(2) qiymatllar: ";
        for (int &x: varr_one) cout << '\t' << x;
    cout << endl;

        varr_one = varr.cshift(-2);
    cout << "Massivning cshift(-2) qiymatllar: ";
        for (int &x: varr_one) cout << '\t' << x;
    cout << endl;

        system("pause");
    return 0;
}
```

1.2-dastur.output

```
Massivning eski qiymatllar:
Massivning shift(-2) qiymatllar:    0    0    5    25    55
Massivning shift(2) qiymatllar:    55    85    115    0    0
Massivning cshift(2) qiymatllar:    55    85    115    5    25
Massivning cshift(-2) qiymatllar:    85    115    5    25    55
```

Slice sinfi. Bu - xizmatchi sinf bo‘lib, valarray sinfning merosxo‘ridir va bir o‘lchovli to‘plam ostilarni yaratishda ishlatiladi. Agar valarray massivni ikki o‘lchovli matritsa sifatida qarasak, undan bir o‘lchovli vektorni hosil qilish mumkin.

1.3-dastur. Slice sinfidan foydalanish.

```
#include "stdafx.h"

#include <valarray>
#include <iostream>
using namespace std;
int main( ){
  int i;
  size_t sizeVA, sizeVAR;
  size_t startVAR, strideVAR;

  valarray<int> va(28), vaResult;

  for ( i = 0 ; i < 28 ; i += 1 )
          va [ i ] =  (rand()%100)+1;

  cout << "valarray elementlari: \n ( ";
    for ( i = 0 ; i < 28 ; i++ )
      cout << va[i] << '\t';
  cout << ")." << endl;

  sizeVA = va.size();
  cout << "valarray elementlar soni: "
      << sizeVA << "." << endl << endl;
```

```
  // slice konstruktori
  slice vaSlice( 0, 6, 4);
  vaResult = va[vaSlice];

  cout << "valarray to`plam ostisi vaResult = "
       << "va[slice( 0, 6, 4)] =\n ( ";
  for ( i = 0 ; i < vaSlice.size() ; i++ )
       cout << vaResult [ i ] << " ";
  cout << ")." << endl;

  sizeVAR = vaSlice.size( );
  cout << "to`plam ostining elementlar soni: "
       << sizeVAR << "." << endl;

  startVAR = vaSlice.start( );
  cout << "to`plam ostining birinchi element indeksi: "
       << startVAR << "." << endl;


  strideVAR = vaSlice.stride( );
  cout << "to`plam ostining elementlari orasidagi masofa: "
       << strideVAR << "." << endl;

  system("pause");
}
```

1.3-dastur. output

```
valarray elementlari:
 ( 42  68   35   1    70   25   79   59   63   65   6    46   82   28   62
92   96   43   28   37   92   5    3    54   93   83   22   17   ).
valarray elementlar soni: 28.

valarray to`plam ostisi vaResult = va[slice( 0, 6, 4)] =
 ( 42 70 63 82 96 92 ).
to`plam ostining elementlar soni: 6.
to`plam ostining birinchi element indeksi: 0.
to`plam ostining elementlari orasidagi masofa: 4.
```

Dasturga sinfga berilgan ta’rif asosida uning konstruktori 3 ta parametr qabul qilar ekan. Birinchi paramert belgilangan elemeni indeksi, ikkinchisi elementlar soni, elementlar indeksi orasidagi masofa.

Gslice sinfi. Xizmatchi sinf bo‘lib, valarray ning ko‘p o‘lchamli to‘plam ositilari bilan ishlashga mo‘ljallangan. Agar valarray massivda ko‘p o‘lchamli to‘plam sifatida qaralsa, Gslice sinfi undan ko‘p o‘lchamli vektorni oladi.

gslice – bir nechta qismdan tashkil topgan valarray uchun to‘plam osti valarrayni aniqlaydi va hammasi belgilangan bir element indeksidan boshlanadi. gslice i slice ning farqlari birgina bir nechta qism valarray lar bilan ishlashdadir.

1.4-dastur. gslice sinfidan foydalanish.

```
#include "stdafx.h"
#include <valarray>
#include <iostream>
using namespace std;
int main(){

  int i;

  valarray<int> va ( 20 ), vaResult;
  for ( i = 0 ; i < 20 ; i+=1 )
    va [ i ] =  i+1;

  cout << "valarray massiv elementlari: " << endl << "(";
  for ( i = 0 ; i < 20 ; i++ )
    cout << '\t' << va [ i ];
  cout << " )" << endl;

  valarray<size_t> Len (2), Stride(2);
  Len [0] = 6;
  Len [1] = 4;
  Stride [0] = 2;
  Stride [1] = 4;

  gslice vaGSlice (0, Len, Stride );
  vaResult = va[vaGSlice];

  cout << " vaGSlice asosida vaResult elementlari:" << endl
      << "va[vaGSlice] = (";

  for ( i = 0 ; i < 10 ; i++ )
    cout << '\t' << vaResult [ i ];
  cout << ")" << endl;

  const valarray <size_t> sizeGS = vaGSlice.size ( );

    cout << "vaResult elementlar soni:"
      << " vaGSlice.size ( ) = ( ";
    for ( i = 0 ; i < 2 ; i++ )
      cout << sizeGS[ i ] << '\t';
  cout << ")." << endl;

  size_t vaGSstart = vaGSlice.start ( );

  cout << "vaResult uchun birinchi element indeksi: "
      << vaGSstart << "." << endl;

  const valarray <size_t> strideGS = vaGSlice.stride ( );

  cout << "vaResult ychun qadamlar:"
```

```
      << " vaGSlice.stride ( ) = ( ";
    for ( i = 0 ; i < 2 ; i++ )
      cout << strideGS[ i ] << '\t';
  cout << ")." << endl;
  system("pause");
}
```

1.4-dastur. Output

```
valarray massiv elementlari:
(       1       2       3       4       5       6       7       8       9       10      11      12      13      14
15      16      17      18      19      20 )
 vaGSlice asosida vaResult elementlari:
va[vaGSlice] = (        1       5       9       13      3       7       11      15      5       9)
vaResult elementlar soni: vaGSlice.size ( ) = ( 6       4       ).
vaResult uchun birinchi element indeksi: 0.
vaResult ychun qadamlar: vaGSlice.stride ( ) = ( 2      4       ).
```

Dastur tahlili bilan mutaqil shug‘ullaning.

Topshiriq №2

1. Valarray sinfidan foydalanib, ikkita massivning yig’indisini toping.
2. Valarray sinfidan foydalanib, ikkita massivning farqini toping.
3. Valarray sinfidan foydalanib, massivning maksimal elementini toping.
4. Valarray sinfidan foydalanib, massivning minimal elementini toping.
5. Valarray sinfidan foydalanib, massiv elementlarini o’sish tartibda saralang.
6. Valarray sinfidan foydalanib, massiv elementlarini kamayish tartibda saralang.
7. Valarray sinfidan foydalanib, massivning barcha elementlari uchun inkrementini (prefiks (++i) va postfiks (i++) lar uchun) amalga oshiring.
8. Valarray sinfidan foydalanib, massivning barcha elementlari uchun dekrementini (prefiks (--i) va postfiks (i--) lar uchun) amalga oshiring.
9. Slice sinfidan foydalanib valarray massiv elementlari sonini aniqlang.
10. Slice sinfidan foydalanib valarray massiv elementlaridan toq qiymatlilar sonini aniqlang.
11. Slice sinfidan foydalanib valarray massiv elementlari har 3 qadamdagilarni aniqlang.
12. Slice sinfidan foydalanib valarray massiv elementlari har 5 qadamdagilarni aniqlang.
13. Slice sinfidan foydalanib valarray massiv elementlari har 1 qadamdagilarni aniqlang.
14. Slice sinfidan foydalanib valarray massiv elementlari har 4 qadamdagilarni aniqlang.
15. Slice sinfidan foydalanib valarray massiv elementlari har 2 qadamdagilarni aniqlang.

№14-AMALIY MASHG'ULOT

MAVZU: MURAKKAB SARALASH ALGORTMLARI

Ishning maqsadi: C++ dasturlash saralsh **algoritmlarini o'rganish va tahlil** qilish va uning dasturini yaratish ko'nikmalarini egallsh.

Bubble Sort usuli

```
#include<iostream>
using namespace std;
int main ()
{
   int i, j,temp;
   int a[5] = {10,2,0,43,12};
   cout <<"Input list ...\n";
   for(i = 0; i<5; i++) {
   cout <<a[i]<<"\t";
}cout<<endl;
for(i = 0; i<5; i++) {
   for(j = i+1; j<5; j++)
   {
      if(a[j] < a[i]) {
         temp = a[i];
         a[i] = a[j];
         a[j] = temp;
      }
   }
}
cout <<"Sorted Element List ...\n";
for(i = 0; i<5; i++) {
   cout <<a[i]<<"\t";
   }
return 0;
```

```
}
```

Chiqish natijasi:
Ro‘yxat…
10 2 0 43 12
Saralangan elementlar ro‘yxati…
0 2 10 12 43

Chiqarilishdan ko'rinib turibdiki, qabariqni tartiblash usulida, har bir pas bilan eng og‘ir element massiv oxiriga qadar puflanadi va shu bilan qatorni to'liq tartiblashtiriladi.

Selection sort amalga oshiramiz.

```
#include<iostream>

using namespace std;

 int findSmallest (int[],int);

int main ()

{

   int myarray[5] = {12,45,8,15,33};

   int pos,temp;

   cout<<"\n Input list of elements to be Sorted\n";

   for(int i=0;i<5;i++)

   {

      cout<<myarray[i]<<"\t";

   }

   for(int i=0;i<5;i++)

   {

      pos = findSmallest (myarray,i);

      temp = myarray[i];

      myarray[i]=myarray[pos];

      myarray[pos] = temp;

   }

   cout<<"\n Sorted list of elements is\n";

   for(int i=0;i<5;i++)

   {
```

```
        cout<<myarray[i]<<"\t";
    }
return 0;
}
int findSmallest(int myarray[],int i)
{
    int ele_small,position,j;
    ele_small = myarray[i];
    position = i;
    for(j=i+1;j<5;j++)
    {
        if(myarray[j]<ele_small)
        {
        ele_small = myarray[j];
        position=j;
        }
    }
    return position;
}
```

Chiqish natijasi:
Saralangan elementlarning kiritish ro'yxati
12 45 8 15 33
Saralangan elementlar ro'yxati bu
8 12 15 33 45

Selection sort, har bir pas bilan, massivning eng kichik elementi kerakli holatda joylashadi. Shunday qilib, tartiblash jarayoni oxirida biz to'liq tartiblangan qatorga ega bo'lamiz.

, **Insertion Sort texnikasini amalga oshiramiz.**

```
#include<iostream>
using namespace std;
int main ()
{
   int myarray[5] = { 12,4,3,1,15};
```

```
    cout<<"\nInput list is \n";
    for(int i=0;i<5;i++)
    {
       cout <<myarray[i]<<"\t";
    }
    for(int k=1; k<5; k++)
    {
       int temp = myarray[k];
       int j= k-1;
       while(j>=0 && temp <= myarray[j])
       {
          myarray[j+1] = myarray[j];
          j = j-1;
       }
    myarray[j+1] = temp;
    }
 cout<<"\nSorted list is \n";
 for(int i=0;i<5;i++)
    {
    cout <<myarray[i]<<"\t";
    }
 }
```

tezkor tartiblash texnikasini amalga oshiramiz.

NAZORAT SAVOLLARI

1. Murakkabligi o‘rtacha va eng yomon hol uchun $O(n^2)$ va eng yaxshi uchun $O(n)$ bo‘lgan saralash algoritmini ayting?
2. Shell caralash algoritmini tasniflab bering?
3. multiset konteynyerdan qaysi saralash algoritmida foydalanish mumkin?
4. Qo‘shish orqali saralashga o‘xshash kabi saralaydigan algoritmlarni sanab bering?
5. Tanlash orqali saralash g‘oyasini rivojlantirilganvarianti bu qaysi algoritm?
6. Massivni ikkiga bo‘lasiz, qismlarni rekursiv ravishda tartiblaysiz va keyin birlashtirish protsedurasini bajarasiz. Bu qaysi saralash algoritmi?

7. Bitli saralash algoritmlarning qanday variantlari bor?
8. Biton ketma-ketlik nimani anglatadi va qaysi saralash algoritmda ishlatiladi.
9. Saralash algoritmlarini nima uchun taqqoslanadi?
10. Tasodifiy sonlar bilan to‘ldirilgan massivni saralash tasodifiy sonlar chegarasiga bog‘liqmi?
11. Tasodifiy sonlarning eng kattasi 10000 bo‘lganda o‘zini eng yomon tutudigan saralash algoritmini ayting?
12. Massiv elementlari soni 100000 bo‘lganda o‘zini yomon tutudigan saralash algoritmlarini ayting?

1-misol. Agar A massiv elementi 0 dan boshlansa o‘sib borish tartibida joylashtiring aks holda kamayib borish tardibida joylashtiring.
2. Katta sonlarni saralovchi dastur tuzing (kamida 15 xonali bo‘lsin)

2-misol. A Massiv elementlarin tasadifiy sonlar bilan to‘ldiring toq o‘rinda turgan elemrntlar olib B massiv hosil qling va o‘sib borish tartibida joylashtiring.

2. Katta sonlarni saralovchi dastur tuzing (kamida 17 xonali bo‘lsin)

3-misol. Massivni tasodif sonlar bilan tuldiring tub elementlarini topib o‘sib borish tardibida chiqaring.

2. Katta sonlarni saralovchi dastur tuzing (kamida 22 xonali bo‘lsin)

4-misol. Berilgan S va V massivdan toq elementlarini A massivga joylash kamayib borish tartibida yozing. Katta sonlarni saralovchi dastur tuzing (kamida 13 xonali bo‘lsin)

5-misol. Tasodifiy sonlar orqali berilgan A massivni 2 ga karrali elementlarini o‘sib borish tartibida V massivga joylash dasturini tuzing.

2. Katta sonlarni saralovchi dastur tuzing (kamida 12 xonali bo‘lsin)

6-misol. Berilgan A va V massivdan juft elementlarini S massivga o‘sib borish tartibida joylash dasturini tuzing tasodifiy sonlardan foydalangan holda.

2. Katta sonlarni saralovchi dastur tuzing (kamida 16 xonali bo‘lsin)

7-misol. Bir o‘lchovli massiv berilgan. Massivni 3 va 2ga karrali elementlaridan bir o‘lchovli boshqa massiv xosil qiling va saralab ekranga chiqaring.
2. Katta sonlarni saralovchi dastur tuzing (kamida 19 xonali bo‘lsin)

8-misol. Ikki o‘lchovli massivning 3ga karrali elementlarining satr bo‘yisha saralang.

2. Katta sonlarni saralovchi dastur tuzing (kamida 22 xonali bo‘lsin)

9-misol. A={a_{ij}} matritsa berilgan. Matritsa ustunlarida bo‘yicha kamayib borish tartibida saralang.
2. Katta sonlarni saralovchi dastur tuzing (kamida 25 xonali bo‘lsin)

10-misol. Berilgan massiv elementlaridan 3 ta massiv hosil kiling. Birinchisiga 2 ga karrali bo‘lgan, ikkinchisiga 5 ga karrali bo‘lgan,3 chisiga esa qolgan elementlarini joylovchi dastur tuzing,hamma masivga o‘sib borish tartibida yozilsin.

2. Katta sonlarni saralovchi dastur tuzing (kamida 11 xonali bo‘lsin)

11-misol. Butun sonlardan iborat bo‘lgan massivning elementlarini har birini kvadratga oshirib o‘sib borish tardibida saralang.
2.Katta sonlarni saralovchi dastur tuzing (kamida 19 xonali bo‘lsin)

12-misol. X son va A={a_{ij}} matritsa berilgan. Matritsa elementlari ichida X sonidan kichik elementlarini saralang.
2.Katta sonlarni saralovchi dastur tuzing (kamida 24 xonali bo‘lsin)

13-misol. A={a_{ij}} matritsa berilgan. Matritsaning juft nomerli ustun elementlari saralansin.

2. Katta sonlarni saralovchi dastur tuzing (kamida 20 xonali bo‘lsin)

14-misol. Birinchi uchragan manfiy va ikkinchi uchragan manfiy sonlar o‘rtasidagi elementlarni kamayib borish tartibida joylashtiring.

2. Katta sonlarni saralovchi dastur tuzing (kamida 18 xonali bo‘lsin)

15-misol. English alifbosida harflar kiritilganda ularni alphabet bo‘yicha saralang.

2. Katta sonlarni saralovchi dastur tuzing (kamida 14 xonali bo‘lsin)

16-misol. Kiritilgan simbollar ichidan unli harflarni topib saralash tartibida yozish.

2. Katta sonlarni saralovchi dastur tuzing (kamida 20 xonali bo‘lsin)

№15-AMALIY MASHG’ULOT

MAVZU: LARGE SINFINING QO`LLANILISHI. JUDA ҲAM KATTA SONLAR BILAN ISHLASH.

Ishning maqsadi: C++ dasturlash tilida Large sinfining qo’llanilishi va juda ham katta sonlar bilan ishlash dasturini yaratish ko‘nikmalarini egallsh.

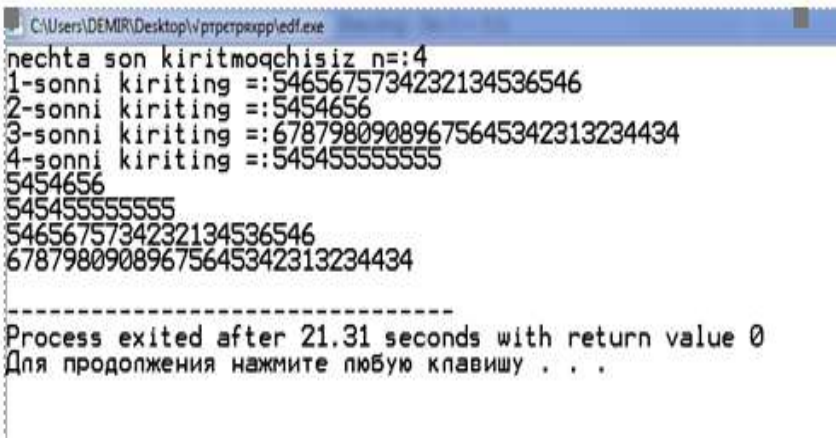
Amaliyot mashg‘ulotini bajarish bo‘yicha variantlar

Saralash dasturi.

- *//C ++ dasturi taqdim etilgan katta raqamlarni saralash uchun*
- *// satr sifatida.*
- **#include<bits/stdc++.h>**
- **using namespace std;**
- *// agar str1 str2 dan kichik bo'lsa, u haqiqiy qiymatni qaytaradi.*
- **bool raqamlarni_taqqoslash(string satr1, string satr2) {**

```
// Ikkala qatorning uzunligini hisoblang
    int n1 = satr1.length(), n2 = satr2.length();
   if (n1 < n2)
     return true;
    if (n2 < n1)
       return false;
  // Agar uzunliklar bir xil bo'lsa
    for (int i=0; i<n1; i++)
    {   if (satr1[i] < satr2[i])
          return true;
       if (satr1[i] > satr2[i])
          return false;
    } return false; }
// Ko'p sonli qatorlarni saralash funktsiyasi
// satrlar shaklida berilgan
void kattasonlarnisaralash(string arr[], int n)
{ sort(arr, arr+n, raqamlarni_taqqoslash); }
  // Boshqaruvchi  dastur
int main()
{   int n;
    cout<<"nechta son kiritmoqchisiz n=:";
    cin>>n;
    string sonlar[n];
    for(int i=0;i<n;i++)
    {cout<<i+1<<"\a-sonni kiriting =:";
      cin>>sonlar[i]; }
    kattasonlarnisaralash(sonlar, n);
    for (int i=0; i<n; i++)
      cout << sonlar[i] <<endl;
}
```

Natija:



NAZORAT SAVOLLARI

1. Murakkabligi o‘rtacha va eng yomon hol uchun $O(n^2)$ va eng yaxshi uchun $O(n)$ bo‘lgan saralash algoritmini ayting?
2. Shell saralash algoritmini tasniflab bering?
3. multiset konteynyerdan qaysi saralash algoritmida foydalanish mumkin?
4. Qo‘shish orqali saralashga o‘xshash kabi saralaydigan algoritmlarni sanab bering?
5. Tanlash orqali saralash g‘oyasini rivojlantirilganvarianti bu qaysi algoritm?
6. Massivni ikkiga bo‘lasiz, qismlarni rekursiv ravishda tartiblaysiz va keyin birlashtirish protsedurasini bajarasiz. Bu qaysi saralash algoritmi?
7. Bitli saralash algoritmlarning qanday variantlari bor?
8. Biton ketma-ketlik nimani anglatadi va qaysi saralash algoritmda ishlatiladi.
9. Saralash algoritmlarini nima uchun taqqoslanadi?
10. Tasodifiy sonlar bilan to‘ldirilgan massivni saralash tasodifiy sonlar chegarasiga bog‘liqmi?
11. Tasodifiy sonlarning eng kattasi 10000 bo‘lganda o‘zini eng yomon tutudigan saralash algoritmini ayting?
12. Massiv elementlari soni 100000 bo‘lganda o‘zini yomon tutudigan saralash algoritmlarini ayting?
13. Katta sonlar bilan ishlash bilan ishlash nima uchun kerak?
14. Nima uchun sonlarga diapozon tushunchasi kiritilgan?
15. Sonlar ustida arifmeik amallarni bajargada xato natija nima uchun chiqishini tushuntirib

bering?

16. int64 tipining diapozonini ayting?

17. Katta sonlar chegarasini tekshirish mumkinmi?

18. 20 xona aniqlikda ishlovchi tipni qanday tip?

19. Katta sonlar bilan ishlaganda xatoliklar qachon chiqib keladi yoki necha xonadan keyin. Nima uchun va misol keltiring?

20. Katta sonlarni mos tiplardan foydalanmasdan ham arifmetik amallarni bajarish mumkinmi, qanday va nima uchun?

21. Katta sonlarni massivga olib, qo‘shish amalini bajarish g‘oyasini ayting?

22. Katta sonlarni massivga olib, ayirish amalini bajarishdagi dastlabki asosiy g‘oya nimadan iborat?

23. Katta sonlarni massiv ostilarga oli bajarish mumkinmi?

24. Katta sonlar uchun turli operatorlarni aniqlash mumkinmi va qanday?

25. OpenSSL qanday kutubxona va nima uchun ishlatiladi?

26. Libgcrypt kutubxonasi qanday kutubxona va nima uchun ishlatiladi?

27. GMP qanday kutubxona va nima uchun ishlatiladi?

28. Boost.Multiprecision kutubxonasi nimasi bilan ustun?

29. C++ standartlashtirish bo‘yicha Milliy ishchi guruhdan rus ishlab chiquvchilari standart kutubxonaga mustaqil ravishda qanday tiplarini kiritish taklifi bilan chiqqan?

30. using int_mb = wide_int<1024*1024>; dastur fragmentini tushuntirib bering?

31. BigInt kutubxonasi haqida nimani bilasiz va undagi stoi funksiyasining vazifasini tushuntirib bering.

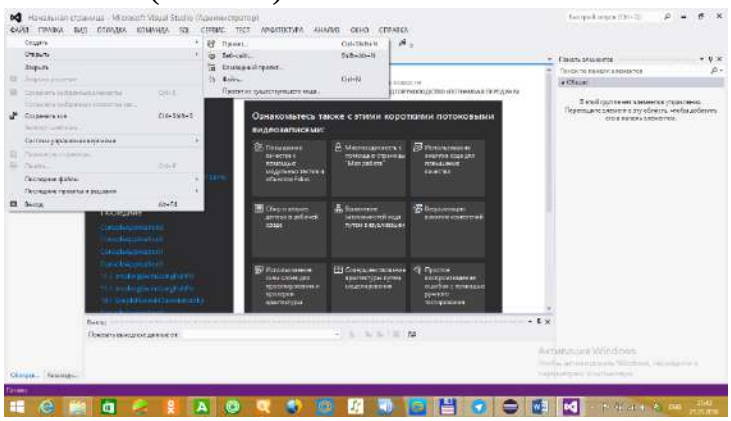
№16-AMALIY MASHG’ULOT

MAVZU: VISUAL C++ MUHITIDA DASTURLASH. VISUAL C++ MUHITIDA MENYULAR VA USKUNALAR PANELI

Ishni bajarishdan maqsad: Windows ilovalar yaratishni, hodisalar va xossalar bilan ishlashni o‘rganish.

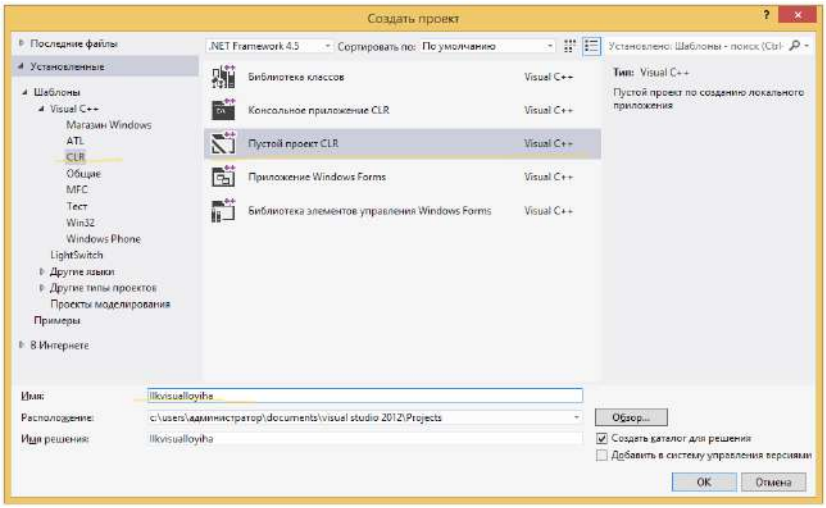
Nazariy qism: Visual Studioda Windows Forms loyihasini yaratish Ma’lumki, ancha vaqtlar davomida Windows operatsion tizimining oynalari bilan ishlaydigan dasturni tuzish har bir dasturlash tizimining oldiga qo‘yilgan eng asosiy masala bo‘lib keldi. Endilikda esa, buni yaratish imkoniyati mavjud bo‘lmagan dasturlash vositasi “eng qoloq” dasturlash vositasi hisoblanadi. Shu jumladan, Visual Studio muhiti ham C++ dasturlash tilida Windows ilovalar yaratishni qo‘llab-quvvatlaydi. Buni amalda ko‘rib chiqish uchun Visual Studio 2012 dasturidan foydalanamiz.

Buning uchun, Visual Studio 2012 dasturini ishga tushiramiz va quyidagi ketma-ketlikni tanlaymiz: *Fayl* → *Sozdat* → *Proekt* (1.1-rasm).



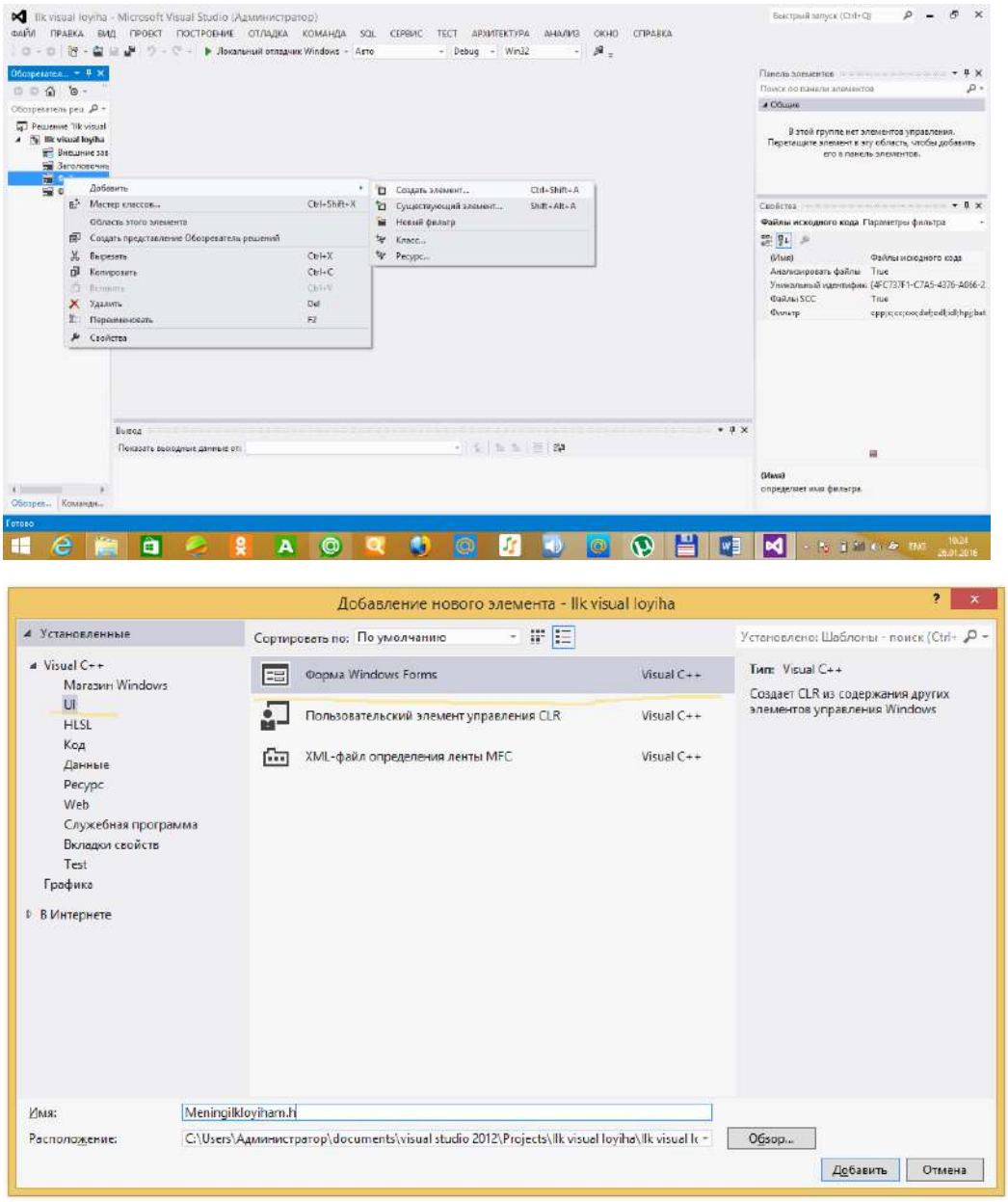
1.1-rasm. Visual Studio 2012 da loyiha yaratish

So‘ngra, quyida ochilgan oynada (1.2-rasm) quyidagi ketma-ketlik amalga oshiriladi:



1.2-rasm. Bo‘sh CLR loyiha yaratish.

Undan so‘ng, CLR loyihaga forma qo‘shish kerak, bo‘ladi. Buning uchun, quyidagi amallar ketma-ketligi amalga oshiriladi (9.3-rasm):



1.3-rasm. Loyihaga forma qo‘shish.

Natijada, ish maydonida windows oynasi hosil bo‘ladi, ammo dastur kompilyatsiya qilinganda xatolik yuzaga keladi. Shuning uchun, quyidagi amallar ketma-ketligini bajarish lozim: 1. **Faylы isxodnogo koda** bandidan, *.cpp faylga o‘tiladi (qaralayotgan misolda, Meningilkloyiham.cpp) 2. *.cpp kod ichida yaratilgan UI fayl, kutubxona sifatida e’lon qilinadi (qaralayotgan misolda Meningilkloyiham.h) 3. Ilkvisualloyiha nomli loyihaning elementlaridan foydalanishimizni inobatga olib, loyihani **using namespace** kalit so‘zi bilan e’lon qilamiz.

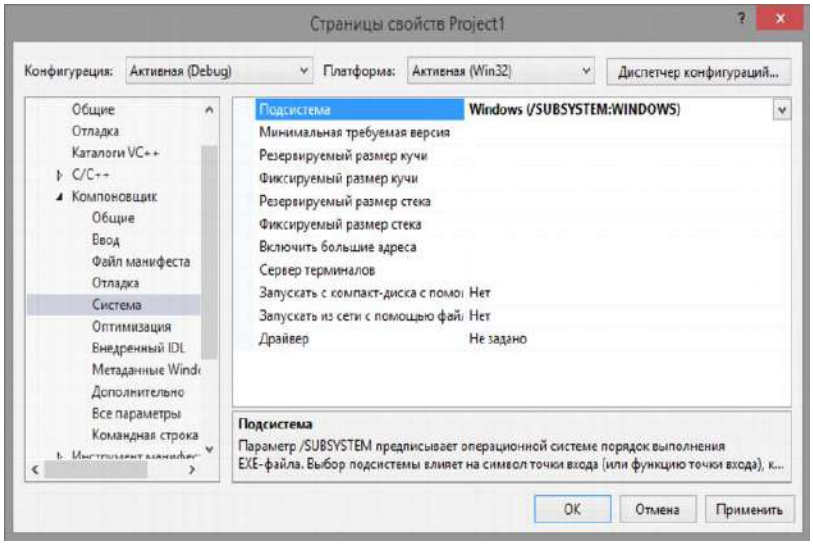
Main funksiyasi ichiga quyidagi kodni kiritamiz: System::Windows::Forms::DialogResult dr; Meningilkloyiham ^ newdlg = gcnew Meningilkloyiham(); dr=newdlg->ShowDialog();

Bir so‘z bilan aytganda, Meningilkloyiham.cpp fayli ichiga quyidagi kod yoziladi:

