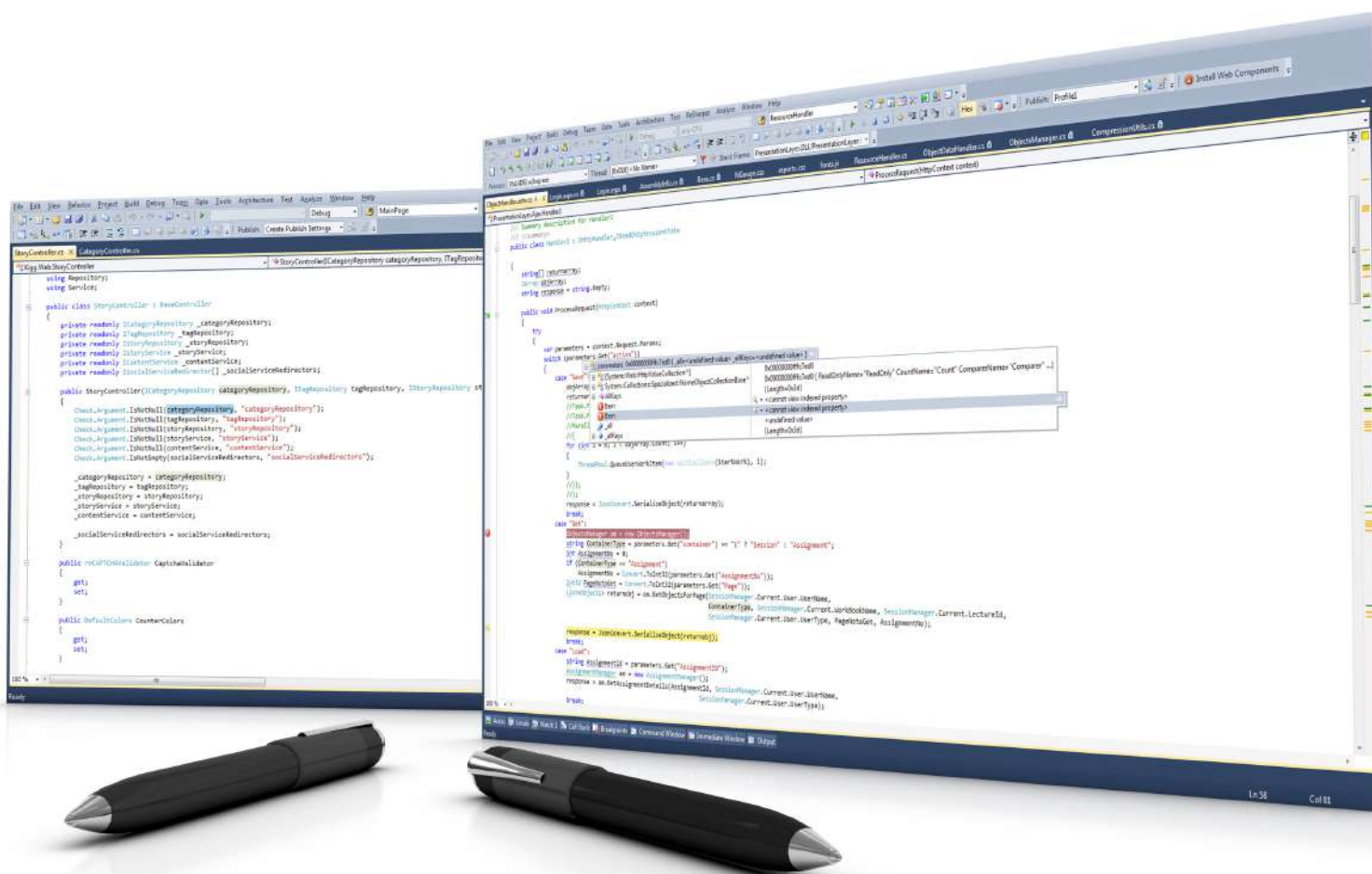




Co-funded by the
Erasmus+ Programme
of the European Union

Toshkent axborot texnologiyalari universiteti

KOMPYUTERDA DASTURLASH 1





Erasmus+

This project is funded by the European Union.



KOMPYUTERDA DASTURLASH 1

=====

COMPUTER PROGRAMMING 1

=====

КОМПЬЮТЕРНОЕ ПРОГРАММИРОВАНИЕ 1

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

The mission of science is to those who study it:

- fields of application of programming languages;
- program structure and alphabet;
- use of constants and variables, data types;
- methods of applying mathematical and logical operations;
- methods of using input - output operators;
- creating algorithms and programs for branching and repetitive processes;
- creation of functions and libraries and their use;
- work with arrays, strings and characters;
- work with static structure and dynamic structure of data;
- graphic programming capabilities;
- working with files;
- methods of applying class, object-oriented programming capabilities in various fields;
- consists of teaching theoretical and practical knowledge of visual programming elements on the basis of coherence and continuity.

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	Algoritmash va dasturlashning asosiy tushunchalari. Tilning bazaviy tushunchalari.	30
2	Dasturlash tillarining tuzilmasi. Ternar operator.	
3	Tarmoqlanish va uzilishlarni tashkil etish operatorlari.	
4	Takrorlanish operatorlari.	
5	Funksiyalar. Funksiya tavsifi.	
6	Massivlar. Massiv tushunchasi.	
7	Ko'rsatkichlar va dinamik xotira bilan ishlash.	
8	Obyektga yo'naltirilgan dasturlash asoslari.	
9	Konstruktorlar va destruktorlar.	
10	Satrlar va kengaytirilgan belgilar.	
11	Fayllar va fayllar bilan ishlash.	
12	Inkapsulyatsiya va merosxo'rlik. Inkapsulyatsiya.	
13	Polimorfizm. Polimorfizm va uning turlari.	
14	Operatorlarni qayta yuklash.	
15	Shablonlar bilan ishlash. Shablon (template) tushunchasi va ularning qo'llanilishi.	

Amaliy mashg'ulot mavzulari:

№	Amaliy mashg'ulot mavzulari	Ajratilgan soat
1	Algoritmalar, xossalari, ularni ifodalash usullari.	44
2	Dasturlashga kirish. C++ dasturlash tilida dastur kompilyatsiyasi va kompilyator turlari. Identifikator va ularning turlari.	
3	Dasturlash tillarining tuzilmasi. O'zgaruvchilarga qiymat kiritish va chiqarish operatorlari. printf(), scanf() funksiyalari. Matematik kutubxona funksiyalari va ular yordamida chiziqli algoritmarni tashkil qilish.	
4	Tarmoqlanuvchi operatorlar. Shartli o'tish operatori if, tanlash operatori (switch).	
5	Takrorlanuvchi operatorlar. Parametrli sikl operatori for, old shart while va so'ng shart do while operatorlari.	
6	Funksiyalar. Qiymat qaytaruvchi va qiymat qaytarmaydigan funksiyalar, funksiya prototiplardan foydalanib, parametrlarni qiymat va adresga binoan jo'natishga doir masalalar yechish.	

7	Funksiyalar. Rekursiv funksiyalarga doir masalalar yechish. Funksiyalarni qayta yuklash. Foydalanuvchi kutubxonasini tashkil qilish.
8	Massivlar. Bir o'lovli massivlarni e'lon qilish va ularning elementlariga murojatlarni tashkil qilishga doir masalalar yechish.
9	Massivlar. Ko'p o'lovli massivlarni e'lon qilish va ularning elementlariga murojatlarni tashkil qilishga doir masalalar yechish.
10	Massivlar. Massiv elementlarini saralash va qidirishga doir masalalar yechish.
11	Ko'rsatkichlar va dinamik xotira bilan ishlash. Ko'rsatkichlar va dinamik xotira bilan ishlashga doir masalalar yechish.
12	Ob'ektga yo'naltirilgan dasturlash asoslari. Sinf (class) yaratish usullari. Sinf a'zolariga murojat usullari. Ob'ektlar yaratish va ulardan foydalanishga doir masalalar yechish.
13	Ob'ektga yo'naltirilgan dasturlash asoslari Sinfning funksiya a'zolarini yaratish va ularga murojat usullari va ulardan foydalanishga doir masalalar yechish.
14	Ob'ektga yo'naltirilgan dasturlash asoslari. Sinf konstruktorini yaratish va ularga murojat usullari va ulardan foydalanishga doir masalalar yechish.
15	Ob'ektga yo'naltirilgan dasturlash asoslari. Tuzilmalar va birlashmalar hamda ulardan foydalanishga doir masalalar yechish.
16	Kengaytirilgan belgilar. Kengaytirilgan belgilarni ulash, solishtirish, belgilarni izlash, satr qismlarini izlashga doir masalalar yechish.
17	Kengaytirilgan belgilar. Kengaytirilgan belgilarni o'zgartirish va o'chirishga doir masalalar yechish.
18	Satrlar. Satrlarni ulash, solishtirish, belgilarni izlash, satr qismlarini izlash.
19	Satrlar. Ularni o'zgartirish va o'chirishga doir masalalar yechish.
20	Fayllar va fayllar bilan ishlash. Matnli fayllarga doir masalalar yechish.
21	Fayllar va fayllar bilan ishlash. Binar fayllarga doir masalalar yechish. Istisno (exception) larni qayta ishlash (throw, try i catch).
22	Fayllar va fayllar bilan ishlash. Matnli va binar fayllar bilan ishlashda istisno (exception) larni qayta ishlash (throw, try va catch)ga doir masalalar yechish.

23	Inkapsulyasiya. Sinf a'zolariga ruxsat (public, private va protected) larni va ularga murojat qilishga doir masalalar yechish.	
24	Merosxo'rlik. Merosxo'rlik turlari (public, private va protected), o'zaro bog'langan sinflar ularga bog'langan sinflar va ularga murojat qilishga doir masalalar yechish.	
25	Poliforfizim. Sinf funksiyalarini qayta yuklashga (OVERRIDE) doir masalalar yechish.	
26	Poliforfizim. Virtual funksiyaga doir masalalar yechish. Abstrakt sinf va funksiyalar.	
27	Operatorlarni qayta yuklash. Operatorlarni qayta yuklashga doir masalalar.	
28	Operatorlarni qayta yuklash. Amallarni qayta yuklashga doir masalalar yechish.	
29	Shablonlar bilan ishlash. Funksiya shablon (template)larni yaratish usullari ularga doir masalalar yechish.	
30	Shablonlar bilan ishlash. Sinf shablon(template)larini usullari ularga doir masalalar yechish.	

Mustaqil ish mavzulari:

№	Mustaqil ish mavzulari	Ajratilgan soat
1	Algoritm tuzish jarayonlarini tashkil etish	90
2	C++ dasturlash tilida dastur kompilyatsiyasi va kompilyator turlari	
3	Tarmoqlanuvchi jarayonlarni tashkil etish.(Algoritm va dastur. if operatori).	
4	Tarmoqlanuvchi jarayonlarni tashkil etish.(Algoritm va dastur. switch operatori).	
5	Tarmoqlanuvchi jarayonlarni tashkil etish.(Algoritm va dastur. goto operatori).	
6	Takrorlanuvchi jarayonlarni tashkil etish.(Algoritm va dastur. for operatori).	
7	Takrorlanuvchi jarayonlarni tashkil etish.(Algoritm va dastur. do while operatori).	
8	Takrorlanuvchi jarayonlarni tashkil etish.(Algoritm va dastur. while operatori).	

9	Ichma-ich joylashgan tsikllar.	
10	Ichma-ich joylashgan tsikllar. (tutli ko'rinishdagi tsiklar yordamida tashkil qilingan tsikllar)	
11	break va continue operatorlari	
12	C++da funsiyalarni tashkil etish.	
13	Rekkrziv funksiya	
14	Funksiyalarni qayta yuklash	
15	Massivlarni tashkil etish. Bit o'lchamli massivlar.	
16	Massivlar. Bir o'lchamli massivlarni saralash (pufakcha usuli).	
17	Massivlar. Bir o'lchamli massivlarni saralash (tanlah usuli).	
18	Massivlar. Bir o'lchamli massivlarda qidirish.(binar qidirish).	
19	Massivlar. Ikki o'lchamli massivlar.	
20	Ko'rstkichlar. Bir o'lchovli massivlarni funksiya parametrlari sifatida qo'llanilishi.	
21	Ko'rstkichlar va dinamik massivlar.	
22	Ob'ektga yo'naltirilgan dasturlash. Asosiy tushunchalar. Mustaqil sinflar tashkil etish.	
23	Satrlar. Belgilar massivi. String sinfi.	
24	Fayllar bilan ishlashning yangi usullari(I/O texnologiyasi)	
25	OYD. Inkapsulyatsiya. public va private sinf modifikatorlari.	
26	Konstruktorlar va ularni tashkil etish usullari.	
27	OYD. Do'stona funksiya.	
28	OYD. Ob'ektlar massivi bilan ish yuritish.	
29	OYD.Voris sinflar yaratish va ular yordamida masalalarni yechish.	
30	OYD.Virtual funksiya va abstract sinflar.	
31	Sinflarni yaratishda funsiyalarni qayta yuklash mexanizmidan umumli foydalanish.	
32	Sinflarni yaratishda operatorlarni qayta yuklash mexanizmidan umumli foydalanish.	
33	Shablon funksiya yaratish va ulardan foydalanish.	
34	Shablon sinflar yaratish va ulardan foydalanish.	
35	Shablon funksiyalarda funsiyalarni qayta yuklash mexanizmi.	

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 qiyinchiliklar 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 uskunolari;
- 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:

Baholash usullari	Onlayn testlar, yozma ishlar, og'zaki so'rov, amaliy topshiriqlar prezentatsiyalar va h.k.		
Baholash mezonlari	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.		
	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.		
	60-69 ball «qoniqarli» Fan mohiyatini tushuntirish. Bilish, aytib berish. Fan haqida tasavvurga ega bo'lish.		
	0-60 ball «qoniqarsiz» Fan haqida aniq tasavvurga ega bo'lmaslik. Bilmaslik.		
Reyting baholash turlari	Joriy nazorat	Maksimal ball (30)	O'tkazish vaqti
	6 ta amaliy ishlarini bajarish va hisobot qilish	5	O'quv jarayoni grafigi bo'yicha
	Oraliq nazorat	Maksimal ball (20)	O'tkazish vaqti
	6 ta amaliy ishlarini bajarish va hisobot qilish	5	O'quv jarayoni grafigi bo'yicha
	Yakuniy nazorat	50	Semestr oxirida
	Fan bo'yicha jami	100	

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.

MA'RUZALAR

1-MAVZU: ALGORITMLASH VA DASTURLASHNING ASOSIY TUSHUNCHALARI. TILNING BAZAVIY TUSHUNCHALARI.

Reja:

1. Tilning bazaviy tushunchalari (til alifbosi, identifikator va leksemalar, kalit so'zlar, konstanta satrlar, ma'lumotlar toifasi, ma'lumotlar toifasini o'zgartirish)
2. Arifmetik ifoda va amallar, siljitish amallari, inkrement va decrement, bitlarga ishlov beruvchi operatorlar
3. Kutubxona funksiyalari
4. Preprotssessor direktivalari va vositalari

Kalit so'zlar: *kommunikatsiya, dasturiy ta'minot, tashxis, teskari aloqa, loyihalash, foydalanuvchi interfeysi, foydalanuvchi, aniqlik, Stereotip, buyurtmachi, dasturchi, samaradorlik.*

1. Tilning bazaviy tushunchalari (til alifbosi, identifikator va leksemalar, kalit so'zlar, konstanta satrlar, ma'lumotlar toifasi, ma'lumotlar toifasini o'zgartirish)

C++ tili Byarn Straustrup tomonidan 1980 yil boshlarida ishlab chiqilgan. C++ tilida yaxshi dastur tuzish uchun "aql, farosat va sabr" kerak bo'ladi. Bu til asosan tizim sathida dasturlovchilar uchun yaratilgan.

C/C++ algoritmik tilining alifbosi:

1. 26 ta lotin va 32 ta kirill harflari (katta va kichik);
2. 0 dan 9 gacha bo'lgan arab raqamlari;
3. Maxsus belgilar: - + * / : ; . , % ? ! = " ' " № < > { } [] () \$ # & ^ va h.k.

Dastur bajarilishi jarayonida o'z qiymatini o'zgartira oladigan kattaliklar o'zgaruvchilar deyiladi. O'zgaruvchilarning nomlari harfdan boshlanuvchi xarf va raqamlardan iborat bo'lishi mumkin. O'zgaruvchilarni belgilashda katta va kichik harflarning farqlari bor. (A va a harflari 2 ta o'zgaruvchini bildiradi) Har bir o'zgaruvchi o'z nomiga, toifasiga, xotiradan egallagan joyiga va son qiymatiga ega bo'lishi kerak. O'zgaruvchiga murojaat qilish uning ismi orqali bo'ladi. O'zgaruvchi uchun xotiradan ajratilgan joyning tartib raqami uning adresi hisoblanadi. O'zgaruvchi ishlatilishidan oldin u aniqlangan bo'lishi lozim.

O'zgaruvchilarning son qiymatlari quyidagi ko'rinishda yoziladi:

- Butun toifali o'nlik sanoq tizimsida: ular faqat butun sondan iborat bo'ladilar. Masalan: 5; 76; -674 va h.k.
- Sakkizlik sanoq tizimsidagi sonlar: 0 (nol) dan boshlanib, 0 dan 7 gacha bo'lgan raqamlardan tashkil topadi. Masalan: $x=0453217$; $s=077$;
- O'n oltilik sanoq tizimsidagi sonlar: 0 (nol) dan boshlanadi va undan keyin x yoki X harfi keladi, so'ngra 0-9 raqamlari va a-f yoki A-F harflaridan iborat ketma-ketliklar bo'ladi. Masalan: 10 s.s.dagi 22 soni 8 s.s. da 026, 16 s.s.da 0x16 shaklida bo'ladi.
- Haqiqiy toifali sonlar: ular butun va kasr qismlardan iborat bo'ladilar. Masalan: 8,1; -12,59 va x.k. Haqiqiy toifali sonlarning bu ko'rinishi oddiy ko'rinish deyiladi. Juda katta yoki juda kichik haqiqiy toifali sonlarni darajali (eksponensial) formada yozish qulay. Masalan: $7,204 \cdot 10^{12}$ yoki $3,567 \cdot 10^{-11}$ kabi sonlar $7.204e+12$ va $3.567e-11$ ko'rinishda yoziladi.
- Simvulli konstantalar. Ular qatoriga dastur bajarilishi ' ' ichida qabul qilinadigan simvollar kiradi.

C/C++ tilida har qanday o'zgaruvchi ishlatilishidan oldin e'lon qilinishi kerak. E'lon qilish degani ularning toifalarini aniqlab qo'yish demakdir.

Identifikatorlar va kalit so'zlar. Dasturlash tilining muhim tayanch tushunchalaridan biri - identifikator tushunchasidir. *Identifikator* deganda katta va kichik lotin harflari, raqamlar va tag chiziq ('_') belgilaridan tashkil topgan va raqamdan boshlanmaydigan belgilar ketma-ketligi tushuniladi. Identifikatorlarda harflarning registrlari (katta yoki kichikligi) hisobga olinadi. Masalan, RUN, run, Run - bu har xil identifikatorlardir. Identifikator uzunligiga chegara qo'yilma-gan, lekin ular kompilyator tomonidan faqat boshidagi 32 belgisi bilan farqlanadi.

Identifikatorlar kalit so'zlar, o'zgaruvchilar, funksiyalar, nishonlar va boshqa obyektlarni nomlashda ishlatiladi.

C++ tilining kalit so'zlariga quyidagilar kiradi:

asm, auto, break, case, catch, char, class, const, continue, default, delete, do, double, else, enum, explicit, extern, float, for, friend, goto, if, inline, int, long, mutable, new, operator, private, protected, public, register, return, short, signed, sizeof, static, struct, swith, template, this, throw, try, typedef, typename, union, unsigned, virtual, void, volatile, while.

Yuqorida keltirilgan identifikatorlarni boshqa maqsadda ishlatish mumkin emas. Protsessor registrlarini belgilash uchun quyidagi so'zlar ishlatiladi:

_AH, _AL, _AX, _EAX, _BH, _BL, _BX, _EBX, _CL, _CH, _CX, _ECX, _DH, _DL, _DX, _EDX, _CS, _ESP, _EBP, _FS, _GS, _DI, _EDI, _SI, _ESI, _BP, _SP, _DS, _ES, _SS, _FLAGS.

Bulardan tashqari «__» (ikkita tagchiziq) belgilaridan boshlangan identifikatorlar kutubxonalar uchun zahiralangan. Shu sababli '_' va «__» belgilarni identifikatorning birinchi belgisi sifatida ishlatmagan ma'qul. Identifikator belgilar orasida probel ishlatish mumkin emas, zarur bo'lganda uning o'rniga '_' ishlatish mumkin: Cilindr_radiusi, ailana_diametiri.

O'zgarmaslar. *O'zgarmas (literal)* - bu fiksirlangan sonni, satrni va belgini ifodalovchi leksemadir.

O'zgarmaslar beshta guruhga bo'linadi - *butun, haqiqiy (suzuvchi nuqtali), sanab o'tiluvchi, belgi (literli) va satr («string», literli satr).*

Kompilyator o'zgarmasni leksema sifatida aniqlaydi, unga xotiradan joy ajratadi, ko'rinishi va qiymatiga (turiga) qarab mos guruhlariga bo'ladi.

Butun o'zgarmaslar. Butun o'zgarmaslar quyidagi formatlarda bo'ladi:

- o'nlik son;
- sakkizlik son;
- o'n oltilik son.

O'nlik o'zgarmas 0 raqamidan farqli raqamdan boshlanuvchi raqamlar ketma-ketligi va 0 hisoblanadi: **0; 123; 7987; 11.**

Manfiy o'zgarmas - bu ishorasiz o'zgarmas bo'lib, unga faqat ishorani o'zgartirish amali qo'llanilgan deb hisoblanadi.

Sakkizlik o'zgarmas 0 raqamidan boshlanuvchi sakkizlik sanoq sistemasi (0,1,...,7) raqamlaridan tashkil topgan raqamlar ketma-ketligi:

023; 0777; 0.

O'n oltilik o'zgarmas 0x yoki 0X belgilaridan boshlanadigan o'n oltilik sanoq sistemasi raqamlaridan iborat ketma-ketlik hisoblanadi:

0x1A; 0X9F2D; 0x23.

Harf belgilar ixtiyoriy registrlarda berilishi mumkin.

Kompilyator sonning qiymatiga qarab unga mos turni belgilaydi. Agar tilda belgilangan turlar dastur tuzuvchini qanoatlantirmasa, u oshkor ravishda turni ko'rsatishi mumkin. Buning uchun butun o'zgarma raqamlari oxiriga, probelsiz l yoki L (long), u yoki U (unsigned) yoziladi. Zarur hollarda bitta o'zgarma uchun bu belgilarning ikkitasini ham ishlatish mumkin:

45lu, 012Ul, 0xA2L.

Haqiqiy o'zgarmlar. Haqiqiy o'zgarmlar - suzuvchi nuqtali son bo'lib, u ikki xil formatda berilishi mumkin:

- o'nlik fiksirlangan nuqtali formatda. Bu ko'rinishda son nuqta orqali ajratilgan butun va kasr qismlar ko'rinishida bo'ladi. Sonning butun yoki kasr qismi bo'lmasligi mumkin, lekin nuqta albatta bo'lishi kerak. Fiksirlangan nuqtali o'zgarmlarga misollar: **24.56; 13.0; 66.; .87;**

- eksponensial shaklda haqiqiy o'zgarma 6 qismdan iborat bo'ladi:

- 1) butun qismi (o'nli butun son);
- 2) o'nli kasr nuqta belgisi;
- 3) kasr qismi (o'nlik ishorasiz o'zgarma);
- 4) eksponenta belgisi 'e' yoki 'E';
- 5) o'n darajasi ko'rsatkichi (o'nli butun son);
- 6) qo'shimcha belgisi ('F' yoki 'f' , 'L' yoki 'l').

Eksponensial shakldagi o'zgarma sonlarga misollar: **1e2; 5e+3; .25e4; 31.4e-1 .**

Belgi o'zgarmlar. Belgi o'zgarmlar qo'shtirnoq (','-apostroflar) ichiga olingan alohida belgilardan tashkil topadi va u char kalit so'zi bilan aniqlanadi. Belgi o'zgarma uchun xotirada bir bayt joy ajratiladi va unda butun son ko'rinishidagi belgining ASCII kodi joylashadi. Quyidagilar belgi o'zgarmlarga misol bo'ladi: **'e', '@', '7', 'z', 'W', '+', 'sh', '*', 'a', 's'.**

1.1-jadval. C++ tilida escape -belgilar jadvali

Escape belgilari	Ichki kod (16 son)	Nomi	Amal
\\	0x5C	\	Teskari yon chiziqni chop etish
\'	0x27	'	Apostrofni chop etish
\"	0x22	"	Qo'shtirnoqni chop etish
\?	0x3F	?	So'roq belgisi
\a	0x07	bel	Tovush signalini berish
\b	0x08	bs	Kursorni 1 belgi o'rniga orqaga qaytarish
\f	0x0C	ff	Sahifani o'tkazish
\?	0x3F	?	So'roq belgisi
\n	0x0A	lf	Qatorni o'tkazish
\r	0x0D	cr	Kursorni ayni qatorning boshiga qaytarish
\t	0x09	ht	Navbatdagi tabulyasiya joyiga o'tish
\v	0x0D	vt	Vertikal tabulyasiya (pastga)
\000	000		Sakkizlik kodi
\xNN	0xNN		Belgi o'n oltilik kodi bilan berilgan

Ayrim belgi o'zgarmlar '\ ' belgisidan boshlanadi, bu belgi birinchidan, grafik ko'rinishga ega bo'lmagan o'zgarmlarni belgi-laydi, ikkinchidan, maxsus vazifalar yuklangan belgilar - apostrof belgisi('), savol belgisini ('?'), teskari yon chiziq belgisini ('\') va ikkita qo'shtirnoq belgisini ('') chop qilish uchun ishlatiladi. Undan tashqari, bu belgi orqali belgini ko'rinishini emas, balki oshkor ravishda uning ASCII kodini sakkizlik yoki o'n oltilik shaklda yozish mumkin. Bunday belgidan boshlangan belgilar escape ketma-ketliklar deyiladi (1.1-jadval).

C++ tilida qo'shimcha ravishda wide harfli o'zgarmlar va ko'p belgili o'zgarmlar aniqlangan.

wide harfli o'zgarmlar turi milliy kodlarni belgilash uchun kiritilgan bo'lib, u wchar_t kalit so'zi bilan beriladi, hamda xotirada 2 bayt joy egallaydi. Bu o'zgarmlar L belgisidan boshlanadi:

L'\013\022', L'cc'

Probel bilan ajratib yozilgan satrlar kompilyator tomonidan yagona satrga ulanadi (konkantenatsiya):

"Satr - bu belgilar massivi" /* bu satr keyingi satrga ko'shiladi */", uning turi char[]";

Bu yozuv

"Satr - bu belgilar massivi, uning turi char[]";

yozuvi bilan ekvivalent hisoblanadi.

Uzun satrni bir nechta qatorga yozish mumkin va buning uchun qator oxirida '\ ' belgisi qo'yiladi:

**"Kompilyator har bir satr uchun kompyuter xotirasida\
satr uzunligiga teng sondagi baytlardagi alohida \
xotira ajratadi va bitta - 0 qiymatli bayt qo'shadi";**

Yuqoridagi uchta qatorda yozilgan satr keltirilgan. Teskari yon chiziq ('\') belgisi keyingi qatorda yozilgan belgilar ketma-ketligini yuqoridagi satrga qo'shish kerakligini bildiradi. Agar qo'shiladigan satr boshlanishida probellar bo'lsa, ular ham satr tarkibiga kiradi. Probel bilan ajratib yozilgan satrlar kompilyator tomonidan yagona satrga ulanadi (konkantenatsiya):

"Satr - bu belgilar massivi" /* bu satr keyingi satrga ko'shiladi */", uning turi char[]";

Bu yozuv

"Satr - bu belgilar massivi, uning turi char[]";

yozuvi bilan ekvivalent hisoblanadi.

Uzun satrni bir nechta qatorga yozish mumkin va buning uchun qator oxirida '\ ' belgisi qo'yiladi:

**"Kompilyator har bir satr uchun kompyuter xotirasida\
satr uzunligiga teng sondagi baytlardagi alohida \
xotira ajratadi va bitta - 0 qiymatli bayt qo'shadi";**

Yuqoridagi uchta qatorda yozilgan satr keltirilgan. Teskari yon chiziq ('\') belgisi keyingi qatorda yozilgan belgilar ketma-ketligini yuqoridagi satrga qo'shish kerakligini bildiradi. Agar qo'shiladigan satr boshlanishida probellar bo'lsa, ular ham satr tarkibiga kiradi.

Berilganlar turlari va o'zgaruvchilar. Dastur bajarilishi paytida qandaydir berilganlarni saqlab turish uchun o'zgaruvchilar va o'zgarmlardan foydalaniladi. O'zgaruvchi - dastur obyekti bo'lib, xotiradagi bir nechta yacheyka-larni egallaydi va berilganlarni saqlash uchun xizmat qiladi. O'zgaruvchi nomga, o'lchamga va boshqa atributlarga - ko'rinish sohasi, amal qilish vaqti va boshqa xususiyatlarga ega bo'ladi. O'zgaruvchilarni ishlatish uchun ular albatta e'lon qilinishi kerak. E'lon natijasida o'zgaruvchi uchun xotiradan qandaydir soha zahiralanadi, soha o'lchami esa o'zgaruvchining konkret turiga bog'liq bo'ladi. SHuni qayd etish zarurki, bitta turga turli apparat platformalarda turlicha joy ajratilishi mumkin.

O'zgaruvchi e'loni uning turini aniqlovchi kalit so'zi bilan boshlanadi va '=' belgisi orqali boshlang'ich qiymat beriladi (shart emas). Bitta kalit so'z bilan bir nechta o'zgaruvchilarni e'lon qilish mumkin. Buning uchun o'zgaruvchilar bir-biridan ',' belgisi bilan ajratiladi. E'lonlar ';' belgisi bilan tugaydi. O'zgaruvchi nomi 255 belgidan oshmasligi kerak.

C++ tilining tayanch turlari, ularning baytlardagi o'lchamlari va qiymatlarining chegaralari 1.2-jadvalda keltirilgan.

Butun son turlari. Butun son qiymatlarni qabul qiladigan o'zgaruvchilar int (butun), short (qisqa) va long (uzun) kalit so'zlar bilan aniqlanadi. O'zgaruvchi qiymatlari ishorali bo'lishi yoki unsigned kalit so'zi bilan ishorasiz son sifatida qaralishi mumkin

Belgi turi. Belgi turidagi o'zgaruvchilar char kalit so'zi bilan beriladi va ular o'zida belgining ASCII kodini saqlaydi. Belgi turidagi qiymatlar nisbatan murakkab bo'lgan tuzilmalar - satrlar, belgilar massivlari va hakoza larni hosil qilishda ishlatiladi.

1.2-jadval. C++ tilining tayanch turlari

Tur nomi	Baytlardagi o'lchami	Qiymat chegarasi
bool	1	true yoki false
unsigned short int	2	0..65535
short int	2	-32768..32767
unsigned long int	4	0..42949667295
long int	4	-2147483648..2147483647
int (16 razryadli)	2	-32768..32767
int (32 razryadli)	4	-2147483648..2147483647
unsigned int (16 razryadli)	2	0..65535
unsigned int (32 razryadli)	4	0..42949667295
unsigned char	1	0..255
char	1	-128..127
float	4	1.2E-38..3.4E38
double	8	2.2E-308..1.8E308
long double (32 razryadli)	10	3.4e-4932..3.4e4932
void	2 yoki 4	-

Haqiqiy son turi. Haqiqiy sonlar float kalit so'zi bilan e'lon qilinadi. Bu turdagi o'zgaruvchi uchun xotirada 4 bayt joy ajratiladi va <ishora><tartib><mantissa> qolipida sonni saqlaydi (1-ilovaga qarang). Agar kasrli son juda katta (kichik) qiymatlarni qabul qiladigan bo'lsa, u xotiradi 8 yoki 10 baytda ikkilangan aniqlik ko'rinishida saqlanadi va mos ravishda double va long double kalit so'zlari bilan e'lon qilinadi. Oxirgi holat 32-razryadli platformalar uchun o'rinli.

Mantiqiy tur. Bu turdagi o'zgaruvchi bool kalit so'zi bilan e'lon qilinadi. U turdagi o'zgaruvchi 1 bayt joy egallaydi va 0 (false, yolg'on) yoki 1 qiymatidan farqli qiymat (true, rost) qabul qiladi. Mantikiy turdagi o'zgaruvchilar qiymatlar o'rtasidagi munosabat-larni ifodalaydigan mulohazalarni rost yoki yolg'on ekanligini tavsiflashda qo'llaniladi va ular qabul qiladigan qiymatlar matematik mantiq qonuniyatlariga asoslanadi.

Matematik mantiq - fikrlashning shakli va qonuniyatlari haqidagi fan. Uning asosini mulohazalar hisobi tashkil qiladi. **Mulohaza** - bu ixtiyoriy jumla bo'lib, unga nisbatan rost yoki yolg'on fikrni bildirish mumkin. Masalan «3>2», «5 - juft son», «Moskva-Ukraina poytaxti» va hakoza. Lekin «0.000001 kichik son» jumlasini mulohaza hisoblanmaydi, chunki «kichik son» tushunchasi juda ham nisbiy, ya'ni kichik son deganda qanday sonni tushunish kerakligi aniq emas. Shuning uchun yuqoridagi jumlaning rost yoki yolg'onligi haqida fikr bildirish qiyin.

Mulohazalarning rostligi holatlarga bog'liq ravishda o'zgapishi mumkin. Masalan «bugun - chorshanba» jumlasini rost yoki yolg'onligi ayni qaralayotgan kunga bog'liq. Xuddi shunday « $x < 0$ » jumlasini x o'zgaruvchisining ayni paytdagi qiymatiga mos ravishda rost yoki yolg'on bo'ladi.

C++ tilida mantiqiy tur nomi angliyalik matematik Jopj Bul shapafiga bool so'zi bilan ifodalangan. Mantiqiy amallar «Bul algebrasi» deyiladi.

Mantiqiy mulohazalar ustida uchta amal aniqlangan:

1) **inkor** - A mulohazani inkori deganda A rost bo'lganda yolg'on va yolg'on bo'lganda rost qiymat qabul qiluvchi mulohazaga aytiladi. C++ tilida inkor - '!' belgisi bilan beriladi. Masalan, A mulohaza inkori «!A» ko'rinishida yoziladi;

2) **konyuksiya**- ikkita A va V mulohazalar kon'yuktsiyasi yoki mantiqiy ko'paytmasi «A && V» ko'rinishga ega. Bu mulohaza faqat A va V mulohazalar rost bo'lgandagina rost bo'ladi, aks holda yolg'on bo'ladi (odatda «&&» amali «va» deb o'qiladi). Masalan «bugun oyning 5 kuni va bugun chorshanba» mulohazasi oyning 5 kuni chorshanba bo'lgan kunlar uchun rost bo'ladi;

3) **diz'yunksiya** - ikkita A va V mulohazalar diz'yunksiyasi yoki mantiqiy yig'indisi «A || V» ko'rinishida yoziladi. Bu mulohaza rost bo'lishi uchun A yoki V mulohazalardan biri rost bo'lishi etarli. Odatda «||» amali «yoki» deb o'qiladi.

Yurqorida keltirilgan fikrlar asosida mantiqiy amallar uchun rostlik jadvali aniqlangan (1.3-jadval)

1.3-jadval. Mantiqiy amallar uchun rostlik jadvali

Mulohazalar		Mulohazalar ustida amallar		
A	B	!A	A && B	A B
false	false	true	false	false
false	true	true	false	true
true	false	false	false	true
true	true	false	true	true

Mantiqiy tur qiymatlari ustida mantiqiy ko'paytirish, qo'shish va inkor amallarini qo'llash orqali murakkab mantiqiy ifodalarni qurish mumkin. Misol uchun, « x -musbat va y qiymati [1..3] sonlar oralig'iga tegishli emas» mulohazasini mantiqiy ifoda ko'rinishi quyidashicha bo'ladi:

$(x > 0) \&\& (y < 1 || y > 3)$.

void turi. void turidagi dastur obyekti hech qanday qiymatga ega bo'lmaydi va bu turdan qurilmaning til sintaksisiga mos kelishini ta'minlash uchun ishlatiladi. Masalan, C++ tili sintaksisi funksiya qiymat qaytarishini talab qiladi. Agar funksiya qiymat qaytarmaydigan bo'lsa, u void kalit so'zi bilan e'lon qilinadi.

Turlangan o'zgarmaslar. Turlangan o'zgarmaslar xuddi o'zgaruvchilardek ishlatiladi va initsializatsiya qilingandan (boshlang'ich qiymat berilgandan) keyin ularning qiymatini o'zgartirib bo'lmaydi.

Turlangan o'zgarmas e'lonida const kalit so'zi, undan keyin o'zgarmas turi va nomi, hamda albatta initsializatsiya qismi bo'ladi. Misol tariqasida turlangan va literal o'zgarmaslardan foydalangan holda radius berilganda aylana yuzasini hisoblaydigan programmani keltiramiz.

```
#include <iostream.h>
int main(){
    const double pi=3.1415;
    const int Radius=3;
    double Square=0;
    Square=pi*Radius*Radius;
    cout<<Square<<"\n";
```



```
return 0;}
```

Dastur bosh funksiyasining boshlanishida ikkita - pi va Radius o'zgarmlari e'lon qilingan. Aylana yuzasini aniqlovchi Square o'zgarma deb e'lon qilinmagan, chunki u dastur bajarilishida o'zgaradi. Aylana radiusini dastur ishlashida o'zgartirish mo'ljallanmagan, shu sababli u o'zgarma sifatida e'lon qilingan.

Sanab o'tiluvchi tur. Ko'p miqdordagi, mantiqan bog'langan o'zgarmlardan foydalanilganda sanab o'tiluvchi turdan foydalanilgani ma'qul. Sanab o'tiluvchi o'zgarmlar enum kalit so'zi bilan aniqlanadi. Mazmuni bo'yicha bu o'zgarmlar oddiy butun sonlardir. Sanab o'tiluvchi o'zgarmlar C++ standarti bo'yicha butun turdagi o'zgarmlar hisoblanadi. Har bir o'zgarмага (songa) mazmunli nom beriladi va bu identifikatorni dasturning boshqa joylarida nomlash uchun ishlatilishi mumkin emas. Sanab o'tiluvchi tur qo'yidagi ko'rinishga ega:

```
enum <sanab o'tiladigan tur nomi> { <nom1>=<qiymat1>,  
    <nom2>=<qiymat2>, ... <nomn>=<qiymatn> };
```

Bu yerda, enum - kalit so'z (inglizcha enumerate - sanamoq); <sanab o'tiladigan tur nomi>- o'zgarmlar ro'yxatining nomi; <nomi> - butun qiymatli konstantalarning nomlari; <qiymati>- shart bo'lmagan initsializatsiya qiymati (ifoda).

Misol uchun hafta kunlari bilan bog'liq masala yechishda hafta kunlarini dush (dushanba), sesh (seshanba), chor (chorshanba), paysh (payshanba), juma (juma), shanba (shanba), yaksh (yakshanba) o'zgarmlarini ishlatish mumkin va ular sanab o'tiluvchi tur yordamida bitta satrda yoziladi:

enum Hafta {dush,sesh,chor,paysh,juma,shanba,yaksh};

Sanab o'tiluvchi o'zgarmlar quyidagi xossaga ega: agar o'zgarma qiymati ko'rsatilmagan bo'lsa, u oldingi o'zgarma qiymatidan bittaga ortiq bo'ladi. Kelishuv bo'yicha birinchi o'zgarma qiymati 0 bo'ladi.

Initsializatsiya yordamida o'zgarma qiymatini o'zgartirish mumkin:

```
enum Hafta {dush=8,sesh,chor=12,paysh=13,juma=16,  
    shanba, yaksh=20};
```

Bu e'londa sesh qiymati 9, shanba esa 17 ga teng bo'ladi. Sanab o'tiluvchi o'zgarmlarning nomlari har xil bo'lishi kerak, lekin ularning qiymatlari bir xil bo'lishi mumkin. O'zgarmaning qiymati ifoda ko'rinishda berilishi mumkin, faqat ifodadagi nomlarning qiymatlari shu qadamdagacha aniqlangan bo'lishi kerak.

Turni boshqa turga keltirish. C++ tilida bir turni boshqa turga keltirishning oshkor va oshkormas yo'llari mavjud.

Umuman olganda, turni boshqa turga oshkormas keltirish ifodada har xil turdagi o'zgaruvchilar qatnashgan hollarda amal qiladi (aralash turlar arifmetikasi). Ayrim hollarda, xususan tayanch turlar bilan bog'liq turga keltirish amallarida xatoliklar yuzaga kelishi mumkin. Masalan, hisoblash natijasidagi sonning xotiradan vaqtincha egallagan joyi uzunligi, uni o'zlashtiradigan o'zgaruvchi uchun ajratilgan joy uzunligidan katta bo'lsa, qiymatga ega razryadlarni yo'qotish holati yuz beradi.

Oshkor ravishda turga keltirishda, o'zgaruvchi oldiga qavs ichida boshqa tur nomi yoziladi:

```
#include <iostream.h>  
int main(){  
    int Integer_1=54;  
    int Integer_2;  
    float Floating=15.854;  
    Integer_1=(int)Floating; // oshkor keltirish;  
    Integer_2=Floating; // oshkormas keltirish;  
    cout<<"Yangi Integer(Oshkor): "<<Integer_1<<"\n";  
    cout<<"Yangi Integer(Oshkormas): "<<Integer_2<<"\n";
```

```
return 0;}
```

2. Arifmetik ifoda va amallar, siljitish amallari, inkrement va decrement, bitlarga ishlov beruvchi operatorlar

Arifmetik amallar. Qiymat berish operatori. Berilganlarni qayta ishlash uchun C++ tilida amallarning juda keng majmuasi aniqlangan. Amal - bu qandaydir harakat bo'lib, u bitta (unar) yoki ikkita (binar) operandlar ustida bajariladi, hisob natijasi uning qaytaruvchi qiymati hisoblanadi.

Tayanch arifmetik amallarga qo'shish (+), ayirish (-), ko'paytirish (*), bo'lish (/) va bo'lish qoldig'ini olish (%) amallarini keltirish mumkin.

Amallar qaytaradigan qiymatlarni o'zlashtirish uchun qiymat berish amali (=) va uning turli modifikatsiyalari ishlatiladi: qo'shish, qiymat berish bilan (+=); ayirish, qiymat berish bilan (-=); ko'paytirish, qiymat berish bilan (*=); bo'lish, qiymat berish bilan (/=); bo'lish qoldig'ini olish, qiymat berish bilan (%=) va boshqalar. Bu holatlarning umumiy ko'rinishi:

<o'zgaruvchi><amal>=<ifoda>;

Quyidagi dastur matnida ayrim amallarga misollar keltirilgan.

```
#include <iostream.h>
int main(){
int a=0,b=4,c=90; char z='t';
a=b; cout<<a<<z; // a=4
a=b+c+c+b; cout<<a<<z; // a= 4+90+90+4 = 188
a=b-2; cout<<a<<z; // a=2
a=b*3; cout<<a<<z; // a=4*3 = 12
a=c/(b+6); cout<<a<<z; // a=90/(4+6) =9
cout<<a%2<<z; // 9%2=1
a+=b; cout<<a<<z; // a=a+b = 9+4 =13
a*=c-50; cout<<a<<z; //a=a*(c-50)=13*(90-50)=520
a-=38; cout<<a<<z; // a=a-38=520-38=482
a%=8; cout<<a<<z; // a=a%8=482%8=2
return 0;}
```

Dastur bajarilishi natijasida ekranga quyidagi sonlar qatori paydo bo'ladi:

4 188 2 12 9 1 482 2

Ifoda tushunchasi. C++ tilida *ifoda* - amallar, operandlar va punktatsiya belgilarining ketma-ketligi bo'lib, kompilyator tomonidan berilganlar ustida ma'lum bir amallarni bajarishga ko'rsatma deb qabul qilinadi. Har qanday ';' belgi bilan tugaydigan ifodaga *til ko'rsatmasi* deyiladi.

C++ tilidagi til ko'rsatmasiga misol

```
x=3*(y-2.45);
y=Summa(a,9,c);
```

Inkrement va dekrement amallari. C++ tilida operand qiymatini birga oshirish va kamaytirish-ning samarali vositalari mavjud. Bular inkrement (++) va dekrement (--) unar amallardir.

Operandga nisbatan bu amallarning prefiks va postfiks ko'ri-nishlari bo'ladi. Prefiks ko'rinishda amal til ko'rsatmasi bo'yicha ish bajarilishidan oldin operandga qo'llaniladi. Postfiks holatda esa amal til ko'rsatmasi bo'yicha ish bajarilgandan keyin operandga qo'llaniladi.

Prefiks yoki postfiks amal tushunchasi faqat qiymat berish bilan bog'liq ifodalarda o'rinni:

```
x=y++; // postfiks
index =--i; // prefiks
count++; // unar amal, "++count;" bilan ekvivalent
abc-- ; // unar amal, "--abc;" bilan ekvivalent
```

Bu yerda y o'zgaruvchining qiymatini x o'zgaruvchisiga o'zlashtiriladi va keyin bittaga oshiriladi, i o'zgaruvchining qiymati bittaga kamaytirib, index o'zgaruvchisiga o'zlashtiriladi.

Razryadli mantiqiy amallar. Dastur tuzish tajribasi shuni ko'rsatadiki, odatda qo'yilgan masalani yechishda biror holat ro'y bergan yoki yo'qligini ifodalash uchun 0 va 1 qiymat qabul qiluvchi bayroqlardan foydalaniladi. Bu maqsadda bir yoki undan ortiq baytli o'zgaruvchilardan foydalanish mumkin. Masalan, bool turidagi o'zgaruvchini shu maqsadda ishlatish bo'ladi. Boshqa tomondan, bayroq sifatida baytning razryadlaridan foydalanish ham mumkin. Chunki razryadlar faqat ikkita qiymatni - 0 va 1 sonlarini qabul qiladi. Bir baytda 8 razryad bo'lgani uchun unda 8 ta bayroqni kodlash imkoniyati mavjud.

Faraz qilaylik, qo'riqlash tizimiga 5 ta xona ulangan va tizim taxtasida 5 ta chiroqcha (indikator) xonalar holatini bildiradi: xona qo'riqlash tizimi nazoratida ekanligini mos indikatorning yonib turishi (razryadning 1 qiymati) va xonani tizimga ulanmaganligini indikator o'chganligi (razryadning 0 qiymati) bildiradi. Tizim holatini ifodalash uchun bir bayt etarli bo'ladi va uning kichik razryadidan boshlab beshtasini shu maqsadda ishlatish mumkin:

7	6	5	4	3	2	1	0
			ind5	ind4	ind3	ind2	ind1

Masalan, baytning quyidagi holati 1, 4 va 5 xonalar qo'riqlash tizimiga ulanganligini bildiradi:

7	6	5	4	3	2	1	0
x	x	x	1	1	0	0	1

Quyidagi jadvalda C++ tilida bayt razryadlari ustida mantiqiy amallar majmuasi keltirilgan.

1.4-jadval. Bayt razryadlari ustida mantiqiy amallar

Amallar	Mazmuni
&	Mantiqiy VA (ko'paytirish)
	Mantiqiy YOKI (qo'shish)
^	Istisno qiluvchi YOKI
~	Mantiqiy INKOR (inversiya)

Razryadli mantiqiy amallarning bajarish natijalarini jadval ko'rinishida ko'rsatish mumkin.

1.5-jadval. Razryadli mantiqiy amallarning bajarish natijalari

A	B	C=A&B	C=A B	C=A^B	C=~A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Yuqoridagi keltirilgan misol uchun qo'riqlash tizimini ifodalovchi bir baytli char turidagi o'zgaruvchini e'lon qilish mumkin:

char q_textasi=0;

Bu yerda q_textasi o'zgaruvchisiga 0 qiymat berish orqali barcha xonalar qo'riqlash tizimiga ulanmaganligi ifodalanadi:

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

Agar 3-xonani tizimga ulash zarur bo'lsa

q_textasi=q_textasi|0x04;

amalni bajarish kerak, chunki 0x0416=000001002 va mantiqiy YOKI amali natijasida q_textasi o'zgaruvchisi bayti quyidagi ko'rinishda bo'ladi:

7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0

Xuddi shunday yo‘l bilan boshqa xonalarni tizimga ulash mumkin, zarur bo‘lsa birdaniga ikkitasini (zarur bo‘lsa barchasini):

q_taxtasi=q_taxtasi|0x1F;

Mantiqiy ko‘paytirish orqali xonalarni qo‘riqlash tizimidan chiqarish mumkin:

q_taxtasi=q_taxtasi&0xFD; // 0xFD16=11111012

Xuddi shu natijani ‘~‘ amalidan foydalangan holda ham olish mumkin. Ikkinchi xona tizimga ulanganligi bildiruvchi bayt qiymati - 000000102, demak shu holatni inkor qilgan holda mantiqiy ko‘paytirishni bajarish kerak.

q_taxtasi=q_taxtasi&(~0x02);

Va nihoyat, agar 3-xona indikatorini, uni qanday qiymatda bo‘lishidan qat‘iy nazar qarama-qarshi holatga o‘tkazishni «inkor qiluvchi YOKI» amali yordamida bajarish mumkin:

q_taxtasi=q_taxtasi^0x04; // 0x0416=000001002

Razryadli mantiqiy amallarni qiymat berish operatori bilan birgalikda bajarilishining quyidagi ko‘rinishlari mavjud:

&= - razryadli VA qiymat berish bilan;

|= - razryadli YOKI qiymat berish bilan;

^= - razryadli istisno qiluvchi YOKI qiymat berish bilan.

Chapga va o‘ngga surish amallari. Baytdagi bitlar qiymatini chapga yoki o‘ngga surish uchun, mos ravishda “<<” va “>>” amallari qo‘llaniladi. Amaldan keyingi son bitlar nechta o‘rin chapga yoki o‘nga surish kerakligini bildiradi.

Masalan:

unsigned char A=12; // A=000011002=0x0C16

A=A<<2; // A=001100002=0x3016=4810

Razryadlarni n ta chapga (o‘nga) surish sonni 2n soniga ko‘paytirish (bo‘lish) amali bilan ekvivalent bo‘lib va nisbatan tez bajariladi. Shuni e‘tiborga olish kerakki, operand ishorali son bo‘lsa, u holda chapga surishda eng chapdagi ishora razryadi takrorlanadi (ishora saqlanib qoladi) va manfiy sonlar ustida bu amal bajarilganda matematika nuqtai-nazardan xato natijalar yuzaga keladi:

char B=-120; // B=100010002=0x8816

B=B<<2; // B=001000002=0x2016=3210

B=-120; // B=100010002=0x8816

B=B>>3; // B=111100012=0xF116=-1510

Shu sababli, bu razryadli surish amallari ishorasiz (unsigned) turdagi qiymatlar ustida bajarilgani ma‘qul.

Taqqoslash amallari. C++ tilida qiymatlarni solishtirish uchun taqqoslash amallari aniqlangan (1.7-jadval). Taqqoslash amali binar amal bo‘lib, quyidagiko‘rinishga ega:

<operand1> <taqqoslash amali> <operand2>

Taqqoslash amallarining natijasi - taqqoslash o‘rinli bo‘lsa, true (rost), aks holda false (yolg‘on) qiymat bo‘ladi. Agar taqqoslashda arifmetik ifoda qatnashsa, uning qiymati 0 qiymatidan farqli holatlar uchun 1 deb hisoblanadi.

1.6-jadval. Taqqoslash amallari va ularning qo‘llanishi

Amallar	Qo‘llanishi	Mazmuni (o‘qilishi)
<	a<b	“a kichik b”
<=	a<=b	“a kichik yoki teng b”
>	a>b	“a katta b”
>=	a>=b	“a katta yoki teng b”

==	a==b	“a teng b”
!=	a!=b	“a teng emas b”

«Vergul» amali

Til qurilmalaridagi bir nechta ifodalarni kompilyator tomonidan yaxlit bir ifoda deb qabul qilishi uchun «vergul» amali qo'llaniladi. Bu amalni qo'llash orqali dastur yozishda ma'lum bir samaradorlikka erishish mumkin. Odatda «vergul» amali if va for operatorlarida keng qo'llaniladi. Masalan, if operatori qo'yidagi ko'rinishda bo'lishi mumkin:

if(i=CallFunc(),i<7)...

Bu yerda, oldin CallFunc() funksiyasi chaqiriladi va uning natijasi i o'zgaruvchisiga o'zlashtiriladi, keyin i qiymati 7 bilan solishtiriladi.

Quyidagi jadvalda C++ tilida ishlatiladigan amallar (operatorlar), ularning ustunlik koefitsientlari va bajarilish yo'nalishlari (<= - o'ngdan chapga, => - chapdan o'ngga) keltirilgan.

1.7-jadval. Amallarning ustunliklari va bajarilish yo'nalishlari

Operator	Tavsifi	Ustunlik	Yo'nalish
::	Ko'rinish sohasiga ruxsat berish	16	=>
[]	Massiv indeksi	16	=>
()	Funksiyani chaqirish	16	=>
.	Struktura yoki sinf elementini tanlash	16	<=
->			
++	Postfiks inkrement	15	<=
--	Postfiks dekrement	15	<=
++	Prefiks inkrement	14	<=
-	Prefiks dekrement	14	<=
sizeof	O'lchamni olish	14	<=
(<tur>)	Turga akslantirish	14	
~	Razryadli mantiqiy INKOR	14	<=
!	Mantiqiy inkor	14	<=
-	Unar minus	14	<=
+	Unar plyus	14	<=
&	Adresni olish	14	<=
*	Vositali murojaat	14	<=
new	Dinamik obyekttni yaratish	14	<=
delete	Dinamik obyekttni yo'q qilish	14	<=
casting	Turga keltirish	14	
*	Ko'paytirish	13	<=
/	Bo'lish	13	<=
%	Bo'lish qoldig'i	13	<=
+	Qo'shish	12	<=
-	Ayirish	12	<=
>>	Razryad bo'yicha o'ngga surish	11	<=
<<	Razryad bo'yicha chapga surish	11	<=
<	Kichik	10	<=
<=	Kichik yoki teng	10	<=
>	Katta	10	<=
>=	Katta yoki teng	10	<=
==	Teng	9	<=
!=	Teng emas	9	<=
&	Razryadli VA	8	<=
^	Razryadli istisno qiluvchi YOKI	7	<=
	Razryadli YOKI	6	<=

&&	Mantiqiy VA	5	<=
	Mantiqiy YOKI	4	<=
?:	SHart amali	3	<=
=	Qiymat berish	2	<=
*=	Ko'paytirish qiymat berish bilan	2	<=
/=	Bo'lish qiymat berish bilan	2	<=
%=	Modulli bo'lish qiymat berish bilan	2	<=
+=	Qo'shish qiymat berish bilan	2	<=
- =	Ayirish qiymat berish bilan	2	<=
<<=	CHapga surish qiymat berish bilan	2	<=
>>=	O'ngga surish qiymat berish bilan	2	<=
&=	Razryadli VA qiymat berish bilan	2	<=
^=	Razryadli istisno kiluvchi YOKI qiymat berish bilan	2	<=
=	Razryadli YOKI qiymat berish bilan	2	<=
throw	Istisno holatni yuzaga keltirish	2	<=
,	Vergul	1	<=

C++ tili dastur tuzuvchisiga amallarning bajarilish tartibini o'zgartirish imkoniyatini beradi. Xuddi matematikadagidek, amallarni qavslar yordamida guruhlariga jamlash mumkin. Qavs ishlatishga cheklov yo'q.

3.Kutubxona funksiyalari

C++ tilidagi dastur quyidagi tarkibdan tashkil topadi:

Direktivalar – # include <file.h> direktiva – instruksiya degan ma'noni beradi. C++ tilida dasturning tuzilishiga, ya'ni ehtiyojiga qarab, kerakli direktivalar ishlatiladi. Ular < > belgisi orasida keltiriladi. Umuman olganda quyidagi direktivalar mavjud (jami 32 ta):

- #include <stdio.h> - S da oddiy kiritish/chiqarish dasturi uchun. Bu yerda std - standart, i – input, o - output degani.
- #include <iostream.h> - C++ da kiritish/chiqarish uchun, oddiy amallar bajarilsa.
- #include <math.h> - standart funksiyalarni ishlatish uchun.
- #include <conio.h> - dasturning tashqi ko'rinishini shakllantirish uchun.
- #include <string.h> - satr toifasidagi o'zgaruvchilar ustida amallar bajarish uchun.
- #include <stdlib.h> - standart kutubxona fayllarini chaqirish uchun.
- #include <time.h> - kompyuter ichidagi soat qiymatlaridan foydalanish uchun.
- #include <graphics.h> - C++ tilining grafik imkoniyatlaridan foydalanish uchun.

Bu fayllar maxsus kutubxona e'lon fayllari hisoblanadilar va ular aloxida INCLUDE deb nomlanadigan papkada saqlanadi. Hozirda C++ kutubxonasini yangilandi va undagi fayllarning nomlaridan .h (head – bosh ma'nosida) kengaytmasi olib tashlandi va oldiga c harfi qo'shildi (C dan qolgan 18 tasiga). Bu fayllarda funksiya prototoifalari, toifalari, o'zgaruvchilar, o'zgaruvchilar ta'riflari yozilgan bo'ladi.

Direktivalar dasturni uni kompilyasiya qilinishidan oldin tekshirib chiqadi.

2. Makroslar - # define makro qiymati. Masalan:

```
#define y sin(x+25) – u = sin(x+25) qiymati berildi;
```

```
#define pi 3.1415 - pi = 3.1415
```

```
#define s(x) x*x - s(x) = x*x (; belgisi qo'yilmaydi)
```

Global o'zgaruvchilarni e'lon qilish. Asosiy funksiya ichida e'lon qilingan o'zgaruvchilar lokal, funksiya tashqarida e'lon qilinganlari esa global o'zgaruvchilar deyiladi. Global o'zgaruvchilar dastur davomida ishlaydi va xotiradan ma'lum joyni egallaydi. O'zgaruvchini bevosita

ishlatishdan oldin e'lon qilsa ham bo'ladi, u holda o'z lokal bo'ladi. Global o'zgaruvchilar nomi lokal o'zgaruvchilar nomi bilan bir xil bo'lishi ham mumkin. Bunday holatda lokal o'zgaruvchining qiymati joriy funksiya ichidagini qiymatini o'zgartiradi, funksiyadan chiqishi bilan global o'zgaruvchilar ishlaydi.

Asosiy funksiya - `main ()` hisoblanadi. Bu funksiya dasturda bo'lishi shart. Umuman olganda C++ dagi dastur funksiyalardan iborat deb qaraladi. `main ()` funksiyasi { boshlanadi va dastur oxirida berkitilishi shart } . `main` – asosiy degan ma'noni beradi. Bu funksiya oldida uning toifasi ko'rsatiladi. Agar `main ()` funksiyasi beradigan (qaytaradigan) javob oddiy so'z yoki gaplardan iborat bo'lsa, hech qanday natija qaytarmasa, `void` so'zi keltiriladi. `main ()` funksiyasi dastur tomonidan emas, balki OS tomonidan chaqiriladi. OSga qiymat qaytarish shart emas, chunki u bu qiymatdan foydalanmaydi. Shuning uchun `main ()` funksiyasining turini void deb ko'rsatganimiz ma'qul. Har bir funksiyaning o'z argumenti bo'ladi, shuning uchun `main` funksiya () lari ichiga uning parametri keltiriladi. Ba'zan u bo'sh bo'lishi ham mumkin. Bu funksiyadan chiqish uchun odatda return operatori ishlatiladi. 0 (nol) qiymatining qaytarilishi operasion tizimga ushbu dastur normal bajarilib turganini bildiradi. return orqali qaytadigan qiymat toifasi funksiya e'lonidagi qaytish toifasi bilan bir xil bo'lishi kerak. Masalan `int main ()` va 0 (nol) qiymat butun toifalidir. Bu funksiyadan so'ng lokal o'zgaruvchilar, qism dasturlar, ularning haqiqiy parametrlar e'lon qilinadi. So'ngra dasturning asosiy operatorlari (kiritish/chiqarish, hisoblash va h.k.) yoziladi. Agar bu operatorlar murakkab toifali bo'lsalar, ularni alohida {} qavslarga olinadi. C++ tilida dastur kichik harflarda yoziladi. Ba'zi operatorlar katta harflar bilan kelishi mumkin, bunday xollarda ular alohida aytib o'tiladi. Operatorlar oxiriga ; belgisi qo'yiladi. Operatorlar bir qatorga ketma-ket yozilishi mumkin. Dasturda izohlar xam kelishi mumkin, ular /**/ belgisi orasiga olinadi. Agar izoh bir qatorda tugasa, uni // belgisidan keyin yoziladi. Masalan:

`main () // C++ tilining asosiy funksiyasi/`

Ifodalarda matematik funksiyalar ham qatnashishi mumkin. Bunday ifodlaning qiymatini hisoblash uchun C++ tilida matematk funksiyalar kutubxonsi `mat.h` mavjud.

C++ tilida matematik standart funksiyalarning yozilishi

Funksiya	Ifodalanishi	Funksiya	Ifodalanishi
$\sin x$	<code>sin(x)</code>	\sqrt{x}	<code>sqrt(x); pow(x,1/2.)</code>
$\cos x$	<code>cos(x)</code>	$ x $	<code>abs(x)</code> yoki <code>fabs(x)</code>
$\operatorname{tg} x$	<code>tan(x)</code>	$\operatorname{arctg} x$	<code>atan(x)</code>
e^x	<code>exp(x)</code>	$\arcsin x$	<code>asin(x)?</code>
$\ln x$	<code>log(x)</code>	$\arccos x$	<code>acos(x)</code>
$\lg x$	<code>log10(x)</code>	x^3	<code>pow(x,3)</code>
x^a	<code>pow(x,a)</code>	$\log_2 x$	<code>log(x)/log(2)</code>

Masalan: $\frac{-b + \sqrt{b^2 - 4ac}}{2a} \rightarrow (-b + \operatorname{sqrt}(b*b-4*a*c))/(2*a);$ yoki

$(-b + \operatorname{pow}(b*b-4*a*c, 1/2.))/(2*a);$

$e^{\sin x} + \operatorname{tg}^2(x+3) \rightarrow \operatorname{exp}(\sin(x)) + \operatorname{pow}(\tan(x+3), 2);$

$k = (m*5) + ((7 \% n) / (9+x));$

Yuqoridagi standart funksiyalardan tashqari yana quyidagi funksiyalar ham ishlatiladi:

- `ceil (x)` - x ni x dan katta yoki unga teng bo'lgan eng kichik butun songacha yaxlitlash.
Masalan: `ceil (12.6) = 13.0`; `ceil (-2.4) = -2.0`;

- floor (x) - x ni x dan kichik bo'lgan eng katta butun songacha yaxlitlash. Masalan: floor (4.8) = 4.0; floor (-15.9) = -16.0; floor(12.1) = 12; floor(-12.1)=-13;
- fmod (x,y) – x / y ning qoldig'ini kasr son ko'rinishida berish. Masalan: fmod(7.3, 1.7) = 0.5;

4. Preprotssessor direktivalari va vositalari

Sodda dastur tuzilishi. Dastur preprotssessor komandalari va bir necha funksiyalardan iborat bo'lishi mumkin. Bu funksiyalar orasida main nomli asosiy funksiya bo'lishi shart. Agar asosiy funksiyadan boshqa funksiyalar ishlatilmasa dastur quyidagi ko'rinishda tuziladi:

Preprotssessor_komandalari

int main() { //Dastur tanasi }

Preprotssessor direktivalari kompilyatsiya jarayonidan oldin preprotssessor tomonidan bajariladi. Natijada dastur matni preprotssessor direktivalari asosida o'zgartiriladi.

Preprotssessor komandalaridan ikkitasini ko'rib chiqamiz.

include <fayl_nomi> Bu direktiva standart bibliotekalardagi funksiyalarni dasturga joylash uchun foydalaniladi.

#define <almashtiruvchi ifoda> <almashinuvchi ifoda>

Bu direktiva bajarilganda dastur matnidagi almashtiruvchi ifodalar almashinuvchi ifodalarga almashtiriladi.

Misol tariqasida C ++ tilida tuzilgan birinchi dasturni keltiramiz:

```
#include <iostream.h>
int main(){
cout << "\n Salom, Dunyo! \n";}
```

Bu dastur ehkranga Salom, Dunyo! Jumlasini chiqaradi.

Define direktivasi yordamida bu dasturni quyidagicha yozish mumkin:

```
#include <iostream.h>
#define pr cout << "\n Salom, Dunyo! \n"
#define begin {
#define end }
int main()
begin
pr;
end
```

Define direktivasidan nomlangan konstantalar kiritish uchun foydalanish mumkindir.

Misol uchun:

#define EULER 2.718282

Agar dasturda quyidagi matn mavjud bo'lsin:

Double mix=EULER

D=alfa*EULER

Preprotssessor bu matnda har bir EULER konstantani uning qiymati bilan almashtiradi, va natijada quyidagi matn hosil bo'ladi.

Double mix=2.718282

D=alfa*2.718282

Dastur matni va preprotssessor. C ++ tilida matnli fayl shaklida tayyorlangan dastur uchta qayta ishlash bosqichlaridan o'tadi. Matnni preprotssessor direktivalari asosida o'zgartilishi. Bu jarayon natijasi Yana matnli fayl bo'lib preprotssessor tomonidan bajariladi.

Kompilyatsiya. Bu jarayon natijasi mashina kodiga o'tkazilgan obektli fayl bo'lib, kompilyator tomonidan bajariladi.

Bog'lash. Bu jarayon natijasi to'la mashina kodiga o'tkazilgan bajariluvchi fayl bo'lib, boglagich(komponent) tomonidan bajariladi.

Preprocessor vazifasi dastur matnini preprocessor direktivalari asosida o'zgartirishdir. Define direktivasi dasturda bir jumlaning ikkinchi jumla bilan almashtirish uchun ishlatiladi. Bu direktivadan foydalanishning sodda misollarini biz yuqorida ko'rib chiqdik. Include direktivasi ikki ko'rinishda ishlatilishi mumkin.

#include fayl nomi direktivasi dasturning shu direktiva urniga qaysi matnli fayllarni qo'shish kerakligini ko'rsatadi.

#include <fayl nomi> direktivasi dasturga kompilyator standart bibliotekalariga mos keluvchi sarlavhali fayllar matnlarini qo'shish uchun muljhallangandir. Bu fayllarda funksiya prototipi, tiplar, o'zgaruvchilar, konstantalar tariflari yozilgan buladi. Funksiya prototipi funksiya qaytaruvchi tip, funksiya nomi va funksiyaga uzatiluvchi tiplardan iborat bo'ladi. Misol uchun cos funkciyasi prototipi quyidagicha yozilishi mumkin: double cos(double). Agar funkciya nomidan oldin void tipi ko'rsatilgan bo'lsa bu funksiya hech qanday qiymat qaytarmasligini ko'rsatadi. Shuni ta'kidlash lozimki bu direktiva dasturga standart biblioteka qo'shilishiga olib kelmaydi. Standart funksiyalarning kodlari bog'lash ya'ni aloqalarni tahrirlash bosqichida, kompilyatsiya bosqichidan so'ng amalga oshiriladi.

Kompilyatsiya bosqichida sintaksis xatolar tekshiriladi va dasturda bunday xatolar mavjud bo'lmasa, standart funksiyalar kodlarisiz mashina kodiga utkaziladi. Sarlavhali fayllarni dasturning ixtiyoriy joyida ulash mumkin bo'lsa ham, bu fayllar odatda dastur boshida qo'shish lozimdir. Shuning uchun bu fayllarga sarlavhali fayl (header file) nomi berilgandir.

Dasturda kiritish va chiqarish funksiyalaridan masalan Cout<< funksiyasidan foydalanish uchun #include <iostream.h> direktivasidan foydalanish lozimdir Bu direktivada iostream.h sarlavhali fayl nomi quyidagilarni bildiradi: st- standart(standartni), i- input, o- output, h – head(sarlavha).

Muhokama savollari

1. Tilning alifbosi.
2. O'zgarmaslar
3. O'zgaruvchilarning toifalari.
4. Standart funksiyalarning ko'rinishi.

Nazorat savollari:

1. C/C++ tilida o'zgarmaslar.
2. C/C++ tilida o'zgaruvchilarning toifalari
3. Kompanovka bosqichlarini ayting.
4. Standart funksiyalarning qo'llanishi.
5. Ifodalar haqida tushuncha.
6. Dastur tuzilishi.
7. Preprocessor direktivalari
8. Identifikator, o'zgaruvchilar va o'zgarmaslar.
9. O'zgaruvchilarning oddiy toifalari.
10. Amallar va ifodalar.

2-MA'RUZA

MAVZU: ALGORITMLASH VA DASTURLASHNING ASOSIY TUSHUNCHALARI

Reja:

1. Kompilyator va uning turlari
2. Unar va binary operatorlari. Qiymat o'zlashtirish operatorlari va ularning ishlash usullari
3. Ternar operatori. sizeof operatori
4. Format modifikatorlari: printf(), scanf() funksiyalari

Kalit so'zlar: *toifalar, xatolik, sintaktik xatolik, dastur ishlashi davomidagi xatolik, testlash, kompilyatsiya paytidagi xatolik, talab, mantiqiy xato, turlar xatosi.*

1. Kompilyator va uning turlari

Kompilyator bu – dastur tuzish uchun yangi kodlarning qonun qoida bo'yicha terilganligini nazorat qiluvchi va dasturning natijasini chiqaruvchi amaliy dasturdir.

Kompilyator turlari:

1. Dev;
2. CodeBlocks;
3. Visual Studio;
4. Borland C++Builder;
5. EmbarCadero.

C++ tilida kata va kichik harflarning farqi bor. Bundan tashqari kalit so'zlar ham bor. Kompilyatorlarni turlari va versiyalariga qarab har hil hatoliklar kelib chiqishi mumkin:

- 1- kalit so'zlarni noto'g'ri ishlatish;
- 2- o'zgaruvchilarni yaratish va foydalanishda;
- 3- ingliz tilini bilish darajasiga ham bog'liq;
- 4- operatorlarni noto'g'ri ishlatish;
- 5- kutubxonalardan foydalanishda.

Preprotessor direktivalari. Preprotessor direktivalari kompilyatsiya jarayonidan oldin preprotessor tomonidan bajariladi. Natijada dastur matni preprotessor direktivalari asosida o'zgartiriladi.

#include <fayl_nomi> bu direktiva standart bibliotekalardagi funksiyalarni dasturga joylash uchun foydalaniladi.

#define <almashtiruvchi ifoda> <almashinuvchi ifoda>

Bu direktiva bajarilganda dastur matnidagi almashtiruvchi ifodalar almashinuvchi ifodalarga almashtiriladi. Misol:

```
#include <stdio.h>
#define begin {
#define end }
#define pr printf("\n Dasturlash \n");
int main(){
begin
pr;
end;}
```

Almashtiruvchi **define** direktivasidan nomlangan konstantalar kiritish uchun foydalanish mumkindir.