```
#include "Meningilkloyiham.h"
using namespace System;
using namespace System::Windows::Forms;
```

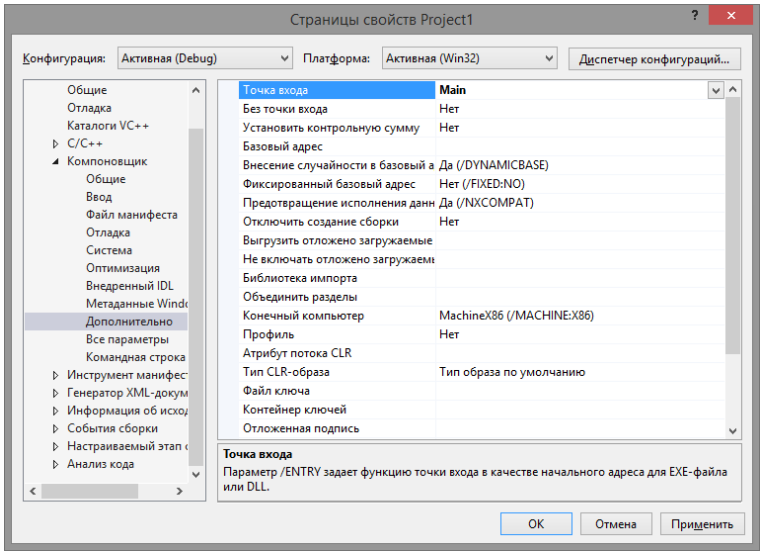
```
using namespace Ilkvisualloyiha;
void Main(array<String^>^ args)
{
  System::Windows::Forms::DialogResult dr;
  Meningilkloyiham ^ newdlg = gcnew Meningilkloyiham();
  dr=newdlg->ShowDialog();
}
```

4. Loyiha ustida konteks menyu chaqiriladi, va undan **Svoystvo** bandi tanlanadi.
5. **Svoystvo** oynasidan, **Komponovщik** bandiga, uning ichidan **Sistema** bandiga o‘tiladi (9.4-rasm).



1.4-rasm. Loyiha uchun dasturning yuklanish shaklini tanlash

Undan so‘ng, dasturga KIRISH nuqtasi sifatida, **Main** funksiyasi nomini beramiz (9.5-rasm):

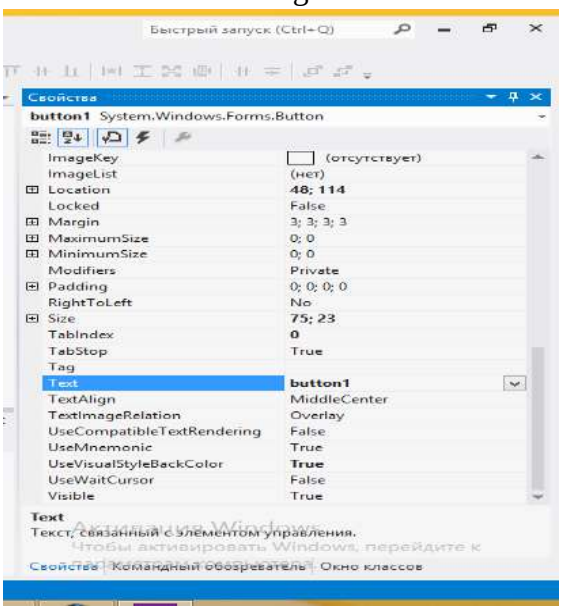


1.5-rasm. Loyihaga KIRISH nuqtasini berish **Ok** tugmasini bosish bilanoq, dastur bajariladi.

Xossalar

Ish maydoniga joylashtirilgan har qanday obyekt o‘zining xususiyatiga egadir. Masalan, **Solution Explorer**dagi fayl o‘zini turgan joyini ifodalaydigan papka yo‘lini ko‘rsatadigan xususiyatga ega. Palitradagi komponent (masalan, tugmacha) ishlash vaqtida ko‘rinuvchi, formadagi o‘zining nomiga, koordinatasiga va h.k. ega.

Faol (ya’ni, ayni vaqtda sichqoncha bilan bosilgan) obyektning barcha xususiyatlari Properties (svoystva) nomli alohida oynada aks etib turadi. Bu oyna ko‘rinib turgan bo‘lishi ham mumkin yoki yo‘q. Oynani ko‘rsatish uchun formadagi istalgan obyektning ustida konteks menyu chiqariladi hamda undan Properties opsiyasi tanlanadi. Ushbu elementning ko‘rinishi 9.6-rasmda aks etgan



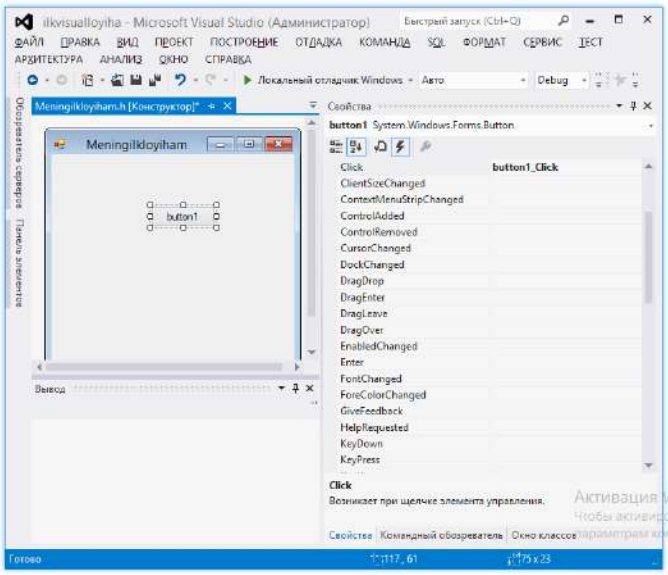
1.6-rasm. Obyektning xususiyatlar oynasi, xossalar bo‘limi

Hodisalar

Hodisalar – obyektga qo‘shiladigan usullar bo‘lib, bu usullar har bir komponenta uchun standart ishlab chiqilgan va har bir usul dasturdan foydalanayotgan shaxsning ma’lum bir harakati amalga oshirilgandagina bajariladi.

Masalan, o‘sha obyektni sichqonchaning chap tugmachasi bilan bir marta chertish.

Obyektga qo‘shiladigan usullarga o‘sha obyektning **Xususiyatlar (Svoystvo)** oynasidan, **Events** (**Sobытiya)** bo‘limiga o‘tiladi. Obyektning Properties oynasining ko‘rinishi 1.7 – rasmda aks ettirilgan.

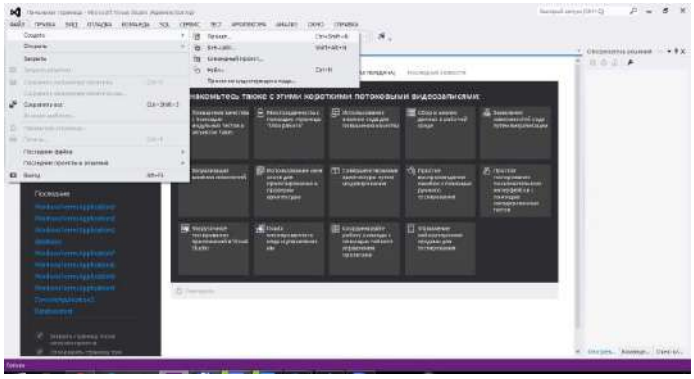


1.7-rasm. Obyektning xususiyatlar oynasi, hodisalar bo‘limi

Form sinfi

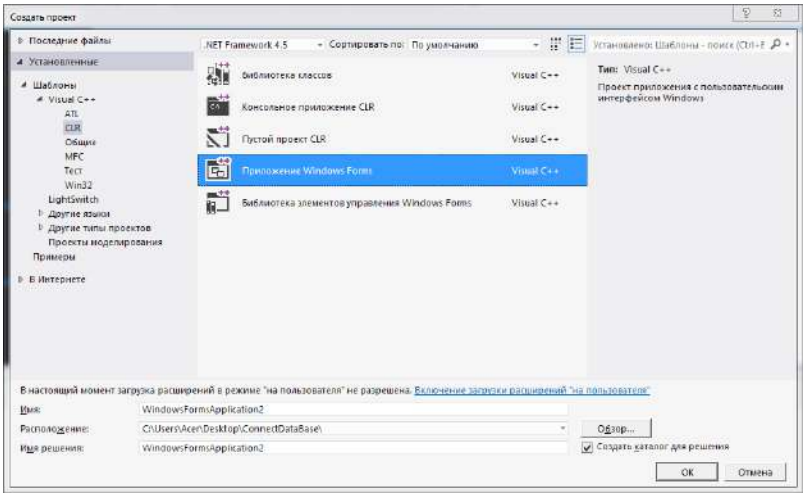
Visual Studio 2012 dasturlash muhitida **Console Application** qismidan tashqari Windows Application dasturlash muhiti ham mavjud. Bu muhitda biz interfeysga ega bo‘lgan dasturlarni yarata olamiz. Hozirgi zamonaviy dasturlash tillarida interfeysli dasturlashga katta ahamiyat qaratilgan. Microsoft Windows operatsion tizim uchun dastur tuzishda Visual Studio 2012 muhitida C++ dasturlash tilidan foydalanishimiz
mumkin.

Visual C++ dasturlash tilida Form sinfini yaratish uchun yangi loyiha yaratiladi va loyihani yaratish quyidagicha amalga oshiriladi: **Fayl->Sozdat->Proekt** yoki Ctrl+Shift+N tugmalarini bosish kerak (1.8 rasm)



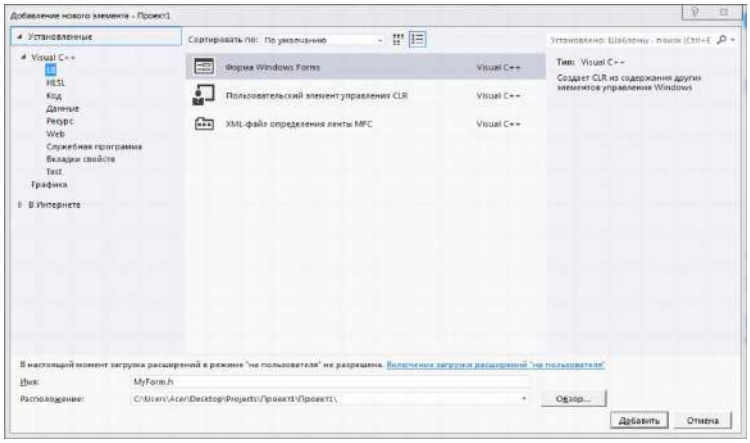
1.8-rasm. Loyiha yaratish

Loyihani yaratishda Visual C++ning CLR bo‘limidan Prilojeniya Windows Forms tanlanadi, loyihaning nomi va kompyuter xotirasiga o‘rnatiladigan joyi tanlanadi va OK tugmasi bosiladi (1.9-rasm).



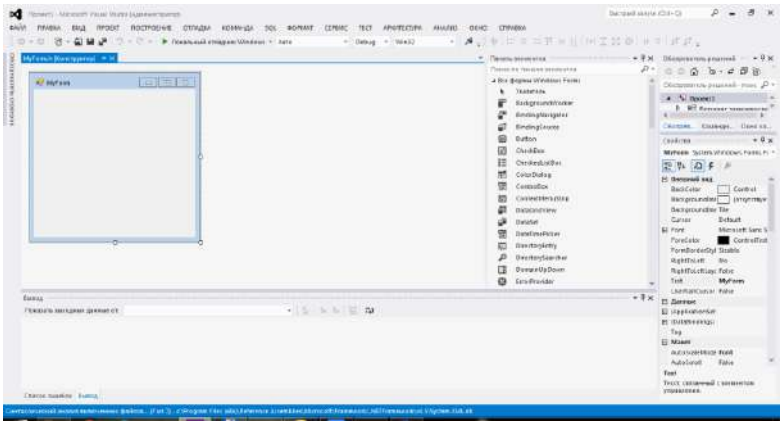
1.9-rasm. Loyiha yaratish

Natijada yangi bo‘sh loyiha yaratiladi va bu loyiha asosida yangi form oynasini hosil qilamiz. Buning uchun *proekt* menyusidan *dobavit novыy element* tugmasi bosiladi yoki **Ctrl+Shift+A** tugmalari bir vaqtda bosilishi lozim. Bunda quyidagi oyna hosil bo‘ladi va ushbu oynada **UI** bo‘limida **Forma Windows Forms** tanlanadi va formaning ismi yozilib **dobavit** tugmasi bosiladi (1.10-rasm)



1.10-rasm. Yangi form oynasini hosil qilish.

Dobavit tugmasi bosilgach quyidagi oyna hosil bo‘ladi.

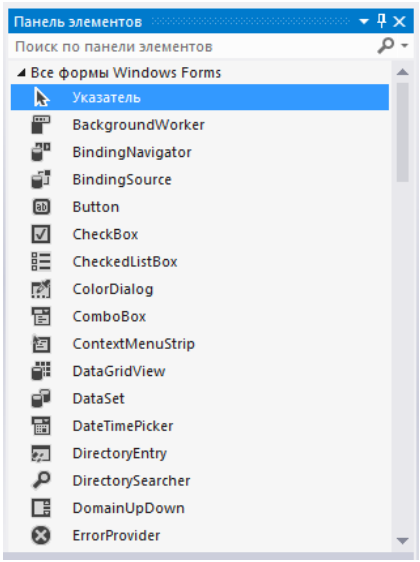


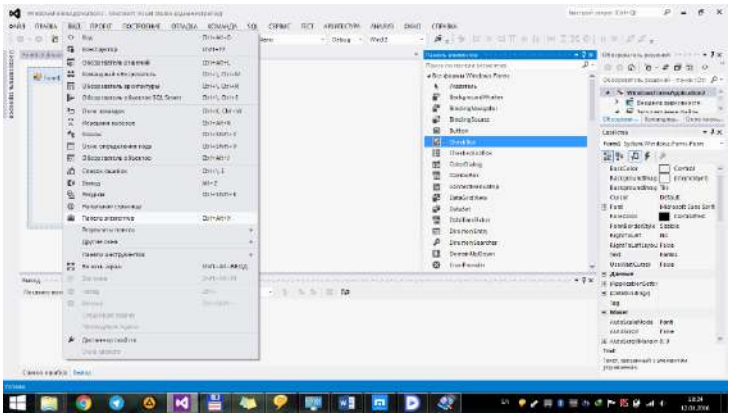
1.11-rasm. MyForm oynasi

Oyna hosil qilingach Ctrl+F5 tugmalari bosilsa, loyiha ishga tushadi.

Microsoft Visual C++ muhitida komponentalar

Komponentalar Form oynasi hosil qilingach paydo bo‘ladi. Komponentalar *Panel elementov* yoki *ToolBox* da joylashgan bo‘ladi. Agarda ekranda ushbu oynacha bo‘lmasa unda biz Ctrl+Alt+X tugmalarini bosishimiz kerak bo‘ladi. Yoki View menyusidan *ToolBoxni*(**VID-> panel elementov**) tanlashimiz kerak.





1.12-rasm. Komponentalar oynasini hosil qilish

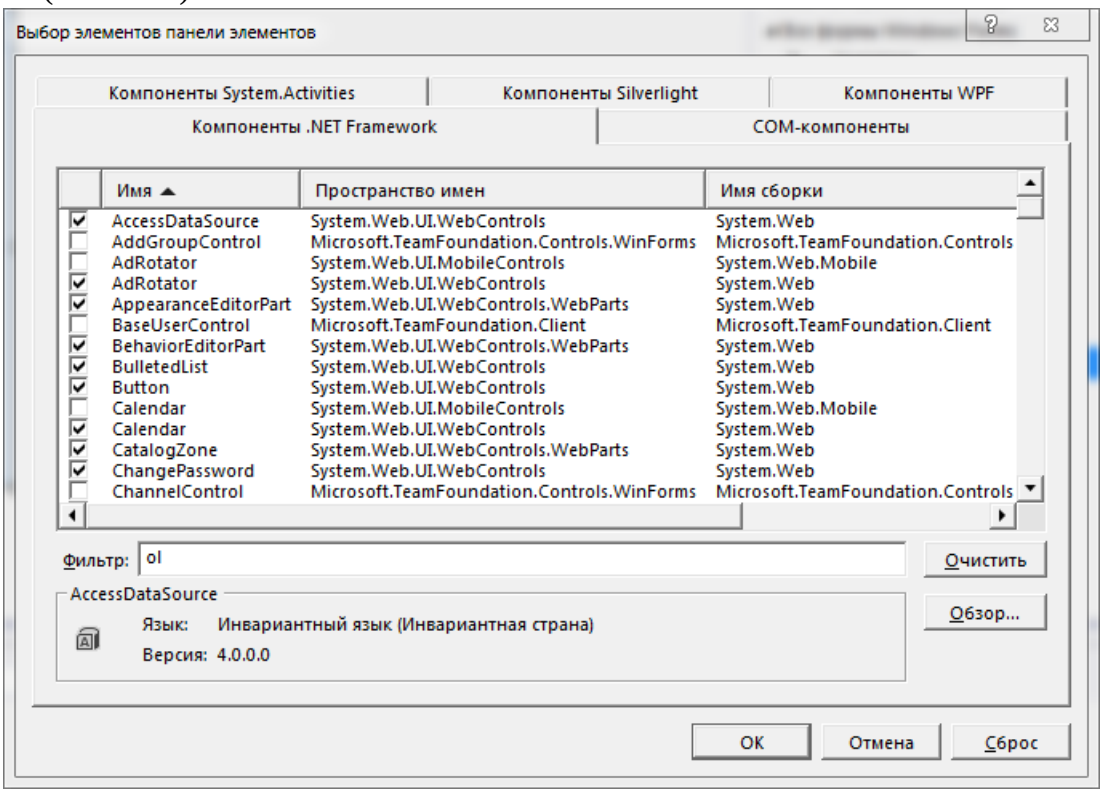
Komponentalar oynasi quyidagi asosiy komponentalardan tashkil topadi:

- TextBox
- Label
- Combobox
- PictureBox
- ProgressBar
- Button
- CheckBox
- RadioButton
- DateTimePicker
- ListBox

- Panel
- TabControl

1.13-rasm. Komponentalar oynasi

Agarda bizga kerak bo‘lgan komponenta ushbu oynada mavjud bo‘lmasa u holda **ToolBox** oynasi ustida sichqonchaning o‘ng tugmasini bosamiz va *Vыbrat elementы*, yoki *choose elements* ni tanlaymiz.(1.13 rasm).



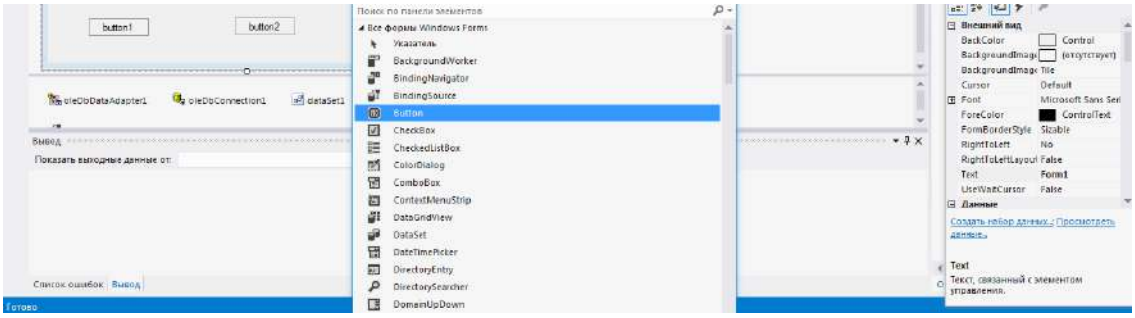
1.14-rasm. Yangi komponenta qo‘shish

Bunda Filter bo‘limiga kerak bo‘lgan komponenta yoziladi va shu komponenta ro‘yxatda paydo bo‘lgach shu komponentani belgilaymiz(checkbox. checked) va ok tugmasini bosamiz. ToolBox panelini yana bir marta tekshirganda bizga kerak bo‘lgan komponenta paydo bo‘lgan bo‘ladi.

Button komponentasi

Button komponentasi (uzb “tugma”, rus “knopka”) **All Windows Forms** komponentalar guruhida joylashgan.

Bu komponenta Form oynasining istalgan joyiga joylashtiriladi va sichqoncha tugmasini Button komponentasining ustida bosilgan paytda biror bir funksiyani amalga oshiradi. Button komponentasi quyidagi rasmda keltirilgan.



Button xossalari (Properties Button)

Button komponentasi asosiy xossalari quyidagilar: Backcolor- ushbu xossa orqali tugmachaning orqa rangini tanlashimiz mumkin.

BackgroundImage- bu xossa orqali tugmachaning orqa foniga rasm tashlash imkoni mavjud. Ammo tanlangan rasm o‘z o‘lchamini o‘zgartirmasdan tashlanadi.

BackgrounImageLayout- bu xossada orqa fonga tashlangan rasmni xajmini sozlashimiz mumkin. Tugmacha o‘lchamiga rasm o‘lchamini mos qilib olishimiz uchun **Tile** ni tanlash kerak bo‘ladi.

Font – ushbu xossada tugmachaning fontini shriftini va shrift o‘lchamini o‘zgartirishimiz mumkin.

ForeColor- Matnning rangini o‘zgartirish.

Button xossalari (Properties Button)

Button komponentasi asosiy xossalari quyidagilar: Backcolor- ushbu xossa orqali tugmachaning orqa rangini tanlashimiz mumkin.

BackgroundImage- bu xossa orqali tugmachaning orqa foniga rasm tashlash imkoni mavjud. Ammo tanlangan rasm o‘z o‘lchamini o‘zgartirmasdan tashlanadi.

BackgrounImageLayout- bu xossada orqa fonga tashlangan rasmni xajmini sozlashimiz mumkin. Tugmacha o‘lchamiga rasm o‘lchamini mos qilib olishimiz uchun **Tile** ni tanlash kerak bo‘ladi.

Font – ushbu xossada tugmachaning fontini shriftini va shrift o‘lchamini o‘zgartirishimiz mumkin.

ForeColor- Matnning rangini o‘zgartirish.

Flat style- Tugmachaning stilini tanlash. Bunda tugmacha stili ancha chioyli

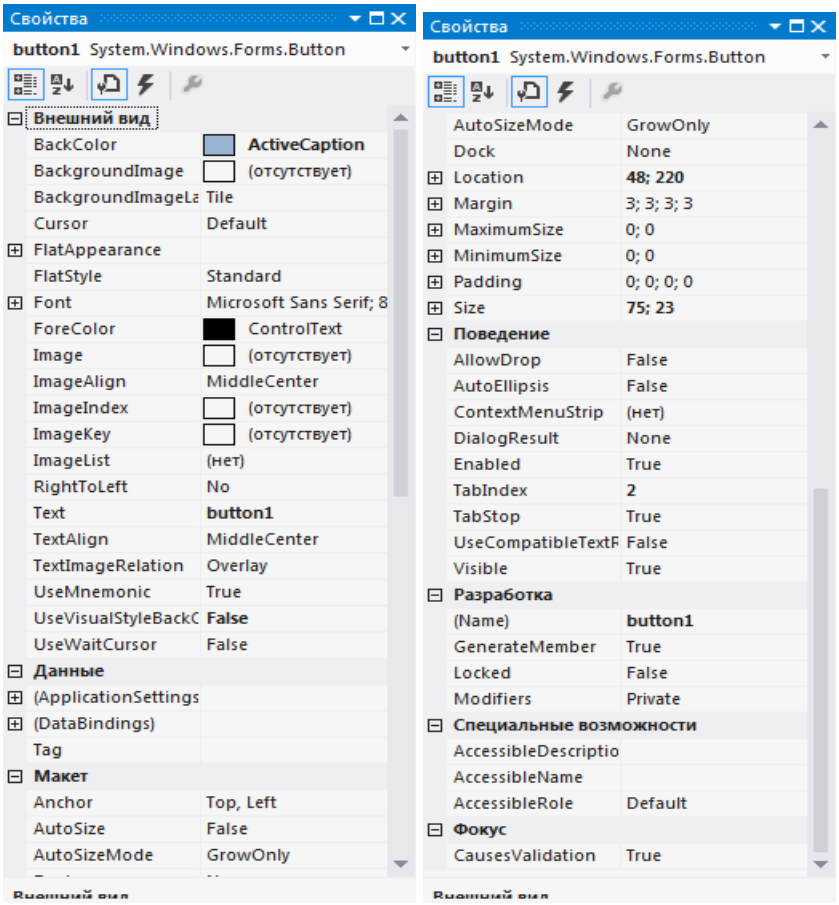
tusga kiradi.

Image- rasm tanlash. Bunda tugmacha ichiga rasm tushadi.

Text- Button1 tugmachasining ko‘rinish nomi- agar text nomi o‘zgartirilsa, dasturda chiquvchi tugmacha nomi o‘zgaradi.