Misol uchun:

#define ZERO 0

Agar dasturda quyidagi matn mavjud bo'lsin:

int d = ZERO;

Preprosessor bu matnda har bir **ZERO** konstantani uning qiymati bilan almashtiradi, va natijada quyidagi matn hosil bo'ladi.

int d = 0;

Preprosessorlarni boshqarish

- oldindan tayyorlangan simvollar ketma ketligi bilan identifikatorlarni almashtirish ;
- ko'rsatilgan fayldagi matnni dasturga ulash(bog'lash) ;
- dasturdan ba'zi qismlarni olib tashlash (shartli kompilyasiya) .

Preprosessor direktivalari:

1. **#define** - makrosning aniqlanishi yoki preprosessorning identifikatori ;
2. **#include** - fayldan tekstni o'qish ;
3. **#undef** - identifikatorni va makrosni aniqlanishini bekor qilish;
4. **#if** -shart ifodani tekshirish;
5. **#ifdef** - identifikator aniqlanishini tekshirish;
6. **#ifndef** - identifikator noaniqligini aniqlash;
7. **#else** - #if uchun alternativ tarmoqning boshlanishi;
8. **#endif** - shart direktivasi #if ning oxiri;
9. **#elif** - tarkibiy direktiva #else/#if;
10. **#line** - keyingi satr nomerini almashtirish;
11. **#error** - translatsiya xatosi haqida xabarni formatlashtirish;
12. **#pragma** – oldindan aniqlangan amallar;
13. **#** -bo'sh direktivalar.

2. Unar va binary operatorlari. Qiymat o'zlashtirish operatorlari va ularning ishlash usullari

Amallar odatda unar ya'ni bitta operandga qo'llaniladigan amallarga va binar ya'ni ikki operandga qo'llaniladigan amallarga ajratiladi.

Binar amallar additiv ya'ni + qo'shuv va – ayirish amallariga , hamda multiplikativ ya'ni * kupaytirish, / bulish va % modul olish amallariga ajratiladi.

Additiv amallarining ustivorligi multiplikativ amallarining ustivorligidan pastroqdir.

Butun sonni butun songa bo'lganda natija butun songacha yahlitlanadi. Misol uchun $20/3=6$; $(-20)/3=-6$; $20/(-3)=-6$.

Modul amali butun sonni butun songa bulishdan hosil buladigan qoldikka tengdir. Agar modul amali musbat operandlarga qo'llanilsa, natija ham musbat bo'ladi, aks holda natija ishorasi kompilyatorga bog'lidir.

Binar arifmetik amallar bajarilganda tiplarni keltirish quyidagi qoidalar asosida amalga oshiriladi:

- short va char tiplari int tipiga keltiriladi;
- Agar operandlar biri long tipiga tegishli bo'lsa ikkinchi operand ham long tipiga keltiriladi va natija ham long tipiga tegishli buladi;
- Agar operandlar biri float tipiga tegishli bulsa ikkinchi operand kham float tipiga keltiriladi va natija ham float tipiga tegishli buladi;
- Agar operandlar biri double tipiga tegishli bo'lsa ikkinchi operand ham double tipiga keltiriladi va natija ham double tipiga tegishli buladi;
- Agar operandlar biri long double tipiga tegishli bo'lsa ikkinchi operand ham long double tipiga keltiriladi va natija ham long double tipiga tegishli bo'ladi;

Unar amallarga ishorani o'zgartiruvchi unar minus – va unar + amallari kiradi. Bundan tashqari ++ va -- amallari ham unar amallarga kiradi.

++ unar amali qiymatni 1 ga oshirishni ko'rsatadi. Amalni prefiks ya'ni ++i ko'rinishda ishlatish oldin o'zgaruvchi qiymatini oshirib so'ngra foydalanish lozimligini, postfiks ya'ni i++ ko'rinishda ishlatish oldin o'zgaruvchi qiymatidan foydalanib so'ngra oshirish kerakligini ko'rsatadi. Misol uchun i qiymati 2 ga teng bo'lsin, u holda 3+(++) ifoda qiymati 6 ga, 3+i++ ifoda qiymati 5 ga teng bo'ladi. Ikkala holda ham i qiymati 3 ga teng bo'ladi.

-- unar amali qiymatni 1 ga kamaytirishni ko'rsatadi. Bu amal ham prefiks va postfiks ko'rinishda ishlatilishi mumkin. Bu ikki amalni faqat o'zgaruvchilarga qo'llash mumkindir.

Unar amallarning ustivorligi binar amallardan yuqoridir.

Qiymat berish amali. Qiymat berish amali = binar amal bo'lib chap operandi odatda o'zgaruvchi ung operandi odatda ifodaga teng bo'ladi. Misol uchun $Z=4.7+3.34$

Bu qiymati 8.04 ga teng ifodadir. Bu qiymat Z o'zgaruvchiga ham beriladi.

Bu ifoda ohiriga nuqta vergul ; belgisi quyilganda operatorga aylanadi.

$Z=4.7+3.34$

Bitta ifodada bir necha qiymat berish amallari qo'llanilishi mumkin. Misol uchun:

$C=y=f=4.2+2.8;$

Bundan tashqari C ++ tili da murakkab qiymat berish amali mavjud bo'lib, umumiy ko'rinishi quyidagichadir:

O'zgaruvchi_nomi amal= ifoda;

Bu erda amal quyidagi amallardan biri *,/,%,+,-, &,^,|, <<,>>.

Misol uchun:

$x+=4$ ifoda $x=x+4$ ifodaga ekvivalentdir;

$x*=a$ ifoda $x=x*a$ ifodaga ekvivalentdir;

$x/=a+b$ ifoda $x=x/(a+b)$ ifodaga ekvivalentdir;

$x>>=4$ ifoda $x=x>>4$ ifodaga ekvivalentdir;

3. Ternar operatori. sizeof operatori

Shartli amal. Shartli amal ternar amal deyiladi va uchta operanddan iborat bo'ladi.

Ternar operatori quyidagi shaklga ega:

$<1\text{-ifoda}>?<2\text{-ifoda}>:<3\text{-ifoda}>$

Shartli amal bajarilganda avval 1- ifoda hisoblanadi. Agar 1-ifoda qiymati 0 dan farqli bo'lsa 2- ifoda hisoblanadi va qiymati natija sifatida qabul qilinadi, aks holda 3-ifoda hisoblanadi va qiymati natija sifatida qabul qilinadi.

Misol uchun modulni hisoblash:

$x<0?-x:x;$ yoki

ikkita son kichigini hisoblash

$a<=b ? a:b;$

Shuni aytish lozimki shartli ifodadan har qanday ifoda sifatida foydalanish mumkin. Agar F float tipga, a N – int tipga tegishli bo'lsa ,

$(N > 0) ? F : N$

ifoda N musbat eki manfiyligidan qat'iy nazar double tipga tegishli bo'ladi.

Shartli ifodada birinchi ifodani qavsga olish shart emas.

Sizeof operatori. Har xil turdagi o'zgaruvchilar kompyuter xotirasida har xil sondagi baytlarni egallaydi. Bunda, hattoki bir turdagi o'zgaruvchilar ham qaysi kompyuterda va qaysi operatsion tizimda amal qilishiga qarab har xil olchamdagi xotirani band qilishi mumkin.

C++ tilida ixtiyoriy turdagi (tayanch va hosilaviy turdagi) o'zgaruvchilarning olchamini sizeof operatori yordamida aniqlanadi. Bu operator konstantaga, turga va o'zgaruvchiga qo'llanishi mumkin.

Quyidagi dastur kompyuterning konkret platformasi uchun tayanch turlarning o'lchamlarini chop qiladi.

```
cout<<"int turining o'lchami:" << sizeof(int)<<"\n";
cout<<"float turining o'lchami:" << sizeof(float)<<"\n";
cout<<"double turining o'lchami:" << sizeof(double)<<"\n";
cout<<"char turining o'lchami:" << sizeof(char)<<"\n";
```

Dastur bajarilishi natijasida *sizeof* operatori yordamida mos turlarning o'lchamlari hisoblanadi va ekranga chop etiladi.

***sizeof* amali operand** sifatida ko'rsatilgan ob'ektning baytlarda xotiradagi hajmini hisoblash uchun ishlatiladi.

Bu amalning ikki ko'rinishi mavjud:

- sizeof ifoda;
- sizeof (tip)

Shuni ta'kidlab o'tish lozimki sizeof funksiyasi preprotssessor qayta ishlash jarayonida bajariladi, shuning uchun dastur bajarilish jarayonida vaqt talab etmaydi.

Misol uchun:

```
#include <stdio.h>
#include <iostream>

using namespace std;
int main() {
    printf("%lu\n", sizeof(char));
    printf("%lu\n", sizeof(int));
    printf("%lu\n", sizeof(float));
    printf("%lu", sizeof(double));
    getchar();
    return 0; }
```

Natija quyidagicha bo'ladi

```
1
4
4
8
```

Ifoda qavslarsiz yoki ko'rsatilmasdan belgilanishi mumkin.

```
// First type
sizeof expression
// Second type
sizeof(expression)
```

Ifoda faqat baholashni emas, balki operanda turini olish uchun ishlatiladi. Masalan, quyidagi kod i ning qiymatini 5 sifatida va ia hajmini ko'rsatadi

```
#include <stdio.h>
int main()
{
    int i = 5;
    int int_size = sizeof(i++);
    // Displaying the size of the operand
    printf("\n size of i = %d", int_size);
    // Displaying the value of the operand
    printf("\n Value of i = %d", i);
    getchar();
}
```

```
return 0; }
```

Natija:

hajmi i = 4

qiymati i = 5

4. Format modifikatorlari: printf(), scanf() funksiyalari

Standart C konsoli funksiyasi printf. Uning tavsifi stdio.h sarlavha faylida mavjud. Ushbu funksiya yordamida siz konsolga ma'lumotlarni yoki maxsus xabarlarni chiqarishingiz mumkin. S tilidagi printf guruhining funksiyalari ma'lumotlarni standart oqimga chiqarishni qayta ishlash va formatlash uchun ishlatiladi.

C tilida belgilar printf funksiyasini chaqirish orqali standart chiqishdan chop etiladi. C printf buyrug'i chiqish ma'lumotlari to'plamini formatlaydi va uni stdout-ga yuboradi. Funktsiyaga argument sifatida berilgan qiymatlar o'z navbatida ikki turdagi elementlarni o'z ichiga olgan belgilangan format qatoriga muvofiq konsolga chop etiladi. Birinchi tur - ekranda ko'rsatilgan belgilar va ma'lumotlar formatining xususiyatlarini aniqlaydigan va chiqishda argumentlarni taqdim etish usuli uchun javob beradigan elementlar ikkinchi turga tegishli.

int tipidagi printf funksiyasi ekranda chop etilgan belgilar sonini ifodalovchi butun son qiymatini qaytaradi. Masalan, siz quyidagilarni belgilashingiz mumkin:

```
int k = printf ("Salom% c% d% s", 'a', 11, "hammaga!"),
```

va keyin k o'zgaruvchining qiymati bo'yicha chiqishda xatolik yuz berganligini aniqlash oson. Agar manfiy qiymat qaytarilsa (agar funksiya "-1" qaytarsa), biz uni bajarishda xatolik yuz berdi degan xulosaga kelishimiz mumkin.

Printf funksiyasidan foydalanish uchun siz "stdio.h" sarlavha faylini quyidagi tarzda kiritishingiz kerak:

```
#include <stdio.h>
```

Funktsiya shabloni quyidagicha ko'rinadi:

```
int printf (const char * formati, ...)
```

Ellips chop etilishi kerak bo'lgan argumentlar ro'yxatiga ishora qiladi. Printf funksiyasidan har xil argumentlar soni bilan foydalanish mumkin, lekin birinchisi har doim ikkala tomondan qo'sh tirnoq bilan ajratiladi va har bir keyingisi oldingi verguldan ajratilishi kerak. Ikki tirnoq ichida yozilgan va format spetsifikatsiyasi bo'lmagan narsa o'zgarmagan holda chop etiladi, aks holda spetsifikatorga duch kelsangiz, uning qiymatining turi nusxalanadi.

Format spetsifikatsiyalarini o'rnatish uchun shakl:

```
% [bayroqlar] [kenglik] [.pozitsiya] [uzunlik] turi
```

Chaquirilayotgan funksiya nomidan keyin qavs ichida ko'rsatilgan format satrini o'qish faqat bitta yo'nalishda sodir bo'ladi: chapdan o'ngga va ushbu satrdan keyin ko'rsatilgan birinchi argumentning o'zi faqat birinchi spetsifikatsiyaga duch kelgan taqdirda chiqariladi. Format satri tugamaguncha, unda ko'rsatilgan spetsifikatsiyalar keyingi argumentlarni konvertatsiya qilish va chop etishni boshlaydi. Format satrida bo'sh joy belgisi oddiy belgi sifatida ko'rib chiqiladi va u format spetsifikatsiyasi ifodasida ishlatilmaganda chiqishga uzatiladi.

"%" belgisi chiqish formati spetsifikatsiyasining boshlanishini, keyin esa format kodini bildiradi. Spetsifikatsiyadagi barcha maydonlar alohida bo'lib, raqamlar yoki belgilar uchun formatlash shartlarini belgilaydi.

C tilida formatlangan printf chiqishi o'ziga xos xususiyatlarga ega. Agar sanab o'tilgan argumentlar soni format spetsifikatsiyalari sonidan oshsa, ular o'tkazib yuboriladi va ko'rsatilmaydi. Aks holda, chop etiladigan argumentlar ro'yxatidagi qiymatlardan ko'ra ko'proq format spetsifikatsiyalari mavjud bo'lsa, funksiya chaqiruvining natijasi aniqlanmagan.

Qaysi argumentni ketma-ketlikda ishlatish kerakligini aniq ko'rsatish uchun "%" o'rniga "% m \$" va " *" o'rniga " * m \$" dan foydalanish mumkin va m, butun o'nlik qiymatini bildiradi. kerakli argumentning pozitsiyasi (indekslash birliklardan boshlanadi).

Format spetsifikatsiyasi	Printf C uchun foydalanish va tavsif	Argument turi
%	Harf yozuvi "%"	
c	Bitta belgining chiqishi. Argument unsigned char turiga aylantiriladi. "l" modifikatoridan foydalanilganda argument belgilar qatoriga aylantiriladi	imzosiz belgi
s	Belgilar qatorini chop etadi. Argument char massivining boshlang'ich elementiga ko'rsatgich bo'lishi kerak	char *
d i	Belgilangan butun son qiymatining o'nli ko'rinishini chiqarish	int
o	Butun qiymatning ishorasiz sakkizlik tasvirini chiqarish	imzosiz int
x X	Belgilanmagan butun qiymatning o'n oltilik ko'rinishini ko'rsatadi. "A", "b", "c", "d", "e", "f" belgilari "x" ni aylantirish uchun ishlatiladi. Va "X" ni aylantirish uchun - "A", "B", "C", "D", "E", "F"	imzosiz int
u	Belgilangan tamsayı qiymatisiz kasrli ayirboshlashni chiqaradi. Agar o'zgartirilgan qiymat va aniqlik 0 bo'lsa, hech qanday belgi chiqmaydi	imzosiz int
f F	Imzolangan suzuvchi nuqta sonining kasrli ko'rinishini ko'rsatish	ikki barobar
e E	O'nli kasr sonining o'nlik eksponensial ko'rinishining chiqishi, yaxlitlanadi va o'nli kasrdan oldin bitta raqam qolishi va o'nli kasrdan keyingi raqamlar soni tasvirlash aniqligiga to'g'ri keladi (sukut bo'yicha, aniqlik 6, va agar 0 belgilansa, vergul belgisi umuman ko'rsatilmaydi). "e" belgisi konversiyaga qarab katta yoki kichik harflarda ko'rsatiladi	ikki barobar
a A	O'zgaruvchan nuqtali raqamning o'n oltilik ko'rinishini ko'rsatish	ikki barobar
g G	Qiymat va aniqlikka qarab suzuvchi nuqta sonining o'nli ko'rinishini yoki uning o'nli eksponensial ko'rinishini ko'rsating	ikki barobar
n	printf tomonidan chop etilgan elementlar sonini qaytaradi. Natija argument tomonidan ko'rsatilgan o'zgaruvchiga yoziladi. BOMda bayroqlar, maydon kengligi yoki aniqligi bo'lmasligi mumkin	int *
p	Pointer chiqishi	

Maydon kengligi modifikatori

printf C formatidagi format qatori foiz belgisidan keyin va format buyrug'idan oldin butun sonni o'z ichiga olishi mumkin. U maydon kengligini o'zgartiradi va ko'rsatilgan ma'lumotlarning taqdimotiga ta'sir qiladi. Qiymat uchun mo'ljallangan eng kichik maydon kengligi ushbu raqam bilan aniqlanadi va bunday modifikatorning mavjudligi, agar argument unga ajratilgan maydondan kichik bo'lsa, natijaga bo'shliqlar yoki nollarning qo'shilishiga olib keladi. Standart to'ldiruvchi bo'sh joy belgisidir, lekin siz uni kenglik spetsifikatsiyasi bilan prefikslash orqali uni nolga o'rnatishingiz mumkin. Modifikator minimal kenglikni belgilaydi va bu minimaldan oshadigan har qanday qiymat tartibsizliklarsiz chop etiladi. Masalan, sakkiz belgidan kam bo'lgan va "% 08d" spetsifikatsiyasi bilan chop etilgan raqam kerakli sakkiz belgigacha nol bilan to'ldiriladi.

Bunday modifikatorlar aniqlik yoki tekislash variantlarini ham belgilashi mumkin.

Aniqlik modifikatori

Aniq modifikator sonli ko'rinishda chop etish uchun kasrlar sonini aniqlash uchun ishlatiladi. Aniqlik modifikatorini qo'shish uchun maydon kengligi spetsifikatsiyasidan keyin nuqta qo'yish va undan keyin kerakli aniqlik qiymatini belgilash kerak. Aniq modifikator "e", "f", "a", "E", "A" va "F" formatlari uchun belgilangan. Butun sonlar uchun modifikator ko'rsatiladigan raqamlar sonini o'rnatadi, agar kerak bo'lsa, chap raqamga nol qo'shadi va ratsional sonlarni ko'rsatishda o'nli kasrlarning kerakli sonini aniqlaydi. Satr o'zgaruvchilari bilan bog'liq holda: aniq o'zgartirgichdagi nuqtadan keyingi raqam chiqishdagi maksimal maydon uzunligini aniqlovchi sifatida xizmat qiladi. Masalan, "% 4.8s" format spetsifikatsiyasini hisobga olgan holda, uzunligi to'rt dan sakkiz belgigacha bo'lgan qator chiqariladi, agar oshib ketgan bo'lsa, ekstremal belgilar o'tkazib yuboriladi.

Boshqa format modifikatorlari

Odatiy tekislash to'g'ri tekislashdir, ammo buni "%" dan keyin "-" qo'yish orqali o'zgartirish mumkin. Ushbu format spetsifikatsiyasi tekislashni chapga o'rnatadi.

Bundan tashqari, printf funksiyasi chop etiladigan butun son qiymatlarining qisqa va uzun turlarini ajrata oladi. Yaroqli spetsifikatsiyalar: "o", "d", "u", "i", "x" va "X". Uzun qiymat turi "l" o'zgartiruvchisi bilan, qisqa tur esa "h" o'zgartiruvchisi bilan o'rnatiladi. Misol uchun, uzun butun sonni va qisqa imzosiz intni chop etishda format spetsifikatsiyalari mos ravishda "% ld" va "% hu" sifatida ko'rinadi.

Uzunlik	Tavsif
h	Qisqa yoki belgisiz qisqa turlar uchun
l	Uzoq yoki imzosiz uzun turlar uchun
L	Uzoq juftlik turi uchun

scanf funksiyasi klaviaturadan kiritilgan ma'lumotlarni o'qish uchun ishlatiladi. C tilida printf va scanf tavsifi uchun "stdio.h" sarlavha fayliga qarang.

scanf ("format ko'rsatkichlari", & qiymat1, & qiymat2, ...);

scanf funksiyasi bilan ishlashning oddiy misoli:

```
#include <stdio.h>

int main () {
    int a;
    float b;
    scanf ("%d%f", &a, &b);
}
```

Nazorat savollari

1. C++ dasturlash tilida toifalar turi va ularning xotiradagi hajmi (sizeof)?
2. Qanday amallar unar amallar deyiladi?
3. Qanday amallar binar amallar deyiladi?
4. Binar amallar turkumiga qaysi amallar kiradi?
5. Qanday amallar ternar amallar deyiladi?
6. Unar amallar turkumiga qaysi amallar kiradi?
7. sizeof amali to'g'risida nimalar ayta olasiz?
8. Formatli chiqarish funksiyasi qanday nomlanadi?
9. Formatli chiqarish spetsifikatorlari tog'risida nima ayta olasiz?
10. Formatli kiritidish funksiyasi qanday nomlanadi?

3- MA'RUZA

MAVZU: TARMOQLANISH VA UZILISHLARNI TASHKIL ETISH OPERATORLARI. TARMOQLANUVCHI OPERATORLAR (IF VA SWITCH) VA ULARNI ISHLASH KETMA KETLIKLARI

Reja:

1. Tarmoqlanuvchi operatorlar
2. Shartli operator - to'liqsiz tarmoqlanish
3. To'liq tarmoqlanish
4. Shartsiz o'tish operatori
5. Tanlash operatori

Annotatsiya:

Tarmoqlanuvchi hisoblash jarayonlarini algoritmlash va dasturlash. Ko'pgina masalalarni yechishda ba'zi bir jarayonlar ma'lum shart yoki shartlarning qo'yilishiga nisbatan bajariladi. Bunday jarayonlar **tarmoqlanuvchi jarayonlar** deb yuritiladi.

Tarmoqlanuvchi hisoblash jarayonlari oddiy va murakkab bo'lishi mumkin. Bu esa jarayondagi tarmoqlar soniga bog'liq. Ma'lum bir tarmoqlanuvchi jarayon tarkibida yana tarmoqlanishlar bo'lishi mumkin. Bunday tarmoqlanishlari bor bo'lgan hisoblash jarayonlari **murakkab tarmoqlanuvchi hisoblash jarayonlari** deb ataladi. C++ tilida tarmoqlanuvchi jarayonlarni dasturlash uchun shartsiz, shartli o'tish va tanlash operatorlaridan foydalaniladi: if, case.

Kalit so'zlar: *Tarmoqlanuvchi algoritmi, if else, switch case, goto, nishon, ternar operatori, break, continue*

1. Tarmoqlanuvchi operatorlar

Agar algoritmi bajarilish ketma-ketligi bir nechta shartlarga bog'liq bo'lsa u **tarmoqlanuvchi** deb ataladi. Oldingi mavzularda misol tariqasida keltirilgan dasturlarda operatorlar yozilish tartibida ketma-ket va faqat bir marta bajarilgan holatlar, ya'ni chiziqli algoritmlar keltirilgan. Amalda esa kamdan-kam masalalar shu tariqa yechilishi mumkin. Aksariyat masalalar esa yuzaga keladigan turli holatlarga bog'liq ravishda mos qaror qabul qilishni (yechimni) talab etadi. C++ tilida dasturning alohida bo'laklarini bajarilish tartibini boshqarishga imkon beruvchi qurilmalarning yetarlicha katta majmuasiga ega. Masalan, dastur bajarilishining birorta qadamida qandaydir shartni tekshirish natijasiga ko'ra dasturning u yoki bo'lagiga boshqaruvni uzatish mumkin (tarmoqlanuvchi algoritmi). Tarmoqlanishni amalga oshirish uchun tarmoqlanuvchi operatorlardan foydalaniladi.

2. Shartli operator - to'liqsiz tarmoqlanish

if operatori. if operatori qandaydir shartni rostlikka tekshirish natijasiga ko'ra dasturda tarmoqlanishni amalga oshiradi:

if (<tekshiriladigan shart>) <operator>1;

Bu yerda <tekshiriladigan shart> har qanday ifoda bo'lishi mumkin, odatda u taqqoslash operatori bo'ladi. Agar tekshiriladigan shart rost (true) bo'lsa, <operator>1 bajariladi, aks holda (false) dastur keyingi operatorlarni bajarishga o'tadi.

C++ tilining qurilmalarida operatorlarni blok ko'rinishida bo'lishiga imkon beradi. Blok '{' va '}' belgi oralig'iga olingan operatorlar ketma-ketligi bo'lib, u kompilyator tomonidan yaxlit bir operator deb qabul qilinadi.

Quyida keltirilgan dasturda if operatoridan foydalanish ko'rsatilgan.

```
#include <iostream.h>
```

```
int main() { int b;
```

```
cin>>b;
```

```
if (b>0) { // b>0 shart bajarilgan holat
```

```
cout << "b – musbat son"<<endl;
```

```
cout<<"Uning ildizi "<< sqrt(b)<< " ga teng "<<endl; }
```

```
if (b<0) cout <<"b – manfiy son"; // b < 0 shart bajarilgan holat
return 0; }
```

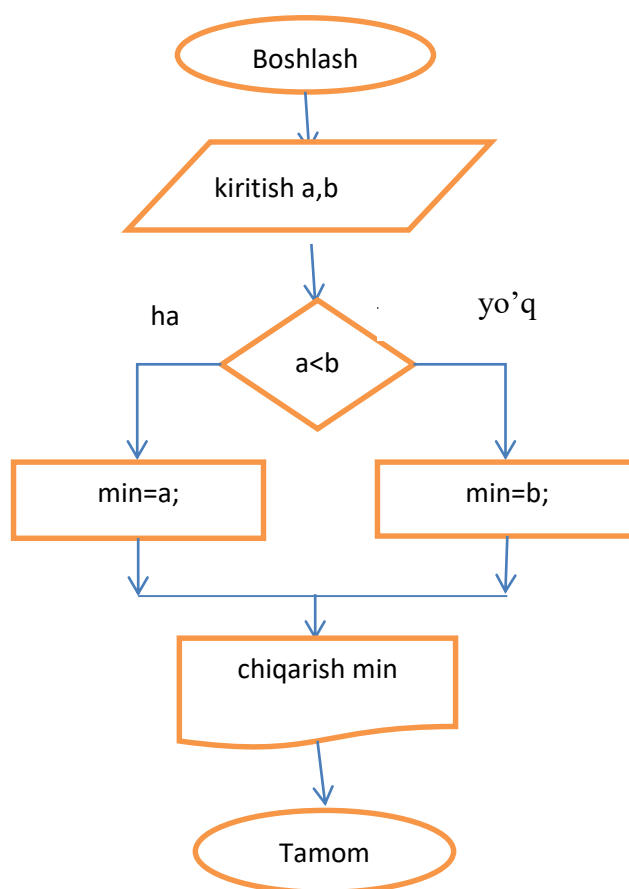
Dastur bajarilishi jarayonida butun turdagi b o'zgaruvchi e'lon qilinadi va klaviaturadan qiymati kiritiladi. Keyin b qiymatini 0 sonidan kattaligi tekshiriladi, agar shart bajarilsa (true) '{' va '}' belgilar ichidagi operatorlar bajariladi va ekranga "b – musbat son" va uning ildizi chiqariladi. Agar shart bajarilmasa, bu operatorlar cheklab o'tiladi. Navbatdagi shart operatori b o'zgaruvchi qiymatini manfiylikka tekshiradi, agar shart bajarilsa yagona cout ko'rsatmasi bajariladi va ekranga "b – manfiy son" xabari chiqadi.

3. To'liq tarmoqlanish.

if – else operatori.

Misol. Ikkita butun sonni kiriting va ulardan kichigini ekranga chiqaring.

.Blok-sxemasi



Shart operatorining if – else ko'rinishi quyidagicha:

```
if (<shart-ifoda>) <operator>1; else <operator>2;
```

Bu yerda <shart-ifoda> rost (true) bo'lsa, <operator>1 bajariladi, aks holda <operator>2 bajariladi. if – else shart operatori mazmuniga ko'ra algoritmning tarmoqlanuvchi blokini ifodalaydi: <shart-ifoda> – shart bloki (romb) va <operator>1 blokning "ha" shoxiga, <operator>2 esa blokning "yo'q" shoxiga mos keluvchi amallar bloklari deb qarash mumkin.

Misol tariqasida determenantni hisoblash usuli yordamida $ax^2+bx+c=0$ ko'rinishidagi kvadrat tenglama ildizlarini topish masalasini ko'raylik.

```
#include <iostream.h>
```

```
#include <math.h>
```

```
int main()
```

```
{
```

```
int a,b,c;
```

```

float D,x1,x2;
cout << "ax^2+bx+c=0 tenglama ildizini topish dastursi! ";
cout<<"\n a - koeffistientni kiriting: ";
cin>>a;
    cout<<"\n b - koeffistientni kiriting: ";
cin>>b;
cout<<"\n c - koeffistientni kiriting: ";
cin>>a;
D = b*b - 4 * a * c;
if (D<0)
{
cout << "Tenglama haqiqiy ildizlarga ega emas";
return 0;
}
if (D==0)
{
cout << "Tenglama yagona ildizga ega: ";
x1=x2= -b / (2 * a);
cout<<"\n x= "<<x1;
return 0;
}
else
{
cout << "Tenglama ikkita ildizga ega: ";
x1 = (- b + sqrt(D)) / (2 * a);
x2 = (- b - sqrt(D)) / (2 * a);
cout<<"\n x1= "<<x1;
cout<<"\n x2= "<<x2;
}
return 0;
}

```

Dastur bajarilishi jarayonida birinchi navbatda tenglama koeffistientlari – a, b, c o‘zgaruvchilar qiymatlari kiritiladi, keyin determenant – D o‘zgaruvchi qiymati topiladi. Keyin D qiymati manfiy ekanligi tekshiriladi. Agar shart o‘rinli bo‘lsa, yaxlit operator sifatida keluvchi ‘{‘ va ‘}’ belgilari orasida operatorlar bajariladi va ekranga “Tenglama haqiqiy ildizlarga ega emas” xabari chiqadi va dastur o‘z ishini tugatadi (return 0 operatorini bajarish orqali). Determenant 0 dan kichik bo‘lmasa, navbatdagi shart operatori uni 0 ga tengligini tekshiradi. Agar D qiymati nolga teng bo‘lsa keyingi qatorlardagi operatorlar bloki bajariladi – ekranga “Tenglama yagona ildizga ega:” xabari, hamda x1 o‘zgaruvchi qiymati chop etiladi va dastur shu yerda o‘z ishini tugatadi, aks holda, ya’ni D qiymati noldan katta bo‘lsa, else keyingi operatorlar bloki bajariladi va ekranga “Tenglama ikkita ildizga ega:” xabari, hamda x1 va x2 o‘zgaruvchilar qiymatlari chop etiladi. Shu bilan shart operatoridan chiqiladi va asosiy funkstiyaning return ko‘rsatmasi bajarish orqali dastur o‘z ishini tugatadi.

4. Shartsiz o‘tish operatori

goto operatori

Shartsiz o‘tish operatorining umumiy ko‘rinishi quyidagicha:

```
goto <nishon>;
```

goto operatoridan keyin boshqarilish <nishon> ga uzatiladi va dasturning bajarilishi shu yerdan davom etadi.

nishon - bu davomida ‘:’ qo‘yilgan identifikator.

Misol uchun: nishon: ;

Nishon har qanday operator oldidan ishlatilishi mumkin, shuningdek shart operatori oldidan ham.

Misol: N natural sonini kiritishni taklif qiluvchi dastur tuzilsin. Agar natural bo‘lmagan son kiritilsa, qayta kiritish taklif qilinsin.

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
    float n;
    nishon:
    cout << “natural son kiriting” << endl;
    cin >> n;
    if(( ceil(n) !=n) or (n <= 0))
        goto nishon;
    cout << “Natural son kiritildi” << endl;
    return 0;
}
```

Dastur bajarilishi jarayonida birinchi navbatda *n* soni kiritiladi, keyin kiritilgan sonni natural son emasligi tekshiriladi. Agar shart rost(true) qiymat qaytarsa nishon ga qaytadi va *n* soni qayta kiritilishi so‘raladi. Aks holda ya‘ni *n* soni natural son bo‘lsa, “Natural son kiritildi” xabari chiqariladi.

5. Tanlash operatori

switch operatori

Shart operatorining yana bir ko‘rinishi switch tarmoqlanish operatori bo‘lib, uning sintaksisi quyidagicha:

```
switch (<ifoda>)
{
    case <konstanta ifoda> :
        <operatorlar guruhi>;
        break;
    case <konstanta ifoda> :
        <operatorlar guruhi>;
        break;
    ...
    default :
        <operatorlar guruhi>;
}
```

Bu operator quyidagicha amal qiladi: birinchi navbatda <ifoda> qiymati hisoblanadi, keyin bu qiymat case kalit so‘zi bilan ajratilgan <konstanta ifoda> bilan solishtiriladi. Agar ular ustma-ust tushsa, ‘:’ belgisidan keyingi break kalit so‘zigacha bo‘lgan <operatorlar guruhi> bajariladi va boshqaruv tarmoqlanuvchi operatoridan keyingi operatorga o‘tadi. Agar <ifoda> birorta ham <konstanta ifoda> ifoda bilan mos kelmasa, qurilmaning default nomidagi operatorlar guruhi bajariladi.

Misol uchun, kirish oqimidan “Jarayon davom etilsinmi?” so‘roviga foydalanuvchi tomonidan javob olinadi. Agar ijobiy javob olinsa, ekranga “Jarayon davom etadi!” xabari chop etiladi va dastur o‘z ishini tarmoqlanuvchi operatoridan keyin davom ettiradi, aks holda “Jarayon tugadi!” javobi beriladi

va programa o'z ishini tugatadi. Bunda, foydalanuvchining 'y','Y','h','H' javoblari jarayonni davom ettirishni bildiradi, boshqa belgilar esa jarayonni tugatishni anglatadi.

```
#include <iostream>
using namespace std;
int main()
{
    char Javob = ' ';
    cout<<"Jarayon davom etsinmi? ('y','Y','h','H'):";
    cin>> Javob;
    switch (Javob) {
        case 'Y' :
        case 'y' :
        case 'h' :
        case 'H' :
            cout<<"Jarayon davom etadi!\n";
            break;
        default :
            cout <<"Jarayon tygadi!\n";
            return 0;
    }
    .... // Jarayon
    return 0;
}
```

Umuman olganda, tarmoqlanuvchi operatorlarda break va default kalit so'zlarini ishlatish shart emas. Lekin bu holda operatorning mazmuni buzilishi mumkin. Masalan, default nomi bo'lmaganda, agar <ifoda> birorta <konstanta ifoda> bilan ustma-ust tushmasa, operator hech qanday amal bajarmasdan boshqaruv navbatdagi operatorga o'tib ketadi. Agar break bo'lmasa dastur "to'xtamasdan" keyingi qatordagi operatorlarni bajarishga o'tib ketadi. Masalan, yuqoridagi misolda break operatori bo'lmasa va jarayonni davom ettirish haqida ijobiy javob bo'lgan taqdirda ekranga

Jarayon davom etadi!

Jarayon tugadi!

xabarlari chiqadi va dastur o'z ishini tugatadi (return 0 operatorini bajarish natijasida).

Tarmoqlanuvchi operator sanab o'tiluvchi turdagi konstantalar bilan birgalikda ishlatilganda samarali bo'ladi. Quyidagi dasturda ranglar gammasini toifalash masalasi yechilgan.