Name- Button1 komponentasining nomi, bunda dasturning kod qismiga o‘tganimizda, button1 nomi bilan emas, balki uning yangi qo‘yilgan nomidan foydalanishimiz mumkin.

Dock- bunda tugmacha joylashgan o‘rnini o‘zgartirish mumkin. Uning o‘rnini ekrannning istalgan burchagiga joylashtirish yoki (fill) butun ekranni egallab qo‘yadigan qilsa bo‘ladi.



Visible- tugmachaning ko‘rinishi. Visible funksiyasi faqat **true** va **false** (rost va yolg‘on) qiymatni qabul qiladi. Agarda true ya’ni rost qiymat qabul qilingan bo‘lsa, u holda tugmacha ekranda ko‘rinib turadi, agarda false qiymat bo‘lsa ekranda

ko‘rinmaydi.

Button komponentasi hodisalari Click – Qachonki sichqoncha tugmasi bosilganda hodisa sodir bo‘lishi.

Enter – Agarda tugmacha faol holatga o‘tgan bo‘lsa, u holda kiritish tugmasini bosishda sodir bo‘luvchi hodisa.

KeyPress – Ushbu hodisada klaviaturani imkonini cheklashda foydalanish mumkin. Ya’ni kerakli klavishlarni faol qilib, qolganlarini ishlatmasdan qoldirish

mumkin.

Agarda button tugmasi ustida sichqonchani ikki marta chertilsa, unda oynaning source (Kod) qismiga o‘tiladi, va quyidagi kodl figurali qavslar ichiga yoziladi. private: System::Void button1_Click_1(System::Object^ sender, System::EventArgs^ e)

{

/*ushbu joyga button button1 tugmasi bajaradigan amalning buyrug‘i yoziladi.

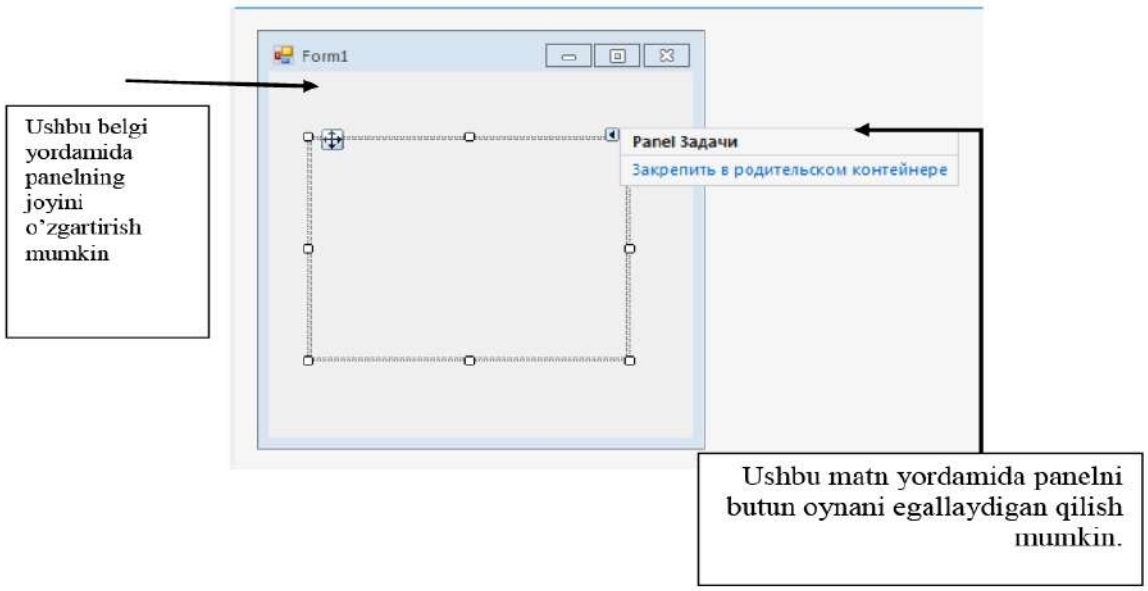
*/

}

Panel komponentasi xossalari Ushbu komponenta **All Windows Forms** palitralar komponentasida

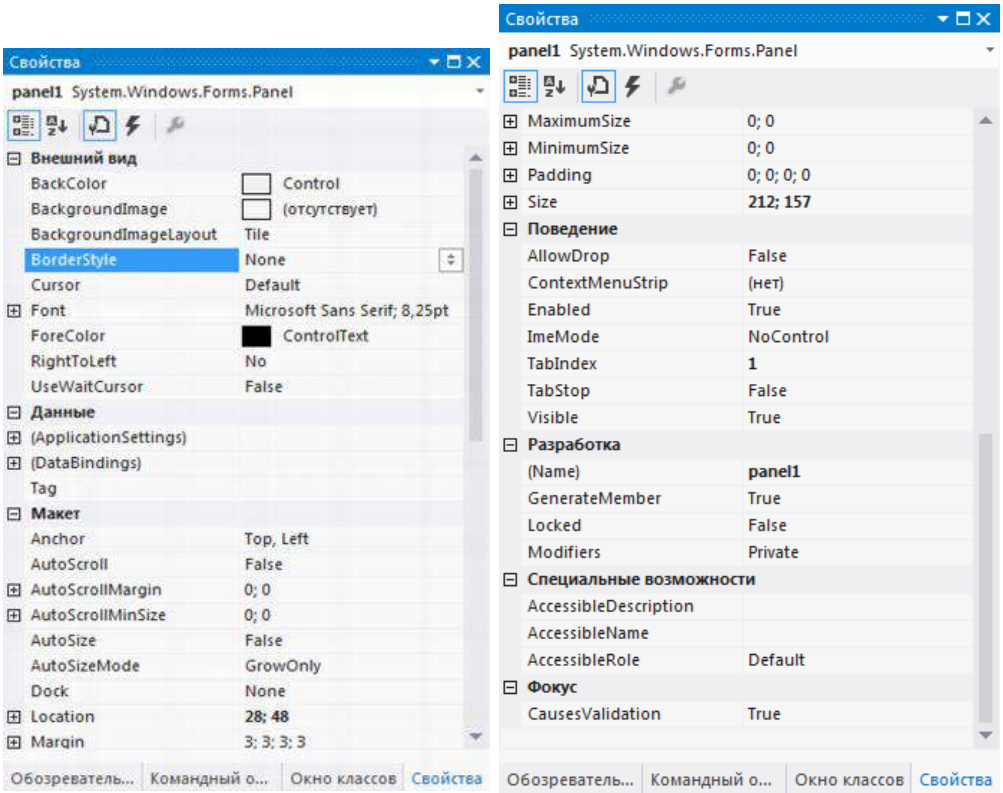
joylashgan.

Panel – Bu komponent, qaysiki, formaga o‘xshash, bitta komponentda yotadi, bu komponentani joylashgan o‘rnini o‘zgartirganimizda, uning ichida joylashgan barcha elementlar ham panel bilan o‘rnini o‘zgartiradi. Panelning ko‘rinishi quyidagi rasmda
keltirilgan.



Panelning bir nechta hodisalari

Panelning hodisalari **properties (svoystva)(Alt+Enter)**oynasi orqali tanishib olishingiz mumkin.



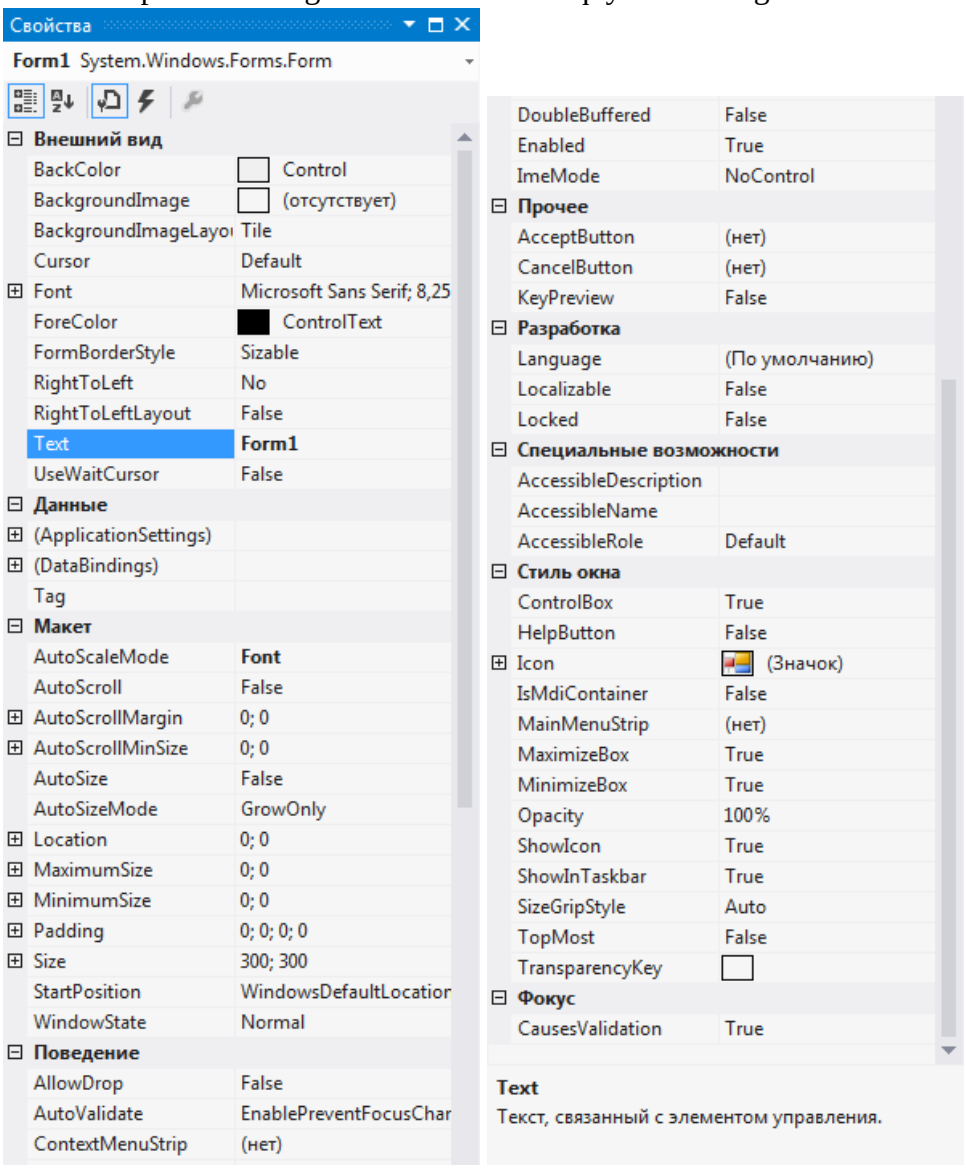
Borderstyle – panel ko‘rinishini o‘zgartirish mumkin. Bunda ko‘rinishni Fixed3D ga o‘zgartirilsa, panel Form oynasida yaqqol ko‘rinib turadi.

Location – panelning joylashgan o‘rnini o‘zgartiradi. Bunda X va Y koordinatalariga qiymat berilsa, shu koordinataga panel komponentasini joylaydi.

Label komponentasi Label komponentasi (metka) **All Windows Forms** palitra komponentalari ro‘yxatida joylashgan. Ushbu component ekranga siz yozgan matn yoki joylagan rasmingizni namoyish etadi, qachonki dastur ishga tushgan payti dastur foydalanuvchisi joylangan ma’lumotni tahlil qila olmaydi. **Labelning bir qancha komponentalari** *Text Align* – Agarda biz label komponentasi o‘lchamini qo‘l rejimiga o‘tkazsak ya’ni AutoSizeni false qilsak u holda uning Text Align xossasidan foydalanganimizda labelning matnini uning o‘lchamidan kelib chiqib hohlagan burchagiga joylay olamiz.

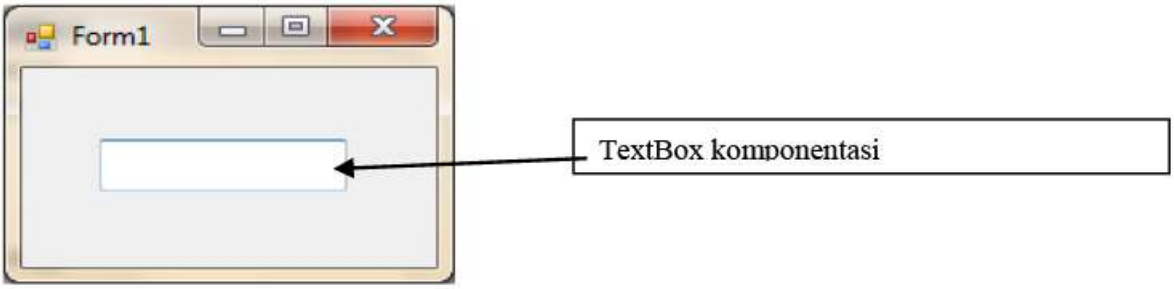
Border Style – Labelni ko‘rinishini o‘zgartirish unga qo‘shimcha dizayn yuklash.

Label komponentasining barcha elementlari quyida keltirilgan.



Textbox komponentasi

TextBox komponentasi ko‘rinishi quyidagicha



TextBox komponentasi xossalari

AcceptsReturn – ushbu xossa textbox komponentasining ko‘p satrli qismi uchun ishlatiladi va u faqat rost va yolg‘on qiymat qabul qiladi. Agarda uning qiymatini rost (true) qilib qo‘ysak u holda ushbu funkisya <Enter> tugmasini bosdganda yangi satr yaratib beradi.

Agarda ushbu xossa yolg‘on qiymat qabul qilsa (false) u holda (odatiy faol holat), foydalanuvchi yangi satr ochish uchun <Ctrl+Enter> tugmalarini bosishi kerak

bo‘ladi.

AcceptsTab – Ushbu xossa odatda false qiymatda turadi. Agarda uni true rost qiymatga o‘tkazsak, u holda u <Tab> tugmachasini ishga tushiradi. Agarda false qiymatida turgan bo‘lsa u holda <Ctrl+Tab> tugmachalrini bosishimiz kerak.

CharacterCasing – ushbu xossa yordamida textbox matnini agarda xossaning Upper qismidan foydalansak faqat katta harflarni yozadi. Agarda Low ni faol qilsak u holda faqat kichik harflar, agarda normal holatida qoldirsak, unda odatiy klaviaturadan matn kiritish mumkin.

MultiLine – Ushbu komponenta textboxni faqat bitta satrli va ko‘p satrli qilish imkoniga ega.

PasswordChar – Ushbu xossaga * belgisini qo‘yib qo‘ysak nima yozishdan qat’iy nazar yozuv faqat * lardan iborat bo‘ladi.

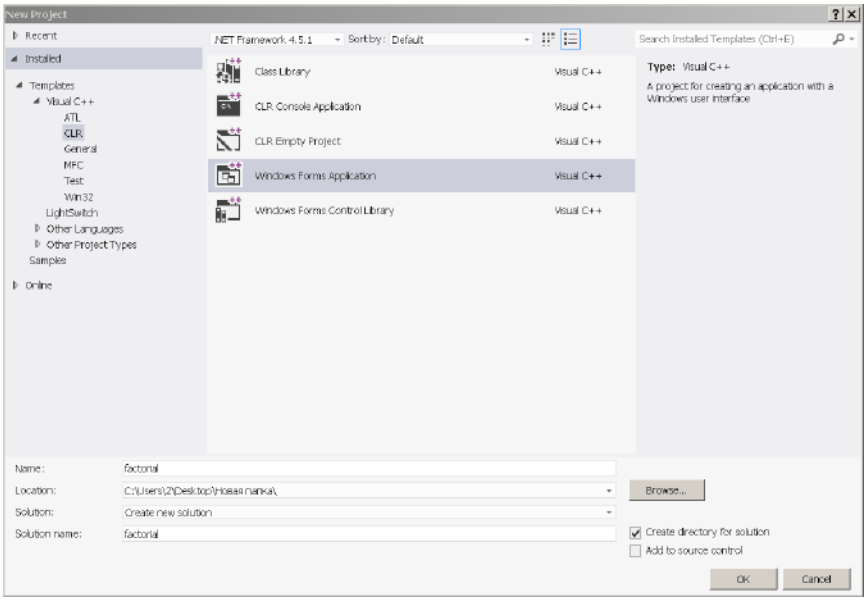
Amaliy qism

Topshiriq. Berilgan natural sonning faktorialini hisoblab berovchi Visual C++ da ilova yarating.

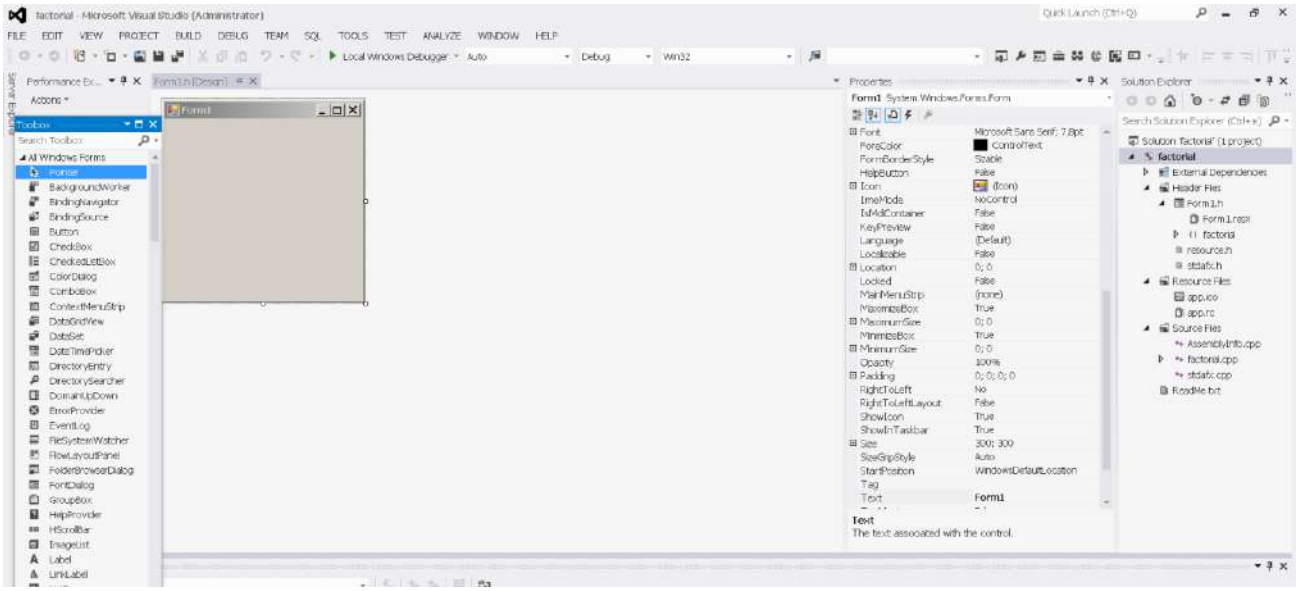
Echih:

1 chi qadam. Visual Studio dasturini ishga tushiramiz. Ochilgan muloqat oyynasidan **Fayl→Sozdat→Proekt (File→New→Project)** buyruqlar ketma-ketligini amalga oshiramiz.

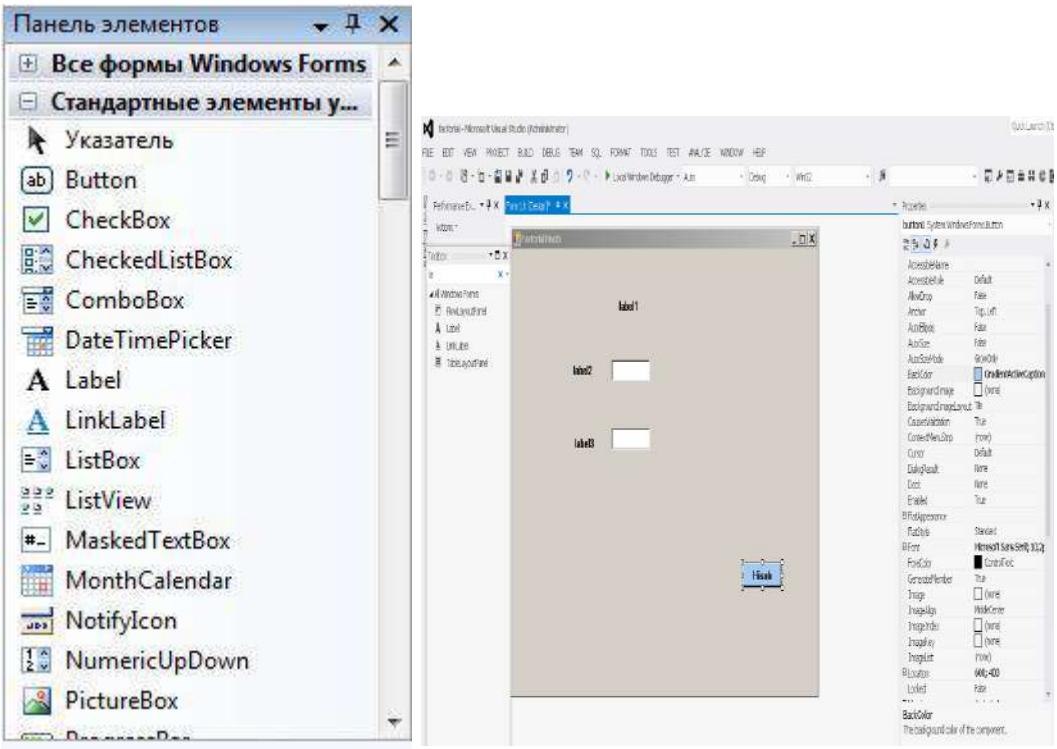
2 chi qadam. Ochilgan muloqat oyynaning tepa chap qismidan **CLR** , o‘ng qismidan esa **Prilojenie Windows Forms** belgilaymiz. Oynaning pastki qismiga proektimizning nomini kiritamiz va diskda joylastirmoqchi bo‘lgan manzilni (**Browse**) ko‘rsatamiz. **OK** tugmasini bosamiz.



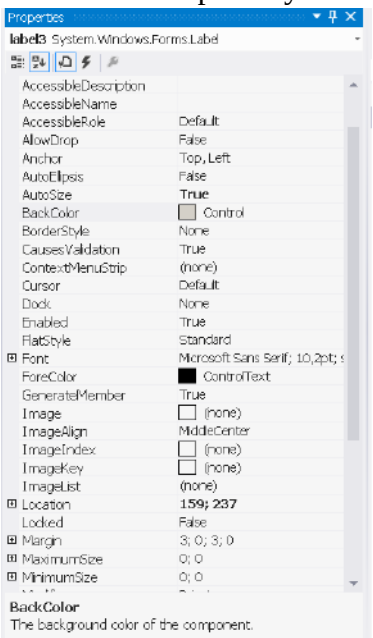
Visual Studio dasturining asosiy ishchi oynasi da bo‘sh forma ko‘rinishda ochiladi.



3 chi qadam. Bo‘sh formaga bizga zarur bo‘lgan komponentalarni joylashtishni boshlaymiz. Bular 2ta Textbox komponentasi, N – natural sonning qiymatini kiritish va Fact – faktorialning qiymatini chiqarish uchun, 3 ta Label – izohlah uchun va 1 ta Button – bajarish buyrug‘ini amalga oshirish uchun. Bularni komponentlar (elementlar) panelidan olami. Agar u ishchi sohada ko‘rinmasa, **Vid→Panel Elementov yoki Ctrl + Alt + X**. buyruqlar yordamida yuklaymiz.

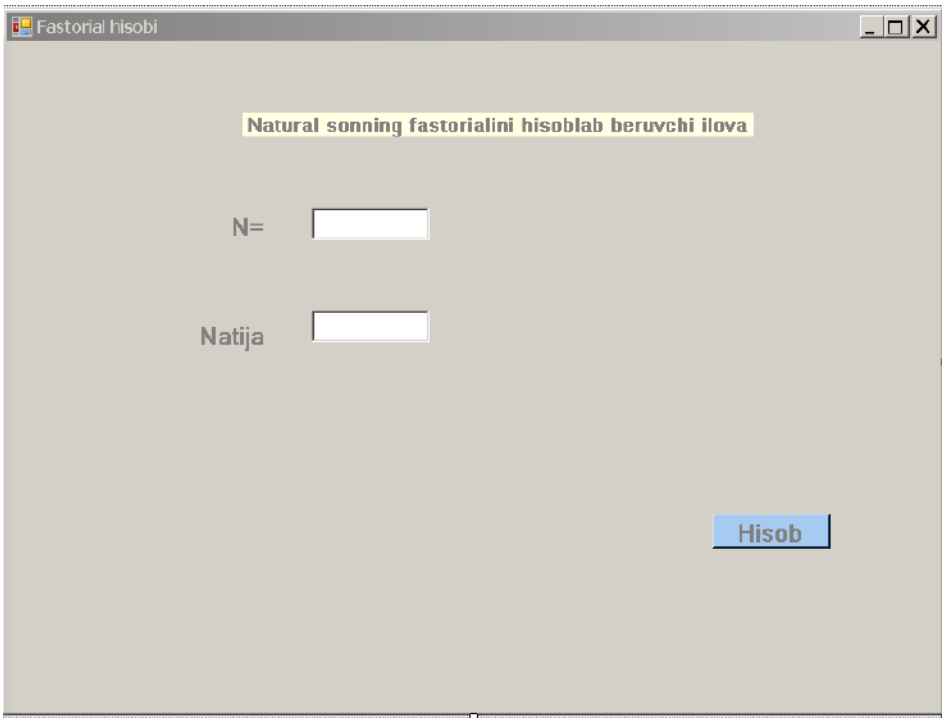


Ularning o‘lchamlarini (Size) sozlaymiz, Formadagi joyini (Location yoki sichqoncha yordamida) ko‘rsatmiz,yozyvni kiritamiz (Text), yozuv shriftini (font) belgilaymiz, kerak bo‘lsa yozuv rangini (ForeColor), fon rangini (BackColor) o‘zgartirishimiz mumkin. Bu ishlarning hammasini komponentaning ustiga siqonchani olib borib, chap tugmasini bosganda shu komponetaning xossalarini aks ettiruvchi xossalar paneli yordamida amalga oshiramiz.

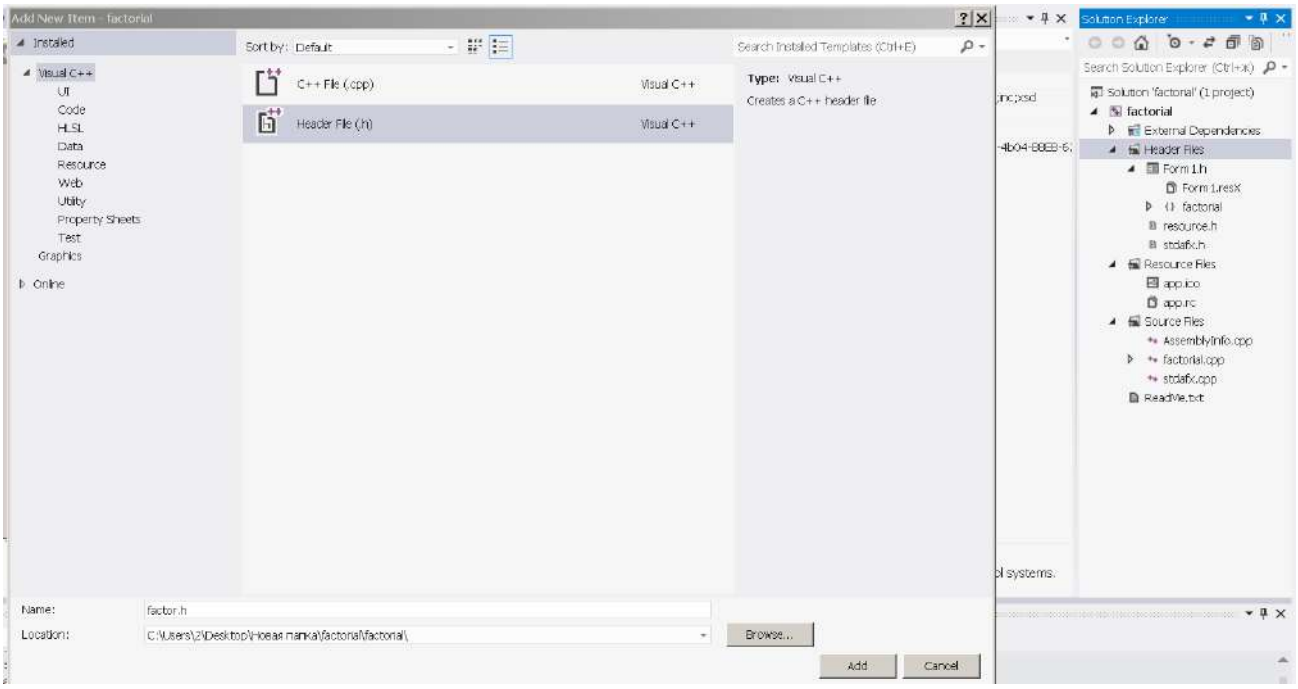


4 chi qadam. Bo‘lajak ilovamizni dizayni shakli shakllandi va uni ishlatib ko‘rsak ham bo‘ladi. Buning uchun oldin kompilyatsiyaga (**Build → Build Solition** yoki **F7)** berish kerak/

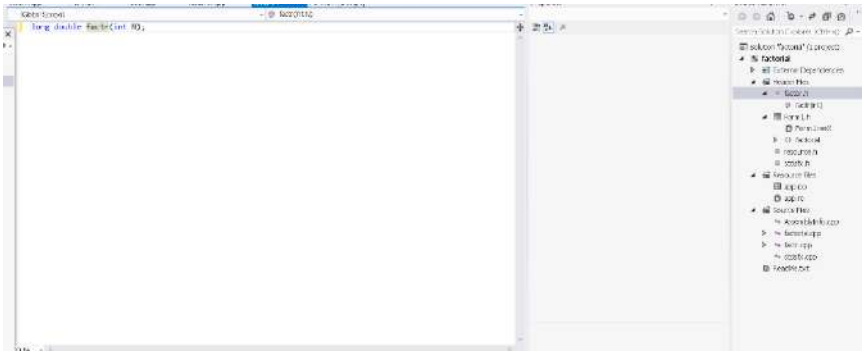
Kompilyatsiyadan yaxshi o‘tsa, **Debug → Star Debugging (**yoki **F5)** buyruqlar yordamida ishga tushiramiz.



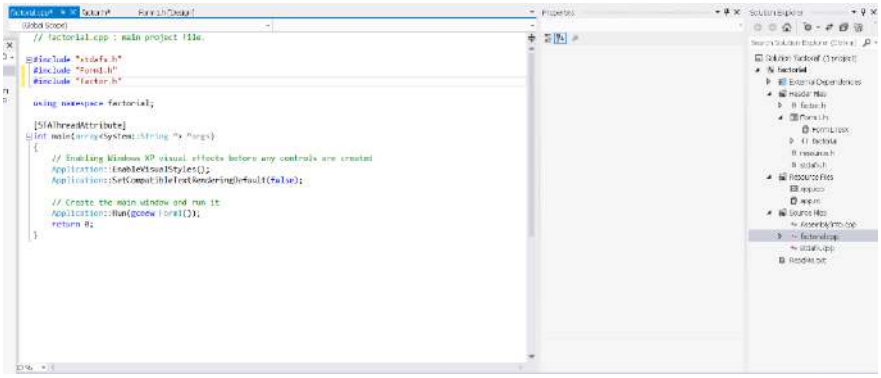
4 chi qadam. Proektga “factor.h” sarlavha faylini qo‘shamiz,



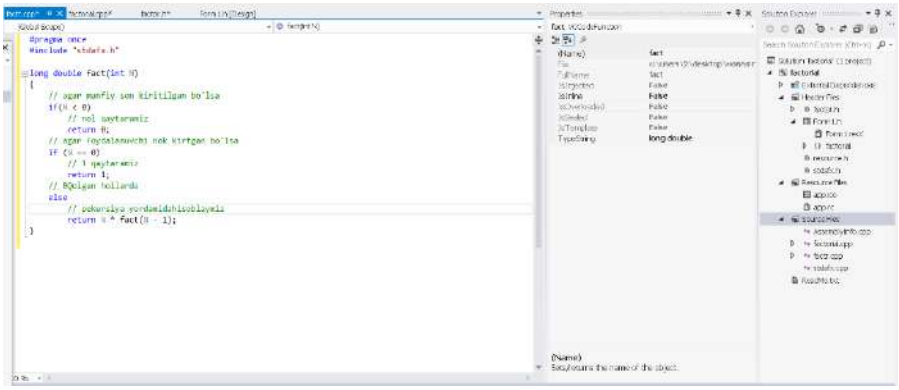
Va ichiga factorial hisoblab beruvchi rekursiv funksiyamizning (factr) prototipini yozamiz.



Asosiy funksiyamiz factorial.cpp ni ochib, boshiga yaratgan sarlavha faylimizni include direktivasi bilan chaqirib olamiz/



Huddi shuday ketma-ketlikda proektga factr funktsiyasini, yani faktr.cpp faylini qo‘shamiz va ichiga kodini yozamiz.



5 chi qadam. **Button** tugmachasini **(Hisob)** bosganda qanday amallar bajarish ketma-ketligini yozamiz. Buning uchun tugmachani 2 marta bosamiz va bizga ajratilgan joyga kodni kiritishni boshlaymiz.