```
#include <iostream>
using namespace std;
int main()
{
    enum Ranglar {Qizil, To'q_sariq, Sariq, Yashil, Ko'k, Zangori, Binafsha};
    Ranglar Rang;
    ...
    switch (Rang)
    {
        case Qizil:
        case To'q_sariq :
        case Sariq :
            cout << "Issiq gamma tanlandi.\n";
            break;
        case Yashil :
```

```

    case Ko‘k :
    case Zangori:
    case Binafsha :
    cout << “Sovuq gamma tanlandi.\n”;
    break;
    default :
    cout <<”Kamalak bunday rangga ega emas.\n”;
    }
    return 0;
}

```

Dastur bajarilishida boshqaruv tarmoqlanuvchi operatorga kelganda, Rang qiymati Qizil yoki To‘q_sariq yoki Sariq bo‘lsa, ‘Issiq gamma tanlandi’ xabari, agar Rang qiymati Yashil yoki Ko‘k yoki Zangori yoki Binafsha bo‘lsa, ekranga ‘Sovuq gamma tanlandi’ xabari, agar Rang qiymati sanab o‘tilgan qiymatlardan farqli bo‘lsa, ekranga ‘Kamalak bunday rangga ega emas’ xabari chop etiladi va dastur o‘z ishini tugatadi.

Nazorat savollari

1. if operatorining nechta turi bor?
2. Tarmoqlanuvchi operatorlar qanday ishlaydi?
3. Tarmoqlanuvchi operatorlar qaysi kutubxonalarga murojaat qiladi?
4. Tanlash operatorlar qanday ishlaydi?
5. Tanlash operatorlarga qandaydir kutubxona kerakmi?
6. switch operatori qavs ichiga nimalar yozish mumkin?
7. Ternar operatori qanday ishlaydi?
8. case deganda nimani tushunasiz?
9. default qaysi vaqtda ishga tushadi?
10. { } figurali bloklar nima uchun kerak?

4-MA'RUZA

MAVZU: TAKRORLANISH OPERATORLARI

Reja:

Kirish

1. Takrorlanuvchi jarayonlar
2. Parametrli takrorlash
 - 2.1. Ichma-ich joylashgan for takrorlanish operatori
3. while takrorlash operatori
4. do-while takrorlash operatori
5. O'tish operatorlari
6. Xulosa
7. Nazorat savollari

Annotatsiya:

Ushbu ma'ruzada keltirilgan materiallar C++ dasturlash tilida muhim ahamiyatga ega bo'lgan takrorlash operatorlari, takrorlanish jarayonini qanday tashkil etish usullariga bag'ishlangan. Ma'ruzada ana shu takrorlanish jarayonini tashkil etuvchi takrorlash operatorlarining turlari hamda ularning qo'llanilish usullari haqida batafsil yoritilgan. Takrorlash operatorlari ishida ba'zan ma'lum bir qiymatlar uchun biror qism bajarilmasligi zarur bo'lib qolganda yordamga kelgan *o'tish* operatorlari haqida ham ma'lumotlar berilgan. Amaliy masalalar yordamida barcha takrorlash operatorlari aniq va to'liq tushuntirilgan.

Kalit so'zlar: *takrorlash, takrorlash tanasi, umepayua, takrorlanish, parametrli, shartoldi takrorlash, shartkeyin takrorlash.*

Kirish

Har qanday dasturning strukturasi tarmoqlanish va takrorlanishlar to'plamining kombinatsiyasidan iborat bo'ladi. Qator masalalarni echish uchun ko'pincha bitta amalni bir necha marotaba bajarish talab qilinadi. Amaliyotda bu rekursiyalar va iterativ algoritmlar yordamida amalga oshiriladi. Iterativ jarayonlar – bu operatsiyalar ketma-ketligini zaruriy sonda takrorlanishidir. Takrorlanuvchi algoritimli dasturlarda aniq bir yoki bir necha amallar takror va takror bajarilish imkoniyati ko'zda tutilgan bo'ladi. Takrorlanishni amalga oshirilishi uchun dasturlash titlining takrorlash operatorlaridan foydalanish mumkin bo'ladi. C++ dasturlash tilida takrorlash operatorlarining bir necha turi mavjud. Takrorlash operatorlari “takrorlash sharti” deb nomlanuvchi ifodaning rost qiymatida dasturning ma'lum bir qismidagi operatorlarni (takrorlash tanasini) ko'p marta takror ravishda bajaradi.

Takrorlash o'zining kirish va chiqish nuqtalariga ega, lekin chiqish nuqtasi bo'lmasligi mumkin, Bunday takrorlashlarga cheksiz takrorlash deyiladi. Cheksiz takrorlash uchun takrorlashni davom ettirish sharti doimo rost bo'ladi.

Takrorlash shartini tekshirish takrorlash tanasidagi operatorlarni bajarishdan oldin tekshirilishi mumkin (for, while operatorlari) yoki tanasidagi operatorlar bir marta bajarilgandan keyin tekshirilishi mumkin (*do-while* operatori).

1. Takrorlanish jarayonlari

Takrorlanish – bu bir xil ketma-ketlikda bajariladigan ko'pqirrali harakat.

Ma'lum qadamlar sonidagi takrorlanish – Noma'lum qadamlar sonidagi takrorlanish (shartli takrorlanish)

Masala. Butun sonlarning kvadratlarini va kublarini ekranga chiqaring 1 dan 8 gacha (a dan b gacha).

Xossa: bir xil harakatlar 8 marta bajariladi.

Takrorlanuvch jarayonga misol: Avval berilgan ma'lumotlar kiritiladi. So'ngra takrorlanuvch jarayonning, ya'ni takrorlashning parametrlari o'rnatiladi. Buni matematikada takrorlanish opalig'i deb ham yuritiladi.

Masalan: $X \in [0;10]$ bo'lsa, takrorlash parametrlari 0 dan 10 gacha hisoblanadi. Keyin hisoblash yoki bir nech hisoblashlar amalga oshiriladi. Natija 1 ta yoki bir nechta chiqishi mumkin, bu masalaning qo'yilishiga bog'liq bo'ladi. Agar masalaning javobi bir nechta chiqadigan bo'lsa, u holda chiqarish blogi ham takrorlash parametri ichida bo'ladi.

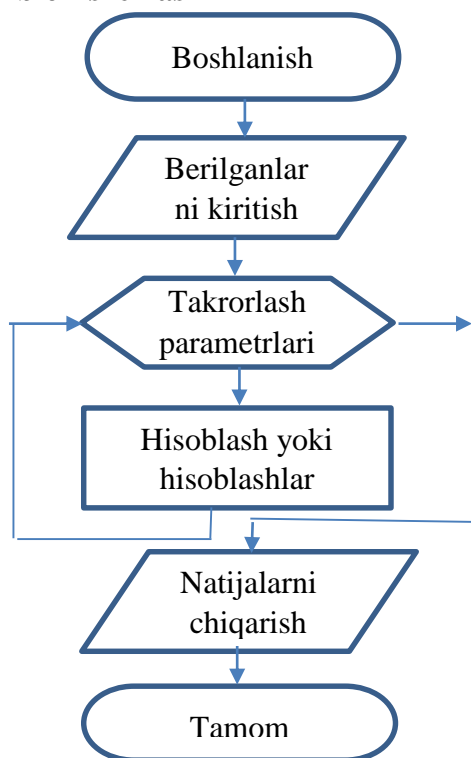
2. Parametrli takrorlash for operatori

for takrorlash operatorining sintaksisi quyidagi ko'rinishga ega:

for (<ifoda1>; <ifoda2>;<ifoda3>) <operator yoki blok>;

for takrorlash operatorining

blok-sxemasi



Bu operator o'z ishini <ifoda1> ifodasini bajarishdan boshlaydi. Keyin takrorlash qadamlari boshlanadi. Har bir qadamda <ifoda2> bajariladi, agar natija 0 qiymatidan farqli yoki true bo'lsa, takrorlash tanasi - <operator yoki blok> bajariladi va oxirida <ifoda3> bajariladi. Agar <ifoda2> qiymati 0 (false) bo'lsa, takrorlash jarayoni to'xtaydi va boshqaruv takrorlash operatoridan keyingi operatorga o'tadi. Shuni qayd qilish kerakki, <ifoda2> ifodasi vergul bilan ajratilgan bir nechta ifodalar birlashmasidan iborat bo'lishi mumkin, bu holda oxirgi ifoda qiymati takrorlash sharti hisoblanadi. Takrorlash tanasi sifatida bitta operator, jumladan bo'sh operator bo'lishi yoki operatorlar bloki kelishi mumkin.

Misol uchun 10 dan 20 gacha bo'lgan butun sonlar yig'indisini

hisoblash masalasini ko'raylik.

```
#include <iostream>
using namespace std;
int main ()
{
    int Summa=0;
    for ( int i= 10 ; i<= 20 ; i++ )
        Summ a+=i;
    cout<<" Yig'indi= " << Summa;
    return 0;
}
```

Dasturdagi takrorlash operatori o'z ishini, i takrorlash parametriga (takrorlash hisoblagichiga) boshlang'ich qiymat - 10 sonini berishdan boshlaydi va har bir takrorlash qadamidan (itaratsiyadan) keyin qavs ichidagi uchinchi operator bajarilishi hisobiga uning qiymati bittaga oshadi. Har bir

takrorlash qadamida takrorlash tanasidagi operator bajariladi, ya'ni Summa o'zgaruvchisiga i qiymati qo'shiladi. Takrorlash sanagichi i qiymati 21 bo'lganda " $i \leq 20$ " takrorlash sharti false (0-qiymati) bo'ladi va takrorlash tugaydi. Natijada boshqaruv takrorlash operatoridan keyingi cout operatoriga o'tadi va ekranga yig'indi chop etiladi. Yuqorida keltirilgan misolga qarab takrorlash operatorlarining qavs ichidagi ifodalarga izoh berish mumkin: <ifoda1> - takrorlash sanagichi vazifasini bajaruvchi o'zgaruvchiga boshlang'ich qiymat berishga xizmat qiladi va u takrorlash jarayoni boshida faqat bir marta hisoblanadi. Ifodada o'zgaruvchi e'loni uchrasehi mumkin va bu o'zgaruvchi takrorlash operatori tanasida amal qiladi va takrorlash operatoridan tashqarida «ko'rinmaydi», <ifoda2> - takrorlashni bajarish yoki yo'qligini aniqlab beruvchi mantiqiy ifoda, agar shart rost bo'lsa, takrorlash davom etadi, aks holda yo'q. Agar bu ifoda bo'sh bo'lsa, shart doimo rost deb hisoblanadi; <ifoda3> - odatda takrorlash sanagichining qiymatini oshirish (kamaytirish) uchun xizmat qiladi yoki unda takrorlash shartiga ta'sir kiluvchi boshqa amallar bo'lishi mumkin.

for operatorida takrorlash tanasi bo'lmasligi ham mumkin. Yuqorida keltirilgan 10 dan 20 gacha bo'lgan sonlar yig'indisini bo'sh tanali takrorlash operatori orqali hisoblash mumkin:

```
for ( int i= 10; i< = 20 ; Summa+=i++)
```

Takrorlash operatori tanasi sifatida operatorlar bloki ishlatishini faktorialni hisoblash misolida ko'rsatish mumkin:

Misol. Faktorialni hisoblash dasturi

```
#include<iostream>  
using namespace std;  
int main()  
{  
    int n,i;  
    long long fact=1;  
    cout<<"n ni kiriting:";  
    cin>>n;  
    for(i=1; i<=n; i++)  
        fact*=i;  
    cout<<"natija="<<fact;  
    return 0 ;}
```

3. Ichma-ich joylashgan for takrorlanish operatori

Misol. Takrorlash operatorining ichma-ich joylashuviga misol sifatida 20 gacha bo'lgan sonlarning tub son yoki murakkab son ekanligi haqidagi ma'lumotni chop qilish masalasini ko'rishimiz mumkin:

```
#include <iostream>  
#include <math.h>  
using namespace std;  
int main()  
{  
    const int m=20;  
    int n[m];  
    int i,j,f;  
    for(i=0; i<=m; i++)  
        n[i]=1;  
    for(i=2; i<=m/2; i++)  
    {  
        if (n[i]==1)  
        {
```

```

        for(j=i+1; j<=m; j++)
        if (n[j]==1)
        if (j%i==0)
            n[j]=0;
    }
}
for(i=2; i<=m; i++)
{
    if (n[i]==1)
        cout<<i<<"=Tub son"<<endl;
    else
        cout<<i<<"=Murakkab son"<<endl;
}
return 0;
}

```

Natija:

```

C:\Users\User\Desktop\wpooEpcioEav\х чрфрэш \U...
2=Tub son
3=Tub son
4=Murakkab son
5=Tub son
6=Murakkab son
7=Tub son
8=Murakkab son
9=Murakkab son
10=Murakkab son
11=Tub son
12=Murakkab son
13=Tub son
14=Murakkab son
15=Murakkab son
16=Murakkab son
17=Tub son
18=Murakkab son
19=Tub son
20=Murakkab son

-----
Process exited after 0.05015 seconds with return va
Для продолжения нажмите любую клавишу . . .

```

Takrorlash operatorida qavs ichidagi ifodalar bo‘lmasligi mumkin, lekin sintaksis ‘;’ bo‘lmasligiga ruxsat bermaydi. Shu sababli, eng sodda ko‘rinishdagi takrorlash operatori quyidagicha bo‘ladi:

```

for (;;)
    cout <<"Cheksiz takrorlash..." ;

```

Agar takrorlash jarayonida bir nechta o‘zgaruvchilarning qiymati sinxron ravishda o‘zgarishi kerak bo‘lsa, takrorlash ifodalarida zarur operatorlarni ‘,’ bilan yozish orqali bunga erishish mumkin:

```

for(int i=10; j=2; i<=20; i++; j=i+10) {...};

```

Takrorlash operatorining har bir qadamida j va i o‘zgaruvchilarning qiymatlari mos ravishda o‘zgarib boradi.

Xossa:

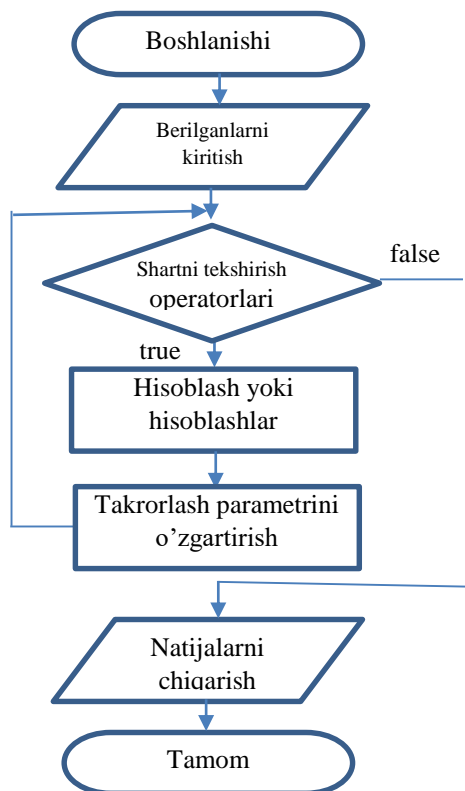
- Shart takrorlashning keyingi qadami boshlanishidan oldin tekshiriladi, agar u yolg‘on bo‘lsa takrorlash bajarilmaydi;
- o‘zgartirish (sarlavhaning uchinchi qismi) takrorlashning navbatdagi qadamining oxirida bajariladi;
- Agar shart yolg‘on bo‘lmasa takrorlash to‘xtovsiz ishlashi mumkin (takrorlash ichiga tushib qoladi)
-

4. while takrorlash operatori

while takrorlash operatori, operator yoki blokni takrorlash sharti yolg'on (false yoki 0) bo'lguncha takror bajaradi. U quyidagi sintaksisga ega:

while (<ifoda>) <operator yoki blok>;

While takrorlash operatorining blok-sxemasi



Agar **<ifoda>** rost qiymatli o'zgarmas ifoda bo'lsa, takrorlash cheksiz bo'ladi. Xuddi shunday, **<ifoda>** takrorlash boshlanishida rost bo'lib, uning qiymatiga takrorlash tanasidagi hisoblash ta'sir etmasa, ya'ni uning qiymati o'zgarmasa, takrorlash cheksiz bo'ladi.

while takrorlash shartini oldindan tekshiruvchi takrorlash operatori hisoblanadi. Agar takrorlash boshida **<ifoda>** yolg'on bo'lsa, **while** operatori tarkibidagi **<operator yoki blok>** qismi bajarilmasdan cheklab o'tiladi.

Ayrim hollarda **<ifoda>** qiymat berish operatori ko'rinishida kelishi mumkin. Bunda qiymat berish amali bajariladi va natija **0** bilan solishtiriladi. Natija noldan farqli bo'lsa, takrorlash davom ettiriladi.

Agar rost ifodaning qiymati noldan farqli o'zgarmas bo'lsa, cheksiz takrorlash ro'y beradi.

Masalan:

while(1); // cheksiz takrorlash

Xuddi **for** operatoridek, ',' yordamida **<ifoda>** da bir nechta amallar sinxron ravishda bajarish mumkin.

Misol. Son va uning kvadratlarini chop qilinadigan dasturda ushbu holat ko'rsatilgan:

```
#include <iostream>
using namespace std;
int main()
{
    int n,n2;
    cout<<"Sonni kiriting (1..10):=";
    cin>>n;
    n++;
    while(n--,n2=n*n,n>0)
    cout<<" n soni = "<<n<<" sonning kvadrati="<<n2<<endl;
    return 0;
}
```

Natija:

```
C:\Users\User\Desktop\project\sonva kvadrat.exe
Sonni kiriting (1..10):=8
n soni = 8 sonning kvadrati=64
n soni = 7 sonning kvadrati=49
n soni = 6 sonning kvadrati=36
n soni = 5 sonning kvadrati=25
n soni = 4 sonning kvadrati=16
n soni = 3 sonning kvadrati=9
n soni = 2 sonning kvadrati=4
n soni = 1 sonning kvadrati=1
Process exited after 3.717 seconds with return code 0
Для продолжения нажмите любую клавишу . . .
```

Dasturdagi takrorlash operatori bajarilishia n soni 1 gacha kamayib boradi. Har bir qadamda n va uning kvadrati chop qilinadi. Shunga e'tibor berish kerakki, shart ifodasida operatorlarni yozilish ketma-ketligining ahamiyati bor, chunki eng oxirgi operator takrorlash sharti sifatida qoraladi va n qiymati 0 bo'lganda takrorlash tugaydi.

Misol: Ixtiyoriy natural sonlar kiritiladi, qachonki char tipidagi biron belgi kiritilguncha va kiritilgan sonlar yig'indisi hisoblanadi.

```
#include <iostream>
#include <string.h>
using namespace std;
int main() {
    int a, ans;
    char s;
    cin >> a;
    ans = a;
    while(cin >> s >> a)
    {
        ans += a;
    }
    cout << ans;
    return 0;}
```

while takrorlash operatori yordamida samarali dastur kodi yozishga yana bir misol bu - ikkita natural sonlarning eng katta umumiy buluvchisini (EKUB) Evklid algoritmi bilan topish masalasini keltirishimiz mumkin:

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main () {
    int a, b ;
    cout<< " A va B natural sonlar EKUBini topish \n " ;
    cout<< " A va B natural sonlarni kiriting : " ;
    cin >> a >> b ;
    while ( a != b ) a > b ? a -= b : b -= a ;
    cout<< " Bu sonlar EKUBi= "<< a ;
    return 0 ; }
```

Bu misolda butun turdagi a va b qiymatlari oqimdan o'qilgandan keyin toki ularning qiymatlari o'zaro teng bo'lmaguncha takrorlash jarayoni ro'y beradi. Takrorlashning har bir qadamida a va b sonlarning kattasidan kichigi ayiriladi. Takrorlashdan keyingi ko'rsatma vositasida a o'zgaruvchisining qiymati natija sifatida chop etiladi.

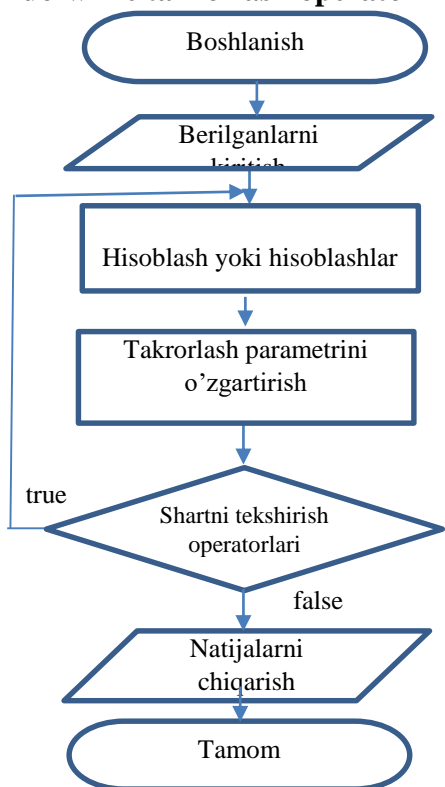
5. do-while takrorlash operatori

do-while takrorlash operatori while operatoridan farqli ravishda oldin operator yoki blokni bajaradi, keyin takrorlash shartini tekshiradi. Bu operator takrorlanish tanasini kamida bir marta bajarilishini ta'minlaydi. **do-while** takrorlash operatori quyidagi sintaksisga ega:

```
do <operator yoki blok>; while (<ifoda>);
```

Bunday takrorlash operatorining keng qo'llaniladigan holatlari - bu takrorlashni boshlamasdan turib, takrorlash shartini tekshirishning iloji bo'lmagan holatlar hisoblanadi.

do-while takrorlash operatorining blok-sxemasi



Masalan, birorta jarayonni davom ettirish yoki to'xtatish haqidagi so'rovga javob olish va uni tekshirish zarur bo'lsin. Ko'rinib turibdiki, jarayonni boshlamasdan oldin bu so'rovni berishning ma'nosi yo'q. Hech bo'lmaganda takrorlash jarayonining bitta qadami amalga oshirilgan bo'lishi kerak:

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    char javob;
```

```
    do {
```

```
        ...    // dastur tanasi
```

```
        cout<<"Jarayonni to'xtatish (N):_ ";
```

```
        cin>>javob;
```

```
    } while(javob !=N)
```

```
    return 0;
```

```
}
```

Dastur toki "Jarayonni to'xtatish (N):_ " so'roviga 'N'

javobi kiritilmaguncha davom etadi.

Bu operator ham cheksiz takrorlanishi mumkin:

```
do; while(1);
```

Misol.

Har qanday 7 katta butun sondagi pul miqdorini 3 va 5 so'mliklarda berish mumkinligi isbotlansin. Qo'yilgan masala $p=3n+5m$ tenglamasi qanoatlantiruvchi m, n sonlar juftliklarini topish masalasidir (**p-pul miqdori**). Bu shartning bajarilishini m va n o'zgaruvchilarining mumkin bo'lgan qiymatlarining barcha kombinatsiyalarida tekshirish zarur bo'ladi.

Dasturi:

```
#include <iostream>
```

```
#include <math.h>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    unsigned int Pul; //Pul- kiritiladigan pul miqdori
```

```
    unsigned n3,m5; //n-3 so'mliklar , m-5 so'mliklar soni
```

```
    bool xato=false; //Pul qiymatini kiritishdagi hatolik
```

```
    do
```

```
    {
```

```
        if(xato)cout<<"Pul qiymati 7 dan kichik!";
```

```
        xato=true; // keyingi takrorlanish xato hisoblanadi
```

```
        cout<<"\nPul qiymatini kiriting (>7): ";
```

```
        cin>>Pul;
```

```
    }
```

```
    while(Pul<=7); // Toki 7 dan katta son kiritilguncha
```

```
    n3=0;          //Birorta ham 3 so'mlik yo'q
```

```
    do
```

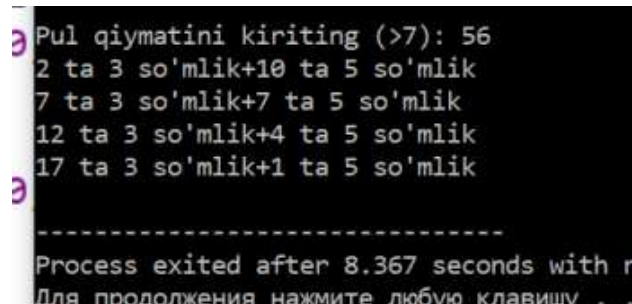
```
    {
```

```

m5=0;          // Birorta ham 5 so'mlik yo'q
do
{
    if (3*n3+5*m5==Pul)
        cout<<n3<<" ta 3 so'mlik+"<<m5<<" ta 5 so'mlik\n";
    m5++;          // 5 so'mliklar 1 taga oshiriladi
}
while(3*n3+5*m5<=Pul);
n3++;          //3 so'mliklar bittaga oshiriladi
}
while(3*n3<=Pul);
return 0;
}

```

Natija:



```

Pul qiymatini kiriting (>7): 56
2 ta 3 so'mlik+10 ta 5 so'mlik
7 ta 3 so'mlik+7 ta 5 so'mlik
12 ta 3 so'mlik+4 ta 5 so'mlik
17 ta 3 so'mlik+1 ta 5 so'mlik
-----
Process exited after 8.367 seconds with r
Для продолжения нажмите любую клавишу ...

```

break operatori

Ba'zi hollarda takrorlanish bajarilishini ixtiyoriy joyda to'xtatishga to'g'ri keladi. Bu vazifani **break** operatori bajarishga imkon beradi. Bu operator darhol takrorlash bajarilishini to'xtatadi va boshqaruvni takrorlashdan keyingi operatorlarga uzatadi. **Break** operatorini takrorlash operatori tanasining ixtiyoriy (zarur) joylariga qo'yish orqali shu joylardan takrorlashdan chiqishni amalga oshirish mumkin.

Misol: Bu misolda **n** o'zgaruvchiga xoxlagancha qiymat kiritishimiz mumkin, qachonki **n** ga 1 yoki 0 kiritilganda **break** operatori ishga tushadi.

Misolning dasturi:

```

#include <iostream>
using namespace std;
int main() {
    int n;
    while (1)
    {
        cin>>n;
        if(n==1 || n == 0)
            break;
    }
    cout<<"Takrorlanish tugadi";
    return 0;
}

```

Bu misolda while(1) operatori yordamida cheksiz takrorlanish hosil qilinadi. Agar 1 yoki 0 raqami kiritilsa, takrorlanish to'xtatiladi.

continue operatori

Takrorlanish bajarilishiga ta'sir o'tkazishga imkon beradigan yana bir operator **continue** operatoridir. Bu operator takrorlanish qadamining bajarilishini to'xtatib, **for** va **while** da ko'rsatilgan shartli tekshirishga o'tkazadi.

Misol:

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    for(;;)
    {
        cin>>n;
        if(n==4 || n == 2)
            continue;
        break;
    }
    cout<<"Takrorlanish tugadi";
    return 0;
}
```

Bu misolda **for(;;)** operatori yordamida cheksiz takrorlanish hosil qilinadi. Agar 4 yoki 2 sonlaridan farqli son kiritilsa, takrorlanish to'xtatiladi.

goto o'tish operatori

O'tish operatorining ko'rinishi:

goto <identifikator>.

Bu operator identifikator bilan belgilangan operatorga o'tish kerakligini ko'rsatadi.

Misol uchun goto A;A: y = 5;

Strukturali dasturlashda **goto** operatoridan foydalanmaslik maslahat beriladi. Lekin ba'zi hollarda o'tish operatoridan foydalanish dasturlashni osonlashtiradi.

Misol. Bir necha takrorlashdan birdan chiqish kerak bo'lib qolganda, to'g'ridan to'g'ri break operatorini qo'llab bo'lmaydi, chunki u faqat eng ichki takrorlashdan chiqishga imkon beradi.

```
#include <iostream>
using namespace std;
int main()
{
    int n = 5, s=0;
    int i, j;
    for(i=1; i<5; i++)
    {
        cout<<endl;
        cout<<"i = "<<i<<endl;
        for(j=1; j<5; j++)
        {
            cout<<"j = "<<j<<" ";
            if(i*j > n) goto A;
            s ++ ;
            cout<<"s = "<<s<<" ";
        }
    }
    A:
    cout<<endl;
    cout<<"Oxirgi natija ="<<s;
```

```
return 0;
}
```

Natija:

```
File C:\Users\User\Desktop\project\goto_exe
i = 1
j = 1 s = 1 j = 2 s = 2 j = 3 s = 3 j = 4 s = 4
i = 2
j = 1 s = 5 j = 2 s = 6 j = 3
1 Oxirgi natija =6
2 -----
3 Process exited after 8.254 seconds with return value 0
4 Для продолжения нажмите любую клавишу . . .
5
```

Quyidagi misolda ma'lum bir rost (true) holatdan yolg'on (false) holatiga o'zgarguncha dastur blokini takrorlashni buyurishi kuzatamiz.

```
#include <iostream>
using namespace std;
int main()
{
    int qadam = 0;
    bool takrorlash = true;
    while ( takrorlash != false )
    {
        qadam++;
        cout << "Takrorlanish bajarildi = " << qadam << " - marta\n";
        if ( qadam == 5 )
            takrorlash = false;
    }
    return 0;
}
```

Natija:

```
Takrorlanish bajarildi = 1 - marta
Takrorlanish bajarildi = 2 - marta
Takrorlanish bajarildi = 3 - marta
Takrorlanish bajarildi = 4 - marta
Takrorlanish bajarildi = 5 - marta
-----
Process exited after 7.591 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

7. Xulosa

Biz C++ dasturlash tilida 3 xil ko'rinishdagi takrorlash operatorlarni borligini ko'rib chiqdik. Bular **for**, **while**, **do-while**. Ular funksional nuqtaiy nazardan bir ish bajaradi, ya'ni ularni funksiyasi ma'lum bir amalni (yoki amallarni) ketma-ket bir necha marta takrorlashdan iborat.

Ammo bu operatorlar mazkur funksiyani (ya'ni takrorlash jarayonini tasiflashni) turlicha amalga oshiradi. Bu operatorlarining asosiy farqi quydagilardan iborat:

- **for** operatori faqat iteratsiyalar (ya'ni takrorlanishlar) soni ma'lum bo'lgan holda va **while** operator esa iteratsiyalar soni noma'lum bo'lgan holda ham ishlaydi.

- do-while operatori ham xuddi **while** operatori kabi iteratsiyalar soni noma'lum bo'lgan holda ham ishlaydi. Ammo u bir iteratsiyadan sung "shart"ni tekshiradi, **while** operatori esa avval "shart"ni tekshiradi va undan so'ng birinchi iteratsiyani amalga oshiradi.

Parametrli takrorlash **for** operatorini parametrning boshlang'ich va oxirgi qiymati hamda o'zgarish qadami aniq bo'lganda qo'llash juda qo'lay!

Nazorat savollari

1. C++ dasturlash tilida takrorlash operatorlari nima uchun kerak?
2. Takrorlash operatorlari qanday ishlaydi?
3. Takrorlash operatorlariga kutubxona kerakmi?
4. while operatorining umumiy ko'rinishi.
5. do while operatori while operatoridan qanday farq qiladi?
6. for operatorining umumiy ko'rinishi.
7. Ichki takrorlanishlar qanday tashkil etiladi?
8. for operatori qanday qo'llaniladi?
9. Qachon operatorlar { } orasiga olinadi?
10. for operatorini qanday qismlari mavjud?
11. for operatorining takrorlanishini tugashi uning qaysi qismiga bo'liq?
12. for operatorida (;) belgi nimani bildiradi?
13. while operatori qanday ishlaydi?
14. Qahday holatda boshqaruv while operatoridan keyingi operatorga uzatiladi?
15. Qahday holatda while operator tanasidagi amallar ketma-ketligi bajariladi va yana shart tekshirishga qaytiladi?
16. for bilan while va do-while operatorlarining qanday farqli tomonlari bor?
17. Takrorlanishlar algoritmlarda qanday ko'rinishda bo'ladi?
18. while operatorida qachon cheksiz takrorlash ro'y beradi?
19. Dasturlashda takrorlash nima?
20. Takrorlanishlar ichma-ich bo'lishi mumkinmi?
21. do-while operatoridan qanday chiqiladi?
22. Takrorlashning takrorlanishlar soni qanday aniqlanadi?
23. for operatorida bir nechta hisoblagich ishlatish mumkinmi?
24. Nima uchun goto operatori ko'p ishlatilmayti?
25. Tanasida biror amal yozilmagan for operatori yordamida takrorlash tashkil etish mumkinmi?
26. for operatori ichida while operatori yordamida takrorlashni tashkil etish mumkinmi?
27. Hech qachon tugallanmaydigan takrorlanishni tashkil etish mumkinmi?
28. while va do-while operatorlarining qanday farqi bor?
29. Takrorlanish tanasiga nechta operator yozish (joylash) mumkin?
30. Takrorlanish tanasidan tashqrida argument qiymatidan foydalanish mumkinmi?
31. Qaysi operator oldin shartni tekshiradi?
32. Qaysi operatorini ishlatganimizda shart tekshirilguncha amallar ketma-ketligi bir marta bajariladi?
33. O'tish operatorlaridan qachon foydalaniladi?

5. MAVZU. FUNKSIYALAR. REKURSIV FUNKSIYALAR. FOYDALANUVCHI KUTUBXONASI

Reja:

Kirish

1. Funksiya tavsifi
2. Qiymatlarni qaytarish
3. Funksiya prototiplari
4. Ko'rinish sohasi. Lokal va global o'zgaruvchilar
5. Funksiyalarni qayta yuklash
6. Rekursiv funksiyalar
7. Nazorat savollari

Annotatsiya:

Ushbu ma'ruza matni C++ da funksiyalarni yaratish va ulardan foydalanish, shuningdek foydalanuvchi kutubxonasini yaratish hamda undan foydalanish haqida ma'lumotlar olish uchun tayyorlangan. Ma'ruzada yuqoridagilardan tashqari funksiyalarning turlari, rekursiv funksiya, lokal va global o'zgaruvchilar batafsil yoritilgan. Funksiyaning turlari bo'yicha to'liq ma'lumotga ega bo'lish uchun amaliy masalalar yordamida aniq va to'liq tushuntirilgan.

Kalit so'zlar. *funksiya, prototip, foydalanuvchi kutubxonasi, local, global, protsedura, qayta yuklash,*

Umuman olganda C/C++ tilida barcha yozuvlar funksiyadan iborat deb qaraladi. Funksiya bu ma'nosiga ko'ra bajariluvchi modul bo'lib hisoblanadi. Funksiyani boshqa dasturlash tillarida qism dastur, prosedura, prosedura funksiya deb yuritiladi. C/C++ tilida funksiya standart formaga asosan quyidagicha ifodalanadi :

funksiya toifasi funksiya nomi (rasmiy parametrlar ro'yxati)

{ funksiya tanasi }

Funksiya toifasi istalgan toifa yoki void (bo'sh) toifa bo'lishi mumkin.

Funksiya nomi istalgan lotin harfi yoki harflaridan iborat bo'lib, xizmatchi so'zlar bilan bir xil bo'lmasligi lozim.

Rasmiy parametrlar ro'yxatida ishlatiladigan parametrlarga mos toifali o'zgaruvchilar toifalari bilan alohida-alohida keltiriladi yoki bu soha bo'sh bo'lishi ham mumkin. Eslatib o'tish lozimki, funksiya aniqlashtirilayotganda nuqta vergul belgisi qo'yilmaydi.

Funksiya tanasi o'zining figurali qavslariga ega bo'lib, o'zida shu funksiyani tashkil etuvchi operatorlar yoki operatorlar blokini mujassamlashtiradi. Bir funksiya tanasi ichida boshqa funksiya aniqlanishi mumkin emas.

Funksiya tanasidan chiqish

return;

yoki

return ifoda;

ko'rinishida bo'ladi. Agar funksiya hech qanday qiymat qaytarmaydigan, ya'ni toifasi void bo'lsa, birinchi ko'rinishdagi chiqish ishlatiladi. Agar funksiya uning toifasiga mos biror qiymat qaytaradigan bo'lsa, ikkinchi ko'rinishdagi chiqish ishlatiladi. C tilida quyidagi ko'rinishlar ekvivalent hisoblanadi, lekin birinchi ko'rinish ko'proq ishlatiladi:

double f (int n, float x)

```
{  
    funksiya tanasi;  
}
```

double f (n, x)