```
rm1.h*  stdafx.cpp  factr.cpp  factorial.cpp  factor.h  Form1.h [Design]*
factorial::Form1                    button1_Click(System::Object ^ sender, System::EventArgs ^ e)
            this->Controls->Add(this->label2);
            this->Controls->Add(this->label1);
            this->Controls->Add(this->button1);
            this->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 10.2F, System::Drawing::Fon
                static_cast<System::Byte>(204)));
            this->ForeColor = System::Drawing::SystemColors::GrayText;
            this->Margin = System::Windows::Forms::Padding(4, 4, 4, 4);
            this->Name = L"Form1";
            this->Text = L"Fastorial hisobi";
            this->ResumeLayout(false);
            this->PerformLayout();

        }
#pragma endregion
    private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {

             }
};
}
```

Formadagi text1Box1 ga kiritilgan son belgi sifatida qabul qilinadi. Shuning uchun uni oldin son turiga o‘girib olishimiz kerak bo‘ladi.

int son=System::Convert::ToDouble(textBox1->Text);

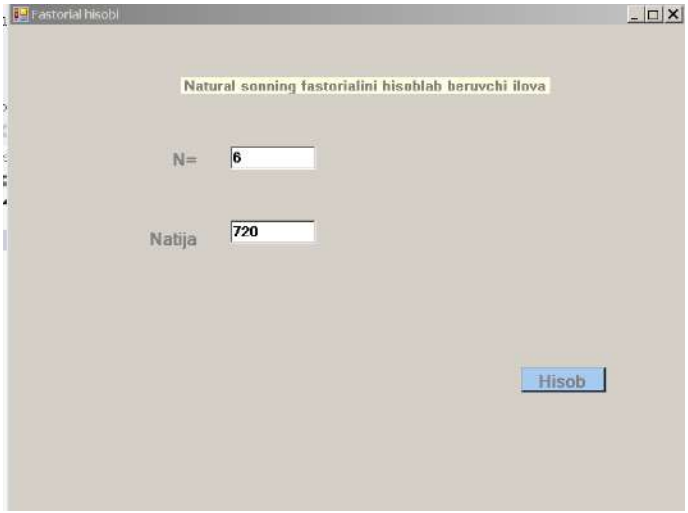
Endi factorial hisoblab beruvchi funksiyamizni chaqirib olsak bo‘ladi

double factor = factr(son);

va olingan natijani textBox2 ga chiqarish uchun qaytatdan belgi toifasiga o‘girishimiz kerak boladi,

textBox2->Text = System::Convert::ToString(factor);

Shu bilan 1-chi ilovamiz tayyor bo‘ldi. Kompilyatsiyaga beramiz, undan keyin ishga tushiramiz va natijani olamiz/



NAZORAT SAVOLLARI

1. Integrallashgan ishlab chiqarish muhitlarining qayday turlarini bilasiz?
2. Qachondan boshlab integrallashgan muhitlar jadal rivojlangan?
3. Turbo muhitlarni sanab bering.
4. Ob'ektga yo'naltirilgan tilni birinchi bo'lib integrallashgan muhitlarga kim va qaysi firma joriy qilgan.
5. Qanday muhitda zamonaviy matn muharrirlari kodi avtomatik bajarilishini ta'minlash (kodi tugatish), muharriri muhitda joriy yozilgan kodi sintaktik to'g'ri va uning davomi bo'lishi mumkin.
6. Taniqli NetBeans integrallashgan muhitini dastlab Java dasturlash uchun qaysi universitet talabasi loyihasi sifatida yaratgan.
7. Visual Studio 97 haqida nimalarni bilasiz?
8. Visual Studio 2012 haqida nimalarni bilasiz?
9. ISO xalqaro standartlari bilan standartlashtirilgan va bir necha implementations ega bo'lgan ob'ektga yo'naltirilgan ko'p tilli dasturlash platformani ayting?
10. MS Visual Studio muhit oynasi nechta qismdan iborat?
12. Windows ilovalarni yaratish uchun qaysi tugmachalar majmuasini bosish yetarli.
13. Qaysi vosita bir necha qo'llab-quvvatlanadigan tillarda har qanday yozilgan dasturlar ijrosini boshqaradi.
14. CLR Empty Project qanday loyiha yaratish uchun kerak.
15. Windows Forms Control Library qanday loyiha yaratish uchun kerak.
16. Loyihaning External Dependencis papkasida qanday fayllar saqlanadi.
17. [FILE] menyusining vazifalarini sanab bering.
18. [TOOLS] menyusining vazifalarini sanab bering.
19. Menyular va uskunalar paneli foydalanuvchilar qanday papka buyruqlarga KIRISH uchun bir yo'ldir.
20. Bir buyruq yaratishda, buning uchun bir avval nima yaratish kerak.
21. Yangi menyular va uskunalar majmuasi aniqlash uchun, Visual Studioda qanday jadval fayl orqali ularni tasvirlash kerak.
22. Qatorlarga izoh qo'yish tugmasini ayting.
23. Satrga izoh qo'shish tugmasini ayting.
24. Tegishli funksiya variantlarini chaqirish tugmasini ayting.
25. [Ctrl+Shift +.] tugmachalar majmuasi qanday amal bajaradi.
26. [Ctrl+Shift+Tab] tugmachalar majmuasi qanday amal bajaradi.
27. Oyna uchun yangi form1 sinfi yaratiladi. Bu sinf qaysi sinfning merosxo'ri hisoblanadi
28. Kontekst menyu orqali formaning qanday funksiyalarini bajarish mumkin.
29. Formaning rejimlarini o'rnatish qayerda joylashgan.
30. Ichki va tashqi ma'lumotlar bilan ishlash hodisalari qayerda joylashgan.
31. MessageBox sinfining nechta argumentlari bor, va har biriga izoh bering.

Amaliyot mashg'ulotini bajarish bo'yicha variantlar

Mavzu: Visual C++ dasturlash muhitida Windows ilovalar yaratish, hodisalar va xossalar bilan ishlash

Ishni bajarishdan maqsad: Windows ilovalar yaratishni, hodisalar va xossalar bilan ishlashni o'rganish.

1 – shi topshiriq. 2 -chi jadval 2 –chi ustunda keltirilgan maydonlarga ega bo‘lgan sinf tuzing. Tashkil qilingan sinfingiz quyidagi vazifalarni bajaruvchi metodlardan iborat bo‘lsin:

- konstruktor;
- ob`yekt to‘g‘risida ma`lumotni ekranga chiqarish;
- individual topshiriqda ko‘rsatilgan vazifani bajaruvchi metod (funksiya).

Ma`lumotlarni kiritish, qayta ishlash va natijani chiqarish ushun textBox, Label, Button va Panel komponentalaridan foydalaning.

1- chi jadval. Individual topshiriqlar variantlari.

№	**Sinf (supper) va uning maydonlari**		**Funksiya-metod** **Sinf(meros) maydon ma`lumotlarini qayta ishlash**
1	Do‘stlar ro‘yxati: *F.I.O., adresi (shahar yoki viloyat), telefoni, tug‘ilgan sanasi*		Navbatdagi tug‘ilgan kungacha necha kun bor
2	Xodim: *F. I.O., bo‘lim, lavozimi, ishga kirgan sanasi*		Korxonada ishlagan vaqti(yil)
3	Dori :*nomi, ishlab chiqilgan sanasi, firma nomi, davlat*		Dorining ishlab chiqarilganiga qancha bo‘ldi (kun)
4	Poezdlar harakatining jadvali: *poezd nomeri, yo‘nalishi, umumiy joylar soni, kupedagi joylar soni, jo‘nash vaqti*		Poezd jo‘nashiga qancha vaqt qoldi (joriy vaqt kiritiladi)
5	Mobil aloqa abonenti: *familiyasi, ismi, pasport seriyasi va nomeri, telefon nomeri, paket turi*		Joriy vaqt imtiyozli vaqt oralig‘iga kiradimi yo yo‘qmi(0 dan 8 gacha)
6	Dars jadvali: *fan, boshlanish vaqti, auditoriya, guruh*		Joriy vaqtda qaysi fan darsi boshlanadi (yoki ketyapti)
7	Kutubxona: *nomi, avtori (avtorlar), saxifalar soni, nashr etilgan yili, soni*		Kitobning 15%ga arzonlashgan narxi
8	Kutubxona do‘koni*: nomi, avtori, nashr etilgan yili, bo‘limi, narxi*		Kitobning “yoshi” 5 yildan katta bo‘lsa, narxini 20% ga tushiring.

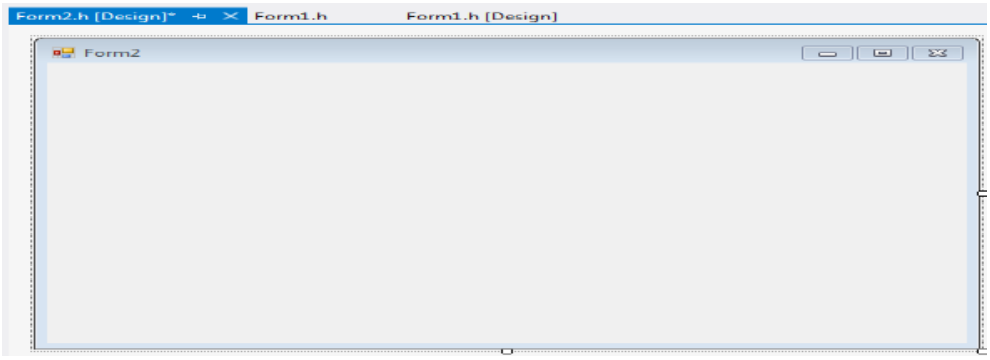
9	Korxona xodimi: *familyasi, ismi, tug‘ilgan yili, oylik maoshi, ishga kirgan yili*		Xodim yoshi 60 yoshdan kishik bo‘lsa, pensiyaga necha yil qolganini, katta bo‘lsa, pensiyada necha yil ishlayotganini aniqlang.
10	Firma xodimi: *familiyasi, ismi, oylik maoshi, tug‘ilgan sanasi, lavozimi.*		Dasturchi lavozimidagi xodimlarning oylik maoshini 20% ga oshirilsin.
11	Vektor va tekislikda nuqta: *vektor koordinatalari (x1,y1,x2,y2) va nuqta koordinatalarii (x3;y3)*		Bir tomoni vektor, qolgan ikki tomoni berilgan nuqtadan o‘tuvchi ushburchak yuzini aniqlang.
12	Ikkita vektor:*birinchi vektor koordinatalari (x1,y1,x2,y2) , ikkinchi_vektor koordinatalari (x1;y1;x3;y3)*		Ikki vektor yig‘indisi vektorining koordinatalarini aniqlang
13	Korxona xodimi: *familiyasi, ismi, lavozimi,oylik maoshi, reyting bali (100 ballik tizimda)*		Reytingi 60 va 75 oralig‘ida bo‘lsa, oylik maoshini 20%ga, 75 va 90 oralig‘ida bo‘lsa, 40%ga, 90 va 100 oralig‘ida bo‘lsa, 60%ga oshirilsin.
14	Firma ofisi: *xona eni, bo‘yi, balandligi (metrda), xonalar soni va koridor maydoni, 1 m^2 ishlatiladigan kraska hajmi*		Ofis devorlarini (koridor va xonalar) bo‘yashga ketatigan kraska hajmi (xar bir xonada bittadan oyna (2x1,5) bor) aniqlang.
15	Firma mahsuloti: *nomi, narxi, ishlab chiqilgan yili, do‘konga kelib tushgan sanasi*		Mahsulot do‘konga kelib tushganiga necha kun bo‘lganini aniqlang
16	Mahsulot: *nomi, narxi (so‘mda), davlat nomi (ishlab chiqqan), ishlab chiqilgan yili, chegirma (protsent hisobida)*		Ishlab chiqilganiga 2 yildan oshgan mahsulotlar narxini chegirma asosida o‘zgartiring.
17	Ikkita sonlar to‘plami: *uchta haqiqiy sonlar x, y, z va uchta haqiqiy sonlar a, b, s*		Ikkita sonlar to‘plamini skalyar ko‘paytmasining qiymatini aniqlang

18	Beshta son: *to‘rtta butun sonlar a, b ,s, d va x soni*		Sonlar kvadratlarining yig‘indisini hisoblang.
19	Izolyatsiyalangan sim: *asos diametri, balandlik, sim og‘irligi (1 sm^3); izolyatsiya qalinligi va uning og‘irligi (1 sm^3).*		Izolyatsiyalangan sim og‘irligini aniqlang
20	Temir (metall) brus: *eni, bo‘yi, balandligi, og‘irligi (1 sm^3)*		Brusning og‘irligini aniqlang.

№17-AMALIY MASHG’ULOT

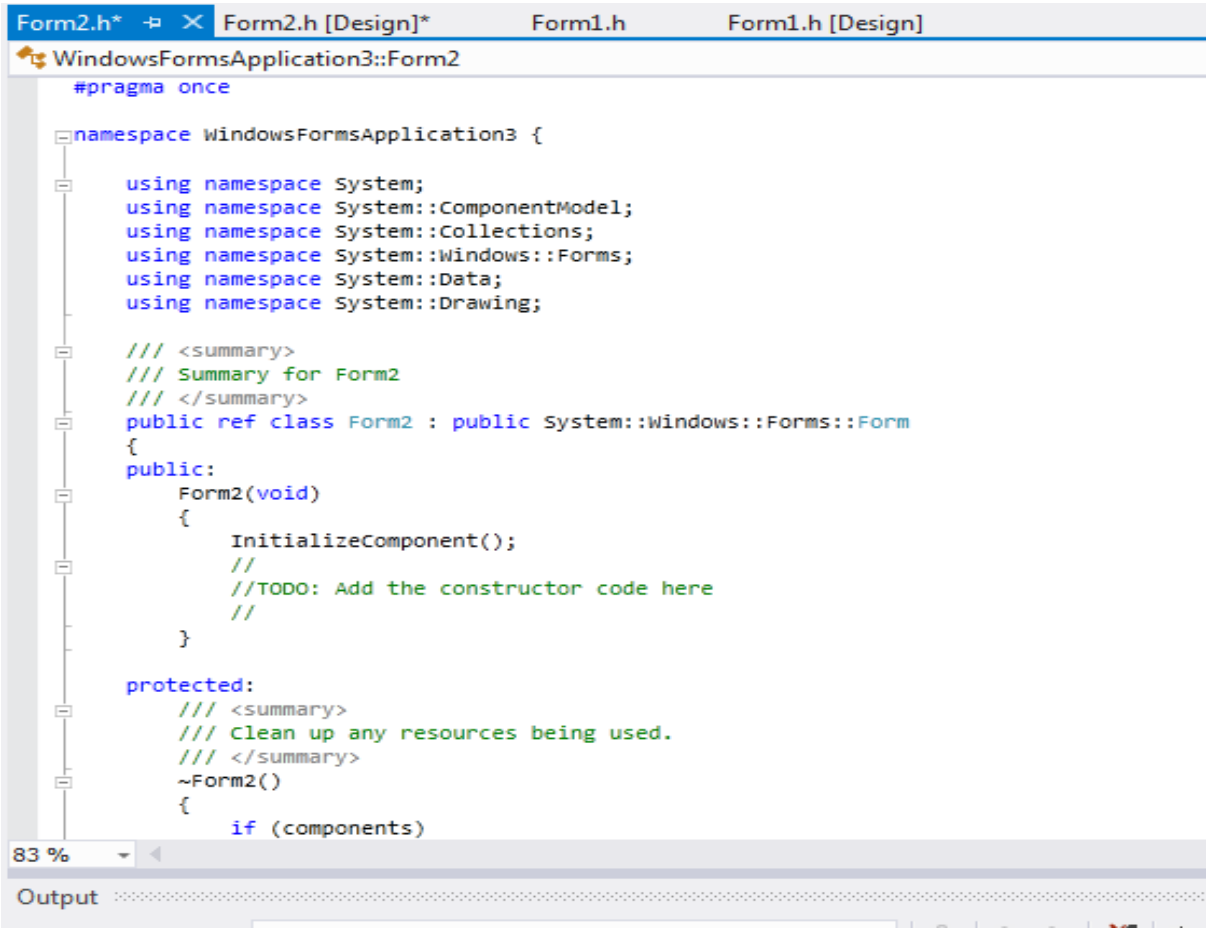
Mavzu: KOMPONENTALAR BILAN ISHLASH. KOMPONENTA TUSHUNCHASI VA XUSUSIYATLARI

Ishning maqsadi: Ekran markazida forma dizayneri joylashgan (1.1-rasm). Bu bo‘lg‘usi dasturning interfeysini hosil qiluvchi oynadir. Oynaning nomi va sarlavhasi uning tepasida yozilgan bo‘ladi. Odatda, **Form1.h** va shuningdek oynani berkituvchi va kichraytiruvchi tugmachalari ekranning o‘ng tepasida joylashgan.



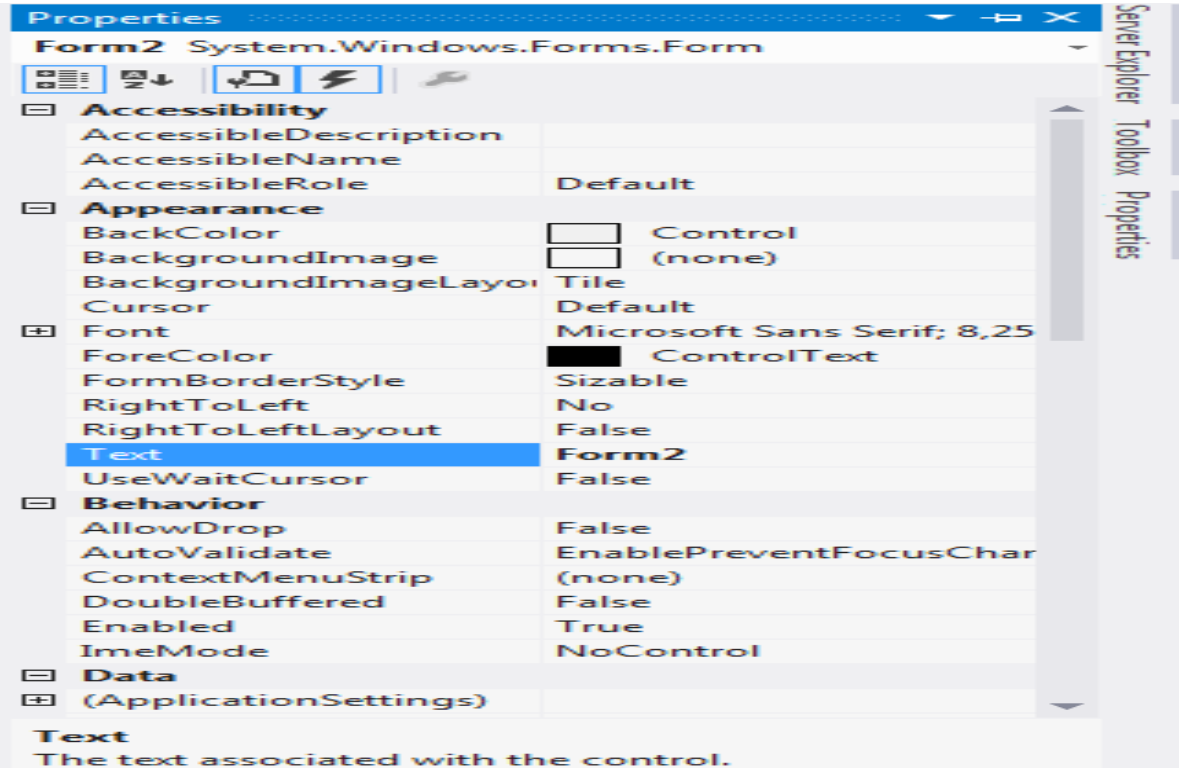
1.1– rasm. Forma oynasi.

Forma dizayneri ostida dastur kodi muharriri joylashgan, odatda u **Form1.h** sarlavha bilan beriladi. Bu oyna dastur kodini kiritish va tahrir qilish uchun mo‘ljallangan (1.2-rasm).



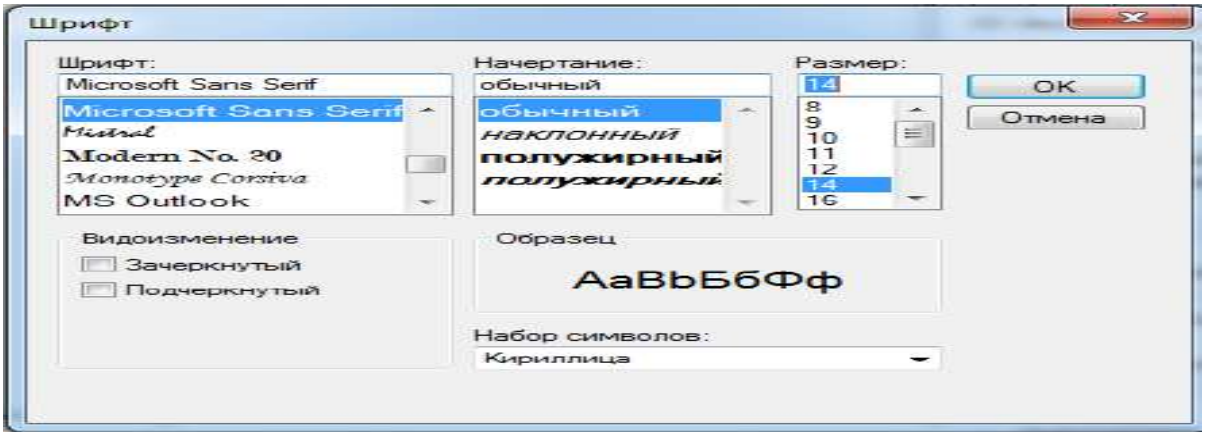
1.2 – rasm. Kodlar oynasi.

1.3 – rasmda forma va unda joylashgan komponentalarning xossalarini va hodisalarini o‘zgartiradigan oynani ko‘rish mumkin.



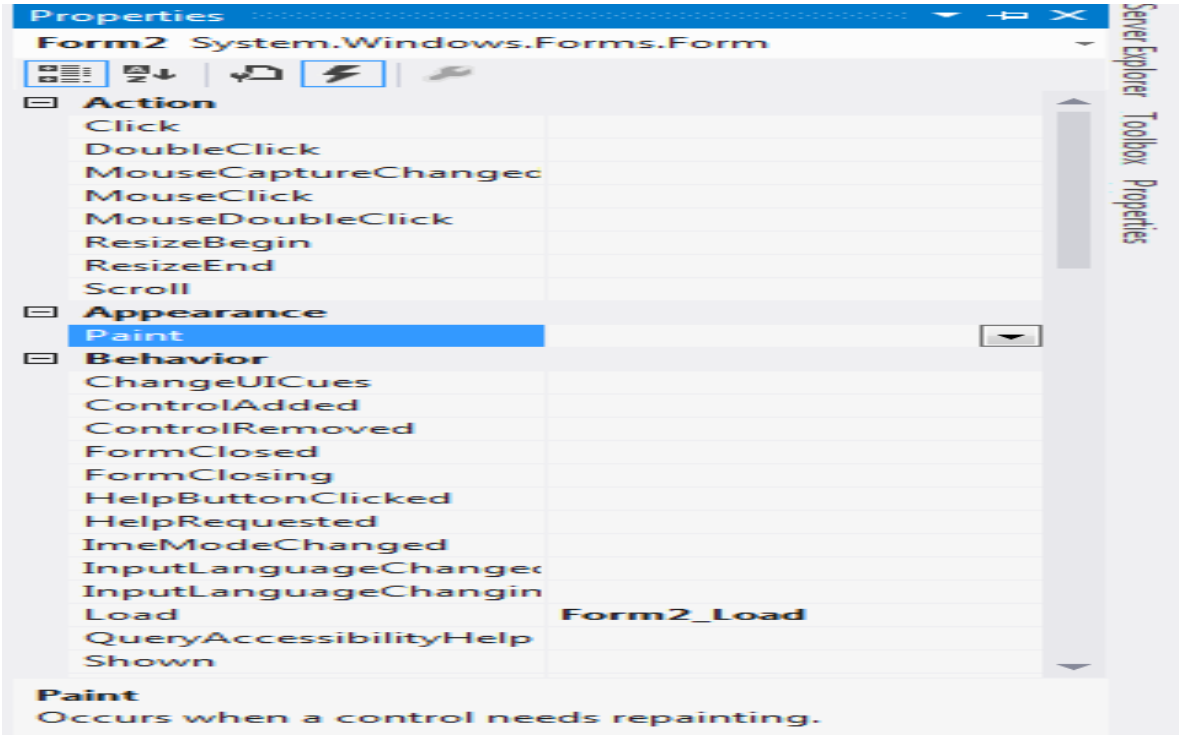
1.3– rasm. Xususiyatlar **“Properties”** va hodisalar **“Events”** oynasi.

Ushbu oyna orqali komponentalarning tashqi ko‘rinishlari va barcha xususiyatlari sozlanadi. Misol uchun komponentaning o‘lchamini o‘zgartirish kerak bo‘lsa, **Font** qismidagi 3 ta nuqtali tugma bosiladi va komponenta o‘lchamlari o‘rnatiladi. U 1.4 – rasmda keltirilgan:



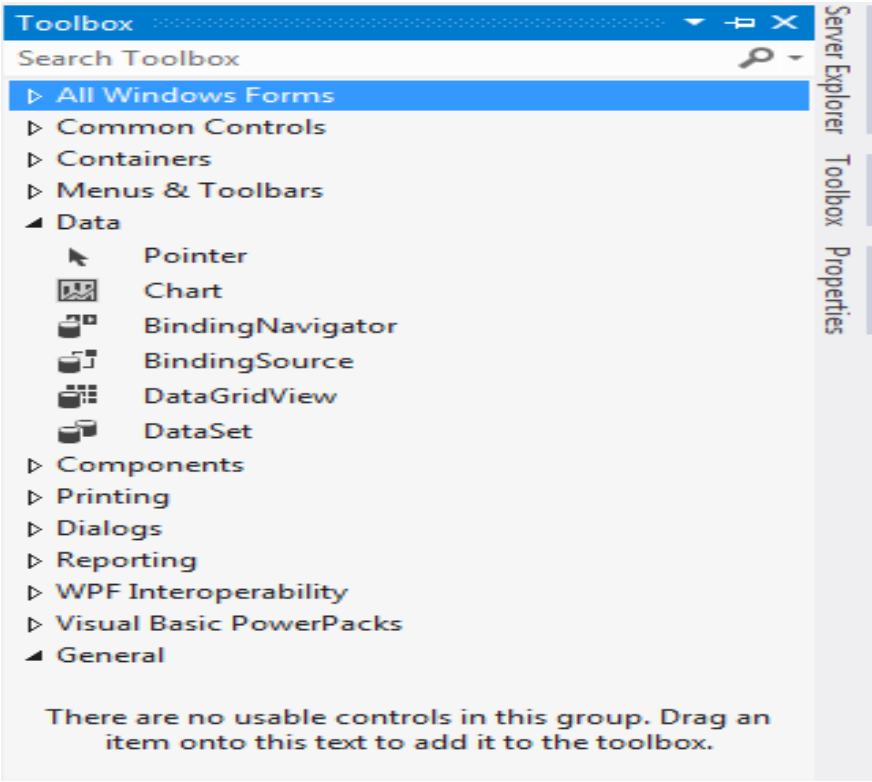
1.4– rasm. Komponenta shiriftlarini sozlash oynasi.

1.5– rasmda. Komponenta hodisalarini ko‘rishingiz mumkin. Misol: Formani yopganda hodisasi “**Form closed**” qismida sichqonchaning o‘ng tugmasi 2 marta tez bosilsa, kodlar oynasida ushbu hodisa ro‘y bergan holdagi jarayonlarning kodini, kodlar oynasida kiritiladi.



1.5– rasm. Hodisalarni sozlash oynasi.

Visual C++ muhitini VCL vizual komponentalari bilan tanishamiz. **Visual C++** muhitida ilova yaratish **VCL** vizual komponentalaridan keraklilarini formaga joylashtirishdan boshlanadi. Komponentalar ilovaning g‘ishtchalari (elementlari, qismlari) deyishimiz mumkin. Barcha **VCL** komponentalari, **ToolBox** oynasida palitralarga ajratilgan holda joylashgan bo‘ladi. Komponentalar palitrasi ma’lum bir vazifalariga ko‘ra komponentalar to‘plamidan iborat komponentalar sahifasiga ajratilgan. Bu sahifalar ma’no va vazifasiga ko‘ra bir - biriga yaqin bo‘lgan komponentalarni bitta guruhga birlashtiradi. 1.6-rasmda komponentalar palitrasi va komponentalar joylashtirilgan oyna keltirilgan.



1.6-rasm. Komponentalar palitrasi va komponentalar oynasi.

Visual C++ 32 razryadli takomillashtirilgan vizual komponentalar kutubxonasi **VCL** (**Visual Component Library**) bilan birgalikda etkazib beriladi. Bu kutubxona eng murakkab ilovalarni qurish uchun mo‘ljallangan, 100 dan ortiq takroran qo‘llanadigan komponentalardan iborat. Kutubxonaning asosiy komponentalari - Palitra komponentalarining instrumental panelida berilgan. Komponentalar belgilari dastur formasiga olib o‘tiladi.

Kutubxona **Windows** operastion tizimlaridagi foydalanuvchi grafik interfeysi, standart interfeys ob’ektlarining to‘liq inkapsulyastiyalanishini o‘z ichiga oladi. Ular orasida, ixtisoslashgan komponentalar bilan bir qatorda, relyastion ma’lumotlar bazasini boshqarish uchun mo‘ljallangan komponentalar alohida o‘rin egallaydi. Ishonchli va samarali dasturlarni yaratishda **Visual C++** OYD imkoniyatlaridan to‘liq foydalanadi. **Visual C++** bu OMD ekan, OLE (OCX) boshqaruvchi elementlarni kiritish uncha qiyinchilik tug‘dirmaydi. O‘z masalalaringiz talablarini kerakli darajada qondirish uchun, Kutubxonaning mavjud komponentalaridan foydalaning va hosila komponentalar imkoniyatlarini kengaytiring.

Visual C++ bosh xususiyati avvalambor, uning dasturini vizual ishlash jarayonida nafaqat tayyor komponentalardan foydalanish, balki yangi komponentalarni yaratish qobiliyatida ham namoyon bo‘ladi. Yangi komponentalar, dastlabki komponentalar kabi, sodda bo‘lishi mumkin, bunda

ularning funkstional imkoniyatlari ozgina kengaytirilgan yoki o‘zining mutlaqo o‘ziga xos ko‘rinishi, xulq-atvori va kodining mazmuni bilan farqlanadigan bo‘ladi. Komponentalarning yaratilishi OYD ning vorislik mexanizmiga tayanadi, cheklanishlarga deyarli ega bo‘lmaydi hamda qo‘yidagi bosqichlardan o‘tadi:

- mavjud komponenta turiga vorislik;
- yangi xususiyatlar, metodlar va voqealarni aniqlash;
- yaratilgan komponentani qayd etish.

Qidirish oson bo‘lishi uchun, palitra funkstional jihatdan o‘xshash komponentalarni birlashtiradigan qo‘shimcha ilovalar bilan bo‘lingan. Tanlab olingan komponentaning kontekst menyusini unga sichqonchaning o‘ng tugmasini bosib ochish mumkin.

Bular ichidan **OK** yozuvli **Button** tugmasini, **A** yozuvli **Label** tugmasini osonlik bilan topish mumkin.

Agar sahifada komponentalar juda ko‘p bo‘lsa, harakatlanuvchi tugmalar orqali o‘ng va chapga panelni harakatlantirish orqali barcha komponentalarni ko‘rishimiz mumkin. Barcha komponentalar biror nom yoki maxsus yorliq orqali berilgan, agar sichqoncha kursorini yorliq ustiga oborsak shu komponenta nomi paydo bo‘ladi. E’tibor bergan bo‘lsangiz, ko‘pgina komponentalar to‘rtburchak yoki dumaloq shaklga ega. Ixtiyoriy komponentani formaga ikki xil yo‘l bilan joylashtirish mumkin. Komponenta ustida sichqonchani olib borib chap tugmasini ikki marta bosilsa, mazkur komponenta forma markazida paydo bo‘ladi. Ikkinchi usul sichqoncha chap tugmasini komponenta ustida bir marta bosilib, siljitish bilan formaga joylashtirish mumkin. Formaga joylashtirilgan komponentani formaning ixtiyoriy qismiga siljitishimiz mumkin. Komponetaning o‘lchamlarini sichqoncha yordamida o‘zgartirish mumkin. Formada turgan ixtiyoriy komponentani aktivlashtirish yoki passivlashtirish mumkin. Aktivlashtirish degani shu komponentaning ustida amal bajarish xususiyatini o‘zgartirish imkoniyatini beradi, buning uchun, mazkur komponentaning ustida sichqoncha tugmasini bosish kifoya. Aktivlik belgisi shu komponentani boshqalaridan ajralib, belgilanib ko‘rinib turadi. Buni ob’ektlar inspektoridan ham ko‘rish mumkin. Sichqoncha yordamida tanlangan komponentani formadan o‘chirib tashlash ham mumkin.

Label

Shaklda tahrir qilib bo‘lmaydigan statik matnning to‘rtburchak sohasini aks ettiradi. Odatda matn boshqa komponenta nomidan iborat bo‘ladi.

Nom matni Text xususiyatining qiymatidir. **Alignment** xususiyati matnni tekislash usulini aniqlaydi. Shrift o‘lchami avtomatik tarzda sohaning maksimal to‘ldirilishiga mos kelishi uchun, **AutoSize** xususiyatining **true** qiymatini o‘rnating. Kalta soha ichida matnning hammasini ko‘rish imkoniga ega bo‘lish uchun, **WordWrap** xususiyatiga **true** qiymatini bering. **Transparent** xususiyatiga **true** qiymatini o‘rnatsangiz, boshqa komponentaning bir qismini to‘g‘ri uning ustida joylashtirilgan nom orasidan ko‘rinib turadigan qilishingiz mumkin.

TextBox

Axborot yakka satrining tahrir qilinayotgan kiritishidagi to‘rtburchak sohani shaklda aks ettiradi. Tahrir sohasining ichidagi boshlang‘ich narsalarni Text xususiyatining qiymati bo‘lgan satr aniqlaydi.

TextBox komponentasi **TtextBox** sinfining to‘g‘ridan-to‘g‘ri hosilasi bo‘lib, uning barcha xususiyatlari, metodlari va voqealariga vorislik qiladi.

Button

Yozuvli to‘rtburchak tugmani yaratadi. Tugmacha bosilganda, dasturda biror-bir hatti-harakat nomlanadi (initstiallashtiriladi).

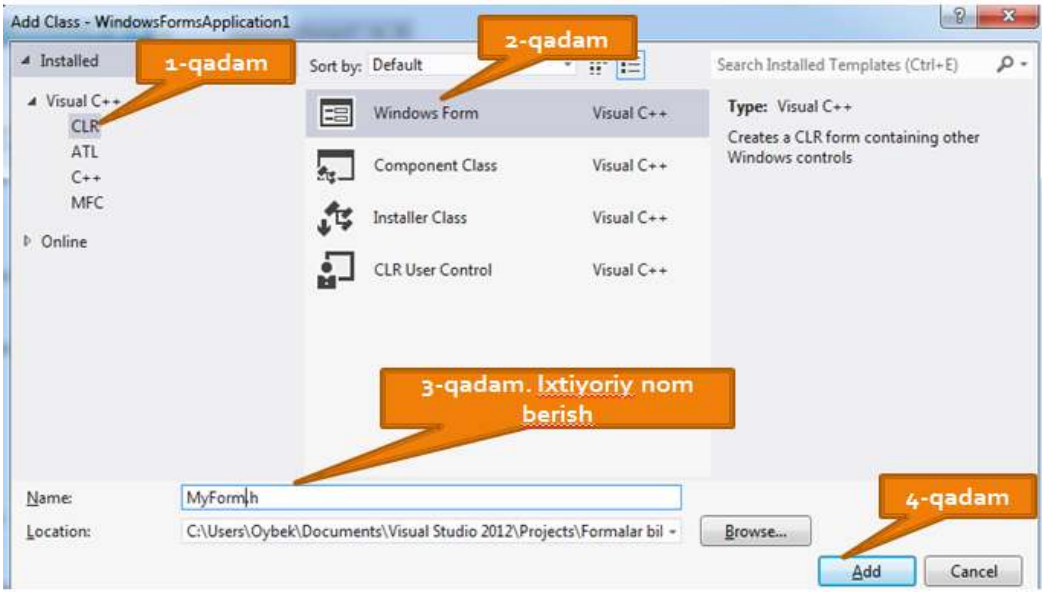
Tugmachalar ko‘proq dialogli darchalarda qo‘llanadi. **Default** xususiyatining **true** qiymati tomonidan tanlab olingan yashirin tugmacha, dialog darchasida har gal enter klavishasi bosilganda **OnClick** voqea qayta ishlatgichini ishga tushiradi. **Cancel** xususiyatining **true** qiymati tanlab olgan bekor qilish tugmachasi, dialog darchasida har gal **Escape** klavishasi bosilganda, **OnClick** voqea qayta ishlatgichini ishga tushiradi.

1- jadvaldagi topshiriqlarni bajarish uchun uslubiy ko‘rsatmalar.

Visual C++ dasturining Windows Application muhitida ishlash:

Masalaning qo‘yilishi: Formada a[N][M] massiv elementlarini [-1000; 1000] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. **Massivni dataGridWiew** komponentasiga joylashtiring. **Massiv elementlaridan qiymati juft bo‘lganlarining yig‘indisini label** komponentasida chiqaring.

1-qadam. Yangi forma ochiladi.



1.7 – rasm. Yangi forma ochish oynasi.

2-qadam. Formaga 2 ta **TextBox** komponetalari (misoldagi N va M uchun) joylashtiriladi;

3-qadam. Formaga 4 ta **label** komponetalari (misol berilishi, N , M lar va natija uchun) joylashtiriladi va **label** Text xususiyatlari mos ravishda o‘zgartirib chiqiladi;

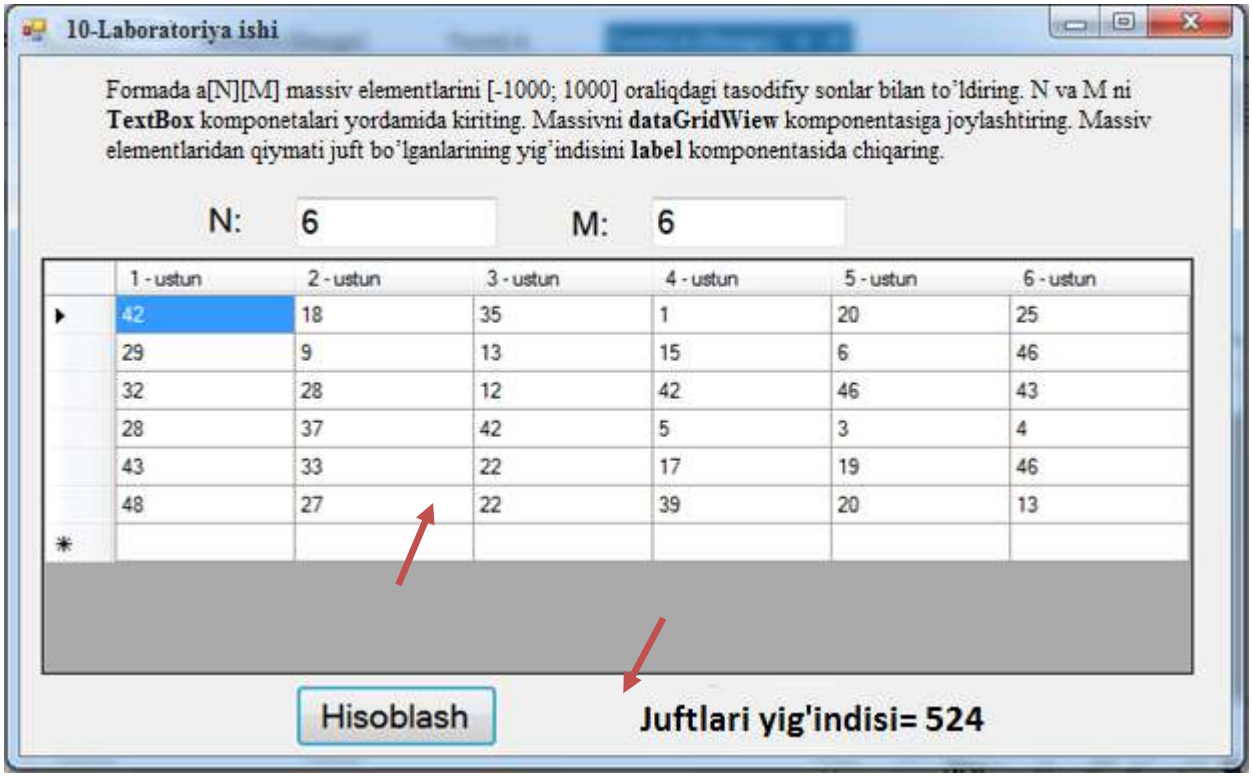
4-qadam. Formaga 1 ta **dataGridWiew** komponetasi (misoldagi N va M o‘lchovli massiv elementlarini chiqarish uchun) joylashtiriladi;

5-qadam. Formaga **button** tugmachasi joylashtiriladi va nomi o‘zgartiriladi. **button1** tugmachasini 2 marta bosib, kodlar oynasida quyidagi kodlar teriladi:

```
#include "iostream"
#include "conio.h"
#include "stdlib.h"
#include "form2.h"
        int g,s;
#pragma endregion
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
```

```
                int a[10][10];
                 s=0;
                 int n=Convert::ToInt32(textBox1->Text);
                 int m=Convert::ToInt32(textBox2->Text);
                 for(int i=0;i<n;i++){
            if(g==0){dataGridView1->Rows->Add();}
            for(int j=0;j<m;j++){
 a[i][j]=rand()%2001-1000;
 dataGridView1->Columns[j]- >HeaderText=(j+1).ToString()+" - ustun";
dataGridView1->Rows[i]->Cells[j]->Value =a[i][j].ToString();
              if(a[i][j]%2==0){s+=a[i][j];}
               }}
               g+=1;
label4->Text="Juftlari yig‘indisi= " + s.ToString();
}
```

Dastur natijasi quyidagi ko‘rinishda bo‘ladi:



1.8 – rasm. Dastur oynasi.

Nazorat savollari

1. Visual C ++ da komponenta nima ?
2. Qanday standart komponentalar mavjud?
3. Qanday qo'shimcha komponentalar mavjud?
4. Tizimning qanday komponentalari mavjud?
5. Button komponentasi nimaga mo'ljallangan ?
6. Button komponentasi xususiyatlarini aytib bering.
7. Button komponentasi hodisalaridan qaysilarini bilasiz.
8. Button komponentasi metodlari qanday qo‘llaniladi?
9. Label komponentasi nima uchun kerak ?
10. Label komponentasi xususiyatlarini aytib bering.
11. Label komponentasi hodisalaridan qaysilarini bilasiz.
12. Label komponentasi metodlari qanday qo‘llaniladi?
13. TextBox komponentasi nimaga mo'ljallangan ?
14. TextBox komponentasi xususiyatlarini aytib bering.
15. TextBox komponentasi hodisalaridan qaysilarini bilasiz.
16. TextBox komponentasi metodlari qanday qo‘llaniladi?
17. ListBox komponentasi nimaga mo'ljallangan ?
18. ListBox komponentasi xususiyatlarini aytib bering.
19. ListBox komponentasi hodisalaridan qaysilarini bilasiz.
20. ListBox komponentasi metodlari qanday qo‘llaniladi?
21. ComboBox komponentasi nimaga mo'ljallangan ?
22. ComboBox komponentasi xususiyatlarini aytib bering.
23. ComboBox komponentasi hodisalaridan qaysilarini bilasiz.
24. ComboBox komponentasi metodlari qanday qo‘llaniladi?
25. CheckBox va RadioButton lar nima uchun ishlatiladi?
26. CheckBox va RadioButton farqini tushuntiring.
27. CheckBox komponentasi xususiyatlarini aytib bering.
28. ImageList komponentasi nima uchun ishlatiladi.
29. ImageList komponentasi xususiyatlarini aytib bering.
30. CheckedListBox komponentasi nimaga mo'ljallangan ?

AMALIYOT MASHG‘ULOTINI BAJARISH BO‘YICHA VARIANTLAR

№17-AMALIYOT MASHG‘ULOTI

TOPSHIRIQ.

Ma’ruza va amaliyot darslarida olingan ma’lumotlarga asoslanib, o‘z variantingiz uchun quyidagilarni bajarinig.

a) 1 jadvaldan o‘z variantingizni ko‘chirib oling;

b) Amaliyot ishini bajarish uchun uslubiy ko‘rsatma asosida o‘z variantingizdagi masalalarni bajaring.

1-jadval

№	Variantlar
1	Formada [N][M] massiv elementlarini [-1000; 1000] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Bu massivning har bir satr elementlarini o‘sish tartibida label komponentasida tartiblangan holda chiqaring.
2	Formada [N][M] massiv elementlarini [-100; 100] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Uning eng katta va eng kichik elementini toping va label komponentasida chiqaring.
3	Formada [N][M] massiv elementlarini [-500; 500] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Uning eng katta elementining nomer(indeks)ini label komponentasida chiqaring .
4	Formada a[N][M] massiv elementlarini [-500; 1000] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Berilgan massiv elementlarining a va b oraliqdagilari sonini label komponentasida chiqaring. a va b ni TextBox komponetasi yordamida kiriting.
5	Formada a[N][M] massiv elementlarini [-300; 300] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Bu massivni ustun elementlarini o‘sish tartibida label komponentasida chiqaring.
6	Formada a[N][M] massiv elementlarini [-1000; 1000] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Bu massivni bosh diagonal elementlarini o‘sish tartibida label komponentasida chiqaring.
7	Formada a[N][M] massiv elementlarini [-3000; 3000] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew

	komponentasiga joylashtiring. Elementini qiymati 200 dan kattalari sonini label komponentasida chiqaring.
8	Formada a[N][M] massiv elementlarini [125; 1000] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Bu massivning bosh diagonal elementlarini kamayish tartibida label komponentasida chiqaring.
9	Formada a[N][M] massiv elementlarini [50; 150] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Elementining qiymatini bir vaqtda 5 ga va 3 ga qoldiqsiz bo‘linadiganlarining soni label komponentasida chiqaring.
10	Formada a[N][M] massiv elementlarini [-300; 300] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Massivni chap diagonal elementlarini kamayish tartibida 2-formaning label komponentasida chiqaring.
11	Formada a[N][M] massiv elementlarini [100; 1000] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Massiv elementlari tublari sonini label komponentasida chiqaring.
12	Formada a[N][M] massiv elementlarini [-1000; 500] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Massivni manfiy elementlarini ularning absolyut qiymati bilan almashtiring va almashtirishlar sonini label komponentasida chiqaring.
13	Formada a[N][M] massiv elementlarini [-200; 200] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. label komponentasida massivning eng oxirgi manfiy elementining nomerini aniqlang va chiqaring.
14	Formada a[N][M] massiv elementlarini [-60; 60] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Massivning eng oxirgi musbat elementining nomerini label komponentasida chiqaring.
15	Formada 1-formada a[N][M] massiv elementlarini [-100; 200] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Massiv elementlaridan qiymati toq bo‘lganlarining yig‘indisini label komponentasida chiqaring.
16	Formada a[N][M] massiv elementlarini [-1000; 100] oraliqdagi tasodifiy sonlar bilan to‘ldiring. N va M ni TextBox komponetalari yordamida kiriting. Massivni dataGridWiew komponentasiga joylashtiring. Massivning musbat elementlarining yig‘indisini label komponentasida chiqaring.

№18-AMALIY MASHG'ULOT

MAVZU: STANDART, ADDITIONAL, SYSTEM KOMPONENTALARI

Ishning maqsadi

Standart komponentasining xususiyatlari va hodisalari. Standart komponentalarga Toolbox oynasidagi [Common Controls] Tab dagi komponentalar kiradi. Ularni tarkibini o‘zgartirish mumkin, ya‘ni yangilarini qo‘shish va keraksizlarini o‘chirib tashlash mumkin. Shuningdek, agar maxsus ishlab chiqilgan yangi komponentalar guruhi bo‘lsa ulardan ham foydalanish

mumkin. Standart komponentalarni quyidagi guruhlarga bo‘linadi va shu asosida ularning hususiyatlari va hodisalari 95% bir xil bo‘ladi.

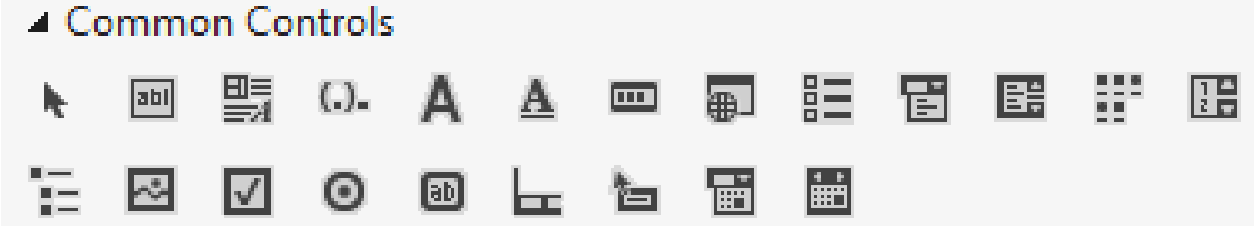
1.1-jadval. Standart komponentalarni guruhlari va vazifalari.

Komponentaning guruhi	**Komponentaning nomi**	**Vazifasi va izoh**
Mantlarni tahrirlash	TextBox	Matnni tahrirlash imkonini beradi, dastur ishlayotgan vaqtda foydalanuvchi yoki algoritm yordamida matnni tahrirlash.
	RichTextBox	Oddiy va RTF formatida Matnni tahrirlash imkonini beradi, dastur ishlayotgan vaqtda foydalanuvchi yoki algoritm yordamida matnni tahrirlash.
	MaskedTextBox	Foydalanuvchi tomonidan kiritilayotgan matnlarni maxsus belgi bilan himoyalaydi
Ma‘lumotlarni ko‘rsatish (faqat o‘qish uchun)	Label	Foydalanuvchi tomonidan to‘g‘ridan- to‘g‘ri tahrir qilinmaydi matnni ko‘rsatadi.
	LinkLabel	Veb-link sifatida matnni ko‘rsatadi va foydalanuvchi joriy matnni bosganda boshqa oynaga yoki veb-saytga havola faollashtiradi.
	ProgressBar	Foydalanuvchi uchun joriy amal bajarilish jarayonini ko‘rsatadi.
Veb-sahifani ko‘rsatish	WebBrowser	Foydalanuvchi oynasida veb-sahifalar orqali harakat qilish imkonini beradi.

Ro‘yxatdan tanlash	CheckedListBox	Har biri katak bilan birga bo‘lgan elementlarning takrorlanadigan ro‘yxatini ko‘rsatadi.
	ComboBox	Qalqib chiquvchi elementlar ro‘yxatini
	ListBox	Matnlar va grafik elementlar ro‘yxatini ko‘rsatadi
	ListView	Elementlarni to‘rt xil ko‘rinishdan birida ko‘rsatadi. Ko‘rinishlarga faqat matn, kichik piktogrammali matn, katta piktogrammali matn va batafsil ko‘rinish kiradi.
	NumericUpDown	Foydalanuvchilar yuqoriga va pastga tugmalari yordamida o‘tiish mumkin bo‘lgan raqamlar ro‘yxatini ko‘rsatadi.
	TreeView	Qo‘shimcha kataklar yoki piktogrammalar bilan matndan iborat bo‘lishi mumkin bo‘lgan tugun ob‘yektlarining iyerarxik to‘plamini ko‘rsatadi.
Grafiklarni tasvirlash	PictureBox	bitmaps va piktogramma kabi tasvir fayllarini ko‘rsatadi.
parametrli qiymat	CheckBox	Matn uchun katakcha va yorliqni ko‘rsatadi. Odatda parametrlarni belgilash uchun ishlatiladi.
	CheckedListBox	Har biri katak bilan birga bo‘lgan elementlarning kaytariladigan ro‘yxatini ko‘rsatadi.
	RadioButton	Yoqilgan yoki o‘chirilishi mumkin bo‘lgan tugmani ko‘rsatadi.
	TrackBar	Foydalanuvchilarga shkala bo‘ylab "Thumb" ni harakatlantirib shkala qiymatlarini o‘rnatish imkonini beradi.
Buyruqlar	Button	Boshlanadi, to‘xtaydi yoki jarayonni bekor qiladi.
	LinkLabel	Veb-link sifatida matnni ko‘rsatadi va foydalanuvchi joriy matnni bosganda boshqa oynaga yoki veb-saytga havola faollashtiradi.

	NotifyIcon	Fonda ishlaydigan dasturni ifodalovchi vazifalarni paneli holati xabarnomasidagi belgini ko‘rsatadi.
Foydalanuvchi uchun yordam	HelpProvider	Elementlari uchun pop-up yordam oynasini yoki tezkor yordam oynasini beradi.
	ToolTip	Foydalanuvchi nazorat bilan hovers nazorat maqsadi bo‘yicha qisqacha tavsifi ko‘rsatadi va bir pop-up oyna beradi.
Vaqt	DateTimePicker	Foydalanuvchilarga sana yoki vaqtni tanlash imkonini beradigan grafik taqvim ko‘rsatadi.
	MonthCalendar	Foydalanuvchilarga sana qatorini tanlash imkonini beradigan grafik taqvim ko‘rsatadi.

Standart komponentalarni guruhlashning asosiy maqsadi, guruhlarga mos xususiyatlar va hodisalari bir xil bo‘ladi. Bu komponentalarning Toolbox oynasida quyidagicha ko‘rinishga ega (1.2rasmga qarang)

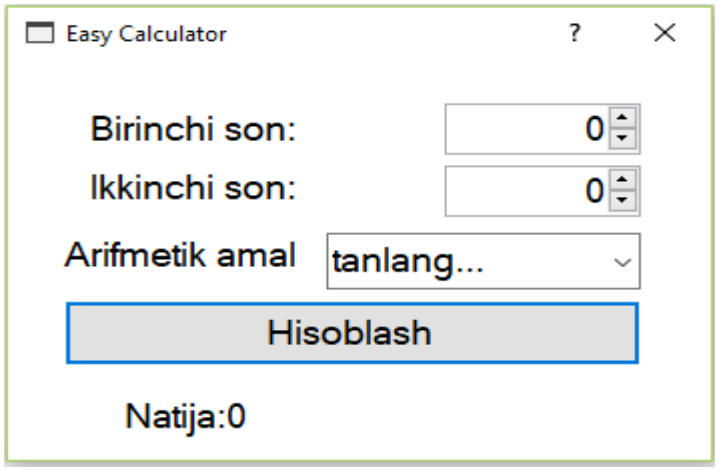


1.2-rasm. Standart komponentalar ko‘rinishi.

Bu komponentalarni oynaga joylashtirish uchun sichqonchaning chap bitta bosiladi va oynaga keltirib chap tomoni yana bir marta bosiladi va oynada tanlangan komponentaning nusxasi paydo bo‘ladi. Bu nusxani ob‘yekt deb aytamiz. Chunki eslab ko‘ring, sinfning nusxasi bu ob‘yekt deb aytiladi. Ob‘yektning ustiga sichqonchaning o‘ng tugmasini bossangiz, u bilan bajarish mumkin bo‘lgan amallarni ko‘rasiz. Agar ob‘yektdan nusxa olsangiz uning super sinfining, ya‘ni shu ob‘yekt tipidagi boshqa bir ob‘yekt yaratiladi. Ikkita bir xil tipdagi ob‘yektlarning xususiyatlari va hodisalariga alohida – alohida ishlov beriladi.

Standart komponentalarni xususiyat va hodisalarini ko‘rib chiqish uchun arifmetik amallarni bajaruvchi dastur yaratish masalasini olamiz.

Mantiqiy jihatdan sonlarni kiritish uchun 2 ta NumericUpDown, matnli yozuvlar uchun 4 ta Label, hisoblash uchun bitta Button, amallar uchun bitta CheckBox komponentlarini olamiz va oynaga quyidagi 1.3-rasmdagidek qilib joylashtiramiz.



1.3-rasm. Standart komponentalarni xususiyat va hodisalaridan foydalanish.

Bu masalani loyiha sifatida amalga oshirish uchun quyidagi qadamlar bajariladi:

1-qadam. Forma xususiyatlarini o‘rnatish: BackColor xususiyatiga oq rang; font xususiyatiga 14 o‘lchamli yozuv; Icon ga maxsus ikinka; MaximizeBox xususiyatiga false qiymat; MinimizeBox xususiyatiga false qiymat; Size xususiyatiga yangi o‘lchamlar; StartPosition – CenterScreen holati; Text xususiyatiga “Easy Calculator” matnni yoziladi;

Buni qanday ajratishsh mumkin, ya‘ni o‘zgargan xususiyatlarni, qiymatining matni yog‘on qora ranga kirganlari o‘zgartirilgan hisoblanadi. Bu o‘zgarishlarni dastur fragment ko‘rinida keltiish ham mumkin. Bu dastur fragmentini qo‘lda yozish kerak emas, ammo tushunarli bo‘lishi uchun keltiramiz. Buni InitializeComponent funksiyasi ichida kerakli izohlar bilan keltirilgan bo‘ladi.

```
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font; this->ClientSize = System::Drawing::Size(351, 225);
this->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point, static_cast<System::Byte>(204)));
this->HelpButton = true;
this->Icon = (cli::safe_cast<System::Drawing::Icon^ >(resources->GetObject(L"$this.Icon")));
this->MaximizeBox = false;
this->MinimizeBox = false;
this->StartPosition = System::Windows::Forms::FormStartPosition::CenterScreen;
this->Text = L"Easy Calculator";
```

2-qadam. Label1 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->label1->Text = L"Birinchi son:";
```

3-qadam. Label2 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->label2->Text = L"Ikkinchi son:";
```

4-qadam. Label3 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->label3->Text = L"Natija:";
```

5-qadam. numericUpDown1 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->numericUpDown1->TextAlign = System::Windows::Forms::HorizontalAlignment::Right;
```

6-qadam. numericUpDown2 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->numericUpDown2->TextAlign = System::Windows::Forms::HorizontalAlignment::Right;
```

7-qadam. comboBox1ob‘yektining xususiyatlarini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(5) {L"[+] (qo‘shish)", L"[-] (ayirish)", L"[*] (ko‘paytirish)",
                                    L"[/] (butun bo‘lish)", L"[%] (qoldiq bo‘lish)"});
this->comboBox1->Text = L"tanlang...";
```

8-qadam. label4 ob‘yektining xususiyatini o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->label4->Text = L"Arifmetik amal";
```

9-qadam. button1 ob‘yektining xususiyati va hodisani o‘rnatish dastur fragmenti sifatida keltiramiz.

```
this->button1->Text = L"Hisoblash";
this->button1->Click += gcnew System::EventHandler(this, &Form1::button1_Click);
```

button1_Click funksiyasiga quyidagi algoritmni yozamiz.

```
int one = Convert::ToInt16(Math::Round(numericUpDown1->Value,
0)); int two = Convert::ToInt16(numericUpDown2->Value);
int result = 0;
label3->Text = "Natija:";

switch(comboBox1->SelectedIndex){
        case 0: result = one + two; break; case
        1: result = one - two; break; case 2:
        result = one * two; break; case 3:
        result = one / two; break; case 4:
        result = one % two; break;
                default: result = 0;
                        break;
 }
 label3->Text += result.ToString();
```

Algoritmga izoh brish shart emas, chunki ko‘p kamchiliklari mavjud, masalan, bo‘lish arifmetik amalida. Ammo ba‘zi xulosalarni chiqarish mumkin.

[numericUpDown1->Value] ob‘yektning tanlangan haqiqiy tipdagi qiymatini qaytaruvchi xususiyat, [Convert::ToInt16] – ixtiyoriy tipni butun tipga almashtirib beradi, [label3->Text] – ob‘yektning matn xususiyatini o‘zgartirishi mumkin, [comboBox1->SelectedIndex] – tanlagichning tanlangan indeksini qaytaradi, [label3->Text += result.ToString();] – ob‘yektning matninga matn qo‘shish va tipni satrli tipga o‘tkazish ko‘rsatilgan.

Dastur fragmentida 3 ta belgi [::] – cinfning usullariga murojaat qilish, [->] – ob‘yektning xususiyatiga murojaat qilish, [.] ob‘yektning qiymati ustida bajarilishi kerak bo‘lgan amal. Visual C++ da juda ko‘p usul, va funksiyalarni ishlatish tamoyllari bor, ularni faqat kompilyatorning imkoniyatiga qarab ajratish mumkin.

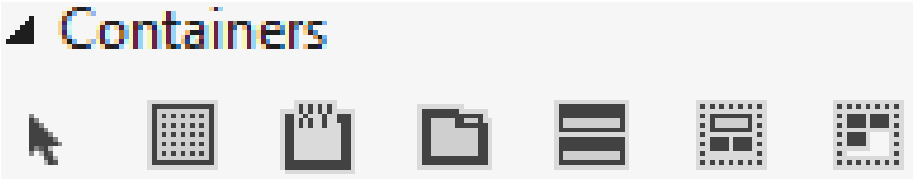
Standart komponentalarning ba‘zi xususiyatlari va hodisalari, ular uchun ba‘zi usullarni ishlatiyu ko‘rdik. Bularni chuqurroq o‘rganish uchun ko‘proq amaliy ishlarni bajarish kerak.

Additional komponentasining xususiyatlari va hodisalari. Bu standart komponentalar bilash ishlash uchun qo‘shimcha komponentlardir. Bular asosan boshqaruv elementlarni guruhlash uchun ishlatiladi. Quyidagi jadvalda qo‘shimacha komponentalarni va vazifalarini keltiramiz.

1.4-jadval. Additional komponentalarining guruhlari va vazifalari

Komponenta-ning guruhi	Komponentaning nomi	Vazifasi va izoh
Boshqaruvlar elementlarni guruhlash	Panel	elementlarni boshqarish uchun to‘plamini guruhlaydi
	GroupBox	Nomlangan elementlarni boshqarish to‘plamini guruhlaydi
	TabControl	Guruhlangan ob‘yektlarni samarali tashkil etish va kiritish uchun sohali sahifani taqdim etadi.
	SplitContainer	Bir biridan ajratilgan ikki paneli beradi.
		Splitcontainer – Splitter o‘rniga mo‘ljallangan.
	TableLayoutPanel	Kontent dinamik satr va ustunlar iborat panelini ifodalaydi.
	FlowLayoutPanel	Dinamik ravishda kontentni gorizontal yoki vertikal joylashtiradigan panelni ifodalaydi.

Bu komponentalar Containers nomli tabda joylashgan. Ularning ko‘rinishi 1.9-rasmda keltirilgan.



1.5-rasm. Containers komponentalari.

Bu komponentlarining xususiyat va hodiralari alohida to‘xtalib o‘tamiz. Vazifalari eelementlari guruhini boshqarishdan iborat, shu maqsadda ishlatiladigan xususiyat va hodisalarini keltiramiz.

Panel komponentasi. Panel – bu oddiy to‘rt tomonidan chegaralangan hududga o‘xshaydi. Uni butun sohani egallashi uchun o‘ng tomonidagi [Panel Task] dagi [dock in parent container]ni

bosish yetarli. [undock in parent container] ni tanlab, oldingi holatiga kelish mumkin. Ob‘yektni harakatlantirish uchun chap tomondagi 4 ta tomonga yo‘naltirilgan tugmachadan foydalaniladi.

AutoSize - xususiyati true/false qiymatlarni qabul qiladi. Odatda chin qiymat ko‘proq ishlatiladi, chunki ob‘yektga joylashtirilgan boshqaruv elementlarininng o‘lchamlariga mos holda o‘lchamni sozlaydi. Joriy holatda yolg‘on qiymatga ega bo‘ladi.

Cursor - xususiyati sichkonchaning kursor ko‘rinishini o‘rnatish imkonini beradi.

BorderStyle – ob‘oektning atrof chegaralarining ko‘rinishlarini o‘rnatish uchun ishlatiladi.

AutoScoroll – ob‘yektda birlashtirilgan elementlar to‘liq ko‘rinmasa, harakatlaguvchi tugmalarni o‘rnatish va olib tashlash uchun ishlatiladi. Bu xususiyat true/false qiymatlarni qabul qiladi.

AutoScorollMargin – bu xususiyat harakatlanuvchi tugmachalar chet masofalarini o‘rnatida, Wight va Height ichki xususiyatlari bor.

AutoScorollMinSize - bu xususiyat harakatlanuvchi tugmachalar eng kichik o‘lchamlarini o‘rnatadi. Wight va Height ichki xususiyatlari bor.

Dock – ob‘yektni oynaning qaysi qismida joylashtirish kerakligini o‘rnatadi.

Margin – ob‘yekt chet qismlarida qoldirilishi kerak bo‘lgan masofalarni o‘rnatadi. Uning 5 ta xususiyati bor, barchasi, chap, o‘ng, yuqori va past.

groupBox komponentasi. groupBox - nomlagan elementlar guruhini birlashtirish uchun ishlatiladi. Ob‘yektni harakatlantirish uchun chap tomondagi 4 ta tomonga yo‘naltirilgan tugmachadan foydalaniladi.

Text - bu xususiyat elementlar guruhini nomini oynaga chiqaradi

AutoSize - xususiyati true/false qiymatlarni qabul qiladi. Elementlarning o‘lchamlariga mos ob‘yektga o‘zlcham ajratadi.

BackColor – ob‘yektning fon xususiyati rangini o‘rnatadi.

BackgroundImage - ob‘yektning fon xususiyati Image ob‘yektlarni o‘rnatadi.

Dock - ob‘yektni oynaning qaysi qismida joylashtirish kerakligini o‘rnatadi.

FlatStyle – ob‘yektning stil xususiyatini o‘rnatadi.

Font – bu xususiyat ob‘yektning yozuvlari uchun aniq bir standart yozuvni o‘rnatish uchun ishlatiladi.

MaximunSize – ob‘yektning eng katta o‘lchamini belgilaydi.

MinimumSize - ob‘yektning eng kichik o‘lchamini belgilaydi.

Padding – ob‘yektgacha bo‘lgan masofalarni o‘rnatish uchun ishlatiladi. Uning 5 ta xususiyati bor, barchasi, chap, o‘ng, yuqori va past.

RightToLeft – ob‘yektning matnini o‘ng tomonidan o‘rnatish.

TabIndex – ob‘yektning tab tugmasi bosilganda, nechinchi navbatda bo‘lishini belgilashni o‘rnatadi.

tabControl komponentasi. tabControl bu oddiy to‘rt tomonidan chegaralangan hududga o‘xshaydi. Uniga yangi sahifa qo‘shish yoki saxifani o‘chirish uchun o‘ng tomonidagi [tabControl Task] dagi [Add Tab] va [Remove Tab] ni bosish yetarli. Ob‘yektni harakatlantirish uchun chap tomondagi 4 ta tomonga yo‘naltirilgan tugmachadan foydalaniladi.

Anchor – ob‘yektni aftor yoki oyna bo‘yiyaa tartiblab joylashtirish xususiyatlarini tanlaydi. Uning to‘rtta qiymati o‘ng, chap, yuqori, past boryu Bularning o‘zaro aralashuvidan foydalaniladi.

Appearance – bu xususiyat ob‘yektning tugmachalarining ko‘rinishini o‘rnatish uchun ishlatiladi. Uning qiymatlari [Normal], [Buttons], [Flat Buttons] qiymatlarni qabul qiladi.

DrawMode – ob‘yektning xususiyatini chizilgan formasini o‘zgartrish uchun ishlvtiladi.

TabPages – bu xususiyat to‘plam qiymatlarni qabul qiladi va uning nechta sahifa bo‘lishini ta‘minlaydi. U bilan ishlaganda har bir sahifaning alohida xususiyatlariga ishlov berish mumkin.

Tabpage1 – bu ob‘yektning saxifa xususiyatlarni o‘rnatish uilatiladiggan ob‘yekt hisoblanadi va panel ob‘yektidek xususiyatlarga ega.

tabStop – ob‘yekt bo‘yicha tab tugmasining harakatlanishi o‘rnatish va o‘chirish qiymatini o‘rnatadi.

splitContainer komponentasi. splitContainer ikkiga ajratilgan elementlarni boshqarish uchun to‘plamni hosil qiladi. Uni butun sohani egallashi uchun o‘ng tomonidagi [splitContainer Task] dagi [dock in parent container]ni bosish yetarli. [undock in parent container] ni tanlab, oldingi holatiga kelish mumkin. Shuningdek, [Horizontal Splitter Orientation] – ob‘yekt ramkalarini gorizontal o‘rnatadi va [Vertical Splitter Orientation] esa vertikal o‘rnatadi. Ob‘yektni harakatlantirish uchun chap tomondagi 4 ta tomonga yo‘naltirilgan tugmachadan foydalaniladi.

Anchor - ob‘yektni aftor yoki oyna bo‘yiyaa tartiblab joylashtirish xususiyatlarini tanlaydi. Uning to‘rtta qiymati o‘ng, chap, yuqori, past boryu Bularning o‘zaro aralashuvidan foydalaniladi.

Backcolor - ob‘yektning fon xususiyati rangini o‘rnatadi.

BorderStyle - ob‘oektning atrof chegaralarining ko‘rinishlarini o‘rnatish uchun ishlatiladi.

FixedPanel – ob‘yekt uchun asosiy panelni o‘rnatish uchun, ya‘ni ikkita paneldan bir asosiy bo‘lishi kerak.

isSplitterFixed - ob‘yekt uchun asosiy panelni o‘rnatish amal qilishi yoki qilmasligini o‘rnatiadi. xususiyati true/false qiymatlarni qabul qiladi.

Orientation – ob‘yektning panellarini gorizontal yoki vertikal o‘rnatish uchun ishlatiladi. Ikkita Horizontal va Vertical qiymat qabul qiladi.

splitContainer1.Panel1 – ob‘yektning panel xususiyatlariga ishlov berish uchun panel xususiyatlarini ochib beradi va u uchun xususiyatlarni o‘rnatish mumkin.

Panel1Coolapsed – ob‘yektda panel1ni to‘liq qilib o‘rnatadi. Bunday xususiyat barcha panellari uchun o‘rinlidir.

Panel1MinSize – ob‘yektdagi panelning eng kichik o‘lchamini o‘rnatishdan iborat.

RightToLeft- ob‘yekt panellarini o‘ngdan o‘rnatish xususiyati.

tableLayoutPanel komponentasi. tableLayoutPanel – bu jadval kabi tartiblangan panellar to‘plami orqali elementlarni birlashtirish ob‘yektidir.

AutoScoroll – ob‘yektda birlashtirilgan elementlar to‘liq ko‘rinmasa, harakatlaguvchi tugmalarni o‘rnatish va olib tashlash uchun ishlatiladi. Bu xususiyat true/false qiymatlarni qabul qiladi.

AutoScorollMargin – bu xususiyat harakatlanuvchi tugmachalar chet masofalarini o‘rnatida, Wight va Height ichki xususiyatlari bor.

AutoScorollMinSize - bu xususiyat harakatlanuvchi tugmachalar eng kichik o‘lchamlarini o‘rnatadi. Wight va Height ichki xususiyatlari bor.

CellBorderStyle – ob‘yektning yacheykalarning chet chegaralarining stilini belgilaydi.

ColumnCount – ob‘yektdagi ustunlar sonini o‘rnatish.

Columns – ob‘yekt ustunlari uchun xususiyatlarni o‘rnatish uchun alohida muloqot oynasi ochiladi.

GrowStyle – ob‘yektga ustun yoki qator yoki ikkalasini qo‘shish xususiyatini o‘rnatish.

RowCount – qatorlar soni

Rows - ob‘yekt qatorlari uchun xususiyatlarni o‘rnatish uchun alohida muloqot oynasi ochiladi.

flowLayoutPanel komponentasi. flowLayoutPanel **–** ketma ketlik kabi tartiblangan panellar to‘plami orqali elementlarni birlashtirish ob‘yektidir.

Anchor - ob‘yektni aftor yoki oyna bo‘yiyaa tartiblab joylashtirish xususiyatlarini tanlaydi. Uning to‘rtta qiymati o‘ng, chap, yuqori, past boryu Bularning o‘zaro aralashuvidan foydalaniladi.

Backcolor - ob‘yektning fon xususiyati rangini o‘rnatadi.

AutoSize - xususiyati true/false qiymatlarni qabul qiladi. Elementlarning o‘lchamlariga mos ob‘yektga o‘zlcham ajratadi.

WrapContents – ob‘yektda sontentlarni ketma ketligini ustuvor qilish yoki qilmaslikni o‘rnatadi. Xususiyat true/false qiymatlarni qabul qiladi.

Bu qo‘shimacha komponentalarning xususiyatlari haqida ma‘lumotlarni berildi. Ularning hodisalari bir biriga o‘xshash bo‘lib, standart komponentalarningdan farqi juda kam. Shuning uchun hodisalarga to‘xtalaib o‘tirmaymiz. Shuningdek, qo‘shimacha komponentalar juda ko‘p, buni ko‘rish uchun Tollbox oynasiga sichqonchaning o‘ng tugmasini bosing va [Shown All] buyrug‘ini tanlang.

System komponentasining xususiyatlari va hodisalari. Bu komponentalarga asosan operatsion tizim bilan ishlaydigan komponentalar kiradi. Ularni xususiyatlari murakab hisoblanadi. Chunki birgina menyu yaratish uchun standart va qo‘shimcha komponentalar ham foydalaniladi. Shuning uchun umumiy holda bu komponentalarning vazifalarini keltirib o‘tamiz.

1.6-jadval. System komponentalarining guruhlari va vazifalari

Komponentaning guruhi	**Komponentaning nomi**	**Vazifasi va izoh**

Buyruqlar	ToolStrip	Microsoft Windows, Microsoft Office, Microsoft Internet Explorer bo‘lishi mumkin uskunalar majmuasi yoki foydalanuvchi interfeysini yaratadi. Toolstrip control Asboblar paneli boshqaruvini almashtirishga mo‘ljallangan.
	ToolStripContainer	Toolstrip elementlarini ajralmas to‘plamini beradi (birikib, sohasida gorizontal yoki vertikal almashish)
Ma‘lumotni aks ettirish (faqat o‘qish elementlari uchun)	StatusStrip	Windowsning holat satrini beradi Statusstrip - Statusbar o‘rnini bosadi. Statusstrip maxsus xususiyatlari tartibini o‘z ichiga oladi,
Menyuni boshqarish elementlari	MenuStrip	Forma uchun menyu tizimini beradi. Menustrip - Mainmenu o‘rnini bosadigan yuqori darajali konteyner hisoblanadi. Bundan tashqari, asosiy ishlov berish va hujjat interfeysi imkoniyatlarini beradi. Funksionaligi Toolstripitem olingan bo‘lsa-da,Toolstripdropdownitem va Toolstripmenuitem bilan birga ishlashlatiladi.
	ContextMenuStrip	Kontekst menyuni ifodalaydi. Contextmenustrip bu Contextmenu o‘rnini bosadi. Contextmenustrip bilan birga ishlatiladi va sichqonchaning o‘ng tugmasini bosishda avtomatik ravishda kontekst menyusi namoyon bo‘ladi.

1.7-rasm. Menyu va uskunalar panellarini yaratish uchun komponentalarning ko‘rinishi.

Menyu va uskunalar panellarini yaratish uchun komponentalardan tashqari tizim bilan ishlaydigan komponentalarga quyidagi 1.8jadvaldagi komponentalar ham qiradi, ular Components tab ga kiritilgan.

1.8-jadval. Components tab ning komponentalari.

Komponentaning nomi	Vazifasi va izoh	
BackgroundWorker	Alohida ajratilgan oqim uchun amal bajara oladigan BackgroundWorker komponentaning nusxasini yaratadi	
DirectoryEtry	Active Directory xizmati bilan o‘zaro bog‘lanib ishlashi va uning iearxiyasidan ob‘yekt yoki tugunlarni inkapsulyatsiya oladigan DirectoryEntry komponentaning nusxasini yaratadi	
DirectorySearcher	Active Directory da so‘rovlarni bajarish imkonini beradigan DirectorySearcher komponentaning nusxasini yaratadi	
ErrorProvider	Formada elementni boshqarishdagi xato foydalanuvchiga ko‘rsatish uchun ErrorProvider komponentaning nusxasini yaratadi	
EventLog	Tizim va foydalanuvchining hodisalar jurnaliga bog‘lanish imkonini beradigan EventLog komponentaning nusxasini yaratadi	
FileSystemWatcher	Ruxsat berilgan papka va fayllarga o‘zgartirishlar kiritish imkonini beradigan FileSystemWatcher komponentaning nusxasini yaratadi	

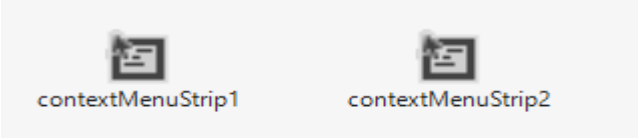
HelpProvider	Tezkor va foydalanuvchining yordam oynasini chaqirish imkonini beruvchi HelpProvider komponentaning nusxasini yaratadi	
ImageList	Image ob‘yektlarining to‘plamini boshqarish usullarini ta‘minlovchi ImageList komponentaning nusxasini yaratadi	
MessageQueue	Navbatdagi xabarlarni o‘qish imkoniyatini beruvchi MessageQueue komponentaning nusxasini yaratadi	
PerformanceCounter	Windowsning hisoblagichlari bilan ishlash uchun foydalaniladi, yangi toifalar va misollar yaratish, hisoblagichlardan qiymatlarni o‘qish va hisoblagich ma‘lumotlariga asoslangan hisob-kitoblarni bajarish uchun PerformanceCounter komponentaning nusxasini yaratadi	
Process	Tizimdagi jarayonlar bilan bog‘liq ma‘lumotlarni to‘xtatish, ishga tushirish va o‘zgartirish uchun Process, komponentaning nusxasini yaratadi	
SerialPort	sinxron va voqea imkoniyatlar uchun SerialPort komponentaning nusxasini yaratadi	
ServiceController	Mavjud xizmatlarni, shu jumladan, boshlang‘ich va to‘xtatish xizmatlarini boshqarish va ularga buyruqlar berish uchun ServiceController komponentaning nusxasini yaratadi	
Timer	Windows ilovalar uchun vaqtga asoslangan vazifalarni kiritish uchun Timer komponentaning nusxasini yaratadi	

1.9-rasm. Components tab ga kiritilgan komponentalarning ko‘rinishi.

Bu komponentalardan kontkst menyuni yaratish va boshqa ob‘yektlarga o‘rnatishni ko‘rib chiqamiz.

Masala. Oynada ikkita kontkst menyu yaratish kerak. Birinchisi oyna uchun, ikkinchisi RichTextBox ob‘yekti uchun bo‘lsin. Birinchi kontekstda oyna yopish, kattalashtirish, kichiklashtirish, yig‘ishtirish, tugmachaolarni yoqish va o‘chirish bo‘yruqlari bo‘lsin. Ikkinchi kontekstda matnni tahrirlash uchun buferga nusxalash, buferga nusxalash va o‘chirish, nusxa qo‘shish, ob‘yektni tozalash, gorozantal va vertikal harakatlanish tugmachalarni qurish va o‘chirish amallari bo‘lsin.

Yangi loyiha yaratamiz va bir oyna bo‘ladi. Oynaning bir chetiga RichtextBox ni o‘rnatamiz. Menyu va uskunalar tabidan contextMenuStrip ob‘yektidan ikkita o‘rnatamiz. Ular formada ko‘rinmasligi mumkin. Ular loyihaning ishchi maydoni pastki qismida joylashadi (1.10-rasmga qarang).

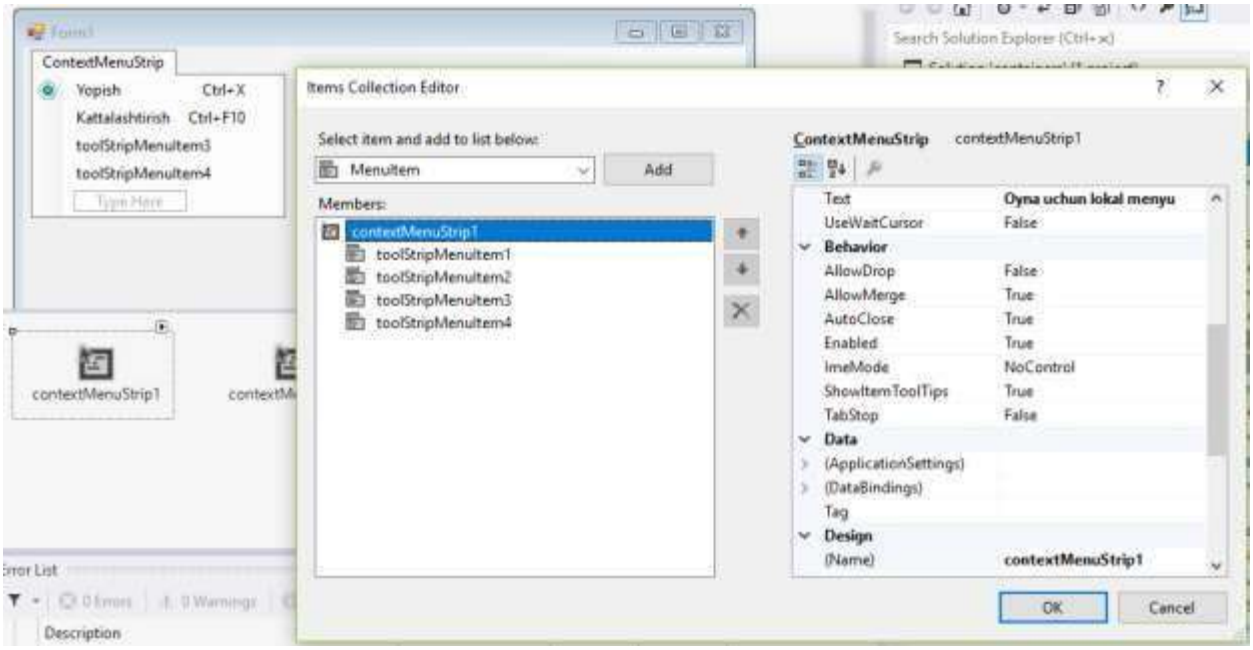


1.10-rasm. contextMenuStrip ob‘yekti o‘rnatilgan ko‘rinishi.

Bularni kerakli ob‘yektlari bilan bog‘laymiz. Oynaga birinchi kontekst menyuni [contextMenuStrip1], [RichTextBox1] ob‘yektga ikkinchi kontekst menyuni [contextMenuStrip2] o‘rnatamiz. Buning uchun oynaning [ContextMenuStrip] xususiyatiga [contextMenuStrip1] ni va RichTextBox1] ob‘yektning [ContextMenuStrip] xususiyatiga [contextMenuStrip2] ni o‘rnatamiz. Agar bitta kontekstni ikkita ob‘yektga o‘rnatilsa, kontekstning buyruqlarga joriy qilingan algoritmlarni bajaraveradi.

O‘rnatilgan kontekst menyu ob‘yekti [ContextMenuStrip] xususiyati uchun ichki xususiyatlarni, ya‘ni [contextMenuStrip] xususiyatlarni o‘rnatish uchun ochib beradi. Bu

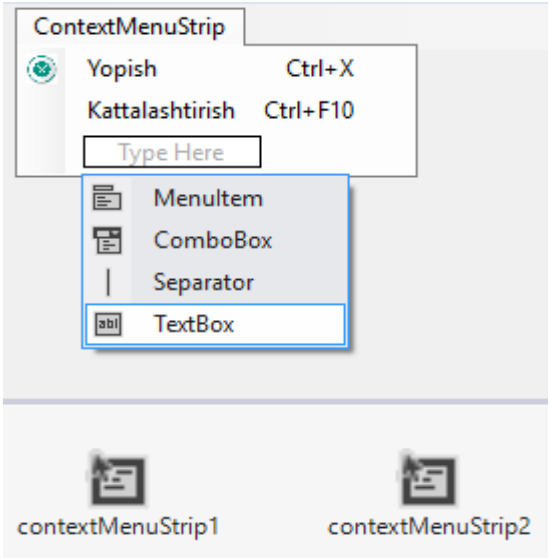
xususiyatlarni hammasiga to‘xtalib o‘tmaymiz. Faqat Items xususiyatiga to‘xtalamiz. Shu xususiyat tanlanganda quyidagicha moloqot oynasi chiqadi(1.13-rasmga qarang).



1.11-rasm. Kontekst menyu buyruqlarini o‘rnatish.

E‘tibor qaratsangiz, muloqot oynasida qilingan o‘rnatishlar muloqot oynasida emas balki asosiy oynada ko‘rinadi. Agar bu muloqot oynasida ishlash noqulay bo‘lsa, asosiy ishchi maydonda ham ishlash mumkin.

Buning kerakli kontekst menyu ob‘yekti uchtiga sichqonchani bir marta bosilsa, oynada kontekst menyu hosil bo‘ladi (1.12-rasmga qarang).



1.12-rasm. Oynada kontekst menyuga ishlov berish.