```
int n; float x;  
{  
    funksiya tanasi; }
```

Dasturda funksiya ishlatiladigan bo'lsa, uni albatta e'lon qilish shart. Funksiyani e'lon qilishda uning toifasi, nomi va qaytaradigan parametrlari haqida xabar beriladi. Dasturda biror funksiya oldindan e'lon qilmasdan turib uni chaqirish mumkin emas. Funksiyani asosiy funksiya main() dan oldin va keyin aniqlanishi mumkin. Agar funksiya asosiy funksiya oldin aniqlansa, u aniqlanishi bilan birga e'lon qilingan deb hisoblanadi va uni alohida main() ichida e'lon qilish shart bo'lmay qoladi. Agar funksiya asosiy funksiya keyin aniqlanayotgan bo'lsa, uni main() ichida albatta e'lon qilish shart bo'ladi. Funksiyani main() ichida e'lon qilinadigan bo'lsa, uning nomi bilan birga ishlatiladigan parametrlarining faqatgina toifalari ko'rsatilishi ham mumkin. Masalan:

```
int myFuncion ( int, float);
```

```
double Area (float, float);
```

Funksiyaga murojaat qilishdan uning rasmiy parametrlari aniqlangan bo'lishi, ya'ni haqiqiy parametrlar berilgan bo'lishi lozim. Funksiyaga murojaat qilish quyidagicha amalga oshiriladi:

```
funksiya_toifasi funksiya_nomi (haqiqiy parametrlar ro'yxati);
```

Masalan:

```
myFuncion (78, 3.0+m);
```

```
Area (a, b);
```

```
g (6.4e-2, 5, 70);
```

Funksiyaning rasmiy va haqiqiy parametrlarining toifasi, parametrlar soni va ularning kelish o'rinlari albatta bir biriga mos kelishi shart!

Funksiyaga murojaat qilinganidan so'ng aniqlangan funksiya tanasi bajariladi va mos toifali qiymat chaqirilgan joyga qaytib keladi.

Masalan: quyidagi funksiya chaqirilganida float toifali natija qaytaradi:

```
float ft (double x, int n)
```

```
{  
  if (x < n) return x;  
  else return n;  
}
```

Funksiyalarga murojaat qilinganida uning uzatiladigan parametrlariga alohida e'tibor berish kerak. Parametrlarning uzatilishi quyidagi bosqichlardan iborat:

- Funksiyani tashkil etadigan rasmiy parametrlar uchun xotiradan joy ajratiladi. Agar parametrlar haqiqiy toifaga ega bo'lsa, ular double toifaga, agar char va short int toifali bo'lsalar, ular int toifasi sifatida tashkil etiladilar. Agar parametrlar massiv shaklida bo'lsalar, massiv boshiga ko'rsatkich qo'yiladi va u funksiya tanasi ichida massiv parametr bo'lib xizmat qiladi.

- Funksiya chaqirilganida kerak bo'ladigan ifodalar yoki haqiqiy parametrlar aniqlanadi va ular rasmiy parametrlar uchun ajratilgan joyga yoziladi;

- Funksiya chaqiriladi va aniqlangan haqiqiy parametrlar yordamida hisoblanadi. Bu yerda ham agar parametrlar haqiqiy toifaga ega bo'lsa, ular double toifaga, agar char va short int toifali bo'lsalar, ular int toifasi sifatida tashkil etiladilar.

- Natija funksiya chaqirilgan joyga qaytariladi.

- Funksiyadan chiqishda rasmiy parametrlar uchun ajratilgan xotira qismi bo'shatiladi.

Funksiyaga murojaat qilish ifodani tashkil etadi, lekin agar funksiyaning qaytaradigan qiymati bo'sh (void) bo'lsa, u ifoda bo'lmasligi ham mumkin. Unda bunday funksiyalarga murojaat qilish quyidagicha bo'ladi:

```
funksiya nomi (haqiqiy parametrlar);
```

Masalan:

```
void print (int gg, int mm, int dd)
```

```
{  
  cout<< "\n yil:"<< gg;
```

```
cout << " \n oy: " << mm;
cout << " \n kun: " << dd;
}
```

ko‘rinishidagi funksiyaga **print (1966, 11, 22);** deb murojaat qilinsa, quyidagi natija chiqadi:

```
yil: 1966
oy: 11
kun: 22
```

Ba‘zan umuman hech qanday parametrsiz funksiyalar ham ishlatiladi. Masalan:

```
void Real_Time (void)
{
    cout << " Hozirgi vaqt: " << TIME "(soat: min: sek)";
}
```

funksiyasiga Real_Time (); deb murojaat qilinsa, ekranga

Hozirgi vaqt: 14: 16: 25 (soat: min: sek) degan axborot chiqadi.

Funksiya - bu mantiqan to‘g‘ri tugatilgan dasturiy qismdir. Ular yordamida katta va murakkab hisoblashlarni qayta - qayta yozish mashaqqatidan xalos bo‘linadi va dastur bajarilishi yengillashadi. Uni bir marta tashkil etib yozib qo‘yiladi va unga dasturning istalgan yeridan murojaat qilish mumkin bo‘ladi. Funksiyani tashkil qilishda funksiyaning toifasi, uning nomi va tashkil etuvchi parametrlari haqida axborot keltiriladi. Bu parametrlar rasmiy parametrlar deb yuritiladi.

Rasmiy va haqiqiy parametrlar soni, ularning toifasi va kelish o‘rni bilan albatta bir biriga mos bo‘lishi shart! Rasmiy va haqiqiy parametrlar nomlari bir xil bo‘lishi mumkin. Funksiyani bosh funksiya ichida e‘lon qilinganida haqiqiy parametrlar nomlarini ko‘rsatmasdan, faqat ularning toifalarini keltirish ham mumkin.

Funksiyalar **main ()** funksiyasidan avval ham, keyin ham aniqlanishi mumkin. Agar bosh funksiya avval aniqlangan bo‘lsa, uni **main ()** funksiyasi ichida alohida e‘lon qilish shart emas, agar bosh funksiya keyin keladigan bo‘lsa, uni **main ()** funksiyasi ichida albatta e‘lon qilish kerak. Masalan: sonning kubini hisoblash uchun funksiya tashkil eting va undan foydalaning.

```
# include <iostream.h>
# include <conio.h>
void main ( )
{ int k, n, kw (int n); // kw - funksiya nomi (ixtiyoriy)
cin>>n; // n - berilayotgan son
k=kw(n); // kw funksiyasiga murojaat qilinmoqda
cout << «k=»<<k<<endl;
getch( );
}
int kw (int a) // funksiya aniqlanmoqda. Bu yerda a rasmiy parametr
{ int c; // lokal o‘zgaruvchi
c=a*a*a; // hisoblash
return c; } // funksiyaga natijani qaytarish
```

Yuqoridagi s lokal o‘zgaruvchisini ishlatmasdan, to‘g‘ridan-to‘g‘ri **return a*a*a;** deb yozsa ham bo‘ladi.

Bu yerda funksiya bosh funksiya keyin aniqlandi, shuning uchun uni bosh funksiya ichida e‘lon qildik. Dasturni yana quyidagicha yozsa ham bo‘ladi:

```
# include <iostream.h>
# include <conio.h>
int kw (int a)
{ return a*a*a; }
void main ( )
```

```

{ int k, n ;
  cin>>n;
  k=kw(n);
  cout << «k=«<<k<<endl;
  getch( );}

```

2-misol. Ikkita sondan eng kattasini topish uchun funksiya tashkil qiling va undan foydalaning.

```

# include <iostream.h>
# include <conio.h>
void main( )
{ float a=7, b=9, c, max(float , float );
  c = max(a, b);
  cout << «c=«<<c<<endl;
  getch( );
}
float max ( float x, float y)
{ if (x > y) return x; else return y; }

```

Funksiyaga yana quyidagicha ham murojaat qilish mumkin:

```

c = max( 7.23, 9.145);
c = max( a, 9.145);

```

3-misol. Uchburchak uchlarining koordinatalari berilgan. Shu koordinatalar yordamida uchburchak qursa bo‘ladimi? Agar mumkin bo‘lsa shu uchburchakning yuzini hisoblash dasturini tuzing.

Demak, berilgan koordinatalar yordamida uchburchak tomonini ko‘rish funksiyasini, shu tomonlar asosida uchburchak qurish mumkinmi yoki yo‘qligini va uning yuzini hisoblash funksiyalarini tuzing.

```

# include <iostream.h>
# include <math.h>
# include <conio.h>
// uchburchak tomonini topish funksiyasi
float line (float x1, float x2, float y1, float y2)
{ (float) p = sqrt ((x1-x2)*(x1-x2)+ (y1-y2)*(y1-y2));
  return p; }
// uchburchak qurib bo‘ladimi? funksiyasi
int uch ( float a, float b, float c)
{ if ( a+b>c && b+c>a && c+a>b ) return 1;
  else return 0; }
// uchburchakning yuzini topish funksiyasi
float s (float a, float b, float c)
{ float p, s ;
  p = ( a + b + c ) / 2; s = sqrt (p*(p-a)*(p-b)*(p-c));
  return s; }
void main ( )
{ float x1, x2, x3, y1, y2, y3, p1, p2, p3; clrscr ( );
  cin >> x1>> x2>> x3>> y1>> y2>> y3;
  p1 = line (x1, x2, y1, y2);
  p2 = line (x1, x3, y1, y3);
  p3 = line (x2, x3, y2, y3);
}

```

```

t = uch (p1, p2, p3);
if ( t == 1)
{
    yuza = s ( p1, p2, p3);
    cout << "yuza = " << yuza << endl;
    else cout << "uchburchak qurib bo'lmaydi !!!" << endl;
}
getch ( );
}

```

Bir funksiya ichida boshqa funksiya aniqlanishi mumkin emas, lekin funksiya ichida o'zini-o'zi chaqirishi mumkin. Bunday holatni rekursiya holati deyiladi. Rekursiya 2 xil bo'ladi: to'g'ri rekursiya va bilvosita rekursiya. Agar funksiya o'zini-o'zi chaqirsa, bu to'g'ri rekursiya deyiladi. To'g'ri rekursiyada funksiyaning nusxasi chaqiriladi. Agarda funksiya boshqa bir funksiyaning chaqirsa va u funksiya o'z navbatida 1-sini chaqirsa, u holda bilvosita rekursiya deyiladi. Rekursiya 2 xil natija bilan yakunlanadi: biror natija qaytaradi yoki hech qachon tugallanmaydi va xatolik yuz beradi. Bunday holatlarda rekursiv funksiyalar uchun rekursiyani to'xtatish shartini berish zarur, chunki rekursiyada xotira yetishmasligi xavfi bor.

4-misol. $F = n!$ ni hisoblash uchun funksiya tashkil eting va undan foydalaning.

```

#include <iostream.h>
#include <conio.h>
int main( )
{
    int n, f, fac(int);
    cout << "sonni kiriting:"; cin >> n;
    f = fac(n); cout << "sonning faktoriali=" << f << endl;
    getch( );
}
int fac(int i)
{
    return i <= 1 ? 1 : i * fac( i - 1);
}

```

5-misol. Fibonacci sonlarini hosil qilish dasturini tuzing. Fibonacci sonlari quyidagicha topiladi:

$$f_0 = 1; f_1 = 1; f_2 = f_1 + f_0; \dots$$

$$f_n = f_{n-1} + f_{n-2};$$

Rekursiv jarayonni to'xtatish sharti $n < 2$ deb olinadi. Masalan 9-o'rindagi Fibonacci sonini topish kerak.

```

#include <iostream.h>
int main ( )
{
    int n, f ; int fib ( int );
    cout << "Nomerni kiriting =";
    cin >> n;
    f = fib (n);
    cout << "Fibonacci soni=" << f << endl;
}
int fib ( int n )
{
    if ( n < 2) return 1;
    else
        return ( fib (n-2) + fib (n-1));
}

```

6-misol. $Z = \frac{a^5 + a^{-4}}{2a^n}$ hisoblash dasturi tuzilsin. Bu yerdagi darajani hisoblash funksiya sifatida

tashkil etilsin. $y = x^n$ ni funksiya deb tashkil etamiz, bu yerda x, n - rasmiy parametrlar

```
#include <iostream.h>
float dar (float x, int n)
{ float y=1;
  for (int i=0; i<=n; i++)
    y = y*x;
  return y; }
int main( )
{
  int n=3 ; float a, z;
  cin>>a;
  z = ( dar(a, 5) + dar(1/a, 4))/( 2* dar(a, n)) ;
  cout << «z=»<<z<<endl; }
```

Bir xil nomdagi funksiyalarni har xil toifali o'zgaruvchilar ro'yxati bilan murojaat qilib chaqirish mumkin. Parametrlar soni ham har xil bo'lishi mumkin. Bunday holatda parametrlar ro'yxati va qiymatlarga qarab kompilyator o'zi qaysi funksiyani chaqirish kerakligini aniqlaydi. Masalan:

```
1. double multi (float x)
{ return x*x*x; }
2. double multi (float x, float y)
{ return x*y*y; }
3. double multi (float x, float y, float z)
{ return x*y*z; }
```

va quyidagi murojaatlarning hammasi to'g'ri yozilgan:

```
multi (0.5);
multi (1.45, 7);
multi (10, 39, 54);
```

Funksiyalarning bir xil nom bilan atalishi polimorfizm deb ataladi. Poli – ko'p, morfe – shakl degan ma'noni bildiradi. Masalan:

```
#include <iostream.h>
int max (int a, int b)
{ if (a>b) return a; else return b;}
float max (float a, float b)
{ if (a>b) return a; else return b;}
int main ( )
{
  int a1, b1; float a2, b2;
  cin >> a1>>b1;
  cout << "butun max="<<max(a1, b1)<<endl;
  cin >>a2>>b2;
  cout << "haqiqiy max="<< max(a2, b2)<<endl;
}
```

1-misol. B va C vektorlarining uzunliklarini hisoblash dasturini tuzing. Vektor uzunligini hisoblash uchun funksiyadan foydalaning.

```
#include <iostream.h>
float vector (int d[ ], int k)
{ float s=0; int i ;
```

```

    for (i=0; i<k; i++)
    s = s + d[i] * d[i];
    s = sqrt (s);
    return s; }
int main ( )
{
    int b[3] = {10,20,30}, c[4] = {14,15,16,17};
    float s1, s2;
    s1 = vector (b, 3);
    s2 = vector (c, 4);
    cout <<"s1=" << s1 <<" s2=" << s2 << endl;
}

```

2-Misol. Butun sonli 4x5 matrisasi berilgan. Aniq bir sondan kichik bo'lgan hadlarining yig'indisini topish dasturini tuzing. Matrisa elementlarini kiritish (tasodifiy sonlar yordamida), chiqarish va yig'indini hisoblash jarayonlarini funksiya sifatida tashkil eting.

Funksiya ichida 2 o'lchovli massivlardan foydalanilganda uning 1-parametrini, ya'ni satrlar sonini ko'rsatmaslik ham mumkin, lekin 2-parametrini, ya'ni ustunlar sonini albatta ko'rsatish shart.

```

# include <iostream.h>
# include <conio.h>
# include <stdlib.h>
# include <time.h>
void kir(int m[ ][5], int k);
void chiq(int m[ ][5], int k);
int summa(int m[ ][5], int k, int x);
int i, j ;
void main ( )
{ int matr[4][5]; int a, s; int b[ ][3];
  cout<< "sonni kiriting="; cin>>a;
  kir(matr, 4); chiq(matr, 4);
  s = summa(matr, 4, a);
  cout<< "s="<<s<< endl;
  getch( ); }
void kir(int m[ ][5], int k)
{ srand(time(0));
  for (i=0;i<k;i++)
  for (j=0;j<5;j++)
  m[i][j]=rand( ) - 200; }
void chiq(int m[ ][5], int k)
{ for (i=0;i<k;i++)
  for (j=0;j<5;j++)
  cout <<m[i][j]<<endl; }
int summa(int m[ ][5], int k, int x)
{ int s1 = 0;
  for (i=0; i<k; i++)
  for (j=0; j<5; j++)
  if (m[i][j] < x) s1 = s1 + m[i][j];
  return s1; }

```

Funksiyalarga murojaat qilish quyidagi bosqichlardan iborat bo'ladi:

1. Funksiya bajarilayotganda rasmiy parametrlar uchun xotiradan joy ajratiladi, ya'ni ular funksiyaning ichki parametrlariga aylantiriladi. Bunda parametr toifasi float toifasi double toifasiga, char va short int toifalari int toifasiga aylantiriladi.

2. Haqiqiy parametrlar qiymatlari qabul qilinadi yoki hisoblanadi.

3. Haqiqiy parametrlar rasmiy parametrlar uchun ajratilgan xotira qismiga yoziladi.

4. Funksiya tanasi ichki parametrlar yordamida bajariladi va qiymat qaytarish joyiga yuboriladi.

5. Funksiyadan chiqishda rasmiy parametrlar uchun ajratilgan xotira qismi bo'shatiladi.

Dasturdagi har bir o'zgaruvchi – Ob'yekt hisoblanadi. Uning nomi va qiymati bo'ladi. Har bir Ob'yekt xotiradan ma'lum joy egallaydi va ular ma'lum adresga ega bo'ladi. Dasturlashning ma'lum bosqichlarida o'zgaruvchining o'ziga emas, balki uning adresiga murojaat qilishga to'g'ri keladi. Bunday paytlarda ko'rsatkichlardan foydalaniladi. Ko'rsatkich - bu biror o'zgaruvchining adresini o'zida saqlovchi o'zgaruvchidir. Adres - bu xotira yacheykasining tartib nomeri. Umuman olganda adres 4 bayt joy oladi. Ko'rsatkichlarni e'lon qilishda uning toifasidan keyin * belgisi va o'zgaruvchi nomi keltiriladi.

Masalan: `int a; char *d; int *p;`

Ko'rsatkichlar ham inisalizasiya qilinishi mumkin. `*r = 6; *d = '$';`

`cout <<*p` bilan `cout <<p` ning farqi bor. `*p` da shu yerdagi qiymat chiqadi, `p` ning o'zini yozsak, shu yerning adres nomeri chiqadi. Masalan:

`int a=10, b=5, e, *m;`

`e = a + b;`

`*m = e;`

`cout <<*m;` deb yozilsa, `m = 15` chiqadi;

`cout <<m;` deb yozilsa, `m = 0xffff2` chiqadi, ya'ni shu yerning adres nomeri chiqadi.

Ularning qiymatlarini "adresini ol!" (&) operatsiyasi orqali amalga oshirsa ham bo'ladi, ya'ni `m=&e; cout <<*m;` deb yozish ham mumkin, u holda `m=15` chiqadi, ya'ni **ye** ning adresidagi son qiymat chiqadi. Buni bilvosita murojaat operatori ham deyiladi. Masalan:

`int h;`

`int *p=35;`

`h = &p;`

Natija: h = 35;

Adres olish amali (&) son yoki ifodalarga qo'llanilmaydi, ya'ni `&3.14` va `&(a+b)` yozuvlari xatodir.

Ko'rsatkichlar ustida quyidagi amallarni bajarish mumkin:

Ko'rsatkichlar ustida arifmetik amallar bajarish:

`*p1-*p2; *p1+*p2`

Ko'rsatkichlarga biror sonni qo'shish yoki ayirish:

`*p1 - 25; *p1+3.45`

Ko'rsatkichlarni bittaga oshirish yoki kamaytirish:

`*p1++ yoki --*p1`

Misol.

`# include <iostream.h>`

`# include <conio.h>`

`int main ()`

`{`

`int x=10, y=10; int *xp, *yp;`

`*xp = &x; *yp = &y;`

`if (xp == yp) cout << "ular teng!"<<endl;`

`else cout << "ular teng emas!"<<endl;`

```

if (*xp == *yp) cout << "ular teng!"<<endl;
else cout << "ular teng emas!"<<endl;
getch( ); }

```

1- if da ular teng emas chiqadi, chunki ularning adres qiymatlari har xil.

2- if da ular teng chiqadi, chunki ularning adreslaridagi son qiymatlari bir xil

2-misol.

```

# include <iostream.h>
int main ( )
{ int m = 5, *p = 0;
  p = &m;
  cout << m << endl;
  cout << *p<<endl;
  *p = 7;
  cout << m << endl;
  cout << *p << endl;
  m = 9;
  cout << m << endl;
  cout << *p << endl;
}

```

Natija:

m = 5

*p = 5

m = 7

*p = 7

Ba'zi masalalarda funksiya bilan ishlaganda funksiya tanasi ichida haqiqiy parametrlar qiymatlarini o'zgartirish zaruriyati tug'iladi, ya'ni natija bir emas, balki bir nechta hosil bo'lishi kerak bo'ladi. Bunday jarayonni proseduralar hosil qilish deyiladi va bu muammoni xal qilish uchun ko'rsatkichlardan foydalaniladi. Funksiyani aniqlashtirishda rasmiy parametrlar bilan bir satrda natijalar nomlari ham ko'rsatiladi. Shuning uchun proseduralar bilan ishlaganda funksiya toifasini bo'sh (void) deb olish maqsadga muvofiqdir, return shart bo'lmay qoladi [10].

Masalan: tomonlari berilgan to'rtburchakning perimetrini va yuzini hisoblash uchun funksiya quyidagicha aniqlashtiriladi:

```

void tt (float a,float b, float* p, float* s)
{ *p = 2*(a+b); *s = a*b; }

```

Bu yerda float a, float b beriladigan kattalik hisoblanadi, float* p, float* s lar esa natijalar hisoblanadi.

Bu funksiya murojaat qilish quyidagicha bo'ladi:

tt (2.3, 4, &p, &s); ya'ni 2*(a+b); va a*b ning qiymatlari adreslar bo'yicha olinadi.

{Proseduralarni beriladigan kattaliklarsiz ham tashkil etish mumkin. Bunda funksiya tanasi ichida ishlatilgan barcha kattaliklar beriladiganlar hisobiga o'tadi. }

Hosil qilingan proseduralarga murojaat qilish adres (&) operatsiyasi orqali amalga oshiriladi.

Masalan: $Z = \frac{a^5 + a^{-4}}{2a^n}$ hisoblash dasturi tuzilsin. Bu yerdagi darajani hisoblash prosedura

sifatida tashkil etilsin. $y = x^n$ ni prosedura deb tashkil etamiz, bu yerda x, n - rasmiy parametrlar

```

# include <iostream.h>
void dar1 (float x, int n, float *y)
{ *y=1;
  for (int i=0; i<=n; i++)
    *y = y*x; }
void main( )
{
  int n=3 ; float a, z, z1, z2, z3;
  cin>>a;
}

```

```

dar1(a, 5,&z1); dar2( 1/a, 4, &z2); dar1(a, n, &z3);
z = ( z1+z2) / z3 ;
cout << «z=»<<z<<endl;
}

```

2-misol. 2 ta vektor berilgan. Vektorlar orasidagi burchak quyidagi formula bilan hisoblanadi:

$$\varphi = \arccos \frac{(x, y)}{\sqrt{(x, x)(y, y)}}$$

bu yerda (x,u), (x,x), (u,u) - vektorlarning skalyar ko'paytmasi. Vektorlarning skalyar ko'paytmasini dasturda prosedura sifatida tashkil eting.

```

# include <iostream.h>
# include <math.h>
typedef float mm[4];
void vec(mm a, mm b, float* s)
{ *s=0;
  for (int i=0; i<4; i++)
    *s=*s+a[i]*b[i]; }
int main ( )
{ float fi, f1, f2, f3; int i;
  mm x, y; // mm x={1,2,3,4}, y={5,6,7,8};
  for (i=0; i<4; i++)
    cin >>x[i] >> y[i];
  vec (x, y, &f1); vec (x, x, &f2); vec (y,y,&f3);
  fi = f1 / sqrt( f2*f3); fi = atan(sqrt( 1-fi*fi) / fi);
  cout << «fi=»<<fi*180/3.1415<<endl; // natija gradusda chiqadi
  getch( ); }

```

Proseduralarni tashkil etishda ko'rsatkichlardan tashqari yana ilovalardan ham foydalaniladi. Bu usul yanada qulay hisoblanadi. Unda (*) amalining o'rniga to'g'ridan-to'g'ri adres olish (&) amali ishlatiladi va proseduraga murojaat qilish osonlashadi. Masalan:

```

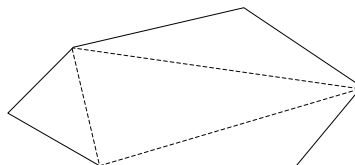
void tt (float a,float b, float& p, float& s)
{ p = 2*(a+b); s = a*b; }

```

Bu funksiyaga murojlat qilish quyidagicha bo'ladi:

```
tt (2.3, 4, p, s);
```

Misol. Bir fermerning yer yuzasini va shu yerga to'laydigan yer solig'ini hisoblash dasturini tuzing. Yer maydoni quyidagi ko'rinishda:



```

# include <iostream.h>
# include <math.h>
# include <conio.h>
# define pi 3.1415
void yuza (int a, int b, int al, float& c, float& s)
{ c = sqrt(a*a+b*b-2*a*b*cos(al*pi/180));
  s = a*b*sin(al*pi/180)/2; }
int main ( )
{ int a1=10, b1=30, a2=40, b2=40, a3=30,b3=50,al1=85, al2=145, al3=125;
  float c1, c2, c3, s1, s2, s3, s4, s, sol, p;
  yuza (a1, b1, al1, c1, s1); // yuza (10, 30, 85, c1, s1) deb yozsa ham bo'ladi

```

```

    yuza (a2, b2, a12, c2, s2);
    yuza (a3, b3, a13, c3, s3);
    p = (c1+ c2 + c3) / 2;
    s4 = sqrt ( p*(p - c1)*(p - c2)*(p - c3));
    s = s1+s2+s3+s4;
    s = s/100; sol = s * 8560; // (som)
    cout <<«er yuzasi=«<< s << endl;
    cout << «soliq=«<<sol<<endl;
    getch( ); }

```

2-misol. Kvadrat tenglamaning haqiqiy yechimlarini topish dasturini tuzing.

```

# include <iostream . h>
# include <math . h>
int kvad (float a, float b, float c, float &x1, float &x2)
{ float d ;
  d = b * b – 4*a*c;
  if ( d < 0 ) return 0;
  x1 = (-b + sqrt (d)) / (2*a);
  x2 = (-b - sqrt (d)) / (2*a);
  if ( x1 == x2) return 1;
  else return 2;
}

int main ( )
{ float a, b, c, x1, x2; int k;
  cin >> a >> b >> c;
  k = kvad (a, b, c, x1, x2);
  switch ( k )
  {
    case 0 : cout << “echimi yo‘q”<< endl; break;
    case 1 : cout << “x=”<< x1 << endl; break;
    case 2 : cout << “x1=”<< x1 << “ x2=” << x2 << endl; break;
  }
}

```

3-misol. 4x4 va 4x5 o‘lchamli matrisalar berilgan. Ulardagi juft ustunlari hadlari yig‘indisini topish dasturini tuzing. (natija vektor ko‘rinishida chiqadi)

```

# include <iostream. h>
typedef float mmm[10][10];
typedef float mm[10]; int i, j;
void nodir (mmm a, int n, mm b)
{
  for (j=0; j<4; j+=2)
  { b[j] = 0;
    for (i=0; i<n; i++)
      b[j] = b[j] + a[i][j]; } }
int main ( )
{ mmm d = {{1,2,3,4},{1,2,3,4},{1,2,3,4},{1,2,3,4}};
  mmm d1 = {{1,2,3,4,5},{1,2,3,4,5},{1,2,3,4,5},{1,2,3,4,5}};
  mm c, c1;
  nodir (d, 4, 4, c);
  nodir (d1, 4, 5, c1);
}

```

```

for (i=0; i<4; i++)
{ cout <<"c="<< c[i];
cout << "c1="<< c1[i] << endl;}
}

```

4 –misol. Ikki o‘zgaruvchining qiymatini ayirboshlash dasturi tuzilsin.

```

#include <iostream.h>
int main( )
{ float x, y;
void aa( float *, float *);
cout <<" x="<< x <<endl; cin >> x;
cout <<" y=" << y << endl; cin >> y;
aa ( &x, &y);
cout << "\nNatija: \n";
cout <<"x="<<x<<"y="<<y;
}
void aa (float *b, float *c)
{ float e;
e = *b; *b = *c; *c = e;
}

```

Asosiy dasturda x va y o‘zgaruvchilarining qiymatlari klaviaturadan kiritiladi. Masalada ikkita son o‘zaro o‘rin almashishi so‘ralmoqda. aa () funksiyaning rasmiy parametrlari sifatida float * tavsiya etilgan. aa () funksiyasiga murojaat qilinganida x va y larning son qiymatlari haqiqiy parametrlar sifatida qabul qilinadi. Bu dasturning ishlashi jarayonida quyidagi natijalar olinadi:

x=33.3 y=66.6 qiymatlar kiritilsa

Natija:

x=66.600000 y=33.300000

5-misol. Uchburchakning perimetri va uning yuzasini hisoblash uchun dastur tuzilsin.

```

#include <iostream.h>
#include <math.h>
int main ( )
{
float x, y, z, pp, ss;
int tria (float, float, float, float *, float *);
cout <<" x="; cin >> x;
cout <<" y="; cin >> y;
cout <<" z="; cin >> z;
if (tria (x, y, z, &pp, &ss)==1)
cout << "Uchburchak yuzasi="<< ss << "va perimetri="<< pp <<endl;
else
cout << "Ma'lumotlar noto'g'ri kiritilgan!" << endl;
}
int tria ( float a, float b, float c, float *pp, float *ss)
{
float e;
if (a+b<=c || a+c<=b || b+c<=a) return 0;
else
{ *pp = a+b+c;
e=*pp/2;
*s=sqrt(e*(e-a)*(e-b)*(e-c));
}
}

```

```

    return 1;
}

```

Dasturning bajarilishiga misol:

```

x=3
y=4
z=5
pp=12.00000
ss=6.00000

```

Funksiyada parametrlar sifatida massivlar va satrlar ishlatilishi mumkin. Agar funksiyaning parametri sifatida massivlar ishlatilsa, funksiya ichida massiv boshlanishining adresi uzatiladi. Bunga misol tariqasida vektorlarning skalyar ko'paytmasini hisoblovchi funksiya sarlavhasini ko'rib chiqamiz:

```

float skalyar( int n, float a[ ], b[ ]) yoki
float skalyar( int n, float *a, float *b)

```

Bu yerda float a[] va float *a yozuvlari parametr sifatida bir xil ma'noni anglatadi.

Satrlar funksiya parametri sifatida.

Satrlar funksiya parametri sifatida ishlatiladigan bo'lsa, char [] yoki char* toifali ko'rsatkichlar kabi tavsiflanadi. Oddiy massiv parametridan farqli o'laroq, satrning uzunligini ko'rsatish shart emas. Bunda \0 belgisi satr oxirini avtomatik ravishda ko'rsatadi. Misol tariqasida satrlarni qayta ishlovchi bir nechta dasturlarni ko'rib chiqamiz. Bu dasturlarning o'xshashi standart kutubxonalarda saqlanadi va ularni ishga tushirish uchun string.h stdlib.h fayllaridan foydalanish kerak bo'ladi.

1. Satr toifali o'zgaruvchining uzunligini aniqlash uchun funksiya:

```

int len(char e[ ])
{ int m;
  for(m=0; e[m]!='\0'; m++)
  return m; }

```

Yoki bu dasturdagi massivni ko'rsatkichlar orqali quyidagicha ifoda etish ham mumkin:

```

int len(char *s)
{ int m;
  for(m=0; *s!='\0'; m++)
  return m; }

```

2. Satr toifali massiv elementlarini teskari ifoda etish uchun funksiya:

```

void invert (char e[ ])
{ char s; int i, j, m;
  for (m=0; e[m]!='\0'; m++)
  for(j=0, j=m-i; i<j; i++, j- -)
  { s=e[i]; e[i] = e[j]; e[j] =s; } }

```

Dasturdagi void toifadan ma'lumki, bu funksiya hech qanday qiymat qaytarmaydi.

Masalan:

```

# include <iostream.h>
int main( )
{ char ct[ ] ="0123456789";
  void invert (char [ ]);
  invert(ct); cout << ct; }
Natija: 9876543210

```

3. Satrning chap tomonidan kiritilgan boshqa satrni qidirish funksiyasi:

```

int index(char *ct1, char *ct2)
{ int i, j, m1, m2;
  for(m1=0; ct1[m1]!='\0'; m1++)
  for(m2=0; ct2[m2]!='\0'; m2++)
  if (m2>m1) return -1;
  for(i=0; i<m1-m2; i++)
  { for(j=0; j<m2; j++)
    if (ct2[j] !=ct1[i+j] ) break;
    if (j==m2) return 1; }
  return -1; }

```

Funksiyaning ishlashiga misol:

```

#include <iostream.h>
int main ( )
{ char c1[ ] ="og`irlik_yig`indisi";
  int index(char[ ], char[ ]);
  char c2[ ] = "non";
  char c3[ ] = "olma";
  cout<< index(c1,c2);
  cout<< index(c1,c3);
}

```

Nazorat savollari

1. Funksiyaga ta'rif bering?
2. Funksiya prototipi nima?
3. Funksiya parametrlariga ta'rif bering?
4. Kelishuv bo'yicha qiymat berish deganda nimani tushunasiz?
5. O'zgaruvchining amal qilish sohasi deyilganda nima tushuniladi?
6. Funksiya bilan protsedurani farqi nimada?
7. Qiymat qaytarmaydigan funksiyalarni yana qanday nomlash mumkin?
8. Signatura deyilganda nima nazarda tutiladi?
9. Rekursiv funksiyaga ta'rif bering?
10. C++ dasturlash tilida kutubxona fayli qanday yaratiladi?

6- MA'RUZA

MAVZU: MASSIVLAR. BIR O'LCHOVLI VA KO'P O'LCHOVLI MASSIVLAR. MASSIV ELEMENTLARINI SARALASH USULLARI

Reja:

1. Massiv tushunchasi
2. Massiv turlari
3. Massiv elementlarini tartiblash

Annotatsiya: Ushbu mavzuda massiv tushunchasi, uning turlari, elementlari, o'lchamlari, elementlarga qiymat o'zlashtirish va ularni qayta qanday ishlash usullari haqida ma'lumotlar keltirilgan.

Kalit so'zlar: *Massiv, index, manzil, element.*

1. Massiv tushunchasi

Biz bu bobni massiv ma'lumotlar turini tanishtirishdan boshlaymiz. Massivlar bir necha qiymatlarni yig'ish uchun C++ da asosiy mexanizm hisoblanadi. Quyidagi bo'limlarda siz massivlarni qanday aniqlashni va massiv elementlaridan qanday foydalanishni o'rganib olasiz. Vektorlarni aniqlash deylik siz qiymatlar ketma -ketligini o'quvchi va ketma-ketlikni chop etuvchi dastur yozmoqchisiz, buning uchun siz eng katta qiymatni quyida berilgan ko'rinishda belgilang:

```
32
54
67.5
29
34.5
80
115 <= eng katta qiymat
44.5
100
65
```

Bu qiymatlarni barchasini ko'rmasdan turib, siz qaysi birini eng katta qiymat deb belgilash kerakligini bilmaysiz. Oxir oqibat, oxirgi qiymat eng kattasi bo'lishi mumkin. Shuning uchun, dastur chop etishdan oldin birinchi navbatda barcha qiymatlarni saqlab olishi kerak. Har qaysi qiymatni alohida o'zgaruvchida oddiygina saqlasangiz bo'lmasmidi? Bilsangiz, o'nta o'zgaruvchini ya'ni qiymat 1 (value1), qiymat 2 (value2), qiymat 3 (value3), ..., qiymat 10(value10) larni o'nta o'zgaruvchida saqlovchi o'nta kirituvchi mavjud. Biroq, bunday o'zgaruvchilarning ketma - ketligi foydalanish uchun noqulay. Siz, har qaysi o'zgaruvchi uchun oddiy kodni o'n marotaba yozishingizga to'g'ri keladi. Bu muammoni hal etish uchun massivdan foydalaning: qiymatlar ketma - ketligini saqlovchi struktura (tuzilma).

Xotirada ketma-ket (regulyar) joylashgan bir xil turdagi qiymatlarga *massiv* deyiladi.

2. Massiv turlari

Odatda massivlarga zarurat, katta hajmdagi, lekin cheklangan miqdordagi va tartiblangan qiymatlarni qayta ishlash bilan bog'liq masalalarni echishda yuzaga keladi. Faraz qilaylik, talabalar guruhining reyting ballari bilan ishlash masalasi qo'yilgan. Unda guruhning o'rtacha reytingini aniqlash, reytinglarni kamayishi bo'yicha tartiblash, aniq bir talabaning reytingi haqida ma'lumot berish va boshqa masala ostilarini echish zarur bo'lsin. Qayd etilgan masalalarni echish uchun berilganlarning (reytinglarning) tartib-langani ketma-ketligi zarur bo'ladi. Bu yerda tartiblanganlik

ma'nosi Shundaki, ketma-ketlikning har bir qiymati o'z o'rniga ega bo'ladi (birinchi talabaning reytingi massivda birinchi o'rinda, ikkinchi talabaniki - ikkinchi o'rinda va hakoza). Berilganlar ketma-ketligini ikki xil usulda hosil qilish mumkin. Birinchi yo'l - har bir reyting uchun alohida o'zgaruvchi aniqlash: $Reyting_1, \dots, Reyting_N$. Lekin, guruhdagi talabalar soni etarlicha katta bo'lganda, bu o'zgaruvchilar qatnashgan dasturni tuzish katta qiyinchiliklarni yuzaga keltiradi. Ikkinchi yo'l - berilganlar ketma-ketligini yagona nom bilan aniqlab, uning qiymatlariga murojaatni, shu qiymatlarning ketma-ketlikda joylashgan o'rnining nomeri (indeksi) orqali amalga oshirishdir. Reytinglar ketma-ketligini *Reyting* deb nomlab, undagi qiymatlariga $Reyting_1, \dots, Reyting_N$ ko'rinishida murojaat qilish mumkin. Odatda berilganlarning bunday ko'rinishiga massivlar deyiladi. Massivlarni matematikadagi sonlar vektoriga o'xshatish mumkin, chunki vektor ham o'zining individual nomiga ega va u fiksirlangan miqdordagi bir turdagi qiymatlardan - sonlardan iboratdir.

Demak, massiv - bu fiksirlangan miqdordagi ayrim qiymatlarning (massiv elementlarining) tartiblangan majmuasidir. Barcha elementlar bir xil turda bo'lishi kerak va bu tur *element turi* yoki massiv uchun *tayanch tur* deb nomlanadi. Yuqoridagi keltirilgan misolda *Reyting* - haqiqiy turdagi *vektor* deb nomlanadi.

Dasturda ishlatiladigan har bir aniq massiv o'zining individual nomiga ega bo'lishi kerak. Bu nomni *to'liq o'zgaruvchi* deyiladi, chunki uning qiymati massivning o'zi bo'ladi. Massivning har bir elementi massiv nomi, hamda kvadrat qavsga olingan va *element selektori* deb nomlanuvchi indeksni ko'rsatish orqali oshkor ravishda belgilanadi. Murojaat sintaksisi:

<massiv nomi>[<indeks>]

Bu ko'rinishga *xususiy o'zgaruvchi* deyiladi, chunki uning qiymati massivning alohida elementidir. Bizning misolda *Reyting* massivining alohida komponentalariga $Reyting[1], \dots, Reyting[N]$ xususiy o'zgaruvchilar orqali murojaat qilish mumkin. Boshqacha bu o'zgaruvchilar *indeksli o'zgaruvchilar* deyiladi.

Massiv indeksi sifatida butun son qo'llaniladi. Umuman olganda indeks sifatida butun son qiymatini qabul qiladigan ixtiyoriy ifoda ishlatilishi mumkin va uning qiymati massiv elementi nomerini aniqlaydi. Ifoda sifatida o'zgaruvchi ham olinishi mumkinki, o'zgaruvchining qiymati o'zgarishi bilan murojaat qilinayotgan massiv elementini aniqlovchi indeks ham o'zgaradi. Shunday qilib, dasturdagi bitta indeksli o'zgaruvchi orqali massivning barcha elementlarini belgilash (aniqlash) mumkin bo'ladi. Masalan, $Reyting[I]$ o'zgaruvchisi orqali *I* o'zgaruvchining qiymatiga bog'liq ravishda *Reyting* massivining ixtiyoriy elementiga murojaat qilish mavjud.

Haqiqiy turdagi (float, double) qiymatlar to'plami cheksiz bo'lganligi sababli ular indeks sifatida ishlatilmaydi.

C++ tilida indeks doimo 0 dan boshlanadi va uning eng katta qiymati massiv e'lonidagi uzunlikdan bittaga kam bo'ladi.

Massiv e'loni quyidagicha bo'ladi:

<tur> <nom> [<uzunlik>]={boshlang'ich qiymatlar}.

Bu erda <uzunlik> - o'zgarmas ifoda. Misollar:

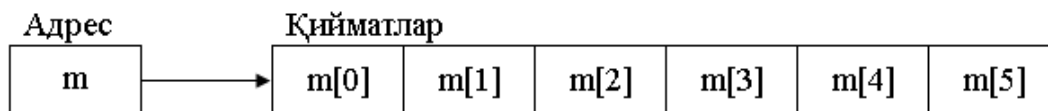
int m[6]={1,4,-5,2,10,3};

float a[4];

Massiv statik va dinamik bo'lishi mumkin. Statik massivning uzunligi oldindan ma'lum bo'lib, u xotirada ma'lum adresdan boshlab ketma-ket joylashadi. Dinamik massivni uzunligi dastur bajarilish jarayonida aniqlanib, u dinamik xotiradagi ayni paytda bo'sh bo'lgan adreslarga joylashadi. Masalan,

int m[6];

ko'rinishida e'lon qilingan bir o'lchamli massiv elementlari xotirada quyidagicha joylashadi:



7.1-rasm. Bir o'lchamli massivning xotiradagi joylashuvi

Massivning i- elementiga $m[i]$ yoki $*(m+i)$ - vositali murojaat qilish mumkin. Massiv uzunligini $\text{sizeof}(m)$ amali orqali aniqladi.

Massiv e'lonida uning elementlariga boshlang'ich qiymatlar berish mumkin va uning bir nechta variantlari mavjud.

1) o'lchami ko'rsatilgan massiv elementlarini to'liq initsializatsiyalash:

```
int t[5]={-10,5,15,4,3};
```

Bunda 5 ta elementdan iborat bo'lgan t nomli butun turdagi bir o'lchamli massiv e'lon qilingan va uning barcha elementlariga boshlang'ich qiymatlar berilgan. Bu e'lon quyidagi e'lon bilan ekvivalent:

```
int t[5];
```

```
t[0]=-10; t[1]=5; t[2]=15; t[3]=4; t[4]=3;
```

2) o'lchami ko'rsatilgan massiv elementlarini to'liqmas initsializatsiyalash:

```
int t[5]={-10,5,15};
```

Bu yerda faqat massiv boshidagi uchta elementga boshlang'ich qiymatlar berilgan. Shuni aytib o'tish kerakki, massivning boshidagi yoki o'rtasidagi elementlariga qiymatlar bermasdan, uning oxiridagi elementlarga boshlang'ich qiymat berish mumkin emas. Agarda massiv elementlariga boshlang'ich qiymat berilmasa, unda kelishuv bo'yicha static va extern modifikatori bilan e'lon qilingan massiv uchun elementlarining qiymati 0 soniga teng deb, automatic massivlar elementlarining boshlang'ich qiymatlari noma'lum hisoblanadi.

3) o'lchami ko'rsatilmagan massiv elementlarini to'liq initsializatsiyalash:

```
int t[]={-10,5,15,4,3};
```

Bu misolda massivni barcha elementlariga qiymatlar berilgan hisoblanadi, massiv uzunligi kompilyator tomonidan boshlang'ich qiymatlar soniga qarab aniqlanadi. Agarda massiv uzunligi berilmasa, boshlang'ich qiymati berilishi shart.

Massivni e'lon qilishga misollar:

```
shar ch[4]={'a','b','c','d'}; //belgilar massivi
```

```
int in[6]={10,20,30,40}; // butun sonlar massivi
```

```
char str[]="abcd";
```

```
//satr uzunligi 5 ga teng, chunki uning oxiriga
```

```
// '\0' belgisi qo'shiladi
```

```
char str[]={ 'a','b','c','d' };
```

```
// yuqoridagi satrning boshqacha yozilishi
```

Masala. Bir oy ichidagi kundalik haroratlar berilgan. Oy uchun o'rtacha haroratni hisoblash dastursi tuzilsin. Dastur matni:

```
int main(){
```

```
const int n=30;
```

```
int temp[n];
```

```
int i,s,temp_urta;
```

```
cout << "Kunlik haroratni kiriting:\n"
```

```
for (i=0;i<n;i++){
```

```
cout << "\n temp[ "<i<< "]=";
```

```
cin >> temp[i]; }
```

```
for (i=0,s=0; i<n;i++)s+=temp[i];
```

```
temp_urta=s/n;
```

```
cout << "Kunlik harorat :\n";
```

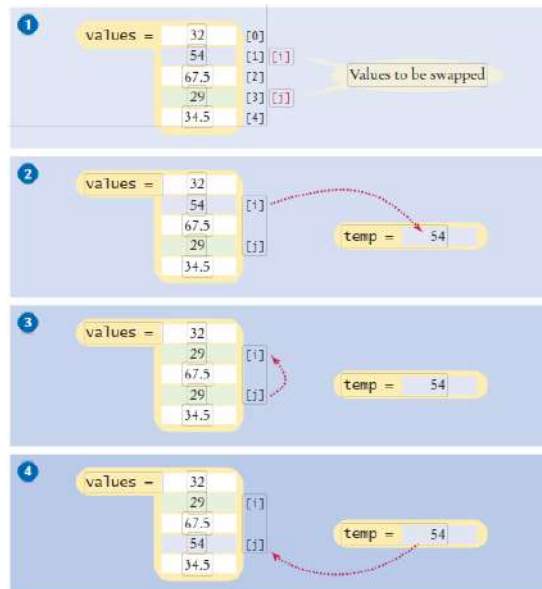
```
for(i=0;i<n;i++)
```

```
cout<<"\t temp[ "<i<< "]="<<temp[i];
```

```
cout<<"Oydagi o'rtacha harorat="
```

```
"<<temp_urta;
```

```
return 0; }
```



```

ch06/largest.cpp
#include <iostream>
using namespace std;
int main(){
    const int CAPACITY = 1000;
    double values[CAPACITY];
    int current_size = 0;
    cout << "Q ni qiymatlarni kiriting:" <<
endl;
    double input;
    while (cin >> input){
        if (current_size < CAPACITY){
            values[current_size] = input;
            current_size++; }
    }
    double largest = values[0];
    <== largest value 44.5

```

```

for (int i = 1; i < current_size; i++){
    if (values[i] > largest){
        largest = values[i]; } }
for (int i = 0; i < current_size; i++){
    cout << values[i];
    if (values[i] == largest){
        cout << " <== largest value"; }
    cout << endl; }
return 0;}

```

Program run

```

Q ni qiymatlarni kiriting:
34.5 80 115 44.5
34.5
80
115

```

3. Massiv elementlarini tartiblash

Mavzuda siz tanish algoritmlarni qo'shish va qabul qilish orqali qanday echishni ko'rdingiz. Lekin agar standard algoritmlarda hech biri sizning vazifangiz uchun yetarli bo'lmasa nima qilasiz? Bu bo'limda siz fizik predmetlarni manipulyasiya qilish orqali algoritmlarni kashf etish texnikalarini o'rganasiz.

Quyidagi topshiriqni faraz qiling. Sizga jadval berilgan, siz uning birinchi va ikkinchi yarmini boshqa holatga keltirishingiz kerak. Masalan, agar jadval 8 ta raqamdan iborat bo'lsa

9 13 21 4 11 7 1 3

siz uni Shunday o'zgartirishingiz kerak

11 7 1 3 9 13 21 4

Ko'pchilik talabalar algoritmni o'sishini biroz murakkab topishadi. Ular tizim

kerakligini bilishi mumkin va elementlarni taxlab chiqilishi yoki almashtirilishi kerakligini tan olishlari mumkin, lekin ularda diagrammalar chizish, tasvirlash va ramziy kodlarni yozishga ularda turtki yetishmadi.

Massivlarni initsializatsiyalash quyidagi misollarda ko'rsatilgan:

```
int a[2][3]={0,1,2,10,11,12};  
int b[3][3]={0,1,2},{10,11,12},{20,21,22}};  
int c[3][3][3]={0},{100,101},{110},  
{200,201,202},{210,211,212},{220,221,222}};
```

Birinci operatorida boshlang'ich qiymatlar ketma-ket yozilgan, ikkinchi operatorida qiymatlar guruhlashgan, uchinchi operatorida ham guruhlashgan, lekin ba'zi guruhlarda oxirgi qiymatlar berilmagan.

Misol uchun, matritsalar va vektor ko'paytmasini - $C = A \times b$ hisoblash masalasini ko'raylik.

Bu erda $A = \{a_{ij}\}$, $b = \{b_j\}$, $c = \{c_i\}$, $0 \leq i < m, 0 \leq j < n$. Hisoblash formulasi - $c_i = \sum_{j=0}^{n-1} a_{ij} b_j$.

Mos dastur matni:

```
int main(){  
const int n=4,m=5;  
float a[m][n],b[n],c[m];  
int i,j; float s;  
for(i=0;i<m;i++)  
for(j=0;j<n;j++)cin>>a[i][j];  
for(i=0;i<m;i++)cin>>b[i];  
for(i=0;i<m;i++) {  
for (j=0,s=0;j<n;j++)s+=a[i][j]*b[j];  
c[i]=s; }  
for (i=0;i<m;i++)cout<<"\t c["<<i<<"]="<<c[i];  
return 0; }
```

4. Ko'p o'lchamli massivlarni initsializatsiyalash

Statik massivlarning kamchiliklari shundaki, ularning o'lchamlari oldindan ma'lum bo'lishi kerak, bundan tashqari bu o'lchamlar berilganlarga ajratilgan xotira segmentining o'lchami bilan chegaralangan. Ikkinchi tomondan, yetarlicha katta o'lchamdagi massiv e'lon qilib, aniq bir masala echilishida ajratilgan xotira to'liq ishlatilmasligi mumkin. Bu kamchiliklar dinamik massivlardan foydalanish orqali bartaraf etiladi, chunki ular dastur ishlashi jarayonida kerak bo'lgan o'lchamdagi massivlarni yaratish va zarurat qolmaganda yo'qotish imkoniyatini beradi.

Dinamik massivlarga xotira ajratish uchun malloc(), calloc() funksiyalaridan yoki new operatoridan foydalanish mumkin. Dinamik ob'ektga ajratilgan xotirani bo'shatish uchun free() funksiyasi yoki delete operatori ishlatiladi.

Yuqorida qayd qilingan funksiyalar «alloc.h» kutubxonasida joylashgan.

malloc() funksiyasining sintaksisi

```
void * malloc(size_t size);
```

ko'rinishida bo'lib, u xotiraning uyum qismidan size bayt o'lchamidagi uzluksiz sohani ajratadi. Agar xotira ajratish muvaffaqiyatli bo'lsa, malloc() funksiyasi ajratilgan sohaning boshlanish adresini qaytaradi. Talab qilingan xotirani ajratish muvaffaqiyatsiz bo'lsa, funksiya NULL qiymatini qaytaradi.

Sintaksisdan ko'rinib turibdiki, funksiya void turidagi qiymat qaytaradi. Amalda esa aniq turdagi ob'ekt uchun xotira ajratish zarur bo'ladi. Buning uchun void turini aniq turga keltirish texnologiyasidan foydalaniladi. Masalan, butun turdagi uzunligi 3 ga teng massivga joy ajratishni quyidagicha amalga oshirish mumkin:

```
int * pInt=(int*)malloc(3*sizeof(int));
```

calloc() funksiyasi malloc() funksiyasidan farqli ravishda massiv uchun joy ajratishdan tashqari massiv elementlarini 0 qiymati bilan initsializatsiya qiladi. Bu funksiya sintaksisi

```
void * calloc(size_t num, size_t size);
```

ko'rinishda bo'lib, num parametri ajratilgan sohada nechta element borligini, size har bir element o'lchamini bildiradi.

free() xotirani bo'shatish funksiyasi o'chiriladigan xotira bo'lagiga ko'rsatkich bo'lgan yagona parametrga ega bo'ladi:

```
void free(void * block);
```

free() funksiyasi parametrining void turida bo'lishi ixtiyoriy turdagi xotira bo'lagini o'chirish imkonini beradi.

Quyidagi dasturda 10 ta butun sondan iborat dinamik massiv yaratish, unga qiymat berish va o'chirish amallari bajarilgan.

```
#include <iostream.h>  
#include <alloc.h>  
int main(){  
  int * pVector;  
  if ((pVector=(int*)malloc(10*sizeof(int)))==NULL) {  
    cout<<"Xotira etarli emas!!!";  
    return 1; }  
  // ajratilgan xotira sohasini to'ldirish  
  for(int i=0;i<10;i++) *(pVector+i)=i;  
  // vektor elementlarini chop etish  
  for(int i=0; i<10; i++) cout<<*(pVector+i)<<endl;  
  // ajratilgan xotira bo'lagini qaytarish (o'chirish)  
  free(pVector);  
  return 0; }
```

Keyingi dasturda $n \times n$ o'lchamli haqiqiy sonlar massivining bosh diagonalidan yuqorida joylashgan elementlar yig'indisini hisoblash masalasi yechilgan.

```
#include <iostream.h>  
#include <alloc.h>  
int main(){  
  int n;  
  float * pMatr, s=0;  
  cout<<"A(n,n): n=";  
  cin>>n;  
  if((pMatr=(float*)malloc(n*n*sizeof(float)))==NULL) {  
    cout<<"Xotira etarli emas!!!";  
    return 1; }  
  for(int i=0;i<n;i++)  
    for(int j=0;j<n;j++)  
      cin>>*(pMatr+i*n+j);  
    for(int i=0;i<n;i++)  
      for(int j=i+1;j<n;j++)  
        s+=*(pMatr+i*n+j);  
    cout<<"Matritsa bosh diagonalidan  
    yuqoridagi ";  
    cout<<"elementlar yig'indisi S="<<s<<endl;  
    return 0; }
```

new operatori yordamida, massivga xotira ajratishda ob'ekt turidan keyin kvadrat qavs ichida ob'ektlar soni ko'rsatiladi. Masalan, butun turdagi 10 ta sondan iborat massivga joy ajratish uchun

```
pVector=new int[10];
```

ifodasi yozilishi kerak. Bunga qarama-qarshi ravishda, bu usulda ajratilgan xotirani bo'shatish uchun

```
delete [] pVector;
```

ko'rsatmasini berish kerak bo'ladi.

Ikki o'lchamli dinamik massivni tashkil qilish uchun

```
int **a;
```

ko'rinishidagi «ko'rsatkichga ko'rsatkich» ishlatiladi.

Boshda massiv satrlari soniga qarab ko'rsatkichlar massiviga dinamik xotiradan joy ajratish kerak:

a=new int *[m] // bu erda m massiv satrlari soni

Keyin, har bir satr uchun takrorlash operatori yordamida xotira ajratish va ularning boshlang'ich adreslarini a massiv elementlariga joylashtirish zarur bo'ladi:

for(int i=0;i<m;i++)a[i]=new int[n];//n ustunlar soni

Shuni qayd etish kerakki, dinamik massivning har bir satri xotiraning turli joylarida joylashishi mumkin (7.1 va 7.3-rasmlar).

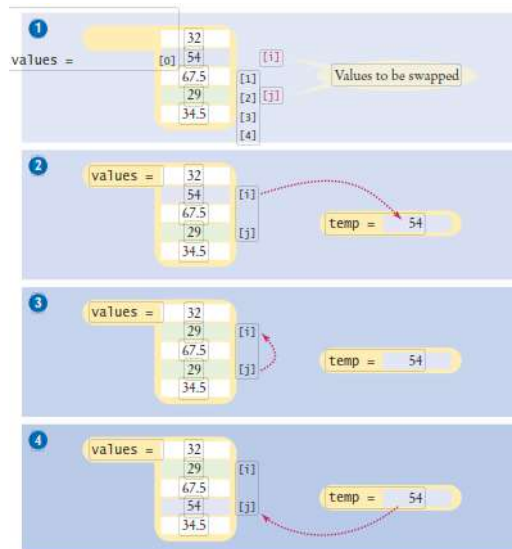
Ikki o'lchamli massivni o'chirishda oldin massivning har bir elementi (satri), so'ngra massivning o'zi yo'qotiladi:

**for(i=0;i<m;i++) delete[]a[i];
delete[]a;**

Matritsani vektorga ko'paytirish masalasi uchun dinamik massivlardan foydalanishga misol:

```
int main (){
    int n,m;
    int i,j; float s;
    cout<<"\n n="; cin>>n; // matritsa
    satrlari soni
    cout<<"\n m="; cin>>m; // matritsa
    ustunlari soni
    float *b=new float[m];
    float *c=new float[n];
    //ko'rsatkichlar massiviga xotira
    ajratish
    float **a=new float *[n] ;
    for(i=0;i<n;i++) // har bir satr uchun
    a[i]=new float[m]; //dinamik
    xotira ajratish

    for(j=0;j<m;j++)cin>>b[j];
    for(i=0;i<n;i++)
    for(j=0;j<m;j++)cin>>a[i][j];
    for(i=0;i<n;i++)
    {
        for(j=0,s=0;j<m;j++)s+=a[i][j]*b[j];
        c[i]=s;
    }
    for(i=0;i<n;i++)cout<<"\t
    c["<i<<"]="<c[i];
    delete[]b;
    delete[]c;
    for (i=0;i<n;i++) delete[]a[i];
    delete[]a;
    return; }
```



ch06/largest.cpp

```
#include <iostream>
using namespace std;
int main(){
    const int CAPACITY = 1000;
    double values[CAPACITY];
    int current_size = 0;
```

```

cout << " Q ni qiymatlarni kiriting:" << endl;
double input;
while (cin >> input) {
if (current_size < CAPACITY){
values[current_size] = input;
current_size++;}
}
double largest = values[0];
for (int i = 1; i < current_size; i++){
if (values[i] > largest){
largest = values[i]; }}
for (int i = 0; i < current_size; i++){
cout << values[i];
if (values[i] == largest){
cout << " <== largest value"; 37 }
cout << endl; }
return 0; }

```

Program run

```

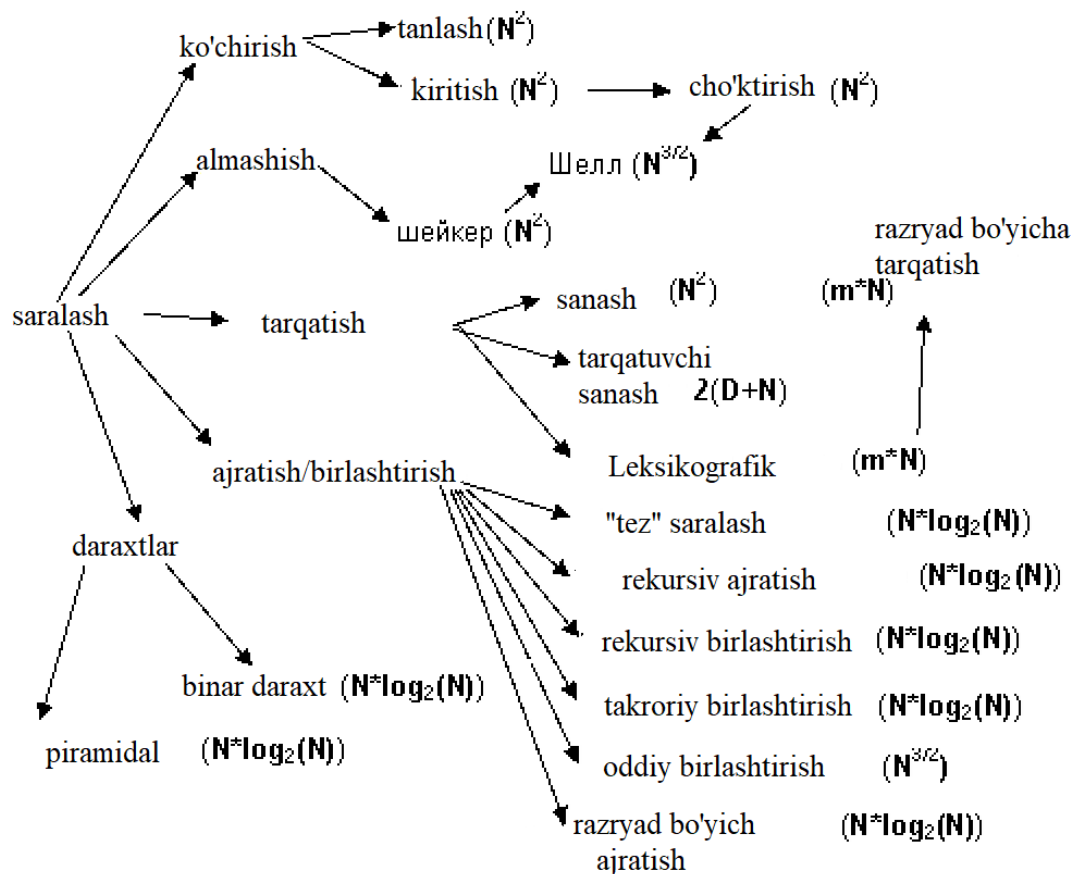
Q ni qiymatlarni kiriting:
34.5 80 115 44.5 Q
34.5
80
115 <== largest value
44.5}

```

Umuman olganda saralashning maqsadi berilgan ob'yektlar to'plamini aniq bir tartibda guruhlab chiqish jarayoni ta'riflanadi.

$a_0, a_1 \dots a_n$ ketma – ketlik berilgan bo'lsin.

Ketma – ketlikning elementlarini saralash (masalan: $a_i \leq a_{i+1}$, $i = 0$ dan $n-1$ gacha) masalasi qo'yilgan bo'lsin. Bu masalani ishlash algoritmini tanlaganda quyidagilarni baholash zarur: *Saralash vaqti* – algoritmnı baholaydigan asosiy parametr hisoblanadi. *Xotira* – algoritm ishlashi uchun ketadigan qo'shimcha xotira hajmi. Bunda berilganlar va dastur kodi uchun ketadigan xotira hajmi hisobga olinmaydi. *Turg'unlik* – dasturnı ketma – ketlikning boshqa qiymatlarda ham turg'un ishlashi tushuniladi. Saralash algoritmlari klassifikatsiyasi. Berilgan ketma – ketlikni saralashda ketma – ketlikning xarakteristikasi xususiyatiga mos ravishda u yoki bu saralash algoritmi olinadi. Aks holda algoritmlar kerakli natijani bermaydi.



Rasm 1. Saralash algoritmi klassifikatsiyasi.

3. Ketma-ketlikni saralash

1. "tez" saralash algoritmi:

"Tez" saralash algoritmi bosqichlari:

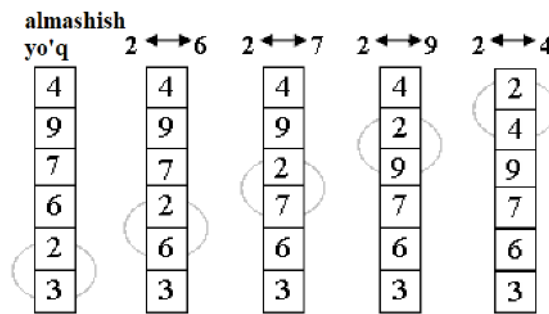
1. Massivning o'rta elementini tanlab olamiz.
2. O'rta element chap tomoniga o'rta elementdan kichik elementlarni joylashtiramiz, o'rta elementning o'ng tomoniga esa o'rta elementdan katta elementlarni joylashtiramiz.

```

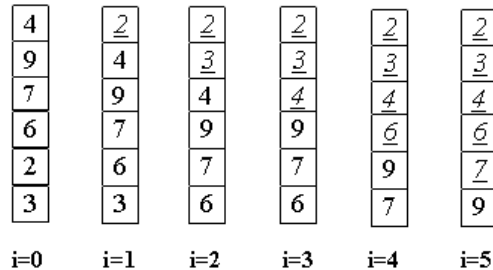
int i=quyi;
int j=yuqori;
int x=A[(quyi+yuqori)/2];
do {
  while(A[i]<x) ++i;
  while(A[j]>x) --j;
  if(i<=j){
    int temp=A[i];
    A[i]=A[j];
    A[j]=temp;
    i++; j--;
  }
} while(i<=j);
  
```

2. Sharsimon saralash algoritmi

Massiv elementlarini tepadan pastga qarab saralaymiz. Bunda faqat juft elementlar $a_i \leq a_{i+1}$ ($i = 0$ dan $n-1$ gacha) shart bilan tekshiriladi, agar shart bajarilmasa ular o'zaro o'rin almashtiriladi.



Bu jarayon ohirgi element qolguncha bajariladi. Natijada massiv elementlari o'sish tartibida saralanadi. Dasturdagi bosqichlar:



```

long i, j,
float x, a[];
for( i=0; i < size; i++)
for( j = size-1; j > i; j-- )
{ if ( a[j-1] > a[j] )
  { x=a[j-1]; a[j-1]=a[j]; a[j]=x; } }

```

Saralashning maqsadi keyinchalik, saralashgan to'plamni qidirilayotgan elementini topishdan iborat. Bu qariyb universal, fundamental jarayon. Biz bu jarayon bilan har kuni uchrashamiz – telefon daftaridagi saralash, kitoblar sarlavhasida, kutubxonalarda, lug'atlarda, pochta va h.k. Hatto yosh bolalar ham o'z narsalarini tartiblashga o'rganadi. Saralashning juda ko'p usullari mavjud. Ular turli to'plamlar uchun turlicha bo'lishi mumkin. Massivlarni saralash uchun ishlatiladigan usul unga berilgan xotirani ixcham holda ishlatish lozim. Boshqacha qilib aytganda, saralanayotgan massiv xuddi shu massivni o'zida amalga oshirilishi lozim. Saralanayotgan a massivni elementlarini kiritib, unda boshqa bir d massivda saralangan holda tashkil topgan bizga hech qanday qiziqish uyg'otmaydi. Biz quyidagi saralash bo'yicha bir nechta sodda va ma'lum usullarni qaraymiz. Ular to'g'ri usullar deb aytiladi. Saralash usullari to'g'risida quyidagi fikrlarni bildirish mumkin:

1. To'g'ri usullar ko'plab saralashning asosiy tamoyillarining xarakterini ochib berishi uchun qulay.
2. Bu usullarni dasturlar oson tushuniladi va ular qisqa. Eslatib o'tamiz, dasturning o'zi ham xotira egallaydi.
3. Murakkab usullar ko'p sondagi amallarni talab qiladi, lekin bu amallarning o'zlari yetarlicha murakkab bo'lganlari uchun, kichik n larda tez va katta n larda sekin ishlaydi. Ammo ularni katta n larda ishlab bo'lmaydi.

Bitta massivni o'zida saralashni ularni mos aniqlangan tamoyillari bilan uch kategoriyaga ajratish mumkin:

1. Qo'shish orqali saralash (by insertion);
2. Ayirish orqali saralash (by selection);
3. Almashish orqali saralash (by exchange).

3. To'g'ridan-to'g'ri qo'shish orqali saralash

Bu usul karta o'yinida ko'p qo'llaniladi.

Kartaning elementlari fikran tayyor holdagi ketma-ketlik qismlarga bo'linadi.

Har qadamda $i=2$ dan boshlab i ta element ketma-ketlikdan chiqariladi va tayyor ketma-ketlikka qo'yiladi. Bunda u har doim kerakli joyga qo'yiladi. i ning qiymati har doim bittaga oshirib boriladi.

Бошланғич	4	↓	4	9	1	0	6		
i = 2	4	↓	5	1	4	9	1	0	6
i = 3	1	4	↓	5	4	9	1	0	6
i = 4	1	4	4	5	↓	9	1	0	6
i = 5	1	4	↓	4	5	9	1	0	6
i = 6	1	↓	1	4	4	5	9	0	6
i = 7	0	1	1	4	4	5	↓	9	6
i = 8	0	1	1	4	4	5	6	9	

To'g'ridan to'g'ri tanlash yordamida saralash

- Eng kichik kalitli element tanlanadi.
- Uni birinchi element a_1 bilan o'rinlari almashtiriladi.

So'ng bu jarayon qolgan $n-1$ element bilan, so'ngra $n-2$ element bilan va h.k. bitta eng katta element qolmaguncha davom ettiriladi.

Бошланғич калитлар	<u>44</u>	55	12	42	94	1	0	67
i = 2		55	12	42	94	18		67
i = 3		<u>12</u>	<u>55</u>	42	94	18		67
i = 4		12	18	42	94			67
i = 5		12	18	42	<u>94</u>			67
i = 6		12	18	42	4			67
i = 7		12	18	42	44	55	<u>94</u>	67
i = 8		12	18	42	44	55	67	94

Pufaksimion saralash:

i =	1	2	3	4	5	6	7	8
	44	06	06	06	06	06	06	06
	55	44	12	12	12	12	12	12
	12	55	44	18	18	18	18	18
	42	12	55	44	42	42	42	42
	94	42	18	55	44	44	44	44
	18	94	42	42	55	55	55	55
	06	18	94	67	67	67	67	67
	67	67	67	94	94	94	94	94

ShEYKER saralash usuli

L =	2	3	3	4	4
R =	8	8	7	7	4
dir =	↑		↑		↑
	44	06	06	06	06
	55	44	44	12	12
	12	55	12	44	18
	42	12	42	18	42
	94	42	55	42	44
	18	94	18	55	55
	06	18	67	67	67
	67	67	94	94	94

Nazorat savollari

1. C++da massiv qanday ishlaydi?
2. Massivga kutubxona kerakmi?
3. Rekursiv funksiya nima?
4. Rekursiv funksiya parametrlari.
5. for operatori funksiyada qanday ishlatiladi?
6. Matematik funksiyalar qanday ishlaydi?
7. Massiv elementlarini Funksiya parametrlarida uzatish nima uchun ishlatiladi?
8. Massivlarni qabday turlari mavjud?

9. Funksiya parametrlari orqali nima uzatiladi?
10. Massivning necha xil turi bor?
11. Massivlar nima uchun qo'llaniladi?

6.2- MAVZU: IKKI O'LCHAMLI MASSIVLAR

Reja:

1. Ko'p o'lchamli massivlarni tavsiflash, ularga ishlov berish;
2. Massiv elementlarini kiritish va chiqarish;
3. Ko'p o'lchamli massivlarni saralash. Misollar.

Annotatsiya: Ushbu mavzuda ko'p o'lchovli massivlar, ularning elementlariga murojaat, elementlarning qiymatlarini chiqarish, elementlariga qiymat o'zlashtirish va qayta ishlov berish yo'llari haqida ma'lumotlar keltirilgan.

Kalit so'zlar: *qator, ustun, elementlar, ichki, jadval, jadval katakchalari, katakchalar manzili.*

C/C++ algoritmik tilida faqat bir o'lchamli massivlar bilan emas, balki ko'p o'lchamli massivlar bilan ham ishlash mumkin. Agar massiv o'z navbatida yana massivdan iborat bo'lsa, demak ikki o'lchamli massiv, ya'ni matrisa deyiladi. Massivlarning o'lchovi kompyuterda ishlashga to'sqinlik qilmaydi, chunki ular xotirada chiziqli ketma-ket elementlar sifatida saqlanadi. Ko'p o'lchamli massivlarni xuddi 1 o'lchamli massivga o'xshab e'lon qilinadi, faqat indeks toifasi sifatida massivning satrlari (qatorlari) va ustunlari toifasi ko'rsatiladi va ular alohida [][] qavslarda ko'rsatiladi. Masalan: A nomli butun sonlardan iborat 2 o'lchamli massiv berilgan bo'lsa va satrlar soni **n** ta, ustunlar soni **m** ta bo'lsa: `int a[n][m]`

Ikki ulchovli massiv elementlarini kiritish-chiqarish, ular ustida amallar bajarish ichma-ich joylashgan parametrli takrorlashlar ichida bo'ladi, ya'ni 1-takrorlash satrlar uchun, 2-takrorlash ustunlar uchun. Masalan:

```
for ( i=0; i<=3; i++)
  for ( j=0; j<=3; j++)
    cin >>a[i][j];
```

Agar ularni klaviaturadan kiritish kerak bo'lsa, ya'ni cin operatori yordamida tashkil etilsa, quyidagicha kiritiladi:

```
1 2 3
4 5 6
7 8 9
```

Bundan tashqari massiv elementlarini e'lon qilish bilan birga ularni inisalizasiya ham qilish mumkin:

```
int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
```

Natijalar chiroyli ko'rinishda bo'lishi uchun chiqarish operatorini quyidagicha qilib tashkil etish kerak:

```
for (int i=0; i<3; i++)
{ for (int j=0; j<3; j++)
  cout <<"a["<<i<<","<<j<<"]="<<a[i][j];
  cout <<endl; }
getch ( );
}
```

1-misol. A va V matrisalari berilgan. Quyidagi formula orqali yangi S matrisasini hosil qiling: $S_{ij} = A_{ij} + B_{ij}$; bu yerda $i=1,3$; $j=1,2$;

$$A = \begin{Bmatrix} 24,3 & -4,15 \\ 0 & 18,4 \\ -8,8 & -15,75 \end{Bmatrix} \quad V = \begin{Bmatrix} 0,1 & -4,8 \\ 6,8 & 7,1 \\ -2,8 & 0,40 \end{Bmatrix}$$

```
#include <iostream.h>
#include <conio.h>
void main ( )
{ float a[3][2]={ {24.3,-4.15},{0,18.4},{-8.8,-15.75}},
  b[3][2]={ {0.1,-4.8},{6.8,7.1},{-2.8,0.40}};
  float c[3][2];
  int i, j;
  for (i = 0; i < 3; i++)
  { for (j = 0; j < 3; j++)
    { c[i][j] = a[i][j] + b[i][j];
      cout <<"c["<<i<<","<<j<<"]="<<c[i][j]; }
    cout <<endl; }
  getch ( );
}
```

Massiv elementlariga son qiymat berishda kompyuter xotirasidagi tasodifiy butun sonlardan foydalanish ham mumkin. Buning uchun standart kutubxonaning rand () funksiyasini ishga tushirish kerak. rand () funksiyasi yordamida 0 ÷ 32767 oraliqdagi ixtiyoriy sonlarni olish mumkin. Bu qiymatlar umuman tasodifiydir. (psevdo – tasodifiy degani).

Agar dastur qayta-qayta ishlatilsa, ayni tasodifiy qiymatlar takrorlanaveradi. Ularni yangi tasodifiy qiymatlar qilish uchun srand () funksiyasini dasturda bir marta e’lon qilish kerak. Dastur ishlashi jarayonida ehtiyojga qarab rand () funksiyasi chaqirilaveradi. Tasodifiy qiymatlar bilan ishlash uchun <stdlib.h> faylini e’lon qilish zarur. srand () funksiyasidagi qiymatni avtomatik ravishda o’zgaradigan holatga keltirish uchun srand (time (NULL)) yozish ma’qul, Shunda kompyuter ichidagi soatning qiymati time () funksiyasi yordamida o’rnatiladi va srand ga parametr sifatida beriladi. NULL yoki 0 deb yozilsa, qiymat sekundlar ko’rinishida beriladi. Vaqt bilan ishlash uchun <time.h> ni e’lon qilish kerak. Misol:

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
int main ( )
{ srand ( time (0));
  int a[5], b[5], i;
  for (i = 0; i < 5; i++) a[i] = rand ( );
  for (i = 0; i < 5; i++)
  { b[i] = a[i] + 64;
    cout << "b="<<b[i]<<endl; } getch ( ); }
```

Izoh: tasodifiy sonlar ichida manfiy sonlarning ham qatnashishini ixtiyor etsak, a[i] = 1050 - rand (); yoki a[i] = rand ()-1000; deb yozish ham mumkin.