[Type Here] tugmasi orqali joriy [MenuItem], [ComboBox], [Separator], [TextBox] - 4 ta elementlarni qo‘shish mumkin. Qo‘shilgan har bir elementning o‘ng tugmasini bosib, u bilan ma‘lum bajarilishi mumkin bo‘lgan amallari bajarish mumkin. Shuningdek, [properties] degan xususiyatlarga olib kiradi. [Text] - xususiyatiga kirib buyruk nomini berish mumkin. Agar harf bilan murojjat qilishni xoxlasangiz, kerakli harf oldiga [&] belgisini qo‘yish kerak. Inteaktiv tugma o‘rnatish uchun esa [ShortcutKeys] xususiyatiga kerakli tugmalar kombinatsiyasini o‘rnatish mumkin. Ikonka o‘rnatish uchun [Image] xususiyatga kerakli rasmni o‘rnatish mumkin. Shunday qilib kerakli barcha buyruqlarni amalga oshirish mumkin. Buyruqlarni guruhlash uchun [Separator] komponentasini tanlash kerak. Buyruqlarga algortim yozishsh uchun ularni ustiga sichqonchani ikki marta bosish yetarli yoki hodisalariga kirib hodisasiga yozish mumkin.

Agar yuqorida kontekst menyuga ishlov berish tushunarli bo‘lgan bo‘lsa, masalani o‘zingiz tugatib qo‘ying. Agar ketma ketlik asosida bajarsangiz albatta masalani hech bo‘lmasa, algoritmlarsiz hal qilishingiz lozim.

NAZORAT SAVOLLARI

1. Toolbox oynasi 5 ta bo‘limdan iborat va nima uchun xizmat qiladi.
1. [List View] va [Show All] buyruqlari nima uchun kerak.
2. Toolbox interaktiv oynasi nechta komponenta muvjud?
3. Toolbox interaktiv oynasi komponentalarining o‘rnini almashtirish va qayta nomlash mumkinmi?
4. Komponenta xususiyatlarining asosiy nechta guruh bor.
5. Komponentalarning Focus guruhi nima uchun ishlatiladi.
6. Asosiy oynaga qo‘shishlgan komponentalar ro‘yxatini qaerdan ko‘rish mumkin.
7. Oynaning InitializeComponent funksiyasida ham komponentalarning xususiyatlari ko‘shilganini ko‘rish mumkinmi?
8. Komponentaning nechta usullari (methods) va hodisalari (events) guruhi bor?
9. Komponentaning Layout guruhi nima uchun ishlatiladi?
10. Standart komponentalarga Toolbox oynasidagi faysi Tab dagi komponentalar kiradi.
11. Mantlarni tahrirlash komponentalarini sanab bering?
12. Ro‘yxatdan tanlash komponentalarini sanab bering?
13. Komponentalarni oynaga joylashtirish uchun qanday amal bajarish kerak?
14. Label1 ob‘yektining text xususiyatini o‘rnatish qanday amalga oshiriladi?

15. comboBox1 ob‘yektining Items xususiyatlarini o‘rnatish qanday amalga oshiriladi?
16. [numericUpDown1->Value] ob‘yektning tanlangan qanday tipdagi qiymatini qaytaruvchi xususiyat.
17. [comboBox1->SelectedIndex] - tanlagichning tanlangan qanday qiymatini qaytaradi.
18. Additional komponentasining xususiyatlari va hodisalari nima uchun ishlatiladi.
19. TabControl komponentasining vazifasini ayting.
20. Panel komponentasi xususiyatlariga misollar keltiring.
21. Panel komponentasini oynaning qaysi qismida joylashtirish kerakligini o‘rnatadigan xususiyatni ayting.
22. Bu xususiyat to‘plam qiymatlarni qabul qiladi va uning nechta sahifa bo‘lishini ta‘minlaydi. U bilan ishlaganda har bir sahifaning alohida xususiyatlariga ishlov berish mumkin. Gap qaysi komponentaning qaysi xususiyati haqida ketmoqda.
23. System komponentasining xususiyatlari va hodisalarini sanab bering.
24. Menyuni boshqarish elementlari uchun ishlatiladigan komponentalarni sanab bering.

№19-AMALIY MASHG’ULOT

MAVZU:MULOQOT OYNALARI BILAN ISHLASH. VISUAL C++MUHITIDA MULOQOT OYNALARI VA ULARNI SOZLASH, BOSHQARISH ELEMENTLARI.

Ishning maqsadi

Button, TextBox va **TextBox** komponetalaridan foydalanib, satrlarga qayta ishlov beruvchi vizual dastur yaratish.

TextBox komponentasining **PasswordChar** xossasi yordamida parollarni kiritishni boshqa shiriftlarga o‘zgartirish mumkin. Ya’ni **text** xossasidan kiritilayotgan ma’lumotlarni ko‘rinishlarini o‘zgartirib, foydalanuvchi tanimaydigan shiriftlarda kiritilishini ta’minlaydi. Bu xususiyat dastur xavfsizligini ta’minlashda katta axamiyat kasb etadi. Bu galgi laboratoriya ishida dasturga parol yordamida kiritish jarayonlarini boshqaruvchi va virtual tarzda komponentalar yaratuvchi amaliy vizual dastur yaratiladi. Ushbu jarayonlarni boshqarishda global e’lon qilinmagan tugmalarni bosgandagi hodisalariga ham to‘xtalib o‘tiladi. Asosiy maqsad **char** turidagi massivlar bilan ishlashda vujudga keladigan muammoli holatlarni bartaraf qilish usullarini o‘rganishdir. Masalan **TextBox** komponentasidan kiritilayotgan ma’lumotlar **String^** tipida. Bunday tipdagi ma’lumotlarni **char** tipiga o‘tkazish. Yoki bo‘lmasa **char** tipidagi massivga o‘tkazishni ko‘rib chiqamiz.

Masalalarni yechishda quyidagi keltirilgan na’munaga ahamiyat berish zarur.

Topshiriq

Ma’ruza va amaliyot darslarida olingan ma’lumotlarga asoslanib, o‘z variantingiz uchun quyidagilarni bajarinig:

a) 1- jadvaldan o‘z variantingizni ko‘chirib oling;

b) quyidagi na’muna asosida o‘z variantingizdagi masalalarni bajaring.

Har bir variantda quyidagilar bajarilishi lozim:

1- Dasturga KIRISHda parol so‘ralsin;
2- yangi parol kiritish va saqlash imkoniyatlari bo‘lsin;
3- dasturni boshqatdan ishga tushirilganda yangi kiritilgan parol saqlanishi lozim.

Topshiriqlarni bajarish uchun uslubiy ko‘rsatmalar.

Visual C++ dasturida 30 – variantdagi masalani Windows Application muhitida ishlanishi:

1-qadam. Formaga 2 ta **TextBox,** 2 ta **label,** 1 ta **Button** komponentalari joylashtiriladi. **Button** komponentasini **onclick** xodisasiga quyidagi kodlar yoziladi:

```
char satr[200]="";
for(int i=0; i<textBox1->Text->Length; i++){
satr[i]=textBox1->Text[i];}
for(int i=0; i<strlen(satr); i++)
{ if(satr[i]==' '){satr[i]='_';}}
String^ s=gcnew String(satr);
textBox2->Text=s;
```

2-qadam. Form1 ning **load** xodisasiga quyidagi kodlar yoziladi:

```
label2->Visible=false;
label1->Text="Parolni kiriting";
textBox2->Visible=false;
button1->Visible=false;
Button^ button4 = gcnew Button();//button yaratish
button4->Font = gcnew System::Drawing::Font("Courier New", 16.0F, FontStyle::Italic);
textBox1->PasswordChar = '*';//kodlar terilganda * //ko‘rinishida bo‘lishini ta’minlaydi
textBox1->Font = gcnew System::Drawing::Font("Courier New",16.0F);
textBox1->Size = Drawing::Size(208, 35);
Form1::Height=200;
Form1::Width=230;
```

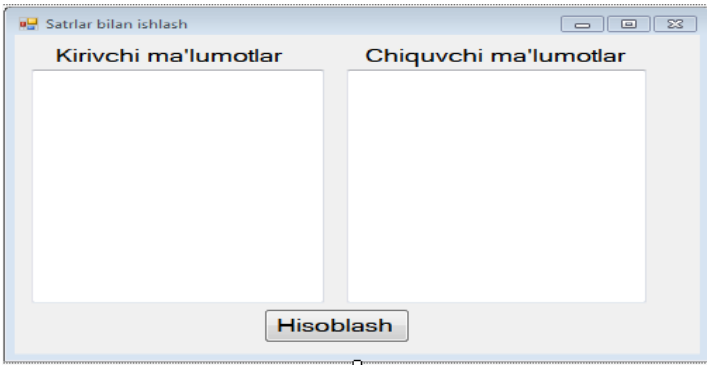
```
button4->Visible = true;
button4->Size = Drawing::Size(100, 30);
button4->Location = Drawing::Point(50, 90);
button4->Text = "Ok";
this->Controls->Add(button4);
parol="Salom";
button4->Click += gcnew EventHandler(this, &Form1::ЩelchokNaKnopke );
```

3-qadam. Satrlarga ishlov beruvchi funksiyalar kutubxonasi e'lon qilinadi va **ЩelchokNaKnopke** funksiyasi tuziladi:

```
#pragma once
#include<string.h>
private: System::Void ЩelchokNaKnopke(System::Object^ sender, System::EventArgs^ e){
newparol=textBox1->Text;
if(parol==newparol){
textBox1->Size = Drawing::Size(208, 242);
Form1::Size=Drawing::Size(503,369);
textBox1->Location = Drawing::Point(20, 35);
textBox1->Text = "";
textBox1->PasswordChar = NULL;
label1->Text="Kiruvchi ma'lumotlar";
                                    label1->Visible=true;
                                    label2->Visible=true;
                                    textBox1->Visible=true;
                                    textBox2->Visible=true;
                                    button1->Visible=true;}
else {
System::Windows::Forms::FormClosingEventArgs^ r;
auto MBox = MessageBox::Show("Yana urunib ko'rasmi?", "Xato",
MessageBoxButtons::YesNoCancel, MessageBoxIcon::Exclamation);
// YES — dialog; NO — выход; CANCEL - redaktirovat
if (MBox == Windows::Forms::DialogResult::No) return;
if (MBox == Windows::Forms::DialogResult::Cancel){ this->Close();}
if (MBox == Windows::Forms::DialogResult::Yes)
```

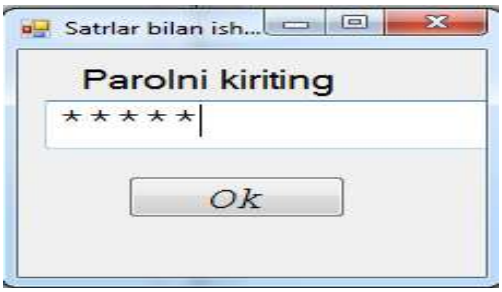
```
{   textBox1->Text="";
}    }              }
```

Dasturning umumiu ko‘rinishi 5.1-rasmda keltirilgan.



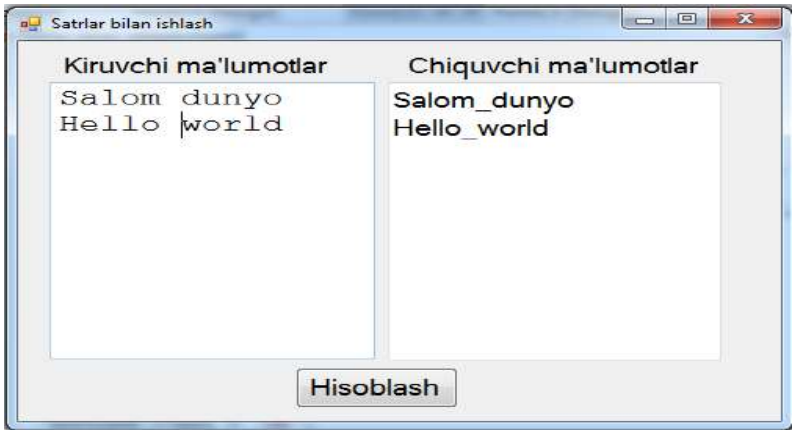
1.1-rasm. Dasturning umumiy ko‘rinishi.

Lekin dastur ishga tushganda **Form1** ning o‘lchami o‘zgaradi. Ya’niy foydalanuvchidan parol kiritishni talab qiluvchi oynaga o‘zgaradi. Uning ko‘rinishi 1.2-rasmda keltirilgan.



1.2-rasm. Dasturning dastlabki ko‘rinishi.

Bu oynada **OK** tugmachasi vurtual yaratilgan va parol kiritish uchun **TextBox** komponentasining **PasswordChar** xossasi = **'*'** ga tenglashtirilgan. Parol to‘g‘ri terilgandan keyin dasturning ishlashi quyidagicha:



1.3-rasm. Dasturning parol to‘g‘ri kiritilgandan keyingi qo‘rinishi.

Dasturning umumiy kodlari quyidagicha:

```
#pragma once
#include<string.h>
namespace Satrlar_bilan_ishlash {
        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;
public ref class Form1:public System::Windows:: Forms::Form{
        public:
Form1(void){
InitializeComponent(); }
        protected:
~Form1(){
if (components){
        delete components;}             }
private: System::Windows::Forms::Button^  button1;
protected:
private: System::Windows::Forms::TextBox^  textBox1;
private: System::Windows::Forms::Label^  label1;
private: System::Windows::Forms::TextBox^  textBox2;
private: System::Windows::Forms::Label^  label2;
private:
System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
void InitializeComponent(void){
// komponentalar e'lon qilinadigan bo'limi}
#pragma endregion
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
```

```
char satr[200]="";
for(int i=0; i<textBox1->Text->Length; i++)
{ satr[i]=textBox1->Text[i]; }
 for(int i=0; i<strlen(satr); i++) { if(satr[i]==' ') {satr[i]='_';}}
String^ s=gcnew String(satr); textBox2->Text=s;}
String^ parol, ^newparol;
private: System::Void Form1_Load(System::Object^  sender, System::EventArgs^  e) {
label2->Visible=false; label1->Text="Parolni kiriting";
textBox2->Visible=false;  button1->Visible=false;
Button^ button4 = gcnew Button();
button4->Font = gcnew System::Drawing::Font("Courier New", 16.0F, FontStyle::Italic);
textBox1->PasswordChar = '*';
textBox1->Font = gcnew System::Drawing::Font("Courier New",16.0F);
textBox1->Size = Drawing::Size(208, 35);
Form1::Height=200;
Form1::Width=230;
button4->Visible = true;  button4->Size = Drawing::Size(100, 30);
button4->Location = Drawing::Point(50, 90);  button4->Text = "Ok";
this->Controls->Add(button4); parol="Salom";
 button4->Click += gcnew EventHandler(this, &Form1::ЩelchokNaKnopke );
                              }
private: System::Void ЩelchokNaKnopke(System::Object^ sender, System::EventArgs^ e)
{            newparol=textBox1->Text;
          if(parol==newparol){
                    textBox1->Size = Drawing::Size(208, 242);
                    Form1::Size=Drawing::Size(503,369);
                    textBox1->Location = Drawing::Point(20, 35);
      textBox1->Text = "";
                    textBox1->PasswordChar = NULL;
                    label1->Text="Kiruvchi ma'lumotlar";
                                        label1->Visible=true;
                                        label2->Visible=true;
                                        textBox1->Visible=true;
```

```
                    textBox2->Visible=true;
                    button1->Visible=true;
        }
                    else {
System::Windows::Forms::FormClosingEventArgs^ r; auto MBox = MessageBox::Show("Yana
urunib ko'rasmi?", "Xato", MessageBoxButtons::YesNoCancel, MessageBoxIcon::Exclamation);
// YES — dialog; NO — выход; CANCEL - redaktirovat
if (MBox == Windows::Forms::DialogResult::No) return;
if (MBox == Windows::Forms::DialogResult::Cancel){ this->Close();}
if (MBox == Windows::Forms::DialogResult::Yes)
{   textBox1->Text=""; }    }
                }}; }
```

Nazorat savollari

1. Axborot tizimlarida muloqot oynalarining qanday guruhlari bor.
2. Muloqot oynalariga qo‘yiladigan talablar bormi va bo‘lsa, nima uchun?
3. Muloqot oynasida teskari aloqa deganda nimani tushunasiz?
4. Muloqot oynada nimalar aniq ko‘rsatilishi kerak?
5. [Dialogs] tab oynasida joylashgan muloqot komponentalarini sanab bering?
6. ColorDialog komponentasining vazifasini ayting?
7. Qaysi komponenta foydalanuvchilar uchun papkalar ko‘rish, yaratish va tanlash imkonini beradigan muloqot oynasini ko‘rsatadi.
8. Interaktiv xabarlarni berish va aniq javoblarni olish Visual C++ da qanday sinfi mavjud.
9. MessageBox sinfnining show funksiyasi paramertlarini sanab bering?
10. MessageBoxOptions qanday ob’ekt tipi va nima vazifani bajaradi.
11. MessageBoxButtons, MessageBoxIcon, MessageBoxDefaultButton ob’ektlarini birini qabul qiluvchi kengaytirilgan ob’ekt tipi nomini ayting.
12. Foydalanuvchi tomonidan yaratiladigan muloqot oynasi qanday yaratiladi.
13. Interaktiv muloqot oynalari kabi nimalarni yaratish va ularni boqarish, kerakli ma’lumotlarni olish uchun ishlatiladi.
14. Qaysi sinfning nomlar fazosi System.Windows va kutubxonasi PresentationFramework.dll hisoblanadi.
15. **ColorDialog muloqot oynasining asosiy vazifasi va ob’ekti qanday yaratiladi.**
16. ColorDialog muloqot oynasidan foydalanish uchun qanday konstruktorni ishga tushirish lozim.
17. CanRaiseEvents xususiyatining vazifasini ayting?
18. komponentaga ilova qilinadigan hodisalar ro‘yxatini oladi xususiyat nomini ayting?
19. Muloqot oynasi ochilganda maxsus ranglarni yaratish uchun ishlatiladigan boshqaruv elementlari ko‘rinib turishini ko‘rsatuvchi qiymatni oladigan yoki o‘rnatadigan xususiyat nomini ayting?
20. Dispose() bu qanday usul?

21. Komponenta yoki uning konteyneri tomonidan taqdim etilgan xizmatni ifodalovchi ob’ektni qaytaradigan hodisani ayting?

22. Disposed hodisaning vazifasini ayting?

23. Foydalanuvchi umumiy muloqot oynasidagi yordam tugmasini bosganda sodir bo‘ladigan usul nomini ayting?

24. Foydalanuvchilarga kerakli komponenta uchun shrift va uning xususiyatlarini o‘rnatish imkonini beruvchi muloqot oynasini nomini ayting?

25. **PrintPreviewDialog muloqot oynasi qaysi tabda joylashgan va vazifasini aniq aytib bering.**

26. Dastur fragmentida bir StreamReader oqimi nima uchun yaratiladi.

27. Win32Window Interface nima uchun ishlatiladi.

28. Muloqot oynasida OK va Cancel tugmalarini o‘rnatish qayday amalga oshiriladi.

29. MessageBoxIcon muloqot oynalarida nima uchun ishlatiladi.

30. Qandaydir hodisa haqida faqat xato amal bajarilganligi haqida ma’lumot beruvchi muloqot oynasi uchun ikonka nomi va yozlishini tushuntirib bering.

31. Muloqot oynasida standart tugmalarni o‘rnatuvchi o‘zgarmaslarni belgilaydigan parametrni ayting?

32. RtlReading qaysi parametr uchun ishlatiladi va muloqot oynasida nima hodisa sodir bo‘ladi.

33. System::Windows::Forms::DialogResult r6 = MessageBox::Show(this, message, caption, MessageBoxButtons::OK, MessageBoxIcon::Question, MessageBoxDefaultButton::Button1, (MessageBoxOptions)0, "mspaint.chm", HelpNavigator::KeywordIndex, "ovals") – dastur fragmentini tushuntirib bering

34. Asosiy formaga yang forma qanday qo‘shiladi

35. Nima uchun [StartPosition] xususiyatining qiymatini [CenterParent]ga tenglashtiriladi.

36. ShowDialog() va Show() usullarining farqini tushuntirib bering.

37. Oddiy ob’ekt yaratish va konstruktor asosida ob’ektni yaratish farqini tushuntirib bering.

38. Muloqot oynalarni yaratishda foydalanuvchi yangi konstruktor yaratishi mumkinmi?

39. [menu] → [project] → [add New item] → [UI] → [Windows Form] ketma ketlik qanday amalni bajaradi.

40. Muloqot oynasidagi ma’lumotlarni qanday qilib asosiy formaga o‘tkazish mumkin.

AMALIYOT MASHG‘ULOTINI BAJARISH BO‘YICHA VARIANTLAR

MAVZU: KICHIK LOYIHALAR YARATISH: MATNLI MAYDONGA PAROL KIRITISH, SATRLARGA QAYTA ISHLOV BERISH

Ishning maqsadi

Button, TextBox va **TextBox** komponetalaridan foydalanib, satrlarga qayta ishlov beruvchi vizual dastur yaratish.

TextBox komponentasining **PasswordChar** xossasi yordamida parollarni kiritishni boshqa shiriftlarga o‘zgartirish mumkin. Ya’ni **text** xossasidan kiritilayotgan ma’lumotlarni ko‘rinishlarini o‘zgartirib, foydalanuvchi tanimaydigan shiriftlarda kiritilishini ta’minlaydi. Bu xususiyat dastur xavfsizligini ta’minlashda katta axamiyat kasb etadi. Bu galgi laboratoriya ishida dasturga parol yordamida kiritish jarayonlarini boshqaruvchi va virtual tarzda komponentalar yaratuvchi amaliy

vizual dastur yaratiladi. Ushbu jarayonlarni boshqarishda global e'lon qilinmagan tugmalarni bosgandagi hodisalariga ham to'xtalib o'tiladi. Asosiy maqsad **char** turidagi massivlar bilan ishlashda vujudga keladigan muammoli holatlarni bartaraf qilish usullarini o'rganishdir. Masalan **TextBox** komponentasidan kiritilayotgan ma'lumotlar **String^** tipida. Bunday tipdagi ma'lumotlarni **char** tipiga o'tkazish. Yoki bo'lmasa **char** tipidagi massivga o'tkazishni ko'rib chiqamiz.

1-jadvaldagi masalalarni yechishda quyidagi keltirilgan na'munaga ahamiyat berish zarur.

Topshiriq

Ma'ruza va amaliyot darslarida olingan ma'lumotlarga asoslanib, o'z variantingiz uchun quyidagilarni bajarinig:

a) 1- jadvaldan o'z variantingizni ko'chirib oling;

b) quyidagi na'muna asosida o'z variantingizdagi masalalarni bajaring.

1-jadvaldagi har bir variantda quyidagilar bajarilishi lozim:

4- Dasturga KIRISHda parol so'ralsin;
5- yangi parol kiritish va saqlash imkoniyatlari bo'lsin;
6- dasturni boshqatdan ishga tushirilganda yangi kiritilgan parol saqlanishi lozim.

1-jadval.

1. Matnli faylda nechta so'z va gap qatnashayotganligini aniqlovchi dastur tuzing.

Kiruvchi	**Chiquvchi**
Bugun burunchi para C++ darsi. 2-para Matematika.	7 2
Toshkent O'zbekiston poytaxti.	3 1
Mening Malibu rusumli mashinam bor edi. Uni sotdim.	8 2

2. Matnli faylda nechta "a" va "." qatnashayotganligini aniqlovchi dastur tuzing.

Kiruvchi	**Chiquvchi**
Mana senga olam olam gul.	5 1
.Toshkent O'zbekiston poytaxti.	1 2
Mning Malibu rusumli mashinam bor.	3 1

3. Berilgan gapdagi simvollar soni n tadan kam bo'lgan so'zlarni o'chiring.

Kiruvchi	**Chiquvchi**
5 Mana senga olam olam gul.	senga
8 Toshkent O'zbekiston poytaxti.	O'zbekiston

7 Mening Malibu rusumli mashinam bor.	rusumli mshinam

4. Matnli satrda katta harflar bilan berilgan satr berilgan. Undagi qism satr necha marta qatnashayotganligini aniqlovchi dastur tuzing.

Kiruvchi	**Chiquvchi**
DEF DEFDEFABCABCZABCDEFDEF	4

5. Matnli fayldagi simvollar ketma-ketligi bir nechta probellar bilan ajratilgan holda berilgan. Undagi eng uzun so‘zning “a” harflarini “b” bilan almashtiring.

Kiruvchi	**Chiquvchi**
Mana senga olam olam gul!	Mana sengb olam olam gul!

6. Matnli fayldagi simvollar ketma-ketligi bir nechta probellar bilan ajratilgan holda berilgan. Undagi eng uzun va eng kalta so‘zlarni sonini probel bilan chiqaring. Masalan, “hi! say biggest” uchun natija “1 2”

Kiruvchi	**Chiquvchi**
Mana senga olam olam gul!	1 1

7. Matnli fayldan ortiqcha probellarni olib tashlovchi dastur tuzing. Har bir so‘z orasida bitta probel tursin.

Kiruvchi	**Chiquvchi**
Mana senga olam olam gul!	Mana senga olam olam gul!

8. Matnli fayldagi gaplarni birinchini ikkinchiga, uchinchini to‘rtinchiga va hokazo so‘zlarini o‘zgartiruvchi dastur tuzing.

Kiruvchi	**Chiquvchi**
Mana senga olam olam gul! avo yaxshi.	senga Mana olam olam gul! yaxshi Havo.

9. Matnli fayldagi birinchini ikkinchiga, uchinchini to‘rtinchiga va hokazo gaplarni o‘rnini o‘zgartiruvchi dastur tuzing.

Kiruvchi	**Chiquvchi**
Mana senga olam olam gul! Havo yaxshi.	Havo yaxshi. Mana senga oam olam gul!

10. Matndagi eng ko‘p qatnashgan harfni topuvchi dastur tuzing. Sonini va harflarni matnda berilgan tartib bo‘yicha chiqaring.

Kiruvchi	Chiquvchi
Mana senga olam olam gul! Havo yaxshi.	7 a
Bugun imtixon. Demak, bayram ekan.	4 a

11. Matndagi eng kam qatnashgan harflarni topuvchi dastur tuzing. Sonini va harflarni matnda berilgan tartib bo‘yicha chiqaring.

Kiruvchi	Chiquvchi
Mana senga olam olam gul! Havo yaxshi.	1 n i x y
Bugun imtixon. Demak, bayram ekan.	1 g t x o D y r

12. So‘z berilgan. Shu so‘zni palindromlikka tekshiring. Boshidan va teskarisiga o‘qilganda bir xil bo‘lsa bu palindrom bo‘ladi. Masalan, aziza. Dasturingiz kiritilgan so‘z palindrom bo‘lsa “Yes”, aks holda “No” chiqarsin.

Kiruvchi	Chiquvchi
kiyik	Yes
aziza	Yes
aziz	No

13. Berilgan so‘zga nechta harf qo‘shsa palindrom bo‘ladi. Shu sonni chiqaring. Masalan,

Kiruvchi	Chiquvchi
aziz	1
olim	3
aslo	3

14. Berilgan so‘z palindrom bo‘lishi uchun qanday bo‘lishi kerak. Masalaga e’tibor bering (1354).

Kiruvchi	Chiquvchi
Aziz	AzizA
Olim	OlimilO
OnLine	OnLineniLnO
iyiK	KiyiK

15. Matnli fayldagi so‘zlarni ulardagi simvollar soni bo‘yicha kamayish va o‘sish tartibida chiqaring. Bunda satr uzunligi 200 va har bir so‘zning uzunligi 10 bo‘lishi mumkin. Masalan,

Kiruvchi	Chiquvchi
2014 yil Sog‘lom bola yili	Sog‘lom 2014 bola yili yil

16. Matnli fayldagi so‘zlar berilgan. Shu so‘zlardagi simvollar soni beshdan oshganlarini oxirgi uchta simvolini “$” bilan almashtiring. Masalan,

Kiruvchi	Chiquvchi
big	big
mouses	mou$
industry	indust$
dastur	das$
klaviatura	klaviat$

17. Matnli fayldagi so‘zlardan faqatgina boshi va oxiridagi simvollari bir xil bo‘lmagan sozlarni qoldiring (Katta va kichik harflar farq qilmaydi). Agarda bunday so‘z bo‘lmasa “NO” chiqaring.

Kiruvchi	Chiquvchi
Mana senga olam olam gul!	NO
Toshkent O‘zbekiston poytaxti.	O‘zbekiston poytaxti.
Mening Malibu rusumli mashinam bor.	Mening Malibu rusumli bor.

18. Berilgan so‘zdagi katta va kichik harflar nisbatini toping. Avval katta harflarni keyin kichik harflarni foizini nuqtadan keyin ikkita aniqlikda chiqaring.

Kiruvchi	Chiquvchi
Ola!	25.00 50.00
alom	20.00 80.00
ToshkenT	25.00 75.00
Toshkent	12.50 87.50

19. Berilgan matndagi so‘zlarning o‘rtacha uzunligini toping.

Kiruvchi	Chiquvchi
Mana senga olam olam gul	4.00

Toshkent O‘zbekiston poytaxti	9.00
Tashkent capital of Uzbekistan	6.75

20. “One” va “puton” bir-biri bilan gaplashyapdi. “One” "out" va "output" so‘zlarini va “puton” so‘zini gapira oladi. “Puton” esa "in", "input" va "one" so‘zlarini gapira oladi. Va ular bir birini probelsiz yozilgan ushbu so‘zlar bilan bemalol tushunishadi. Sizga N ta satrda ularning dialogi berilgan. Shu dialoglardan qaysi biri dialog bo‘lishini aniqlang. Agar berilgan dialog faqat “one” va “puton” ning so‘zlaridan iborat bo‘lsa “YES” aks holda “NO” chiqaring (1102). Masalan,

Kiruvchi	**Chiquvchi**
6 puton inonputin oneputonininputoutoutput oneininputwooutoutput outpu utput	YES NO YES NO NO NO

21. Siz maqolalarni qabul qilyapsiz. Maqolalardagi senzuraga oid so‘zlarni tekshirishingiz kerak. Berilgan senzuraga oid so‘zlarni maqoladan qidiring. Birinchi satrda n ($1 \le n \le 10000$) soni berilgan. Keyingi n ta satrda esa senzuraga oid so‘zlar berilgan. Undan keyin esa maqoladagi satrlar soni m berilgan. Birinchi marta topilgan senzuraga oid so‘zni satrini va o‘rnini chiqaring. Agar bunday so‘zlar topilmasa, “NO” chiqaring (1269). Masalan,

Kiruvchi	**Chiquvchi**
5 dear sweetie angel dream baby 8 Had I the heavens' embroidered cloths, Enwrought with golden and silver light,	6 33

The blue and the dim and the dark cloths Of night and light and the half-light, I would spread the cloths under your feet: But I, being poor, have only my dreams; I have spread my dreams under your feet; Tread softly because you tread on my dreams.	

22. Berilgan matndagi takrorlanuvchi so‘zlarni olib tashlang.

Kiruvchi	**Chiquvchi**
Mana senga olam olam gul.	Mana senga olam gul.
I am going at home. They are going at home too.	I am going at home. They are home too.

23. Berilgan matndagi juft simvolli so‘zlarning birinchi yarmini chiqaring.

Kiruvchi	**Chiquvchi**
Mana senga olam olam gul.	Ma senga ol ol gul.
I am going at home. They are going at home too.	I a going a ho. Th are going t ho too.

24. Berilgan matndagi juft simvolli so‘zlarning ikkinchi yarmini chiqaring.

Kiruvchi	**Chiquvchi**
Mana bugun bayram.	na bugun ram.
I am going at home. They are going at home too.	I m going t me. ey are t me too.

25. Berilgan matndagi juft simvolli so‘zlarning birinchi va ikkinchi yarimlarini o‘rnini almashtiring.

Kiruvchi	**Chiquvchi**
Mana bugun bayram.	naMa bugun rambay.
I am going at home. They are going at home too.	I ma going ta meho. eyTh are ta meho too.

26. Berilgan matndagi juft simvolli so‘zlarning birinchi va ikkinchi yarimlarini o‘rnini almashtiring.

Kiruvchi	**Chiquvchi**
Mana bugun bayram.	naMa bugun rambay.
I am going at home. They are going at home too.	I ma going ta meho. eyTh are ta meho too.

27. Berilgan matndagi toq simvolli so‘zlarni teskari tartibda chiqaring.

Kiruvchi	Chiquvchi
Mana bugun bayram.	Mana nugub bayram
I am going at home. They are going at home too.	I am gniog at home. They era gniog at home oot.

28. Berilgan matnda necha xil unli tovush qatnashayotganligini aniqlovchi dastur tuzing.

Kiruvchi	Chiquvchi
Mana bugun bayram.	2
Bugun rosa qiyin kun bo‘ldi.	4

29. Berilgan matnda necha xil undosh tovush qatnashayotganligini aniqlovchi dastur tuzing. Katta va kichik harflar farq qilmaydi.

Kiruvchi	Chiquvchi
Mana bugun bayram.	6
Bugun rosa qiyin kun bo‘ldi.	10

30. Berilgan matnda barcha probellarni “_” bilan almashtiring.

Kiruvchi	Chiquvchi
Mana bugun bayram.	Mana_bugun_bayram.
Bugun rosa qiyin kun bo‘ldi.	Bugun_rosa_qiyin_kun__bo‘ldi.

20-AMALIY MASHG’ULOT

MAVZU: VISUAL C++NING GRAFIK IMKONIYATLARI. VISUAL C++NING GRAFIK IMKONIYATLARI (GRAFIK HOLAT, TASVIRLARNI VA FUNKSIYA GRAFIKLARINI QURISH). CHART KOMPONENTLARI.

Ishning maqsadi

TextBox va **dataGridView1** komponetalaridan foydalanib, jadvalli ma’lumotlar tiplari yordamida turli Diagrammalar yaratuvchi vizual dastur yaratish.

Jadval tipidagi ma’lumotlar yaratish uchun **DataTable** tipidan foydalaniladi. Bu tipda yaratilgan o‘zgaruvchi ustun va satrlardan tashkil topadi. Jadval ko‘rinishidagi ma’lumotlar asosida grafik Diagrammalar qurish oson bo‘ladi. Formaning grafik imkoniyatlarini o‘zida jamlagan komponenta bu **Chart** komponentasidir. Ushbu komponenta yordamida jadvalli ma’lumotlarni ekranga grafik Diagrammalar ko‘rinishida tasvirlash mumkin. Ushbu laboratoriya ishini mazmunini har oyda sotib olingan mahsulotlar miqdorlarini jadvallarda shakillantirib, grafik Diagramma ko‘rinishida tasvirlovchi vizual dastur yaratish orqali tushintirib o‘tamiz. Topshiriqdagi variantingiz

masalasining dasturini tuzishda ushbu dastur na’muna sifatida hizmat qiladi. Lekin Diagrammalarni ko‘rinishlari va turlari har xil bo‘lganligi uchun ushbu turlardan ham foydalanib tuzilsa dasturimizning tashqi ko‘rinishlari chiroyli va ko‘rgazmali bo‘ladi.

Topshiriqlar

Ma’ruza va amaliyot darslarida olingan ma’lumotlarga asoslanib, o‘z variantingiz uchun quyidagilarni bajaring:

a) 1- jadvaldan o‘z variantini ko‘chirib oling;

b) Quyidagi namuna asosida o‘z variantingizdagi masalalarni bajaring.

Har-bir variantning 1- formasida 2 ta masalani yechish dasturini tuzing.

1-jadval.

№	Masala sharti
1	**1-formada** ishlab chiqarish do‘koniga kiruvchi maxsulotlarni xajmi, ularga to‘lanadigan pul miqdorini haftalik Diagrammasini va **2-formada** oylar kesimidagi Diagrammasini yaratuvchi vizual dastur tuzing.
2	**1-formada** ishlab chiqarish do‘konining ishchilariga to‘lanadigan usta xaqlarni summasini oylar kesimidagi Diagrammasini va **2-formada** esa sof daromatni oylar kesimidagi Diagrammasini yaratuvchi vizual dastur tuzing.
3	**1-formada** kompyuter qurilmalarninig narxlari kesimidagi Diagrammasini va **2-formada** esa sifat kesimidagi Diagrammasini yaratuvchi vizual dastur tuzing.
4	**1-formada** gurhdagi talabalarning fanlardan olgan baholari kesimidagi Diagrammasini va **2-formada** esa fanlarga ajratilgan soatlar kesimidagi Diagrammasini yaratuvchi vizual dastur tuzing.
5	**1-formada** Antivirus dasturlarining narxlari kesimidagi Diagrammasini va **2-formada** esa sifat kesimidagi Diagrammasini yaratuvchi vizual dastur tuzing.
6	**1-formada** Kopyuter qurilmalarini sotuvchi do‘konlarning narxlari kesimidagi Diagrammasini va **2-formada** esa xajmi kesimidagi Diagrammasini yaratuvchi vizual dastur tuzing.
7	**1-formada** NoteBook larning parametrlari kesimidagi Diagrammasini va **2-formada** esa NoteBook ishlab chiqaruvchi firmalarning ishlab chiqarish ko‘rsatgichlari bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
8	**1-formada** Guruh talabalarining yoshlari bo‘yicha Diagrammasini va **2-formada** esa S++da dasturlash fanidan olgan baholari bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
9	**1-formada** Ob-havoning namlik miqdorini xafta kunlari bo‘yicha Diagrammasini va **2-formada** esa issiqlik darajalari bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
10	**1-formada** Ishlab chiqarish fabrikasida tikuv mashinalariga sarflanadigan xarajatlarni oylar bo‘yicha Diagrammasini va **2-formada** esa tushumni va sof foydani hafta kunlari bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
11	**1-formada** AQSh valyuta kursini bir haftalik ko‘rsatgichlarini sumga nisbatan Diagrammasini va **2-formada** esa rublga nisbatan Diagrammasini yaratuvchi vizual dastur tuzing.

12	**1-formada** Bozorda guruch narxini oylar bo‘yicha Diagrammasini va **2-formada** esa shakar narxlarini oylar bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
13	**1-formada** Supermarketda xaridorlar kelishini hafta kunlari bo‘yicha Diagrammasini va **2-formada** esa sotilgan maxsulotlarni xajmi bo‘yicha oylarga nisbatan Diagrammasini yaratuvchi vizual dastur tuzing.
14	**1-formada** Toshkentdagi kinoteatrlarga keluvchi tomoshabinlar sonini hafta kunlari bo‘yicha Diagrammasini va **2-formada** esa qaysi kinolarga ko‘proq qiziqishlar bo‘layotganligini aniqlovchi Diagrammasini yaratuvchi vizual dastur tuzing.
15	**1-formada** Toshkentdagi serqatnov ostanovkalarda qaysi raqamdagi avtobuslarga talablar ko‘proq bo‘layotganligini aniqlovchi Diagrammasini va **2-formada** esa qaysi raqamdagi avtobuslar oraliq masofasi eng kam bo‘layotganlarini kunlik Diagrammasini yaratuvchi vizual dastur tuzing.
16	**1-formada** O‘zbekistonda qaysi oynai jaxon telekanallarini ko‘proq tamosha qilinayotgnligini aniqlovchi Diagrammasini va **2-formada** esa qaysi teledasturlarga tomoshabinlardan qiziqish ko‘p bo‘layotganlarini anqlovchi oylik Diagrammasini yaratuvchi vizual dastur tuzing.
17	**1-formada** TATUning fakultetlaridagi talabalar soni bo‘yicha ko‘rsatgichlar Diagrammasini va **2-formada** esa abuturentlardan qaysi fakultetlarga ko‘proq qiziqishlar bo‘layotganligini bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
18	**1-formada** TATUda S++dasturlash tiliga bo‘lgan qiziqishlarni fakultetlar bo‘yicha Diagrammasini va **2-formada** esa S++ da dasturlash fanidan talabalarni reyting ko‘rsatgichlarini yo‘nalishlar bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
19	**1-formada** Interaktiv hizmatlar ko‘rsatuvchi veb saytlarga bo‘lgan qiziqishlarni oylar bo‘yicha Diagrammasini va **2-formada** esa elektron pochta qutilaridan foydalanish ko‘rsatkichlari bo‘yicha Diagramma yaratuvchi vizual dastur tuzing.
20	**1-formada** Toshkentda oylik yo‘l chiptalarini sotilishi xajmini oylar bo‘yicha Diagrammasini va **2-formada** esa oylik yo‘l chiptalarinining qaysi turiga ko‘proq talab bo‘layotganlik ko‘rsatkichlari bo‘yicha Diagramma yaratuvchi vizual dastur tuzing.
21	**1-formada** TATU kutubxonalarida qanday turdagi kitoblarga bo‘lgan qiziqishlar Diagrammasini va **2-formada** esa qaysi kutbxona bo‘limlariga bo‘layotganlik qiziqishlari bo‘yicha Diagramma yaratuvchi vizual dastur tuzing.
22	**1-formada** Universitet bitiruvchilarining sifat ko‘rsatkichlarini yillarga nisbatan Diagrammasini va **2-formada** esa qaysi yo‘nalishdagi bituruvchilarga bo‘lgan qiziqishlarni firmalar bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
23	**1-formada** Shakar narxining o‘zgarishlarini oylar kesidagi Diagrammasini va **2-formada** esa oziq yog‘i narxining oylarga nisbatan ko‘tarilishi va pasayishi ko‘rsatgichlarini oylar bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
24	**1-formada** do‘konda CocaCola ichimligidan tushadigan sof foydani kunlik Diagrammasini va **2-formada** esa do‘konlardan undiriladigan soliqlarni topiladigan sof daromatga nisbatan Diagrammasini yaratuvchi vizual dastur tuzing.
25	**1-formada** Guruh talabalarini ON va JN natijalari Diagrammasini va **2-formada** esa YaN Diagrammasini yaratuvchi vizual dastur tuzing.
26	**1-formada** Bozorlarda go‘sht va guruch maxsulotlarining narxlarni oylar bo‘yicha o‘sish ko‘rsatgichlari Diagrammasini va **2-formada** esa sut maxsulotlarni oylar bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.

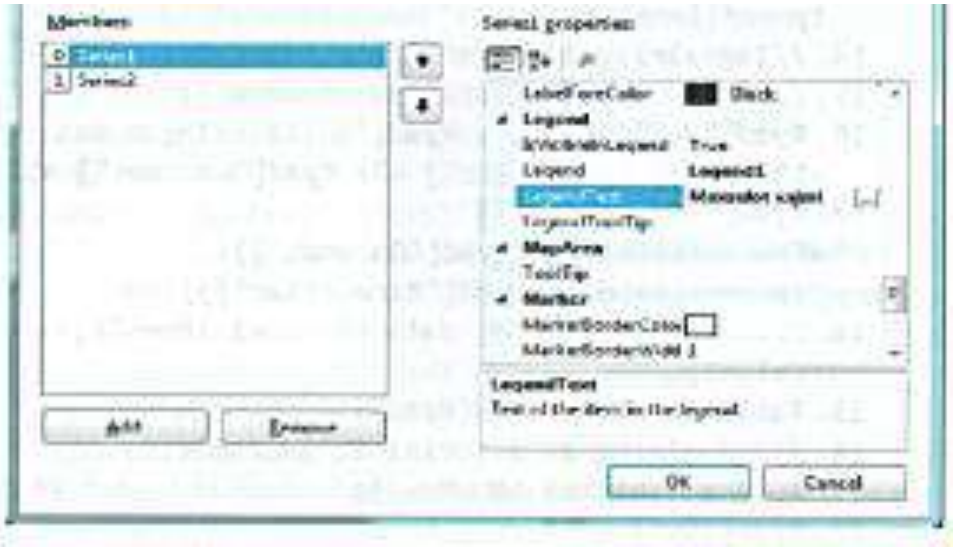
27	**1-formada** Zargarlik do‘konida tillaning narxini oylar bo‘yicha o‘sish ko‘rsatgichlari Diagrammasini va **2-formada** esa kumushnining narxini oylar bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
28	**1-formada** Toshkentda oylik yo‘l chiptalarining sotilish sonini oylar bo‘yicha Diagrammasini va **2-formada** esa internet kartochkalarini sotilishi xajmini oylar bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
29	**1-formada** Ishlab chiqarish korxonasida maxsulot ishlab chiqarish uchun sarflanadigan xarajatlarni haftalik Diagrammasini va **2-formada** esa sof foydani xajmini oylar bo‘yicha Diagrammasini yaratuvchi vizual dastur tuzing.
30	**1-formada** Firmada har oyda sotib olingan mahsulotlar miqdorlarini va sof daromatini oylik Diagrammasini yaratuvchi vizual dastur tuzing.

Jadvalli ma’lumotlar asosida va Chart komponentasidan yordamida diagrammalar yaratish

Ishning maqsadi. **Textbox** va **dataGridViev** komponentalaridan foydalanib, jadvalli ma’lumltlar tiplari yordamida turli diagrammalar yaratuvchi vizual dasturlar yaratish.

Jadval tipidpgi ma’lumotlar yaryataish uchun **DataTable** tipidan foydalaniladi. Bu tipda yaratilgan o‘zgaruvchi ustun va satrlardan tashkil topadi. Jadval ko‘rinishdagi ma’lumotlar asosida grafik diagrammalar qurish oson bo‘ladi. Formaning grafik imkoniyatlari o‘zida jamlangan komponenta bu **Chart** komponentasidir. Ushbu komponenta yordamida jadvalli ma’lumotlarnini ekranga grafik diagrammalar ko‘rinisha tasvirlash mumkin. Ushbu Amaliyot ishini mazmunini o‘ar oyda sotib olingan mahsulotlar miqdorini jadvallarda shakllantirib, grafik diagrammalar ko‘rinisha tasvirlovchi vizual dastur yaratish orqali tushuntirib o‘tiladi. Berilgan topshiriqlarni dasturini tuzishda ushbu dasturdan namuna sifatida foydalanish mumkin.

1-qadam. Formaga **Chart** komponentasi joylashtiriladi. Uning **Series** xossasiga kirib, 2-**Series** qo‘shiladi. **Series1** tanlanadi va uning **legen**dText xossasiga “Maxsulot hajmi” matni yoziladi. **Series2** tanlanadi va uning **legen**dText xossasiga “Sof daromad” matni yoziladi. Ushbu **Series** qo‘shish oynasi 2-rasmda keltrilgan.



2-rasm. **Series** qo‘shish oynasi

2-qadam. Formada **dataGridViev, listBox df button** komponentalari joylashtiriladi. Ularning hech qanday xossasi o‘zgartirilmaydi.

3-qadam. Formaning **load** hodisasining kodlari quyidagicha:

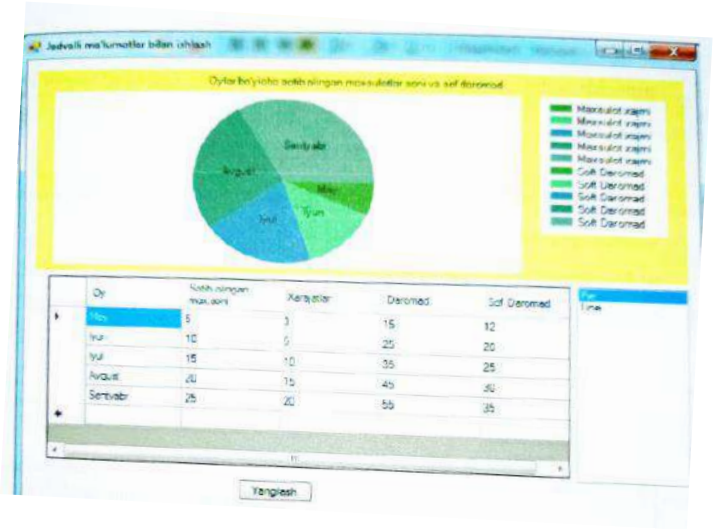
```
DataTable ^ Tablitsa;
private: System::Void Form1_Load(System::Object^
sender, System::EventArgs^  e) {chart1->Titles-
>Add("Oylar bo'yicha sotib olingan maxsulotlar soni va
sof daromat");
listBox1->Items->Add("Pie");
listBox1->Items->Add("Line");
this->Text = "Grafik ma'lumotlar";
Tablitsa = gcnew DataTable();
//Ushbu jadvalda 2 ta ustun "Oy"va "Sotib olingan
//max.soni":
Tablitsa->Columns->Add("Oy", String::typeid);
// V C#: Tablitsa.Columns.Add("Mesyats",
//typeof(String));
//2-ustun bo'yicha ma'lumotlar long tipida bo'lasi:
// V C#: Tablitsa.Columns.Add("Ob'em prodaj",
typeof(long));
//Jadvalning 1- qatorini to'ldiramiz:
DataRow ^ Ryad = Tablitsa->NewRow();
Ryad["Oy"] = "May"; Ryad["Sotib olingan max.soni"]
=10; Ryad["Xarajatlar"] =3; Ryad["Daromat"] =15;
Ryad["Sof daromat"]
=Convert::ToInt64(Ryad["Daromat"])-
Convert::ToInt64(Ryad["Xarajatlar"]);
//Convert::ToInt64( dataGridView1->Rows[0]->Cells[0]-
>Value);
Tablitsa->Rows->Add(Ryad);
//Jadvalning 2- qatorini to'ldiramiz:
Ryad = Tablitsa->NewRow();
Ryad["Oy"] = "Iyun"; Ryad["Sotib olingan max.soni"] =
20; Ryad["Xarajatlar"] =5; Ryad["Daromat"] =25;
Ryad["Sof daromat"]
=Convert::ToInt64(Ryad["Daromat"])-
Convert::ToInt64(Ryad["Xarajatlar"]);
Tablitsa->Rows->Add(Ryad);
//3- qatorni qo'shamiz:
Ryad = Tablitsa->NewRow();
Ryad["Oy"] = "Iyul"; Ryad["Sotib olingan max.soni"] =
30; Ryad["Xarajatlar"] =10; Ryad["Daromat"] =35;
Ryad["Sof daromat"]
=Convert::ToInt64(Ryad["Daromat"])-
Convert::ToInt64(Ryad["Xarajatlar"]);
Tablitsa->Rows->Add(Ryad);
//4- qatorni qo'shamiz:
```

4-qadam. Jadvaldan kiritilgan qiymatlar natijaga darhol tasir qilish uchun **dataGridViev1** ning **CellEndEdit** hodisasi dio() funksiyasi chaqiriladi.

5-qadam. **ListBox1** da diagrammani ko‘rinishlarini o‘zgartirish mumkin. Buning uchun **ListBox** ning **SelectedIndexChanged** hodisasining kodlari quyidagicha bo‘lishi kerak.

```
63. Ryad["Xarajatlar"] =dataGridView1->Rows[0]->Cells[2]-
    >Value; Ryad["Daromat"] =dataGridView1->Rows[0]-
    >Cells[3]->Value;
64. Ryad["Sof daromat"] =Convert::ToInt64(dataGridView1-
    >Rows[0]->Cells[3]->Value) -
    Convert::ToInt64(dataGridView1->Rows[0]->Cells[2]-
    >Value);
65. Tablitsa->Rows->Add(Ryad);
66. Ryad = Tablitsa->NewRow();
67. Ryad["Oy"] = dataGridView1->Rows[1]->Cells[0]->Value;
    Ryad["Sotib olingan max.soni"] = dataGridView1->Rows[1]-
    >Cells[1]->Value;
68. Ryad["Xarajatlar"] =dataGridView1->Rows[1]->Cells[2]-
    >Value; Ryad["Daromat"] =dataGridView1->Rows[1]-
    >Cells[3]->Value;
69. Ryad["Sof daromat"] =Convert::ToInt64(dataGridView1-
    >Rows[1]->Cells[3]->Value) -
    Convert::ToInt64(dataGridView1->Rows[1]->Cells[2]-
    >Value);
70. Tablitsa->Rows->Add(Ryad);
71. Ryad = Tablitsa->NewRow();
72. Ryad["Oy"] = dataGridView1->Rows[2]->Cells[0]->Value;
    Ryad["Sotib olingan max.soni"] = dataGridView1->Rows[2]-
    >Cells[1]->Value;
73. Ryad["Xarajatlar"] =dataGridView1->Rows[2]->Cells[2]-
    >Value; Ryad["Daromat"] =dataGridView1->Rows[2]-
    >Cells[3]->Value;
74. Ryad["Sof daromat"] =Convert::ToInt64(dataGridView1-
    >Rows[2]->Cells[3]->Value) -
    Convert::ToInt64(dataGridView1->Rows[2]->Cells[2]-
    >Value);
75. Tablitsa->Rows->Add(Ryad);
76. Ryad = Tablitsa->NewRow();
77. Ryad["Oy"] = dataGridView1->Rows[3]->Cells[0]->Value;
    Ryad["Sotib olingan max.soni"] = dataGridView1->Rows[3]-
    >Cells[1]->Value;
78. Ryad["Xarajatlar"] =dataGridView1->Rows[3]->Cells[2]-
    >Value; Ryad["Daromat"] =dataGridView1->Rows[3]-
    >Cells[3]->Value;
79. Ryad["Sof daromat"] =Convert::ToInt64(dataGridView1-
    >Rows[3]->Cells[3]->Value) -
    Convert::ToInt64(dataGridView1->Rows[3]->Cells[2]-
    >Value);
```

Dasturda diagrammaning ko‘rinishi quyidagicha:



3-rasm. Diagrammaning sozlash oynasi

Nazorat savollari

1. CHiziqlar va shakllar chizish uchun maxsus qanday usullarni o‘rganish lozim
2. DrawLine, DrawArc, DrawClosedCurve, DrawPolygon va DrawRectangle o‘z ichiga oladigan sinfni ayting.
3. Grafika sinfi xususiyatlari sanab bering.
4. grafika sahifa koordinatalarini uchun ishlatiladigan o‘lchov birligi sozlash xususiyatini ayting.
5. GraphicsUnit qiymatlari sanab bering.
6. - butun chizma yuzasini tozalaydi va belgilangan fon rangi bilan to‘ldiradigan usulning nomini ayting.
7. Bu funksiya berilgan parametrlar asosida tasvirni belgilangan joyda asl kattaligidan foydalanib chizadi. Funksiya nomini ayting.
8. CHiziq chizishni qanday amalga oshiriladi.
9. Uchburchak chizishni qanday amalga oshiriladi
10. Ellipis chizishni qanday amalga oshiriladi
11. Turli shakllarni bo‘yyashni qanday amalga oshiriladi
12. Ko‘p burchaklarni chizishni qanday amalga oshiriladi
13. Hodisalar orqali shakl chizishni qanday amalga oshiriladi
14. Rasmlarni o‘zgartirishni qanday amalga oshiriladi
15. Matnlarni tasvir kabi joylashtirishni qanday amalga oshiriladi
16. Matnli rasmlarni yaratishni qanday amalga oshiriladi
17. Chart komponenta xususiyati va hodisalari haqida gapirib bering.
18. BorderSkin nima.
19. Qaysi xususiyat yordamida komponentaning infografikani ko‘rsatadigan shaklini tanlash mumkin
20. Qaysi xususiyat asosiy bo‘lib, infografikaning qiymatlarini belgilovchi, har bir qiymat tegishliligini bildiradi
21. ChartType infografikaning turlarini belgilash uchun xizmat qiladi. Uning qiymatlarini bilasizmi.

22. Bir infografikada nechtagacha gistogramma joylashtirish mumkin.
23. Chart komponentasi yordamida foydlanuvchining ixtiyoriy ma’lumotlarini infografikasini yaratish mumkinmi
24. Nimalar dastlab bepul plagin-smaylik sifatida ishlatilgan va endi Visual Studio tarkibiga kiritilgan
25. Visual Basic Power Packs komponentalarini sanab bering.
26. Visual Basic Power Packs komponentalarining asosiy vazifalari nimadan iborat.
27. Komponentaning chegara rang qalinligini o‘rnatadi va qiymatlari sonlardan iborat bo‘lgan xususiyat nomini ayting.
28. SHakllarning FillColor, FillGradientColor, FillGradientStyle, FillStyle xususiyatlaridan foydalanib, niam qilish mumkin.
29. Tugmning ko‘rinishini o‘zgartirish uchun nima amallar bajario‘ kerak.
30. Rasmlarni harakatlantirishda oqimlardan qanday foydalanamiz.

Amaliyot mashg‘ulotini bajarish bo‘yicha variantlar

1. Berilgan jadvaldagi topshiriqlardan o‘z variantingizni ko‘chirib oling.
2. Namunadagi dastur asosida o‘z variantingiz masalalarini dasturini tuzing.

1-jadval

№	Topshiriqlar
1.	1-formada ishlab chiqaruvchi tashkilotning ishchilariga to‘lanadigan oylik maoshlari summasini oylar kesimidagi diagrammasini va 2-formada esa sof daromadini oylar kesimidagi diagrammasini yaratuvchi vizual dastur tuzing.
2.	1-formada kompyuter qurilmalarining narxlari kesimidagi diagrammasini va 2-formada esa sifat kesimidagi diagrammasini yaratuvchi vizual dastur tuzing.
3.	1-formada guruxdagi talabalarning fanlardan olgan baxolari kesimidagi diagrammasini va 2-formada esa fanlarga ajratilgan soatlar kesimidagi diagrammasini yaratuvchi vizual dastur tuzing.
4.	1-formada Antivirus dasturiningnarxlari kesimidagi diagrammasini va 2-formada esa sifat kesimidagi diagrammasini yaratuvchi vizual dastur tuzing.
5.	1-formada Do‘konda kompyuter qurilmalarining narxlari kesimidagi diagrammasini va 2-formada esa hajmi kesimidagi diagrammasini yaratuvchi vizual dastur tuzing.
6.	1-formada gurux talabalarning yoshi bo‘yicha diagrammasini va 2-formada esa Dasturlash fanidanolgan baxolari bo‘yicha diagrammani yaratuvchi vizual dastur tuzing.
7.	1-formada Ob-havoning namlik miqdorini hafta kunlari bo‘yicha diagrammasini va 2-formada esa issiqlik darajalari bo‘yicha diagrammasini yaratuvchi vizual dastur tuzing
8.	1-formada Ishlab chiqaish fabrikasida tikuv mashinalariga sarf-xarajatlarning oylar bo‘yicha diagrammasini va 2-formada esa olingan foyda va tushumlarni hafta kunlari bo‘yicha diagrammasini yaratuvchi vizual dastur tuzing
9.	1-formada Bozorda guruch narxlari oylar kesimidagi diagrammasini va 2-formada esa shakar narxlari oylar kesimidagi diagrammasini yaratuvchi vizual dastur tuzing.
10.	1-formada Supermarketda xaridorlar kelishini hafta kunlari bo‘yicha diagrammasini va 2-formada sotilgan mahsulotlarni hajmi bo‘yicha oylar kesimidagi diagrammasini yaratuvchi vizual dastur tuzing.

11.	1-formada Toshkentdagi qaysi yo‘nalishdagi (raqamlari bo‘yicha) avtobuslarda talabalar ko‘proq foydalanilayotganligi bo‘yicha diagrammasini va 2-formada va qaysi raqamdagi avtobusning oraliq masofasi eng uzoq ekanligini kunlik kesimidagi diagrammasini yaratuvchi vizual dastur tuzing.
12.	1-formada O‘zbekistonda oynai jahonning qaysi telekallarini ko‘proq tomosha qilinayotganligini aniqlash bo‘yicha diagrammasini va 2-formada qaysi teledasturga tomoshabinlarda ko‘proq qiziqish bo‘layotganligini aniqlovchi oylar kesimidagi diagrammasini yaratuvchi vizual dastur tuzing.
13.	1-formada TATUning fakultetlaridagi talabalar sonini ko‘rsatish bo‘yicha diagrammasini va 2-formada esa abuturentlardan qaysi fakultetlarga ko‘proq qiziqish bo‘layotganligi bo‘yicha diagrammani yaratuvchi vizual dastur tuzing.
14.	1-formada TATUda Dasturlash faniga bo‘lgan qiziqishlarni fakultetlaridagi bo‘yicha diagrammasini va 2-formada esa Dasturlash fanidan talabalarning yo‘nalishlar kesimida reyting ko‘rsatkichlari bo‘yicha diagrammani yaratuvchi vizual dastur tuzing.
15.	1-formada TATU kutubxonalarida qanday turdagi kitoblarga bo‘lgan qiziqishlar bo‘yicha diagrammasini va 2-formada esa kutubxonaning qaysi bo‘limlarida talabalarning aktivlik darajasi bo‘yicha diagrammani yaratuvchi vizual dastur tuzing.
16.	1-formada Universitet bitiruvchilarining ishga joylashishini yillar bo‘yicha diagrammasini va 2-formada esa qaysi yo‘nalish bitiruvchilariga bo‘lgan talablar yillar kesimida diagrammasini yaratuvchi vizual dastur tuzing.
17.	1-formada SHakar narxining o‘zgarishini oylar kesimida diagrammasini va 2-formada esa yog‘ narxini oylarga nisbatan ko‘tarilishi yoki pasayshini ko‘rsatuvchi diagrammasini yaratuvchi vizual dastur tuzing.
18.	1-formada gurux talabalarining ON va JN natijalari bo‘yicha diagrammasini va 2-formada esa YAN natijalari kesimida diagrammasini yaratuvchi vizual dastur tuzing.
19.	1-formada go‘sht va guruch mahsulotlarini narxining o‘zgarishini oylar kesimida diagrammasini va 2-formada esa sut mahsulotlarini narxining oylarga nisbatan ko‘tarilishi yoki pasayshini ko‘rsatuvchi diagrammasini yaratuvchi vizual dastur tuzing.
20.	1-formada zargarlik do‘konida tillaning narxini o‘zgarishini oylar kesimida diagrammasini va 2-formada esa kumushni narxining oylarga nisbatan ko‘tarilishi yoki pasayshini ko‘rsatuvchi diagrammasini yaratuvchi vizual dastur tuzing.

№21-AMALIY MASHG’ULOT

MAVZU: KICHIK LOYIHALARNI YARATISH. VISUAL C++ MUHITIDA TURLI SOHAGA OID MASALALARNI YECHISH UCHUN (MINI) KICHIK LOYIHALARNI LOYIHALASH USULLARI. KICHIK LOYIHALAR ALGORIMLARINI YOZISH USULLARI. VISUAL C++ MUHITIDA HISOBOT SHAKLINI TAYYORLASH VA CHOP ETISH ELEMENTLARI.

MAVZU: KICHIK LOYIHALAR YARATISH: ODDIY MATN MUXARRIRI. FAYL OCHISH VA SAQLASH. FORMANING CLOSING XODISASI

Ishning maqsadi

TextBox, MenuStrip, OpenFileDialog va **SaveFileDialog** komponetalaridan foydalanib, matnli va **rtf** kentgaytmali fayllarga qayta ishlov beruvchi vizual dastur yaratish.

Bizga ma’lumki matnli fayllarga ma’lumotlarni yozuvchi va o‘quvchi maxsus funksiyalvr mavjud. Shunday bo‘lsa ham **Visual C++** muxarririning ham maxsus shunday funksiyalar mavjud. Ushbu funksiyalarni vizual komponentalar bilan ishlatib, kichik matn muharririrni yaratish mumkin. Ushbu laboratoriya ishida fallarga ishlov berishning bir necha usullari bilan tanishib chiqiladi.

Topshiriqlar

Ma’ruza va amaliyot darslarida olingan ma’lumotlarga asoslanib, o‘z variantingiz uchun quyidagilarni bajarinig:

a)1- jadvaldan o‘z variantingizni ko‘chirib oling;

b) Quyidagi namuna asosida o‘z variantingizdagi masalalarni bajaring.

Har-bir variantning **1- formasida 2 ta masalani yechish dasturini tuzing.**

1-jadval.

№	Masala sharti
1.	**1.** N va K butun musbat sonlar va fayl nomi berilgan. Yangi matnli fayl hosil qilinsin va unga N ta satr va har bir satr K ta "*" (yulduzcha) belgisidan iborat bo‘lsin. 2. 10 ta raqamdan iborat S satr va shifrlangan matnli fayl berilgan. sezar usulidagi algoritm bo‘yicha shifrlangan matnni deshifrlovchi programma tuzilsin. **2-formada matematika fanidan 2 ta variantli test dasturini tuzing.**
2.	**1.** N (0<N<27) butun son va fayl nomi berilgan. Berilgan nomdagi matnli fayl hosil qilinsin va unga: birinchi satri "a" kichik lotin harfi, ikkinchisiga"ab", uchinchisiga "abc" va h.k satrlarni saqlovchi N ta satr yozilsin. **2.** 10 ta raqamdan iborat S satr va lotin harflaridan iborat matnli fayl berilgan. Matnli fayli quyidagicha shifrlang: Matnli fayl satrining K - belgisini, shu belgining kodiga S satridagi K -raqamini qo‘shishdan hosil bo‘lgan kod belgisiga almashtiring. Agar K=11 bo‘lsa, ya’na satrning birinchi raqamidan boshlang. **2-formada fizika fanidan 3 ta variantli test dasturini tuzing.**
3.	1. N (0<N<27) butun son va fayl nomi berilgan. Berilgan nomdagi fayl hosil qilinsin va unga uzunligi N ga teng bo‘lgan N ta satr quyidagicha yozilsin; K-nomerdagi satr (K=1N) katta lotin harflarining boshlang‘ich K ta harfini va undan o‘ngda "*" belgisidan iborat bo‘lsin. Masalan N**=4** uchun fayl quyidagi satrlardan iborat bo‘lishi kerak. "A***", "AB**", "ABC*", "ABCD". 2. Matnli fayl berilgan. Undagi har bir uchragan kichik lotin harflarining uchrashlar miqdori sanalsin va tarkibi quyidagi ko‘rinishda bo‘lgan matnli fayl hosil qilinsin "<harf>-<uchrashlar soni>"(masalan, "a-25"). Matnda uchramagan harflar hisobga olinmasin. Satrlarni harflarning

	uchrash sonini kamayish bo‘yicha, teng sondagi uchrashlarni esa ularning kodlari bo‘yicha o‘sish tartibida joylashtirilsin. **2-formada Kimyo fanidan 4 ta variantli test dasturini tuzing.**
4.	1. Matnli fayl berilgan. Uning tarkibiga kiruvchi satrlar va belgilar soni (miqdori) chop qilinsin. (satrning oxiri EOLN va EOF fayl oxirlari markerlari belgilarni sanayotganda hisobga olinmasin). 2. Matnli fayl berilgan. Undagi har bir uchragan kichik lotin harflarining uchrashlar miqdori sanalsin va tarkibi quyidagi ko‘rinishda bo‘lgan matnli fayl hosil qilinsin "<harf>-<uchrashlar soni>"(masalan, "a-25"). Matnda uchramagan harflar hisobga olinmasin. Satrlar kodi bo‘yicha o‘sish tartibida joylashsin. **2-formada Ona tili fanidan 2 ta variantli test dasturini tuzing.**
5.	1. Satr va matnli fayl berilgan. S satr fayl oxiriga qo‘shilsin. 2. Matnli fayl berilgan. Matnli fayldagi barcha uchragan belgilarni, probel va tinish belgilariga ega bo‘lgan (takrorlanishsiz) belgili fayl hosil qilinsin. Belgilar kodi bo‘yicha kamayish tartibida joylashsin. **2-formada Ingliz fanidan 3 ta variantli test dasturini tuzing.**
6.	1. Ikkita matnli fayl berilgan. Birinchi fayl oxiriga ikkinchi fayl qo‘shilsin. 2. Matnli fayl berilgan. Matnli fayldagi barcha uchragan belgilarni, probel va tinish belgilariga ega bo‘lgan (takrorlanishsiz) belgili fayl hosil qilinsin. Belgilar kodi bo‘yicha o‘sish tartibida joylashsin. **2-formada Nemis tili fanidan 4 ta variantli test dasturini tuzing.**
7.	1. S satr va matnli fayl berilgan. S satr fayl boshiga qo‘shilsin. 2. Matnli fayl berilgan. Matnli fayldagi barcha uchragan belgilarni, probel va tinish belgilariga ega bo‘lgan (takrorlanishsiz) belgili fayl hosil qilinsin. Belgilar matnda birinchi joylashgan tartibida joylashtirilsin. **2-formada rus tili fanidan 2 ta variantli test dasturini tuzing.**
8.	1. Ikkita matnli fayl berilgan. Birinchi fayl boshiga ikkinchi fayl qo‘shilsin. 2. Matnli fayl berilgan. Matnli fayldagi barcha uchragan tinish belgilariga ega bo‘lgan belgili fayl hosil qilinsin. **2-formada Jismoniy tarbiya fanidan 3 ta variantli test dasturini tuzing.**
9.	1. K butun soni va matnli fayl berilgan. K- nomerdagi satrdan oldin bo‘sh satr qo‘yilsin. Agar bunday nomerli satr mavjud bo‘lmasa, u holda fayl o‘zgartirishsiz qoldirilsin. 2. Butun sonlardan iborat uchta ustunli jadvalga ega bo‘lgan matnli fayl berilgan. Jadvaldagi har bir ustunining boshiga va oxitiga hamda ular orasiga ajratuvchi belgi joylashtirilgan. Jadvaldagi ustunlar kengligi va ularning tekislanishi hamda ajratuvchi belgilar ko‘rinishi

	ixtiyoriy. Boshlang‘ich jadvalning har bir satridagi sonlar yig‘indisiga ega bo‘lgan yangi butun sonlar fayli hosil qilinsin. **2-formada Axborot xavfsizligi fanidan 4 ta variantli test dasturini tuzing.**
10.	1. K butun soni va matnli fayl berilgan. K- nomerdagi satrdan keyin bo‘sh satr qo‘yilsin. Agar bunday nomerli satr mavjud bo‘lmasa, u holda fayl o‘zgartirishsiz qoldirilsin. 2. Haqiqiy sonlardan iborat uchta ustunli jadvalga ega bo‘lgan matnli fayl berilgan. Ustun kengligi, tekislash usullari ixtiyoriy ravishda berilgan, maxsus ajratuvchi belgilarga ega emas. Uchta haqiqiy sonlar fayli hosil qilinsin va har bir fayl jadvalning mos ustunidagi sonlarni o‘zida saqlasin. **2-formada S++da dasturlash fanidan 2 ta variantli test dasturini tuzing.**
11.	1. Matnli fayl berilgan. Undagi barcha bo‘sh satrlar ikkilantirilsin. 2. Matnli fayl berilgan. Har bir satrning birinchi 30 ta belgisi matndan, qolgani esa haqiqiy sonlardan iborat. Boshlang‘ich faylning barcha matn qismiga ega bo‘lgan mantli fayl va boshlang‘ich faylning barcha haqiqiy sonlar qismiga ega bo‘lgan haqiqiy sonlar fayli hosil qilinsin. **2-formada Mexnat ta’lim fanidan 3 ta variantli test dasturini tuzing.**
12.	1. S satr va matnli fayl berilgan. Fayldagi barcha bo‘sh satrlar S satrga o‘zgartirilsin. 2. Butun sonlar fayli va matnli fayl berilgan. Matn faylining har bir satrining oxiriga butun sonlar faylidagi mos sonlari joylashtirilsin. Agar butun sonlar fayli matn faylidan qisqa bo‘lsa, u holda matn faylidagi qolgan satrlar o‘zgartirmasdan qoldirilsin. **2-formada Java dasturlash fanidan 4 ta variantli test dasturini tuzing.**
13.	1. Bo‘sh bo‘lmagan matnli fayl berilgan. Undagi birinchi satr o‘chirilsin. 2. Har bir satrida probellar bilan ajratilgan bir nechta sonlarni tasvirlovchi matnli fayl berilgan (haqiqiy sonlar nol bo‘lmagan kasr qismiga ega). Tarkibida boshlang‘ich faylning barcha butun sonlariga ega bo‘lgan butun sonlar fayli hosil qilinsin. **2-formada Falsafa fanidan 2 ta variantli test dasturini tuzing.**
14.	1. Bo‘sh bo‘lmagan matnli fayl berilgan. Undagi oxirgi satr o‘chirilsin. 2. Har bir satri o‘ng va chap tomonidan bir nechta probellar bialn to‘ldirilgan butun yoki haqiqiy sonlarni tasvirlovchi matnli fayl berilgan. (haqiqiy sonlar nol bo‘lmagan kasr qismiga ega). Butun sonlar miqdori va ularning yig‘indisi chop qilinsin. **2-formada UZDJQ fanidan 3 ta variantli test dasturini tuzing.**
15.	1. K butun soni va matnli fayl berilgan. Undaki K-nomerdagi satr o‘chirilsin. Agar faylda bunday nomerdagi satr mavjud bo‘lmasa, u holda fayl o‘zgartirishsiz qoldirilsin. 2. Har bir satrida probellar bilan ajratilgan bir nechta sonlarni tasvirlovchi matnli fayl berilgan (haqiqiy sonlarning kasr qismi noldan farqli). Boshlang‘ich faylning barcha noldan farqli kasr