2-misol. 2 ta matrisa berilgan. Ularni o’zaro ko’paytirib yangi matrisa hosil qiling. Bu yerda 1-matrisaning ustunlar soni 2-matrisaning satrlar soniga teng bo’lishi kerak.

```
#include <iostream.h>
#include <conio.h>
```

```

# include <stdlib.h>
# include <time.h>
int main ( )
{
    { srand ( time (0));
    int a[3][3], b[3][3],c[3][3], i, j, k;
    for (i=0; i<3; i++)
    for (j=0; j<3; j++)
        a[i][j] = rand ( );
    for (i=0; i<3; i++)
    for (j=0; j<3; j++)
        b[i][j] = rand ( );
    for (i=0; i<3; i++)
    { for (j=0; j<3; j++)
        { c[i][j] = 0;
        for (k=0; k<3; k++)
            c[i][j] = c[i][j] + a[i][k]*b[k][j];
        cout <<"c="<<c[i][j]<<"\t"; }
        cout << endl; }
    getch ( );}

```

3-misol. A matrisani V vektorga ko'paytirish algoritmi. $S_i = \sum_{j=1}^n a_{ij} * b_j$

Izoh: matrisaning satrlari soni vektorning satrlariga teng bo'lishi kerak.

Masalan:

$$A = \begin{Bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{Bmatrix} \quad B = \begin{Bmatrix} 1 \\ 3 \\ 6 \end{Bmatrix}$$

$$C_1 = 1*1+2*3+3*6 = 25$$

$$C_2 = 4*1+5*3+6*6 = 55$$

$$C_3 = 7*1+8*3+9*6 = 85$$

```

# include <iostream.h>
# include <conio.h>
int main ( ){
    int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}}, b[3] = {1,3,6}, c[3], i, j;
    for (i=0; i<3; i++)
    { c[i] = 0;
    for (j=0; j<3; j++)
        c[i] = c[i] + a[i][j] * b[j];
    cout <<"c="<<c[i]<<endl; }
    getch ( );
}

```

4-misol. Matrisani transponerlash algoritmini tuzing. Matrisani transponerlash deb, ustun va satr elementlarini o'zaro o'rin almashtirishga aytiladi, ya'ni $A_{ij} = B_{ji}$

```

# include <iostream.h>
# include <conio.h>
int main ( ){

```

```

int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}}, b[3][3],
i, j;
for ( i=0; i<3; i++)
{ for ( j=0; j<3; j++)
{ b[i][j] = a[j][i];
cout <<"b["<<i<<","<<j<<"]="<<b[i][j]; }
cout << endl; }
getch ( );}

```

5-misol. 3 ta qator va 4 ta ustunga ega A matrisa berilgan. Undagi eng kichik elementni va uning indeksini topish, hamda o'sha qatorni massiv shaklida chiqarish dasturini tuzing.

```

#include <iostream.h>
#include <conio.h>
int main ( ){
int a[3][4] = {{1,2,3,4},{4,5,6,7},{7,8,9,10}}, i, j, k, h, min;
int b[4];
min = a[0][0];
for (i=0; i<3; i++)
for (j=0; j<4; j++)
{ if ( a[i][j] > min) { min = a[i][j]; k = i; h = j; } }
cout << "min="<<min<<" k="<<k<<" h="<<h<<endl;
for ( j=0; j<4; j++)
{ b[j] = a[k][j];
cout <<"b="<<b[j]; }
getch ( ); }

```

6-misol. Saralash masalasi. Massiv elementlarini o'sib borish tartibida saralash algoritmini tuzing. (Pufaksimon saralash usuli)

Avval 1 o'lchovli massiv elementlarini saralashni ko'rib o'tamiz.

```

#include <iostream.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
int main ( ){
srand (time (0));
float a[10] , b; int i, j;
for (i = 0; i<10; i++)
a[i] = rand( ) /33.;
for( j = 0; j<10; j++)
for ( i = 0; i < 10; i++){
if (a[i] < a[i+1])
{ b = a[i];
a[i] = a[i+1];
a[i+1] = b; } }
cout. precision (3);
for (i = 0; i < 10; i++)
cout << a[i]<<endl; getch ( ); }

```

Endi 2 o'lchamli massiv elementlarini saralashni ko'ramiz:

```

#include <iostream.h>
#include <conio.h>

```

```

int main ( ){
float a[3][3] = {{.....},{.....},{.....}}, b;
int i, j, k;
for ( k=0; k<3; k++)
for ( i=0; i<3; i++)
for ( j=0; j<2; j++)
{ if (a[i][j] > a[i][j+1] )
{ b = a[i][j]; a[i][j] = a[i][j+1]; a[i][j+1] = b; } }
for ( i=0; i<3; i++)
{ for ( j=0; j<3; j++)
cout <<"a="<<a[i][j]; cout << endl; } getch ( ); }

```

Yuqoridagi dastur saralashni qator bo'yicha olib borish uchun. Agar saralashni ustun bo'yicha qilish kerak bo'lsa, quyidagicha yozish kerak bo'ladi:

```

for ( i=0; i< 2; i++)
for ( j=0; j<3; j++)
{ if ( a[i,j] > a[i+1, j] )
{ b:= a[i, j]; a[i, j]:= a[i+1, j]; a[i+1, j]:= b; }

```

Agar saralashni o'sib borish tartibida qilish kerak bo'lsa, if operatoridagi solishtirish belgisi > bo'lishi kerak, agar kamayish tartibida kerak bo'lsa, solishtirish belgisi < ko'rinishida bo'lishi kerak.

7-misol. Matrisaning izini hisoblash dasturini tuzing. Matrisaning izi deb bosh diagonal elementlarining yig'indisiga aytiladi. Shu dasturda teskari (qo'shimcha) diagonal elementlarining yig'indisini ham hisoblashni ko'rib o'ting.

```

#include <iostream.h>
#include <conio.h>
int main ( ){
float a[3][3] = {{.....},{.....},{.....}}, s1=0, s2=0;
int i, j;
for ( i=0; i<3; i++)
s1 = s1 + a[i][i];
for ( i=0; i<3; i++)
for ( j=0; j<3; j++)
if ( i+j == 2) s2 = s2 + a[i][j];
cout <<"s1="<<s1<<" s2="<< s2<< endl;
getch ( ); }

```

8-misol. Har bir hadi $a_n = \frac{n!}{(2n)!}$ formulasi orqali hisoblanadigan satr yig'indisini 0,0001 aniqlikda

hisoblash dasturini tuzing.

```

#include <iostream.h>
#include <conio.h>
int main ( )
{ int n =1; float s1 = 0, s2 = 0;
float p1 =1, p2 = 1;
while (s2 > 0.0001)
{ p1 = p1 * n; // p1*=n;
p2 = p2 * 2*n*(2*n-1); // p2*= 2*n*(2*n-1);
s2 = p1 / p2;
s1 = s1 + s2; // s1+= s2;
n ++; }

```



```
cout<<precision(3);
cout << "s1="<<s1 <<" n=" << n << endl;
}
```

Nazorat savollari

1. Ikki o'lchamli massivlar.
2. Saralash usullari. To'g'ri tanlash usuli.
3. Eng katta element joylashgan satr yoki ustunni o'chirish algoritmi.
4. Matrisani matrisaga ko'paytirish algoritmi.
5. C++da massiv qanday ishlaydi?
6. Massivga kutubxona kerakmi?
7. Rekursiv funksiya nima?
8. Massiv elementlarini bilan massiv indekslarini farqi nimada?
9. for operatori funksiyada qanday ishlatiladi?
10. Matematik funksiyalar qanday ishlaydi?
11. Massiv elementlarini Funksiya parametrlarida uzatish nima uchun ishlatiladi?
12. Massivlarni qanday turlari mavjud?
13. Funksiya parametrlari orqali nima uzatiladi?
14. Massivning necha xil turi bor?

6.3- MAVZU: IKKI O'LCHAMLI MASSIVLAR (KO'RSATKICHLARNING BOG'LANISHI)

Reja:

1. Ko'rsatgichli massivlar haqida;
2. Funksiya va massivlar.

Annotatsiya: Ushbu ma'ruzada ikki o'lchovli massivlarni ko'rsatkichlar yordamida aks ettirish va ularni parameter sifatida qo'llanilishi haqida ma'lumotlar keltirilgan.

Kalit so'zlar: *ro'yxat, manzil, nolinchi ko'rchsatkich, tugun, adres olish &, bo'shatish, ko'rsatkich, virtual destruktork, xotira, xotira chiqishi, destruktork, toifani o'zlashtirish, resurslar chiqishi, a'zo destruktork.*

1. Ko'rsatkichli massivlar bilan ishlash

Statik massivlarning kamchiliklari shundaki, ularning o'lchamlari oldindan ma'lum bo'lishi kerak, bundan tashqari bu o'lchamlar berilganlarga ajratilgan xotira segmentining o'lchami bilan chegaralangan. Ikkinchi tomondan, yetarlicha katta o'lchamdagi massiv e'lon qilib, aniq masala echilishida ajratilgan xotira to'liq ishlatilmasligi mumkin. Bu kamchiliklar dinamik massivlardan foydalanish orqali bartaraf etiladi, chunki ular dastur ishlashi jarayonida kerak bo'lgan o'lchamdagi massivlarni yaratish va zarurat qolmaganda yo'qotish imkoniyatini beradi.

Dinamik massivlarga xotira ajratish uchun malloc(), calloc() funksiyalaridan yoki new operatoridan foydalanish mumkin. Dina-mik ob'ektga ajratilgan xotirani bo'shatish uchun free() funksiyasi yoki delete operatori ishlatiladi.

Yuqorida qayd qilingan funksiyalar «alloc.h» kutubxonasida joylashgan.

malloc() funksiyasining sintaksisi

```
void * malloc(size_t size);
```

ko'rinishida bo'lib, u xotiraning uyum qismidan size bayt o'lchamidagi uzluksiz sohani ajratadi. Agar xotira ajratish muvaffaqiyatli bo'lsa, malloc() funksiyasi ajratilgan sohaning boshlanish adresini qaytaradi. Talab qilingan xotirani ajratish muvaffaqiyatsiz bo'lsa, funksiya NULL qiymatini qaytaradi.

Sintaksisdan ko'rinib turibdiki, funksiya void turidagi qiymat qaytaradi. Amalda esa aniq turdagi ob'ekt uchun xotira ajratish zarur bo'ladi. Buning uchun void turini aniq turga keltirish texnologiyasidan foydalaniladi. Masalan, butun turdagi uzunligi 3 ga teng massivga joy ajratishni quyidagicha amalga oshirish mumkin:

```
int * pInt=(int*)malloc(3*sizeof(int));
```

calloc() funksiyasi malloc() funksiyasidan farqli ravishda massiv uchun joy ajratishdan tashqari massiv elementlarini 0 qiymati bilan initsializatsiya qiladi. Bu funksiya sintaksisi

```
void * calloc(size_t num, size_t size);
```

ko'rinishda bo'lib, num parametri ajratilgan sohada nechta element borligini, size har bir element o'lchamini bildiradi.

free() xotirani bo'shatish funksiyasi o'chiriladigan xotira bo'la-giga ko'rsatkich bo'lgan yagona parametrga ega bo'ladi:

```
void free(void * block);
```

free() funksiyasi parametrining void turida bo'lishi ixtiyoriy turdagi xotira bo'lagini o'chirish imkonini beradi.

Quyidagi dasturda 10 ta butun sondan iborat dinamik massiv yaratish, unga qiymat berish va o'chirish amallari bajarilgan.

```
#include <iostream.h>  
#include <alloc.h>  
int main(){  
  int * pVector;  
  if ((pVector=(int*)malloc(10*sizeof(int)))==NULL) {  
    cout<<"Xotira yetarli emas!!!";  
    return 1; }  
  // ajratilgan xotira sohasini to'ldirish  
  for(int i=0;i<10;i++) *(pVector+i)=i;  
  // vektor elementlarini chop etish  
  for(int i=0; i<10; i++) cout<<*(pVector+i)<<endl;  
  // ajratilgan xotira bo'lagini qaytarish (o'chirish)  
  free(pVector);  
  return 0; }
```

Keyingi dasturda $n \times n$ o'lchamli haqiqiy sonlar massivi-ning bosh diagonalidan yuqorida joylashgan elementlar yig'indi-sini hisoblash masalasi echilgan.

```
#include <iostream.h>  
#include <alloc.h>  
int main(){  
  int n;  
  float * pMatr, s=0;  
  cout<<"A(n,n): n=";  
  cin>>n;  
  if((pMatr=(float*)malloc(n*n*sizeof(float)))==NULL) {  
    cout<<"Xotira yetarli emas!!!";  
    return 1; }  
  for(int i=0;i<n;i++)  
    for(int j=0;j<n;j++)cin>>*(pMatr+i*n+j);  
  for(int i=0;i<n;i++)
```

```

for(int j=i+1;j<n;j++)s+=*(pMatr+i*n+j);
cout<<"Matritsa bosh diagonalidan yuqoridagi ";
cout<<"elementlar yig'indisi S="<<s<<endl;
return 0;
}

```

new operatori yordamida, massivga xotira ajratishda ob'ekt turidan keyin kvadrat qavs ichida ob'ektlar soni ko'rsatiladi. Masalan, butun turdagi 10 ta sondan iborat massivga joy ajratish uchun

```
pVector=new int[10];
```

ifodasi yozilishi kerak. Bunga qarama-qarshi ravishda, bu usulda ajratilgan xotirani bo'shatish uchun

```
delete [] pVector;
```

ko'rsatmasini berish kerak bo'ladi.

Ikki o'lchamli dinamik massivni tashkil qilish uchun

```
int **a;
```

ko'rinishidagi «ko'rsatkichga ko'rsatkich» ishlatiladi.

Boshida massiv satrlari soniga qarab ko'rsatkichlar massivga dinamik xotiradan joy ajratish kerak:

```
a=new int *[m] // bu erda m massiv satrlari soni
```

Keyin, har bir satr uchun takrorlash operatori yordamida xotira ajratish va ularning boshlang'ich adreslarini a massiv elementlariga joylashtirish zarur bo'ladi:

```
for(int i=0;i<m;i++)a[i]=new int[n];//n ustunlar soni
```

Shuni qayd etish kerakki, dinamik massivning har bir satri xotiraning turli joylarida joylashishi mumkin (7.1 va 7.3-rasmlar).

Ikki o'lchamli massivni o'chirishda oldin massivning har bir elementi (satri), so'ngra massivning o'zi yo'qotiladi:

```
for(i=0;i<m;i++) delete[]a[i];
```

```
delete[]a;
```

Matritsani vektorga ko'paytirish masalasi uchun dinamik massivlardan foydalanishga misol:

```
int main (){
```

```
int n,m;
```

```
int i,j; float s;
```

```
cout<<"\n n="; cin>>n; // matritsa
```

```
satrlari soni
```

```
cout<<"\n m="; cin>>m; //matritsa
```

```
ustunlari soni
```

```
float *b=new float[m];
```

```
float *c=new float[n];
```

```
// ko'rsatkichlar massivga xotira
```

ajratish

```
float **a=new float *[n] ;
```

```
for(i=0;i<n;i++) // har bir satr uchun
```

```
a[i]=new float[m]; //dinamik
```

xotira ajratish

```
for(j=0;j<m;j++)cin>>b[j];
```

```
for(i=0;i<n;i++)
```

```
for(j=0;j<m;j++)cin>>a[i][j];
```

```
for(i=0;i<n;i++) {
```

```
for(j=0,s=0;j<m;j++)s+=a[i,j]*b[j];
```

```
c[i]=s; }
```

```
for(i=0;i<n;i++)cout<<"\t
```

```
c["<<i<<"]="<<c[i];
```

```
delete[]b;
```

```
delete[]c;
```

```
for (i=0;i<n;i++) delete[]a[i];
```

```
delete[]a;
```

```
return;} 
```

3. Funksiya va massivlar

Funksiyalar massivni parametr sifatida ishlatishi va uni funksiyaning natijasi sifatida qaytarishi mumkin.

Agar massiv parametr orqali funksiyaga uzatilsa, elementlar sonini aniqlash muammosi tug'iladi, chunki, massiv nomidan uning uzunligini aniqlashning iloji yo'q. Ayrim hollarda, masalan, belgilar massivi sifatida aniqlangan satr (ASCIIZ satrlar) bilan ishlaganda massiv uzunligini aniqlash mumkin, chunki satrlar '\0' belgisi bilan tugaydi.

Misol uchun:

```
#include <iostream.h>
int len(char s[])
//massivni parametr sifatida ishlatish
{
    int m=0;
```

```
while(s[m++]);
return m-1; }
int main (){
    char z[]="Ushbu satr uzunligi = ";
    cout<<z<<len(z); }
```

Funksiya parametri satr bo'lmagan hollarda fiksirlangan uzunlikdagi massivlar ishlatiladi. Agar turli uzunlikdagi massivlarni uzatish zarur bo'lsa, massiv o'lchamlarini parametr sifatida uzatish mumkin yoki bu maqsadda global o'zgaruvchidan foydalanishga to'g'ri keladi.

Misol:

```
#include <iostream.h>
float sum(int n,float *x) //bu ikkinchi usul
{
    float s=0;
    for (int i=0;i<n;i++)s+=x[i];
    return s;}
int main(){
    float E[]={1.2,2.0,3.0,4.5,-4.0};
    cout<<sum(5,E);
}
```

Massiv nomi ko'rsatkich bo'lganligi sababli massiv elementlarini funksiyada o'zgartirish mumkin va bu o'zgartirishlar funksiyadan chiqqandan keyin ham saqlanib qoladi.

```
#include <iostream.h>
void vector_01(int n,int*x,int * y) {
//bu ikkinchi usul
    for (int i=0;i<n;i++)
        y[i]=x[i]>0?1:0; }
int main(){
```

```
int a[]={1,2,-4,3,-5,0,4};
int c[7];
vector_01(7,a,c);
for(int i=0;i<7;i++)
    cout<<'t'<<c[i]; }
```

Masala. Butun turdagi va elementlari kamaymaydigan holda tartiblangan bir o'lchamli ikkita massivlarni yagona massivga, tartiblanish saqlangan holda birlashtirish amalga oshirilsin.

Dastur matni:

```
#include <iostream.h>
\\butun turdagi massivga ko'rsatkich
qaytaradigan. funksiya
int * massiv_ulash(int,int*,int,int*);
int main(){
    int c[]={-1,2,5,10},d[]={1,7,8};
    int * h;
    h=massiv_ulash(5,c,3,d);
    for(int i=0;i<8;i++) cout<<'t'<<h[i];
    delete[]h; }
```

```
int * massiv_ulash(int n, int *a,
int m,int *b) {
    int * x=new int[n+m];
    int ia=0,ib=0,ix=0;
    while (ia<n && ib<m)
        a[ia]>b[ib]?x[ix++]=b[ib++]:x[ix++]=a[ia++];
    while(ib<m)x[ix++]=b[ib++];
    while(ia<n)x[ix++]=a[ia++];
    return x; }
```

Ko'p o'lchamli massivlar bilan ishlash ma'lum bir murakkablikka ega, chunki massivlar xotirada joylash tartibi turli variantda bo'lishi mumkin. Masalan, funksiya parametrlar ro'yxatida $n \times n$ o'lchamdagi haqiqiy turdagi $x[n][n]$ massivga mos keluvchi parametрни

```
float sum(float x[n][n])
```

ko‘rinishda yozib bo‘lmaydi. Muammo echimi - bu massiv o‘lchamini parametr sifatida uzatish va funksiya sarlavhasini quyidagicha yozish kerak:

```
float sum(int n,float x[][]);
```

Ko‘p o‘lchamli massivlarni parametr sifatida ishlatishda bir nechta usullardan foydalanish mumkin.

1-usul. Massivning ikkinchi o‘lchamini o‘zgaras ifoda (son) bilan ko‘rsatish:

```
float sum(int n,float x[][10])
```

```
{float s=0.0;
```

```
for(int i=0;i<n;i++)
```

```
for(int j=0;j<n;j++)
```

```
s+=x[i][j];
```

```
return s;}
```

2-usul. Ikki o‘lchamli massiv ko‘rsatkichlar massivi ko‘rinishida aniqlangan holatlar uchun ko‘rsatkichlar massivini (matritsa satrlar adreslarini) berish orqali:

```
float sum(int n,float *p[]){
```

```
float s=0.0;
```

```
for(int i=0;i<n;i++)
```

```
for(int j=0;j<n;j++)
```

```
s+=p[i][j];\\"p[i][j]" emas,chunki
```

massivga murojat

```
return s; }
```

```
int main() {
```

```
float x[][4]={ {11,-
```

12,13,14},{21,22,23,24},

```
{31,32,33,34},{41,42,43,44}};
```

```
float *ptr[4];
```

```
for(int i=0;i<4;i++) ptr[i]=(float
```

*)&x[i];

```
cout<<sum(4,ptr)<<endl; }
```

3-usul. Ko‘rsatkichlarga ko‘rsatkich

ko‘rinishda aniqlangan dinamik massivlarni ishlatish bilan:

```
float sum(int n,float **x) {
```

Navbatdagi dasturda funksiya tomonidan natija sifatida ikki o‘lchamli massivni qaytarishiga misol keltirilgan. Massiv elementlarning qiymatlari tasodifiy sonlardan tashkil topadi. Tasodifiy sonlar «math.h» kutubxonasidagi random() funksiya yordamida hosil qilinadi:

```
float s=0.0;
```

```
for(int i=0;i<n;i++)for(int
```

```
j=0;j<n;j++)s+=x[i][j];
```

```
return s;
```

```
}
```

```
int main(){
```

```
float **ptr;
```

```
int n;
```

```
cin>>n;
```

```
ptr=new float *[n];
```

```
for(int i=0;i<n;i++){
```

```
ptr[i]=new float [n];
```

```
for(int j=0;j<n;j++)
```

```
ptr[i][j]=(float)((i+1)*10+j);
```

```
}
```

```
cout<<sum(n,ptr);
```

```
for(int i=0; i<n;i++) delete ptr[i];
```

```
delete[]ptr;
```

```
}
```

```

#include <iostream.h>
#include <math.h>
int **rmatr(int n,int m){
    int ** ptr;
    ptr=new int *[n];
    for(int i=0;i<n;i++) {
        ptr[i]=new int[m];
    }
    for(int j=0;j<m;j++) ptr[i][j]=random(100); }
    return ptr;}
int sum(int n,int m,int **ix){
    float s=0;
    for(int i=0;i<n;i++)
        for(int j=0;j<m;j++) s+=ix[i][j];
    return s; }
int main(){
    int n,m;
    cin>>n>>m;
    int **matr;
    randomize();
    matr=rmatr(n,m);
    for(int i=0;i<n;i++) {
        cout<<endl<<i<<" - satr:"
        for (int j=0;j<m;j++) cout<<"\t"<<matr[i][j];
    }
    cout<<endl<<"Summa="<<sum(n,m,matr);
    for(int i=0;i<n;i++) delete matr[i];
    delete[]matr; }

```

Nazorat savollari

1. C++da massiv qanday ishlaydi?
2. Massivga kutubxona kerakmi?
3. Massiv elementlarini bilan massiv indexlarini farqi nimada?
4. for operatori funksiyada qanday ishlatiladi?
5. Matematik funksiyalar qanday ishlaydi?
6. Massiv elementlarini Funksiya parametrlarida uzatish nima uchun ishlatiladi?
7. Massivlarni qanday turlari mavjud?
8. Funksiya parametrlari orqali nima uzatiladi?
9. Massivning necha xil turi bor?

7- MA'RUZA

MAVZU: KO'RSATKICHLAR VA DINAMIK XOTIRA BILAN ISHLASH. DINAMIK MASSIV

Reja:

1. Ko'rsatkichlar haqida;
 - 2
 3. void ko'rsatkich;
 4. Dinamik xotira bilan ishlash;
 5. Ko'rsatkich ustida amallar;
 - 6
 7. Ko'rsatkichlar va adres oluvchi o'zgaruvchilar funktsiya parametri sifatida;
 8. Dinamik massiv va ularni funktsiya parametri sifatida qo'llanilishi.
- ;

Annotatsiya: Ushbu mavzuda ko'rsatkichlar orqali dinamik xotira bilan ishlash, xotiraga qanday murojat qilish, berilgan o'zgaruvchining adresi orqali qanday o'zlashtirish va ular ustida amallar bajarish, ko'rsatkichlarni e'lon qilishda unga boshlang'ich qiymatlar berish vazifalari ko'rib chiqiladi. Ko'rsatkich turlari o'rganiladi bular, birorta ob'yektga, xususan o'zgaruvchiga ko'rsatkich, funktsiyaga ko'rsatkich, void ko'rsatkich. Xotiraning ob'yektlar o'rtasidan dinamik taqsimlanuvchi sohasidan joy ajratish uchun new operatori ,o'zidan keyin ko'rsatkich nomi yoziladigan delete operatori yordamida qanday ishlanishi keltirib o'tilgan.

Kalit so'zlar: Xotira, dinamik, ko'rsatkich, ob'yekt, void, main, delete, new, operator, funktsiya, adres, parameter, matrisa, pointer, massiv.

1. Ko'rsatkich haqida

Ko'rsatkich – bu kompyuter xotirasi yacheykasining adresi yozilgan o'zgaruvchidir. Kompyuter xotirasi nomerlangan yacheykalar ketma-ketligidan iboratdir. Har bir o'zgaruvchining qiymati uning adresi deb ataluvchi alohida xotira yacheykasida saqlanadi.

Dasturdagi o'zgarmaslar, o'zgaruvchilar, funktsiyalar va sinf ob'yektlar adreslarini

Ko'rsatkichni uch xil turda bo'lish mumkin:

- birorta ob'yektga, xususan o'zgaruvchiga ko'rsatkich;
- o'rsatkich;
- void ko'rsatkich.

Ko'rsatkichning bu xususiyatlari uning qabul qilishi mumkin bo'lgan qiymatlarida farqlanadi. Ko'rsatkich albatta birorta turga bog'langan bo'lishi kerak, ya'ni u ko'rsatgan adresda qandaydir qiymat joylanishi mumkin va bu qiymatning xotirada qancha joy egallashi oldindan ma'lum bo'lishi shart.

Ko'rsatkichlarni e'lon qilish:

char *p; //ixtiyoriy simvol yoki satrni adresi

```
int *pI; // Butun sonni adresi
float *pF; // Xaqiqiy sonni adresi
```

Butun o'zgaruvchilar va massivlar:

```
int n = 6, A[5] = {0, 1, 2, 3, 4};
int *p; // Butun songa ko'rsatgich
p = &n; // n manzilini yozish
*p = 20; // n = 20
p = A + 2; // A[2] (&A[2]) adresni yozish
*p = 99; // A[2] o'zgartirish
p++; // A[3] ga o'tish
printf("Adres: %p, qiymat %d", p, *p);
```

2.Ob'yektga ko'rsatkich

Agar bir turda bir nechta ko'rsatkichlar e'lon qilinadigan bo'lsa, har bir ko'rsatkich uchun '*' belgisi qo'yilishi shart:

```
int *i, *j, *k;
float x, *y, *z;
```

K

e

l

3.void ko'rsatkich

Bu ko'rsatkich ob'yekt turi oldindan noma'lum bo'lganda ishlatiladi. void ko'rsatkichining muhim afzalliklaridan biri - unga har qanday turdagi ko'rsatkich qiymatini yuklash mumkinligidir. void ko'rsatkich adresidagi qiymatni ishlatishdan oldin, uni aniq bir turga oshkor ravishda keltirish kerak bo'ladi. void ko'rsatkichni e'lon qilish quyidagicha bo'ladi:

```
void ko'rsatkich
```

```
butun o'zgaruvchi
```

```
// butun o'zgarmas
```

```
// butun o'zgaruvchiga ko'rsatkich
```

```
// butun o'zgarmasga ko'rsatkich
```

```
// butun o'zgaruvchiga o'zgarmas ko'rsatkich
```

```
// butun o'zgarmasga o'zgarmas ko'rsatkich
```

- butun turdagi ko'rsatkichlar va j - butun turdagi o'zgaruvchi, ikkinchi operatorida x - haqiqiy o'zgaruvchi va y, z - haqiqiy turdagi ko'rsatkichlar e'lon qilingan.

4. Dinamik xotira bilan ishlash.

Ko'rsatkichlar ko'pincha **dinamik xotira** (boshqacha nomi «uyum» yoki «heap») bilan bog'liq holda ishlatiladi. Xotiraning dinamik deyilishiga sabab, bu sohadagi bo'sh xotira dastur ishlash jarayonida kerakli paytida ajratib olinadi va zarurat qolmaganida qaytariladi (bo'shatiladi).

Dinamik xotiraga faqat ko'rsatkichlar yordamida murojaat qilish mumkin. Bunday o'zgaruvchilar *dinamik o'zgaruvchilar* deyiladi va ularni yashash vaqti yaratilgan nuqtadan boshlab dastur oxirigacha yoki oshkor ravishda yo'qotilgan (bog'langan xotira bo'shatilgan) joygacha bo'ladi.

Ko'rsatkichlarga dastlabki qiymat kiritish

Ko'rsatkichlarni e'lon qilishda unga boshlang'ich qiymatlar berish mumkin. Boshlang'ich qiymat (initsializator) ko'rsatkich nomidan so'ng yoki qavs ichida yoki '=' belgidan keyin beriladi. Boshlang'ich qiymatlar quyidagi usullar bilan berilishi mumkin:

- *Ko'rsatkichga mavjud bo'lgan ob'yektning adresini berish:*
- *Oshkor ravishda xotiraning absolyut adresini berish:*
- *Bo'sh qiymat berish:*
- *Dinamik xotirada new amali bilan joy ajratish va uni adresini ko'rsatkichga berish:*

Ko'rsatkichning adreslarni saqlash vositasi sifatida qo'llanilishi.

- **Ko'rsatkichga mavjud bo'lgan ob'yektning adresini berish:**

- a) adresni olish amali orqali:**

- int i=5,k=4; // butun o'zgaruvchilar*

- int *p=&i; // p ko'rsatkichga i o'zgaruvchining adresi yoziladi*

- int *p1(&k); // p1 ko'rsatkichga k o'zgaruvchining adresi yoziladi*

- b) boshqa initsializatsiyalangan ko'rsatkich qiymatini berish:**

- int *r=p; // p oldin e'lon qilingan va qiymatga ega bo'lgan ko'rsatkich*

- v) massiv yoki funksiya nomini berish:**

- int b[10]; // massivni e'lon qilish*

- int *t=b; // massivning boshlang'ich adresini berish*

- void f(int a){/* ... */} // funksiyani aniqlash*

- void (*pf)(int); // funksiyaga ko'rsatkichni e'lon qilish*

- pf=f; // funksiya adresini ko'rsatkichga berish*

- **Oshkor ravishda xotiraning absolyut adresini berish:**

- char *vp = (char *)0xB8000000;**

Bunda **0xB8000000** - o'n oltilik o'zgarmas son va (char*) - turga keltirish amali bo'lib, u **vp** o'zgaruvchisini xotiraning absolyut adresidagi baytlarni **char** sifatida qayta ishlovchi ko'rsatkich turiga aylantirilishini anglatadi.

- **Bo'sh qiymat berish:**

- int *suxx=NULL;**

- int *r=0;**

Birinchi satrda maxsus **NULL** o'zgarmasi ishlatilgan, ikkinchi satrda **0** qiymat ishlatilgan. Ikkala holda ham ko'rsatkich hech qanday ob'yektga murojaat qilmaydi. Bo'sh ko'rsatkich asosan ko'rsatkichni aniq bir ob'yektga ko'rsatayotgan yoki yo'qligini aniqlash uchun ishlatiladi.