	qismiga ega bo‘lgan sonlaridan iborat (taribini o‘zgartirmagan holda) haqiqiy sonlar fayli hosil qilinsin. **2-formada Algoritmlash fanidan 4 ta variantli test dasturini tuzing.**
16.	1. Matnli fayl berilgan. Undagi barcha bo‘sh satrlar o‘chirilsin. 2. Har bir satrida bittadan butun yoki haqiqiy son bo‘lgan matnli fayl berilgan. Sonlar o‘ng va chap tomonidan bir nechta probellar bilan to‘ldirilgan. (haqiqiy sonlarning kasr qismi noldan farqli). Kasr qismi nol bo‘lmagan sonlar miqdori va ularning yig‘indisi chop qilinsin. **2-formada Kompyuter grafikasi fanidan 2 ta variantli test dasturini tuzing.**
17.	1. Ikkita matnli fayl berilgan. Birinchi faylning har bir satridan so‘ng ikkinchi fayldagi mos satrlar qo‘shilsin. Agar ikkinchi fayl birinchi fayldan kalta bo‘lsa, u holda qolgan satrlar o‘zgartirishsiz qoldirilsin. 2. Har bir satrida bittadan butun son bo‘lgan matnli fayl berilgan. Butun sonlar o‘ng va chap tomonidan bir nechta probellar bilan to‘ldirilgan. Shu sonlar miqdori va ularning yig‘indisi chop qilinsin. **2-formada Axborot texnologiyalari fanidan 3 ta variantli test dasturini tuzing.**
18.	1. K butun soni va matnli fayl berilgan. Faylning har bir satridan birinchi k ta belgi o‘chirilsin. (agar satr uzunligi K dan kichik bo‘lsa, u holda satrning hamma belgilari o‘chirilsin). 2. N butun soni va A, V haqiqiy sonlari berilgan. Sin(x) va cos(x) funksiyasining [A, B] oraliqdagi (B-A)/N qadam bilan hosil bo‘luvchi qiymatlari jadvalini saqlovchi matnli fayl hosil qilinsin. Jadval uchta ustundan iborat: x argumentli (8 ta pozitsiya va uning 4 tasi kasr qismi) va sin(x) hamda cos(x) ning qiymatlari (12 ta pozitsiyadan va ulardan 8 tasi kasr qismi). Ustunlar o‘ng tarafdan tekislanadi. **2-formada Oila psixologiyasi fanidan 4 ta variantli test dasturini tuzing.**
19.	1. Matnli fayl berilgan. Fayldagi hamma katta lotin harflari kichik harflarga va aksincha, barcha kichik lotin harflari katta harflarga almashtirilsin. 2. N butun soni va A, V haqiqiy sonlari berilgan. funksiyasining [A,B] oraliqdagi (B-A)/N qadam bilan hosil bo‘luvchi qiymatlari jadvalini saqlovchi matnli fayl hosil qilinsin. Jadval ikkita ustundan iborat: x argumentli (10 ta pozitsiya va uning 4 tasi kasr qismi) va uning qiymatlari (15 ta pozitsiya va ulardan 8 tasi kasr qismi). Ustunlar o‘ng tarafdan tekislanadi. **2-formada Metrologiya fanidan 2 ta variantli test dasturini tuzing.**
20.	1. Matnli fayl berilgan. Undagi barcha ketma-ket kelgan probellar bitta probelga almashtirilsin. 2. Bir xil o‘lchamdagi butun sonlardan iborat ikkita fayl berilgan. Shu sonlardan, kengligi 30 ta belgidan iborat bo‘lgan ikkita ustunga ajratilgan matnli fayl hosil qilinsin. Birinchi ustunda birinchi boshlang‘ich fayl sonlari, ikkinchisida esa ikkinchi boshlang‘ich fayl sonlari

	joylashadi. Matnli fayldagi har bir satrning boshi va oxiriga "|" ajratuvchi (kod 124) qo‘shilsin. Sonlar ustunning o‘ng tarafiga tekislanadi. **2-formada Oliy matematika fanidan 3 ta variantli test dasturini tuzing.**
21.	1. Uchtadan ko‘p bo‘lgan satrga ega bo‘lgan matnli fayl berilgan. Shu fayldan oxirgi uchta satr o‘chirilsin. 2. Bir xil o‘lchamdagi butun sonlardan iborat ikkita fayl berilgan. Shu sonlardan, kengligi 30 ta belgidan iborat bo‘lgan ikkita ustunga ajratilgan matnli fayl hosil qilinsin. Birinchi ustunda birinchi boshlang‘ich fayl sonlari, ikkinchisida esa ikkinchi boshlang‘ich fayl sonlari joylashadi. Matnli fayldagi har bir satrning boshi va oxiriga "|" ajratuvchi (kod 124) qo‘shilsin. Sonlar ustunning o‘ng tarafiga tekislanadi. **2-formada Biznesni boshqarish asoslari fanidan 4 ta variantli test dasturini tuzing.**
22.	1. K (0<K<10) butun son va K ta dan ko‘p bo‘lgan satrga ega matnli fayl berilgan. Shu faylning oxirgi K ta satri o‘chirilsin. 2. K (>25) butun soni va chap tarafdan tekislangan matnli fayl berilgan. Abzas qizil satr (Text26 masalaga qarang) orqali ajratiladi, bo‘sh satrlar esa mavjud emas. Matnni shunday formatlangki, uning kengligi K ta belgidan oshmasin va abzaslarga bo‘linganligini saqlagan holda chap tarafga tekislansin. Satrning oxiridagi probellar o‘chirilsin. Formatlangan matn yangi faylda saqlansin. **2-formada Ma’lumotlar tuzilmasi fanidan 2 ta variantli test dasturini tuzing.**
23.	1. K (0<K<10) butun son va K ta dan ko‘p bo‘lgan satrga ega matnli fayl berilgan. Boshlang‘ich faylning oxirgi K ta elementidan iborat bo‘lgan yangi matnli fayl hosil qilinsin. 2. K (>25) butun soni va chap tarafdan tekislangan matnli fayl berilgan. Matn abzasi bitta bo‘sh satr orqali ajratiladi. Matnni shunday formatlangki, uning kengligi K ta belgidan oshmasin va abzaslarga bo‘linganligini saqlagan holda chap tarafga tekislansin. **2-formada XFX fanidan 3 ta variantli test dasturini tuzing.**
24.	1. Matnli fayl berilgan. Agar abzas bitta yoki bir nechta bo‘sh satrlar bilan ajratilgan bo‘lsa, u holda matndagi abzaslar soni aniqlansin. 2. Chap tomonidan tekislangan matnga ega bo‘lgan matnli fayl berilgan. Abzaslar bitta bo‘sh satr orqali ajratiladi. Har bir bo‘sh bo‘lmagan satrdagi eng oxirgi probeldan boshlab satr so‘zlari orasidagi probellarni qo‘shish orqali matn kenglik bo‘yicha tekislansin (ham chap, ham o‘ng tomondan tekislansin). Matn kengligi 50 ta belgiga teng deb olinsin. **2-formada Telekommunikatsiya fanidan 4 ta variantli test dasturini tuzing.**
25.	1. K butun son va matnli fayl berilgan. Fayldan K-nomerdagi abzas o‘chirilsin (abzas bir-biridan bitta yoki bir nechta bo‘sh satrlar bilan ajratiladi). O‘chirilgan abzasdan oldin va keyin keluvchi bo‘sh satrlar o‘chirilmasin. Agar berilgan nomerdagi abzas mavjud bo‘lmasa u holda fayl o‘zgartirishsiz qoldirilsin.

	2. O‘ng tomonidan tekislangan matnga ega bo‘lgan matnli fayl berilgan. Har bir bo‘sh bo‘lmagan satrning boshlang‘ich probellarini yarmini o‘chirish orqali matn markazga tekislantirilsin. Toq uzunlikka ega bo‘lgan satrlarga, markazlashtirishdan oldin chap tomondan bitta probel o‘chirilsin. **2-formada Aloqa linyalari fanidan 2 ta variantli test dasturini tuzing.**
26.	1. Matnli fayl berilgan. Agar fayl matnidagi har bir abzasning birinchi satri 5 ta probeldan boshlansa ("qizil satr"), u holda shu matndagi abzaslar soni aniqlansin. Abzaslar orasidagi bo‘sh satrlar hisobga olinmasin. 2. Chap tomonidan tekislangan matnga ega bo‘lgan matnli fayl berilgan. Har bir bo‘sh bo‘lmagan satr boshiga kerakli miqdorda probel qo‘shish orqali matnni markazga tekislanilsin. (Matn kengligi 50 ta belgiga teng deb olinsin). Toq uzunlikka ega bo‘lgan satrlarga, markazlashtirishdan oldin chap tomondan probel qo‘shilsin. **2-formada Kompyuter arxitekturasi fanidan 3 ta variantli test dasturini tuzing.**
27.	1. Matnli fayl berilgan. Fayl matnidagi har bir abzasning birinchi satri 5 ta probeldan boshlangan. K butun soni va mantli fayl berilgan. Fayldan K-nomerdagi abzas o‘chirilsin. Abzas qizil satr orqali ajratiladi. 2. Chap tomonidan tekislangan matnga ega bo‘lgan matnli fayl berilgan. Har bir bo‘sh bo‘lmagan satr boshiga kerakli miqdorda probel qo‘shish orqali matnni o‘ng tomondan tekislanilsin. (Matn kengligi 50 ta belgiga teng deb olinsin). **2-formada Tizimli dasturlash fanidan 4 ta variantli test dasturini tuzing.**
28.	1. Matnli fayl berilgan. Fayl matnidagi har bir abzasning birinchi satri 5 ta probeldan boshlangan. Abzas *qizil satr* yordamida ajratiladi. Faylda bo‘sh satrlar mavjud emas. Har bir qo‘shni abzaslar orasiga bittadan bo‘sh satr joylashtirilsin. (faylning boshiga va oxiriga bo‘sh satr qo‘shilmasin). 2. Matnli fayl va lotin harflarining kichik harflaridan S belgi berilgan. Matnli fayl hosil qilinsin va unga boshlang‘ich fayldagi barcha shu S harfidan boshlanuvchi so‘zlar yozilsin (katta yoki kichik harflar bilan boshlanuvchi). So‘z deb, probellarga, tinish belgilariga ega bo‘lmagan va probellar, tinish belgilari yoki satrning boshi/oxiri bilan chegaralangan belgilar to‘plamiga aytiladi. Agar boshlang‘ich fayl tarkibida mos so‘zlar mavjud bo‘lmasa, u holda natijaviy fayl bo‘sh holda qoldirilsin. **2-formada Kompyuter tarmoqlari fanidan 3 ta variantli test dasturini tuzing.**
29.	1. Matnli fayl berilgan. Matnning eng uzun bo‘lgan birinchi so‘zi chop qilinsin. So‘z deb, probellar bilan chegaralangan yoki satrning boshi/oxiri bo‘lgan belgilar to‘plamiga aytiladi. 2. Matnli fayl va lotin harflarining bosh harflaridan S belgi berilgan. Matnli fayl hosil qilinsin va unga boshlang‘ich fayldagi barcha shu S harfidan boshlanuvchi so‘zlar yozilsin (katta yoki kichik harflar bilan boshlanuvchi). So‘z deb, probellarga, tinish belgilariga ega bo‘lmagan va probellar, tinish belgilari yoki satrning boshi/oxiri bilan chegaralangan belgilar to‘plamiga

	aytiladi. Agar boshlang‘ich fayl tarkibida mos so‘zlar mavjud bo‘lmasa, u holda natijaviy fayl bo‘sh holda qoldirilsin. **2-formada Veb texnologiyalar fanidan 2 ta variantli test dasturini tuzing.**
30.	1. Matnli fayl berilgan. Matnning eng qisqa bo‘lgan oxirgi so‘zi chop qilinsin. So‘z deb, probellar bilan chegaralangan yoki satrning boshi/oxiri bo‘lgan belgilar to‘plamiga aytiladi. 2. K butun soni va matnli fayl berilgan. Yangi satrli fayl hosil qilinsin va unga boshlang‘ich fayldagi uzunligi K ga teng bo‘lgan barcha so‘zlar yozilsin. So‘z deb, probellarga, finish belgilariga ega bo‘lmagan va probellar, tinish belgilari yoki satrning boshi/oxiri bilan chegaralangan belgilar to‘plamiga aytiladi. Agar boshlang‘ich fayl K uzunlikdagi so‘z bo‘lmasa, u holda natijaviy fayl bo‘sh holda qoldirilsin. **2-formada Informatika fanidan 3 ta variantli test dasturini tuzing.**

№22-AMALIY MASHG’ULOT

Foydalanuvchi interfeysini loyihalash va testlash usullari.

Foydalanuvchi interfeysini loyihalash usullari, talablari, ma‘lumotlarni eksport qilish vositalari va komponentalari, .NET texnologiyasining hisobotni yaratish konstruktori, testlash va instruksiya yozish usullari, sinovchi xodim xususiyatlari bo‘yicha nazariy bilimlar keltirilgan..

Kalit so‘zlar. Interfeys, UI, WUI, amaliy dastur, ma‘lumotlarni kiritish,

PUI, integratsiya, o‘zaro bog‘lanish, xisobot, testlash.

Inson-mashina interfeysi (IMI) keng tushunchadir. Inson-operator va ular boshqaradigan mashinalar bilan o‘zaro aloqasi ta‘minlashning muhandislik echimlarni o‘z ichiga oladi. Inson - mashina interfeysi tizimlarini yaratishda yaqindan ergonomika tushunchalari bilan bog‘liq. IMI loyihalashtirish quyidagilarni o‘z ichiga oladi:

- ish joyini yaratish: ishchi maydon va boshqaruv paneli, qurilmalar va boshqaruvlarni joylashtirish, ish joyi va ehtimol ranglar uyg‘unligi.
- operatorning barcha boshqaruv bilan o‘zaro aloqasini ta‘minlash hisoblanadi: ularning mavjudligi va zarur harakatlari, KIRISH samaradorligi va tezligi, mustahkamligi, nazorat harakatlarining (izchilligi), joylashgan o‘rni , displeylar va ulardagi teglar hajmi (bularning barchasi mavjudligi).

Inson - kompyuter o‘zaro hamkorligi ilmiy ko‘p yo‘nalishli konteksti kompyuter grafika, muhandislik psixologiyasi, ergonomika, tashkilot nazariyasi, kognitiv fan, informatika va boshqalarni o‘z ichiga oladi.

Foydalanuvchi interfeysi-bu foydalanuvchi kompyuter bilan o‘zaro aloqa qilish imkonini beruvchi dasturiy va apparat vositalar majmuasidir. Bu o‘zaro muloqotlar dialoglarga asoslanadi. Bu holda dialog deganda inson va kompyuter o‘rtasida Real vaqtda amalga oshiriladigan va muayyan vazifani birgalikda hal etishga qaratilgan axborot almashuvi tushuniladi. Axborot almashish va harakatlarni muvofiqlashtirish har bir dialog foydalanuvchi va kompyuterni fizik ulaydigan alohida i/o jarayonlardan iborat. Xabarlarni uzatish orqali axborot almashiladi va nazorat signallari. Xabar-dialog almashinuvida ishtirok etuvchi axborot qismidir. Ular quyidagilardir:

- odam tomonidan kiritish vositalari yordamida hosil qilinadigan KIRISH xabarlari: klaviatura, manipulyatorlar, sichqoncha kabi va boshqalar;
- ompyuter tomonidan matnlar, audio signallar va/yoki tasvirlar ko‘rinishida hosil qilinadigan va foydalanuvchiga monitor ekranida yoki boshqa chiqish qurilmalarida ko‘rsatiladigan chiqish xabarlari.

Asosan foydalanuvchi quyidagi turdagi xabarlarni hosil qiladi: axborot so‘rovi, yordamchi so‘rovi, operatsiya yoki funksiya so‘rovi, ma‘lumot kiritish yoki o‘zgartirish, shablon maydonini tanlash va boshqalar. Bunga javoban, u qabul qiladi: maslahatlar yoki yordam, javob talab qilmaydigan axborot xabarlari, harakatni talab qiladigan so‘rovlar, javob kerak bo‘lgan xato xabarlari va shablon formatidagi o‘zgarishlar.

Dasturlashga protsessual va ob‘yektga yo‘naltirilgan yondashuvlar bilan taqqoslaganda, interfeysni ishlab chiqishga protsessual yo‘naltirilgan va ob‘yektga yo‘naltirilgan yondashuvlar mavjud. Protseduraga yo‘naltirilgan interfeyslar "protsedura" va "operatsiya" tushunchalariga asoslangan foydalanuvchi o‘zaro aloqasining an‘anaviy modelidan foydalanadi. Bu model doirasida dasturiy ta‘minot foydalanuvchiga tegishli ma‘lumotlarni va uning natijasini foydalanuvchi aniqlaydigan ma‘lum amallarni bajarish imkoniyatini beradi.

Ob‘yekt yo‘naltirilgan interfeyslarni foydalanuvchi o‘zaro bir oz boshqacha modelini foydalanib, manipulyatsiya domen ob‘yektlarini qaratishi mumkin.

Zamonaviy realliklardan farqli o‘laroq, birinchi kompyuterlar grafik foydalanuvchi interfeyslari juda zaif bo‘lgan. Shuning uchun, eng boshida, odamlar faqat buyruq qatordan foydalanishingan (CLI yoki buyruq qatorni interfeysi), unda buyruqlar so‘rovlar yordamida

belgilangan edi. Keyinchalik, bu TUI interfeyslarni ishlab, endi operatsion tizimlari o‘rnatish jarayonida ishlatiladi. Kompyuterlarning mavjudligi foydalanuvchilarga qulay interfeysni ishlab chiqish zarurligiga olib keladi.

Grafik foydalanuvchi interfeysi-bu kompyuterning tobora ortib borayotgan ishlashi bilan birga mustahkam o‘rnatilgan interfeys turi. Yaqin kelajakda, odamlar nutq yordamida kompyuter bilan o‘zaro imkonini beradi foydalanuvchi audio interfeyslarni (VUI yoki ovozli foydalanuvchi interfeysi) bo‘lishi mumkin.

Turli kompyuter o‘yinlari tabiiy foydalanuvchi interfeysi (NUI yoki tabiiy foydalanuvchi interfeysi) yaratib beradi. Uning tizimi insonning harakatlarini tahlil qiladi va ularni o‘yindagi harakatlarga aylantiradi. Perseptual foydalanuvchi interfeysi (PUI) va aqlli-kompyuter interfeysi (BCI yoki brain-kompyuter interfeysi) hozirda ishlab chiqilmoqda. eng so‘nggi rivojlanish odamlarga kompyuterlarni fikri kuchi bilan boshqarish qobiliyatini ta‘minlashga qaratilgan.

Nazorat savollari

1. Inson va mashinaning o‘zaro muloqotini ta‘minlash nima deb aytiladi.
2. Inson - kompyuter o‘zaro hamkorligi ilmiy ko‘p yo‘nalishlarini sanab bering.
3. Foydalanuvchi interfeysi nima?
4. Real vaqtda amalga oshiriladigan va muayyan vazifani
5. birgalikda hal etishga qaratilgan axborot almashuvi nima deb aytiladi.
6. Kiritish vositalari yordamida hosil qilinadigan KIRISH xabarlarining texnik vositalarini sanib bering.
7. Interfeys yaratishning asosiy tamoyillari sanab bering.
8. Ma‘lumotlarni eksport qilish vositalari va komponentalari deganda nima nazarda tutiladi.
9. Turli xil xususiyatlarga ega bo‘lgan ma‘lumotlarni birlashtiruvchi vosita – bu nima.
10. Crystal Reports for Microsoft Visual Studio .NET tizimi haqida nimalarni bilasiz.
11. Surish modelining vazifasi
12. Tortish modelining vazifasi nimadan iborat.
13. Testlash deganda nima tushuniladi.
14. Testlashning rivojlanish bosqichlarini sanab bering.
15. Ikki yo‘nalishli testlash jarayoni qaysi bosqichga tegishli.
16. Zamonaviy testlashning qulayliklari va kamchiliklari.

17. Sinovchi xodim bo‘lishning xususiyatlari sanab bering?

MUSTAQIL TA’LIM MAVZULARI

1- Chiziqli, tarmoqlanuvchi va siklik jarayonli masalalarni algoritmlashtirish
2- Takrorlanuvchi masalalarni ishlash
3- Tarmoqlanuvchi masalalarni ishlash
4- Mantiqiy masalalarni ishlash
5- Massivlarga doir masalalarni ishlash
6- Strukturaga doir masalalarni ishlash
7- Saralash algoritmlariga doir masalalarni ishlash
8- Qidirish algoritmlariga doir masalalarni ishlash
9- Rekursiyaga doir masalalarni ishlash
10- Stekka doir masalalarni ishlash
11- Navbatga doir masalalarni ishlash
12- Uzun sonlarga doir masalalarni ishlash
13- Satrli massivlarga doir masalalarni ishlash
14- Alfavit va xizmatchi so’zlar haqida tushunchalar bering.
15- O’zgaruvchilar haqida bilimlaringizni bayon eting.
16- Konstantalar haqida tushunchalar bering.
17- Operatorlar to’g’risida to’liq ma’lumot bering.
18- Ma’lumotlarni kiritish va chiqarish haqida ma’lumot bering.
19- Arifmetik amallar nima, ularga misollar keltiring.
20- Mantiqiy amallar, qiymat berish amallari nima, ularga misollar keltiring
21- Razryadli va shartli amallar nima, ularga misollar keltiring.
22- Nisbat amallar va amallar ustivorligi nima ularga misollar keltiring.
23- Tiplar bilan ishlashga misollar keltiring.
24- Funksiyalar haqida ma’lumotlar bering.

GLOSSARIY

Termin	O’zbek tilidagi sharhi	Ingliz tilidagi sharhi
!=	Teng emas operatori; mantiqiy inkor amali qiymati bilan birhil.	The inequality operator; compares values for inequality returning a bool.
#define	Makro deriktivalarni belgilash	a directive that defines a macro.
#include	Bir source fayl ichida boshqa bir faylag murojatni amalga oshirish mexanizmi	a mechanism for textual inclusion of one source file into another.
+=	add-and-assign operatori; masalan a+=b vazifazi jihatdan a=a+b bilan bir xil	add-and-assign operator; a+=b is roughly equivalent to a=a+b.
.c file	Dastur jodini o’zida jamlovchi fayl	file containing definitions.
.cpp file	Dastur jodini o’zida jamlovchi fayl	file containing definitions.
.h file	Sarlavha fayli	header file
address	Hotira manzili	a memory location
aggregate	Konstruktorsiz massiv yoki struktura	an array or a struct without a constructor
algorithm	Hisoblashning aniq ketma-ketligi	a precise definition of a computation.
and	&& mantiqiy “va” (ko’paytirish) operatori sinonimi	synonym for &&, the logical and operator.
ANSI	Amerika milliy standart agentligi.	The American national standards organization.
application	Umumiy maqsadga ega dasturlar to’plami	a collection of programs seen as serving a common purpose (usually providing a common interface to their users)
bit	0 yoki 1 qiymatga ega birlik hotira	a unit of memory that can hold 0 or 1.

bool	Mantiqiy tip. Bu tip faqat rost yoki yolg'on qiymat qabul qiladi	the built-in Boolean type. A bool can have the values true and false.
Borland C++ Builder	C++ tilida visual dasturlashga ihtisoslashtirilgan IDE muhiti.	Borland's implementation of C++ together with proprietary libraries for Windows programming in an IDE
bug	Xatolik termini	colloquial term for error.
byte	Xotiradagi bir nechta belgilar yig'indisi	a unit of memory that can hold a character of the C++ representation character set.
C++	Tizimli dasturlashni protsedurali qo'llab quvvatlovchi dasturlash tili.	a general-purpose programming language with a bias towards systems programming that supports procedural programming, data abstraction, object-oriented programming, and generic programming.C++ was designed and originally implemented by Bjarne Stroustrup.
char	Belgili tip. Har bir belgi 8 bit, ya'ni baytga teng.	character type; typically an 8-bit byte.
char*	Char massiviga ko'rsatkich	pointer to a char or an array of char. Typically assumed to point to a C-style string.
cin	Standart kiritish oqimi	standard istream.
class	Foydalanuvchi belgilaydigan tur, sinf. Sinf foydalanuvchi funksiyasi, foydalanuvchi ma'lumotlari va kontentlari bo'lishi mumkin.	a user-defined type. A class can have member functions, member data, member constants, and member types.
compiler	C++ da yozilgan buyruqlarni mashina tiliga o'girib beruvchi vosita.	the part of a C++ implementation that produces object code from a translation unit.
const	Faqatgina bir marta qiymat berish mumkin bo'lgan o'zgaruvchilarni e'lon qilsh.	attribute of a declaration that makes the entity to which it refers readonly.

copy()	Nusxa olish operatori	Copy operator
UML	Унификация қилинган моделлаштириш тили	Unified Modeling Language
OMG	Объектларни бошқариш гуруҳи	Object Management Group
4GL	Тўртинчи авлод тили	Fourth-Generation Language
ANSI	Америка миллий стандартлаш институти	American National Standards Institute
AMPS		Advanced Mobile Phone Service
ERP	Корхона ресурсларини режалаштириш	Enterprise Resource Planning
CRM	Мижозлар билан ўзаро муносабатларни бошқариш	Customer Relations Management
SQL	Тузилмалашган сўровлар тили	Structured Query Language
OLAP	Хақиқий вақтда маълумотларга аналитик ишлов бериш	On-Line Analytical Processing
OLTP	Хақиқий вақтда транзакцияларга ишлов бериш	On-Line Transaction Processing
TCO	Эгалик қилишнинг ялпи қиймати	Total Cost of Ownership
JIT	Айни вақтида	Just-In-Time
LAN	Локал ҳисоблаш тармоғи	Local Area Network
MAN	Махаллий ҳисоблаш тармоғи	Metropolitan Area Network
WAN	Худудий ҳисоблаш тармоғи	Wide Area Network
ISO	Ҳалқаро стандартлаштириш ташкилоти	International Organization for Standardization
API	амалий дастурлаштириш махсус интерфейси	Application Programming Interface
WWW	Умумжаҳон ўргамчак тўри	World Wide Web
ASCII	Ахборот алмашишнинг Америка стандарти	American Standard Code for Information Interchange
LIFO	«Охирида келди, биринчи кетди» принципи	Last In, First Out

FIFO	«Биринчи келди, биринчи кетди» принципи	First In, First Out
PDA	персонал рақамли котиб	Personal Digital Assistant

IX. ADABIYOTLAR RO'YXATI

I. Me'yoriy- huquqiy xujjatlar.

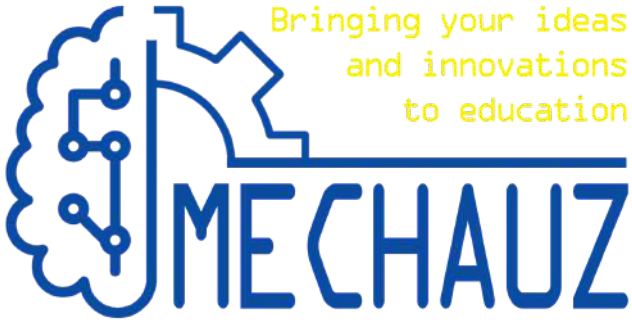
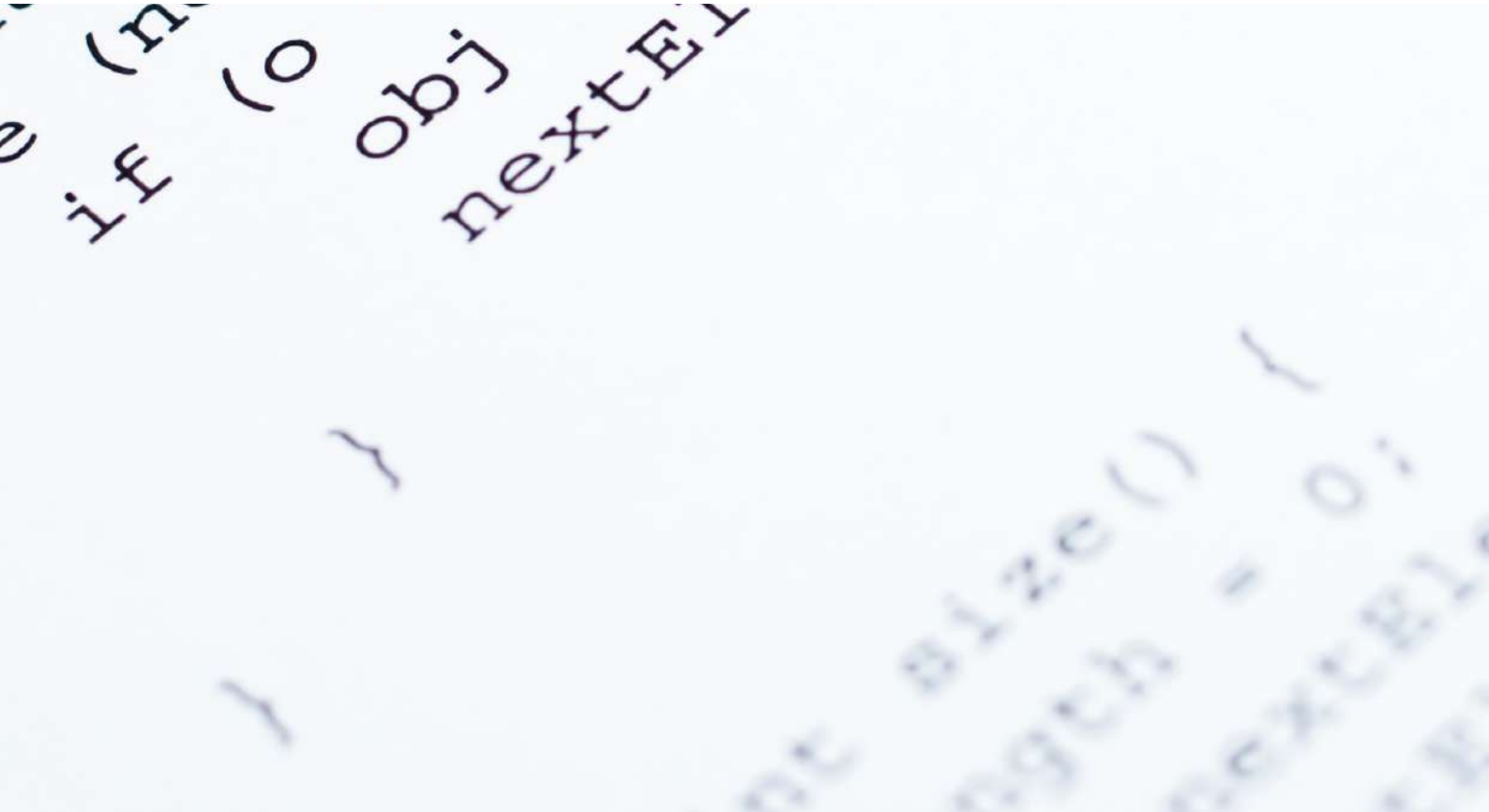
1. O'zbekiston Respublikasi Prezidentining «Oliy ta'lim muassasalarining rahbar va pedagog kadrlarini qayta tayyorlash va malakasini oshirish tizimini yanada takomillashtirish chora-tadbirlari to'g'risida» 2015 yil 12 iyundagi PF-4732-son Farmoni.
2. O'zbekiston Respublikasi Prezidentining 2010 yil 2 noyabrdagi "Oliy malakali ilmiy va ilmiy-pedagogik kadrlar tayyorlash tizimini yanada takomillashtirish chora-tadbirlari to'g'risida"gi PQ-1426-sonli Qarori.
3. Kadrlar tayyorlash milliy dasturi. O'zbekiston Respublikasi Oliy Majlisining Axborotnomasi, 1997 yil. 11-12-son, 295-modda.
4. O'zbekiston Respublikasi Prezidentining 2012 yil 24 iyuldagi "Oliy malakali ilmiy va ilmiy-pedagog kadrlar tayyorlash va attestasiyadan o'tkazish tizimini yanada takomillashtirish to'g'risida"gi PF–4456-son Farmoni.
5. O'zbekiston Respublikasi Vazirlar Mahkamasining 2012 yil 28 dekabrdagi "Oliy o'quv yurtidan keyingi ta'lim xamda oliy malakali ilmiy va ilmiy pedagogik kadrlarni attestasiyadan o'tkazish tizimini takomillashtirish chora-tadbirlari to'g'risida"gi 365- sonli Qarori.

II. Maxsus adabiyotlar.

1. Bjarne Stroustrup. Programming: Principles and Practice Using C++ (2nd Edition). Person Education, Inc. 2014. second printing, January 2015.
2. Harry Hariom Choudhary, Bjarne M Stroustrup. C++ Programming Professional.: Sixth Best Selling Edition for Beginner's & Expert's 2014.
3. Bjarne Stroustrup. The C++ Programming Language, 4th Edition. Person Education, Inc. 2013. Third printing, April 2014.
4. Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. "Voris-nashriyot" MCHJ, Toshkent 2013. 488 b.
5. Horstmann, Cay S. C++ for everyone / Cay S. Horstmann. Printed in the United States of America - 2nd ed. 2010. – P. 562.
6. Horton I. - Beginning Visual C++ 2012 / I.Horton. Published simultaneously in Canada. – 2012. –P. 988.

III. Internet resurslar.

1. http://www.stroustrup.com/4th.html, http://www.cplusplus.com/
2. http://acm.tuit.uz/ - дастурий ечим тўғрилигини автоматик тестловчи тизим.
3. http://acm.tuit.uz/forum/, http://acm.timus.ru/ – дастурларни тестловчи тизим.
4. http://codeforces.com/ - дастурий ечим тўғрилигини автоматик тестловчи тизим



www.mechauz.uz