- new operatori**

Xotiraning ob'yektlar o'rtasidan dinamik taqsimlanuvchi sohasidan joy ajratish uchun new operatori ishlatiladi. new operatoridan keyin xotiraga joylashtiriladigan ob'yekt tipini ko'rsatish lozim. Bu ob'yektni saqlash uchun talab etiladigan xotira sohasi o'lchovini aniqlash uchun kerak bo'ladi. Masalan, new **unsigned short int** deb yozish orqali biz dinamik taqsimlanuvchi xotiradan ikki bayt joy ajratamiz. Xuddi shuningdek, **new long** satri orqali to'rt bayt joy ob'yektlar o'rtasida dinamik taqsimlanuvchi sohadan ajratiladi.

new operatori natija sifatida belgilangan xotira yacheykasining adresini qaytaradi. Bu adress ko'rsatkichga o'zlashtirilishi lozim. Masalan, **unsigned short** tipidagi o'zgaruvchi uchun dinamik sohadan joy ajratish uchun quyidagi dastur kodi yoziladi:

- unsigned short int *pPointer;**

- pPointer = new unsigned short int;**

Yoki xuddi shu amalni bitta satrda ham yozish mumkin.

- unsigned short int * pPoiner = new unsigned short int;**

Ikkala holatda ham pPointer ko'rsatkichi **unsigned short int** tipidagi qiymatni saqlovchi dinamik soha xotirasining yacheykasini ko'rsatib turadi. Endi pPointer ko'rsatkichini shu tipdagi ixtiyoriy o'zgaruvchiga ko'rsatkich sifatida qo'llash mumkin. Ajratilgan xotira sohasiga biror bir qiymat joylashtirish uchun quyidagicha yozuv yoziladi:

- * pPointer = 72 ;**

Bu satr quyidagi ma'noni anglatadi: «pPointer ko'rsatkichida adresi saqlanayotgan xotiraga 72 sonini yozing». Dinamik xotira sohasi albatta chegaralangan bo'ladi. U to'lib qolganda **new** operatori orqali xotiradan joy ajratishga urinsak xatolik yuz beradi.

delete operatori

Agarda o'zgaruvchi uchun ajratilgan xotira kerak bo'lmasa uni bo'shatish zarur. Bu o'zidan keyin ko'rsatkich nomi yoziladigan **delete** operatori yordamida amalga oshiriladi. **delete** operatori ko'rsatkich orqali aniqlangan xotira sohasini bo'shatadi. Shuni esda saqlash lozimki, dinamik xotira sohasidagi adresni o'zida saqlovchi ko'rsatkich lokal o'zgaruvchi bo'lishi mumkin. Shuning uchun bu ko'rsatkich e'lon qilingan funksiyadan chiqishimiz bilan ko'rsatkich ham xotiradan o'chiriladi. Lekin new operatori orqali bu ko'rsatkichga dinamik xotiradan ajratilgan joy bo'shatilmaydi. Natijada xotiraning bu qismi kirishga imkonsiz bo'lib qoladi. Dasturchilar bu holatni xotiraning sirqib ketishi, yoki yo'qolishi (utechka pamyati) deb tavsiflaydilar. Bu tavsif haqiqatga butunlay mos keladi, chunki dastur ishini yakunlaguncha xotirani bu qismidan foydalanib bo'lmaydi.

Xotirani ajratilgan qismini bo'shatish uchun **delete** kalitli so'zidan foydalaniladi. Masalan:

```
delete pPointer;
```

Bunda ko‘rsatkich o‘chirilmaydi, balki unda saqlanayotgan adresdagi xotira sohasi bo‘shatiladi. Belgilangan xotirani bo‘shatilishi ko‘rsatkichga ta’sir qilmaydi, unga boshqa adresni o‘zlashtirish ham mumkin.

Dinamik xotirada new amali bilan joy ajratish va uni adresini ko'rsatkichga berish:

```
int * n=new int;           // birinchi operator
```

```
int * m=new int(10); // ikkinchi operator
```

```
int * q=new int[5];    // uchinchi operator
```

Birinchi operatorida new amali yordamida dinamik xotirada **int** uchun etarli joy ajratib olinib, uning adresi **n** ko'rsatkichga yuklanadi. Ko'rsatkichning o'zi uchun joy kompilyasiya vaqtida ajratiladi.

delete amali

- Ikkinchi operatorida joy ajratishdan tashqari **m** adresiga boshlang'ich qiymat - **10** sonini joylashtiradi.
- Uchinchi operatorida **int** turidagi **5 ta element** uchun joy ajratilgan va uning boshlang'ich adresi **q** ko'rsatkichga berilayapti.
- Xotira **new** amali bilan ajratilgan bo'lsa, u **delete** amali bilan bo'shatilishi kerak. Yuqoridagi **dinamik o'zgaruvchilar** bilan bog'langan xotira quyidagicha bo'shatiladi: **delete n; delete m; delete[q];**
- Agarda xotira **new[]** amali bilan ajratilgan bo'lsa, uni bo'shatish uchun **delete []** amalini o'lchovi ko'rsatilmagan holda qo'llash kerak.
- Xotira bo'shatilganligiga qaramasdan ko'rsatkichni o'zini keyinchalik qayta ishlatish mumkin.

5.Ko‘rsatkich ustida amallar

Ko'rsatkich ustida quyidagi amallar bajarilishi mumkin:

- ✓ ob'jektga vositali murojaat qilish amali;
- ✓ qiymat berish amali;
- ✓ ko'rsatkichga o'zgarmas qiymatni qo'shish amali;
- ✓ ayirish amali;
- ✓ inkrement va dekrement amallari;
- ✓ solishtirish amali;
- ✓ turga keltirish amali.

Vositali murojaat qilish

Vositali murojaat qilish amali ko'rsatkichdagi adres bo'yicha joylashgan qiymatni olish yoki qiymat berish uchun ishlatiladi:

```
char a;           // char turidagi o'zgaruvchi e'loni.
```

```
char *p=new char; // Ko'rsatkichni e'lon qilib, unga
// dinamik xotiradan ajratilgan/ xotiraning adresini berish
*p='b'; // p adresiga qiymat joylashtirish
a=*p; // a o'zgaruvchisiga p adresidagi qiymatni berish
```

Shuni qayd qilib o'tish kerakki, xotiraning aniq bir joyidagi adresni bir paytning o'zida bir nechta va har xil turdagi ko'rsatkichlarga berish mumkin va ular orqali murojaat qilinganda berilganning har xil turdagi qiymatlarini olish mumkin:

```
unsigned long int A=0Xcc77ffaa;
unsigned short int * pint=(unsigned short int*)&A;
unsigned char* pchar=(unsigned char*)&A;
cout<<hex<<A<<' '<<hex<<*pint<<' '<<hex<<(int)*pchar;
```

Ekranga har xil qiymatlar chop etiladi:

```
cc77ffaa ffaa aa
```

O'zgaruvchilar bitta adresda joylashgan holda yaxlit qiymatning turli bo'laklarini o'zlashtiradi. Bunda, bir baytdan katta joy egallagan son qiymatining xotirada «teskari» joylashishi inobatga olinishi kerak. Agar har xil turdagi ko'rsatkichlarga qiymatlar berilsa, albatta turga keltirish amalidan foydalanish kerak:

```
int n=5;
float x=1.0;
int *pi=&n;
float *px=&x;
void *p;
int *r,*r1;
px=(float *)&n;
p=px;
r=(int *)p;
r1=pi;
```

Arifmetik amallar

- Ko'rsatkich turini **void** turiga keltirish amalda ma'noga ega emas. Xuddi shunday, turlari bir xil bo'lgan ko'rsatkichlar uchun turni keltirish amalini bajarishga hojat yo'q.
- Ko'rsatkich ustidan bajariladigan arifmetik amallarda avtomatik ravishda turlarning o'lchami hisobga olinadi.
- Arifmetik amallar faqat bir xil turdagi ko'rsatkichlar ustidan bajariladi va ular asosan, massiv tuzilmalariga ko'rsatkichlar ustida bajariladi.
- Inkrement amali ko'rsatkichni massivning keyingi elementiga, dekrement esa aksincha, bitta oldingi elementining adresiga ko'chiradi. Bunda ko'rsatkichning qiymati sizeof(<massiv elementi-ning turi>) qiymatiga o'zgaradi. Agar ko'rsatkich k o'zgarmas qiymatga oshirilsa yoki kamaytirilsa, uning qiymati k*sizeof(<massiv elementining turi>) kattalikka o'zgaradi.

Masalan:

```
short int *p=new short[5];
long * q=new long [5];
p++; // p qiymati 2 ga oshadi
q++; // q qiymati 4 ga oshadi
q+=3; // q qiymati 3*4=12 oshadi
```

Ko'rsatkichlarning ayirmasi deb, ular ayirmasining tur o'lchamiga bo'linishiga aytiladi. Ko'rsatkichlarni o'zaro qo'shish mumkin emas.

6.Adresni olish amali

Turli kompyuterlarda xotirani adreslash turlicha qoida asosida tashkil etiladi. Ko‘p hollarda dasturchilar uchun biror bir o‘zgaruvchini aniq adresini bilish zarur emas. Zarurat tug‘ilganda bunday axborotni adres operatori (&) yordamida olish mumkin. Dasturning har bir o‘zgaruvchisi o‘zining adresiga egadir. Bu adresni saqlash uchun esa o‘zgaruvchiga ko‘rsatkich e‘lon qilish kerak. Adresning o‘zining qiymatini bilish esa unchalik shart emas.

Adresni olish quyidagicha e‘lon qilinadi:

<tur> & <nom>;

Bu erda **<tur>** - adresi olinadigan qiymatning turi, **<nom>**- adres oluvchi o‘zgaruvchi nomi. O‘rtadagi **‘&’** belgisiga **adresni olish amali** deyiladi.

Bu ko‘rinishda e‘lon qilingan o‘zgaruvchi shu turdagi o‘zgaruvchining sinonimi deb qaraladi. Adresni olish amali orqali bitta o‘zgaruvchiga har xil nom bilan murojaat qilish mumkin bo‘ladi. Misol:

int kol;

int & pal=kol; // *pal murojaati, u kol o‘zgaruvchisining alternativ nomi*

const char & cr=’\n’; // *cr - o‘zgarmasga murojaat*

Adresni olish amalini ishlatishda qoidalarga rioya qilish

- Adresni olish amalini ishlatishda quyidagi qoidalarga rioya qilish kerak: adres oluvchi o‘zgaruvchi funksiya parametri sifatida ishlatilgan yoki **extern** bilan tavsiflangan yoki sinf maydoniga murojaat qilingandan holatlardan tashqari barcha holatlarda boshlang‘ich qiymatga ega bo‘lishi kerak.

- Adresni olish amali asosan funksiyalarda adres orqali uzatiluvchi parametrlar sifatida ishlatiladi.

- Adres oluvchi o‘zgaruvchining ko‘rsatkichdan farqi shundaki, u alohida xotirani egallamaydi va faqat o‘z qiymati bo‘lgan o‘zgaruvchining boshqa nomi sifatida ishlatiladi.

7. Ko‘rsatkichlar va adres oluvchi o‘zgaruvchilar funksiya parametri sifatida

- Funksiya prototipida yoki aniqlanish sarlavhasida ko‘rsatilgan parametrlar **formal parametrlar** deyiladi, funksiya chaqirishida ko‘rsatilgan argumentlarga **faktik parametrlar** deyiladi.

- Funksiya chaqirilishida faktik parametrning turi mos o‘rindagi formal parametr turiga to‘g‘ri kelmasa yoki shu turga keltirishning iloji bo‘lmasa kompilyasiya xatosi ro‘y beradi.

- Faktik parametrlarni funksiyaga ikki xil usul bilan uzatish mumkin: **qiymati** yoki **adres** bilan.

Faktik parametrlarni funksiyaga qiymat bilan uzatish

Funksiya chaqirilishida argument qiymat bilan uzatilganda, argument yoki uning o‘rnidagi kelgan ifoda qiymati va boshqa argumentlarning nusxasi (qiymatlari) stek xotirasiga yoziladi. Funksiya faqat shu nusxalar bilan amal qiladi, kerak bo‘lsa bu nusxalarga o‘zgartirishlar qilinishi mumkin, lekin bu o‘zgarishlar argument-ning o‘ziga ta’sir qilmaydi, chunki funksiya o‘z ishini tugatishi bilan nusxalar o‘chiriladi (stek tozalanadi).

- Agar parametr adres bilan uzatilsa, stekka adres nusxasi yoziladi va xuddi shu adres bo‘yicha qiymatlar o‘qiladi (yoziladi). Funksiya o‘z ishini tugatgandan keyin shu adres bo‘yicha qilingan o‘zgarishlar saqlanib qolinadi va bu qiymatlarni boshqa funksiyalar ishlatishi mumkin.

- Argument qiymat bilan uzatilishi uchun mos formal parametr sifatida o‘zgaruvchini turi va nomi yoziladi. Funksiya chaqirilishida mos argument sifatida o‘zgaruvchining nomi yoki ifoda bo‘lishi mumkin.

- Faktik parametr adres bilan uzatilganda unga mos keluvchi formal parametrni ikki xil usul bilan yozish mumkin: *ko'rsatkich orqali yoki adresni oluvchi parametrlar orqali*.

Misol

```
#include <iostream>
using namespace std;
void f(int, int*, int &);
int main()
{
    int i=1, j=2, k=3; cout<<i<<" "<<j<<" "<<k<<endl;
    f(i, &j, k); cout<<i<<" "<<j<<" "<<k;
}
void f(int i, int *j, int &k)
{
    i++;
    (*j)++;
    k++;
    *j=i+k;
    k=*j+i;
}
```

Misol: $ax^2+bx+c=0$ ko'rinishidagi kvadrat tenglama ildizlarini funksiya parametrlari vositasida olish masalasi

```
#include "iostream"
#include "math.h"
using namespace std;
int kvadrat_ildiz (float a,float b,float c, float & x1, float & x2)
{
    float d; d=b*b-4*a*c;
    if(d<0) return 0;
    if(d==0){ x1=x2=-b/(2*a); return 1;
}
else
{
    x1=(-b+sqrt (d))/(2*a);
    x2=(-b-sqrt(d))/(2*a); return 2;
}
}

int main()
{
    float a,b,c,d,x1,x2;
    cout<<"ax^2+bx+c=0 tenglama ildizini topish. ";
    cout<<"\n a-koeffitsiyentini kiriting="; cin>>a;
    cout<<"\n b-koeffitsiyentini kiriting="; cin>>b;
    cout<<"\n c-koeffitsiyentini kiriting="; cin>>c;
    switch (kvadrat_ildiz(a, b,c, x1, x2))
    {
        case 0: cout<<" Tenglama haqiqiy ildizga ega emas!"; break;
        case 1: cout<<" Tenglama yagona ildizga ega:";
            cout<<"\n x="<<x1;break;
        default: cout<<" Tenglama ikkita ildizga ega : ";
            cout<<"\nx1="<<x1; cout<<"\nx2="<<x2;}
    return 0;
}
```

}

Parametrlar soni noma'lum bo'lgan funksiyalar

Bunday funksiyalar sarlavhasi quyidagi formatda yoziladi:

<funksiya turi> <funksiya nomi> (<oshkor parametrlar ro'yxati>, ...)

Bu yerda **<oshkor parametrlar ro'yxati>** - oshkor ravishda yozilgan parametrlar nomi va turi. Bu parametrlar *majburiy parametrlar* deyiladi. Bunday parametrlardan kamida bittasi bo'lishi shart. Qolgan parametrlar soni va turi noma'lum hisoblanadi. Ularni aniqlash va ishlatish to'la ravishda dastur tuzuvchi zimmasiga yuklanadi.

O'zgaruvchan sondagi parametrlarni tashkil qilish usuli:

1-usul. Parametrlar ro'yxati oxirida yana bir maxsus parametr yoziladi va uning qiymati parametrlar tugaganligini bildiradi. Kompilyator tomonidan funksiya tanasida parametrlar birma-bir aniqlashtiriladi. Barcha parametrlar turi oxirgi maxsus parametr turi bilan ustma-ust tushadi deb hisoblanadi;

2-usul. Birorta maxsus parametr sifatida noma'lum parametrlar soni kiritiladi va unga qarab parametrlar soni aniqlanadi.

Ikkala usulda ham parametrlarga murojaat qilish uchun ko'rsatkichlar ishlatiladi.

Misol:

```
#include "iostream"
using namespace std;
float summa (int k,...)
{
    float p=0;
    int *prt=&k;
    if(*prt==0.0) cout<<" 0 ";
    for(; *prt; prt++) { p+=*prt;
    }
    return p;
}
int main()
{
    cout<<"\n "<<summa(10,20,30,40,0.0);
    cout<<"\n "<<summa(1,2,3,4,0.0);
}
```

Har xil turdagi parametrlarni ishlatish uchun turni aniqlaydigan funksiya

```
#include "iostream"
int Summa(char ,int,...);
using namespace std;
int main()
{
    cout<<"1="<<Summa('i',3,10,20,30)<<endl;
    cout<<"2="<<Summa('f',3,10.0,20.0,5.0)<<endl;
    cout<<"3="<<Summa('d',3,10,20,30)<<endl;
}
int Summa(char z, int k,...){
    switch(z){
        case 'i':{
            int * ptr=&k+1;
            int s=0;
```

```

for(;k--;ptr++) s+=*(ptr);
return (int)s;
}
case 'f':{
float*ptr=(float *)(&k+1); float s=0.0;
for(;k--;ptr++) s+=*(ptr);
return s;
}
default:{
cout<<"\n parametr hato berilgan ";
return 9999999.0;
break;
}
}
}

```

8.Dinamik xotira bilan ishlash.Dinamik massiv va ularni funksiya parametri sifatida qo'llanilishi.

Statistik massivlarning kamchiliklari shundaki, ularning o'lchami oldindan ma'lum bo'lishi kerak, undan tashqari bu o'lcham berilganlarga ajratilgan xotira segmentining o'lchami bilan chegaralangan. Ikkinchi tomondan, yetarlicha katta o'lchamdagi massiv e'lon qilib, konkret masala yechilishida ajratilgan xotira to'liq ishlatilmasligi mumkin. Bu kamchiliklar dinamik massivlardan foydalanish orqali bartaraf etiladi, chunki ular dastur ishlashi jarayonida zarur bo'lganda kerak o'lchamdagi massivlarni yaratish va zarurat qolmaganda yo'qotish imkoniyatini beradi.

Funksiyaga ko'rsatkich dastur joylashgan xotiradagi funksiya kodining boshlang'ich adresini ko'rsatadi, ya'ni funksiya chaqirilganda boshqaruv ayni shu adresga uzatiladi. Ko'rsatkich orqali funksiyaning oddiy yoki vositali chaqirish amalga oshirish mumkin. Bunda funksiya uning nomi bo'yicha emas, balki funksiya ko'rsatuvchi o'zgaruvchi orqali chaqiriladi. Funksiyaning boshqa funksiya argument sifatida uzatish ham funksiya ko'rsatkichi orqali bajariladi. Funksiyaga ko'rsatkichning yozilish sintaksisi quyidagicha:

<tur> (* <nom>) (<parametrlar ro'yxati>);

Bunda **<tur>**- funksiya qaytaruvchi qiymat turi; ***<nom>** - ko'rsatkich o'zgaruvchining nomi; **<parametrlar ro'yxati>** - funksiya parametrlarining yoki ularning turlarining ro'yxati.

Masalan:

```
int (*fun)(float,float);
```

Funksiya va massivlar

Ko'p o'lchamli massivlarni parametr sifatida ishlatishda bir nechta usullardan foydalanish mumkin:

1-usul. Massivning ikkinchi o'lchamini o'zgarmas ifoda (son) bilan ko'rsatish:

```

float sum (int n, float x[][10])
{
float s=0.0;
for(int i=0;i<n;i++)
for(int j=0;j<n;j++)
s+=x[i][j];
return s;
}

```

2-usul. Ikki o'lchamli massiv ko'rsatkichlar massivi ko'rinishida aniqlangan holatlar uchun ko'rsatkichlar massivini (matritsa satrlar adreslarini) berish orqali:

```
float sum (int n, float x[][10])
{
    float s=0.0;
    for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
    s+=x[i][j];
    return s;
}
int main()
{
    float x[][4]={{11,-12,13,14},{21,22,23,24},{31,32,33,34},{41,42,43,44}};
    float *ptr[4];
    for(int i=0;i<4;i++)
    ptr[i]=(float*)&x[i];
    cout<<sum(4,ptr)<<endl;
}
```

3-usul. Ko'rsatkichlarga ko'rsatkich ko'rinishida aniqlangan dinamik massivlarni ishlatish bilan:

```
float sum(int n, float **x)
{
    float s=0.0;
    for(int i=0;i<n;i++)
    for(int j=0;j<n;j++)
    s+=x[i][j];
    return s;
}
int main()
{
    float **ptr;
    int n;
    cin>>n;
    ptr=new float * [n];
    for(int i=0;i<n;i++)
    {
        ptr[i]=new float [n];
        for(int j=0;j<n;j++)
        ptr[i][j]=(float)((i+1)*10+j);
    }
    cout<<sum(n,ptr);
    for(int i=0;i<n;i++)
    delete ptr[i];
    delete []ptr;
}
```

Dinamik massivlar

Masala: Massivni o'lchamini kiriting va unga dastur davomida xotiradan joy ajrating.

Muammo : Massiv o'lchami oldindan ma'lum emas.

Yechish usullari:

✓ Har bir satr uchun alohida xotira blokini ajratish;

✓ Butun Massiv uchun bir yo‘la xotira ajratish.

Xar bir satr uchun xotira bloki

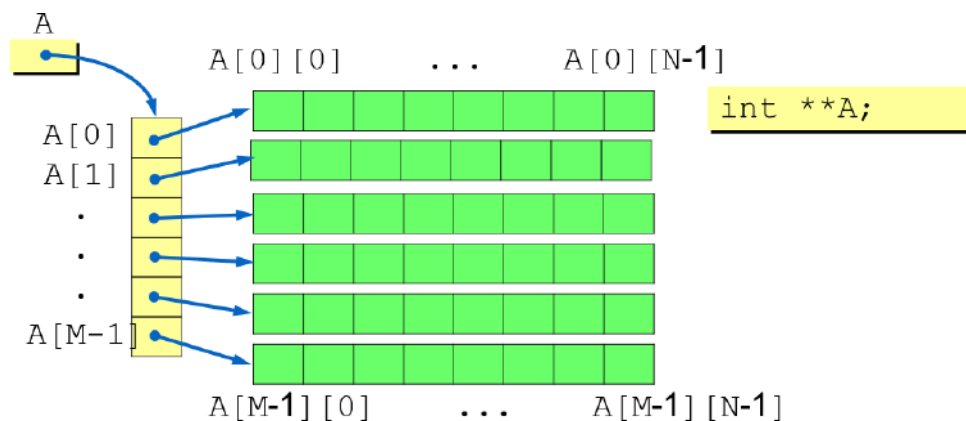
Massiv adresi:

✓ Massiv = satr massivi;

✓ Massiv adresi = massiv adresi, bunda satrlar adresi saqlanadi;

✓ Satr adresi = ko‘rsatgich;

✓ Massiv = ko‘rsatgichlar massivi adresi.



```
typedef int *pInt;
```

```
int main()
```

```
{
```

```
int M, N, i;
```

```
pInt *A;
```

```
A = new pInt[M]; // Ko‘rsatgichlar massivini ajratish
```

```
for ( i = 0; i < M; i ++ )
```

```
A[i] = new int[N]; // Massivning har- bir satri uchun
```

```
for ( i = 0; i < M; i ++ )
```

```
delete A[i]; // Har – bir satr o‘chiriladi
```

```
delete A; // Massiv o‘chiriladi
```

```
}
```

Xulosa

Ko‘rsatgich bu – qiymat sifatida o‘zi joylashgan operativ xotira adresini qaytaradigan identifikatordir. Dinamik massiv – ko‘rsatgich yordamida operativ hotira yacheykalarni ishlatishni boshqarish imkoniyatini beradi.

Nazorat savollari

1.Ko‘rsatkich nima?

2. Ko‘rsatkichlar qanday e‘lon qilinadi va qo‘llaniladi ?

3. Xotira bilan qanday ishlash mumkin?

4. Ob’ektda ko‘rsatkich nima vazifani bajaradi?

5.void ko‘rsatkich nima vazifani bajaradi?

6

7. Dinamik xotira bilan qanday ishlanadi ?

qanaday. Dinamik massiv va ularni funksiya parametri sifatida qo‘llanilishi ...?

9. Ko‘rsatkichlar qanday turlarga bo‘linadi?

10. Dinamik xotiraga qanday murojaat qilish mumkin?
11. Xotiraning dinamik deyilishiga sabab nima?
12. Xotiraning ob'ektlar o'rtasidan dinamik taqsimlanuvchi sohasidan joy ajratish uchun qanday operator ishlatiladi?
13. new unsigned short int deb yozish orqali biz dinamik taqsimlanuvchi xotiradan qancha joy ajratamiz?
14. new operatori vazifasi?
15. Dinamik xotirada new amali bilan joy ajratish va uni adresini ko'rsatkichga berish qanday amalga oshiriladi?
16. delete operatori vazifasi?
17. Xotirani ajratilgan qismini bo'shatish uchun qaysi operatoridan foydalanish mumkin?
18. Dinamik massiv tushunchasi?
19. Ko'rsatkichlarni o'zlashtirish?
20. Funksiyaga ko'rsatkichning yozilish sintaksisi?
21. Adresni olish amalini ishlatish qoidalari?
22. Ko'rsatkichdagi adres bo'yicha joylashgan qiymatni olish yoki qiymat berish uchun qanday amal ishlatiladi?
23. C++ dasturlash tilida cout<<&b; qatorining vazifasi nima?
24. Kompyuter xotirasi yacheykasining adresi yozilgan o'zgaruvchi bu...?
25. Funksiya prototipida yoki aniqlanish sarlavhasida ko'rsatilgan parametrlar qanday parametrlar ataladi?
26. Funksiya chaqirishida ko'rsatilgan argumentlar qanday parametrlar deb ataladi?
27. Faktik parametrlarni funksiyaga ikki xil usul bilan uzatish mumkin bular...?
28. Adresni olish qanday amalga oshiriladi?
29. C++ da ko'rsatkichlarni o'zaro qo'shish imkoniyati...?
30. Ko'rsatkichlarning ayirmasi deb?

8- MA'RUZA

MAVZU: OBYEKTGA YO'NALTIRILGAN DASTURLASH TUSHUNCHASI

Reja:

1. Inkapsulyasiya, polimorfizm, vorislik
2. Obektga yo'naltirilgan dasturlash tushunchasi
3. Obyektlar trassirovkasi
4. Nusxalash konstruktori

Annotatsiya: Ushbu ma'ruzada obyektga yo'naltirilgan dasturlash tamoyillari (inkapsulyatsiya, polimorfizm, vorislik) haqida ma'lumotlar keltirilgan.

Kalit so'zlar: *inkapsulyatsiya, sinf, OYD, konstruktor va destruktors, ro'yxat, manzil, tugun, adres olish &, bo'shatish, ko'rsatkich, inkapsulyatsiya, sinf, OYD, konstruktor va destruktors.*

1. Inkapsulyasiya, polimorfizm, vorislik

Obyektga yo'naltirilgan dasturlash (OYD) – bu dasturlashga yangi bir yondashuvdir. Hisoblash texnikasining rivojlanishi va yechilayotgan masalalarni tobora murakkablashuvi dasturlashning turli modellarini (paradigmalarini) yuzaga kelishiga sabab bo'lmoqda. Birinchi kompilyatorlarda (masalan, FORTRAN tili) dasturlashning funksiyalardan foydalanishga

asoslangan prosedura modelini qo'llab quvvatlagan. Bu model yordamida dastur tuzuvchi bir nechta ming satrli dasturlarni yozishi mumkin edi. Rivojlanishning keyingi bosqichida dasturlarning strukturali modeli paydo bo'ldi va ALGOL, Pascal va C tillari kompilyatorlarida o'z aksini topdi. Strukturali dasturlashning mohiyati – dasturni o'zaro bog'langan proseduralar (blokklar) va ular qayta ishlaydigan berilganlarning majmuasi deb qarashdan iborat. Ushbu model dastur blokklari keng qo'llashga, GOTO operatoridan imkon qadar kam foydalanishga tayangan va unda dastur tuzuvchi o'n ming satrdan ortiq dasturlarni yarata olgan. Yaratilgan dasturni prosedurali modelga nisbatan sozlash va nazorat qilish oson kechgan.

Murakkab masalalarni yechish uchun dasturlashning yangi uslubiga zarurat paydo bo'ldiki, u OYD modelida amalga oshirildi. OYD modeli bir nechta tayanch konsepsiyalarga asoslanadi.

Berilganlarni abstraksiyalash – berilganlarni yangi turini yaratish imkoniyati bo'lib, bu turlar bilan xuddi berilganlarning tayanch turlari bilan ishlagandek ishlash mumkin. Odatda yangi turlarni berilganlarning abstrakt turi deyiladi, garchi ularni soddaroq qilib “foydalanuvchi tomonidan aniqlangan tur” deb atash mumkin.

Inkapsulyasiya – bu berilganlar va ularni qayta ishlovchi kodni birlashtirish mexanizmidir. Inkapsulyasiya berilganlar va kodni tashqi ta'sirdan saqlash imkonini beradi.

2. Obyektga yo'naltirilgan dasturlash tushunchasi

Yuqoridagi ikkita konsepsiyani amalga oshirish uchun C++ tilida *sinf*lar ishlatiladi. *Sinf* termini bilan Obyektlar turi aniqlanadi. Sinfning har bir vakili (nusxasi) *Obyekt* deb nomlanadi. Har bir Obyekt o'zining alohida holatiga ega bo'ladi. Obyekt holati uning *berilganlar-a'zolarining* ayni paytdagi qiymati bilan aniqlanadi. Sinf vazifasi uning *funksiya-a'zolarining* *sinf* Obyektlari ustida bajaradigan amallar imkoniyati bilan aniqlanadi.

Berilgan *sinf* Obyektini yaratish *konstruktor* deb nomlanuvchi maxsus funksiya-a'zo tomonidan, o'chirish esa *destruktor* deb nomlanuvchi maxsus funksiya-a'zo orqali amalga oshiriladi.

Sinf ichki berilganlarini murojaatni cheklab qo'yishi mumkin. Cheklov berilganlarni ochiq (*public*), yopiq (*private*) va himoyalangan (*protected*) deb aniqlash bilan tayinlanadi.

Sinf, shu turdagi Obyektning tashqi dunyo bilan o'zaro bog'lanishi uchun qat'iy muloqot shartlarini aniqlaydi. Yopiq berilganlarga yoki kodga faqat shu Obyekt ichida murojaat qilish mumkin. Boshqa tomondan, ochiq berilganlarga va kodlarga, garchi ular Obyekt ichida aniqlangan bo'lsa ham, dasturning ixtiyoriy joyidan murojaat qilish mumkin va ular Obyektni tashqi olam bilan muloqotni yaratishga xizmat qiladi. Yaratilgan Obyektlarni, ularni funksiya-a'zolariga oddiygina murojaat orqali amalga oshiriluvchi *xabarlar* (yoki *so'rovlar*) yordamida boshqarish mumkin. Keyinchalik Windows xabarlari bilan adashtirmaslik uchun *so'rov* termini ishlatiladi.

Vorislik – bu shunday jarayonki, unda bir Obyekt boshqasining xossalarini o'zlashtirishi mumkin bo'ladi. Vorislik orqali mavjud sinflar asosida hosilaviy sinflarni qurish mumkin bo'ladi. Hosilaviy *sinf* (*sinf-avlod*) o'zining ona sinfidan (*sinf-ajdod*) berilganlar va funksiyalarni vorislik bo'yicha oladi, hamda ular satriga faqat o'ziga xos bo'lgan qirralarni amalga oshirishga imkon beruvchi berilgan va funksiyalarni qo'shadi. Ajdod sinfdagi himoyalangan berilgan-a'zolariga va funksiya-a'zolariga ajdod sinfdan murojaat qilish mumkin bo'ladi. Bundan tashqari, hosilaviy sinfdan ona *sinf* funksiyalari qayta aniqlanishi mumkin. Demak, vorislik asosida bir-biri bilan “ona-bola” munosabatidagi sinflar shajarasini yaratish mumkin. *Tayanch sinf* termini sinflar shajarasidagi ona *sinf* sinonimi sifatida ishlatiladi. Agar Obyekt o'z atributlarini (berilganlar-a'zolar va funksiyalar-a'zolar) faqat bitta ona sinfdan vorislik bilan olsa, *yakka* (yoki *oddiy*) *vorislik* deyiladi. Agar Obyekt o'z atributlarini bir nechta ona sinflardan olsa, *to'plamli vorislik* deyiladi.

Polimorfizm – bu kodning, bajarilish paytidan yuzaga keladigan holatga bog'liq ravishda o'zini turlicha amal qilish xususiyatidir. Polimorfizm – bu faqat Obyektlar xususiyati bo'lmasdan, balki funksiyalar-a'zolar xususiyatidir va ular xususan, bitta nomdagi funksiya-a'zoni, har xil turdagi argumentlarga ega va bajaradigan amali unga uzatiladigan argumentlar turiga bog'liq bo'lgan funksiyalar uchun (o'rnida) foydalanish imkoniyatida namoyon bo'ladi. Bu holatga

funksiyalarni qayta yuklash deyiladi. Polimorfizm amallarga ham qo'llanishi mumkin, ya'ni amal mazmuni (natijasi) operand (berilgan) turiga bog'liq bo'ladi. Polimorfizmning bunday turiga *amallarni qayta yuklash* deyiladi.

Polimorfizmning yana bir ta'rifi quyidagicha: polimorfizm – bu tayanch sinfga ko'rsatgichlarning (murojaatlarning), ularni virtual funksiyalarni chaqirishdagi turli shakl (qiymatlarni) qabul qilish imkoniyatidir. C++ tilining bunday imkoniyati *kechiktirilgan bog'lanish* natijasidir. Kechiktirilgan bog'lanishda chaqiriladigan funksiya-a'zolar adreslari dastur bajarilishi jarayonida dinamik ravishda aniqlanadi. An'anaviy dasturlash tillarida esa bu adreslar statik bo'lib, ular kompilyasiya paytida aniqlanadi (*oldindan bog'lanish*). Kechiktirilgan bog'lanish faqat virtual funksiyalar uchun o'rinli.

Dasturda ishlatiladigan har bir o'zgaruvchi o'z toifasiga ega va u quyidagilarni aniqlaydi:

1. Xotiradagi o'lchovini;
2. Unda saqlanayotgan ma'lumotlarni;
3. Uning yordamida bajarilishi mumkin bulgan amallarni.

C++ tilida dasturchi o'ziga kerakli ixtiyoriy toifani hosil qilishi mumkin. Bu yangi toifa ichki toifalarning xossalari va ularning funksional imkoniyatlarini o'zida ifodalaydi. Yangi toifa sinfni e'lon qilish orqali tuziladi. Sinf bu – bir-biri bilan funksional bog'angan o'zgaruvchilar va usullar (funksiyalar) to'plamidir.

Masalan: Mushuk nomli sinf tuzmoqchimiz. Bu yerda uning yoshi, og'irligi kabi o'zgaruvchilar va miyovlash, sichqon tutish kabi funksiyalardan ishdatiladi. Yoki Mashina sinfi g'ildirak, eshik, o'rindiqliq, oyna kabi o'zgaruvchilar va xaydash, to'xtatish kabi funksiyalardan iborat.

Sinfidagi o'zgaruvchilar – sinf a'zolari yoki sinf xossalari deyiladi.

Sinfidagi funksiyalar odatda o'zgaruvchilar ustida biror bir amal bajaradi. Ularni sinf usullari (metodlari) deb ham ataladi.

Sinfni e'lon qilish uchun class so'zi , { } ichida esa shu sinfning a'zolari va usullari keltiriladi. Masalan:

```
class non  
{ int ogirlik ;  
int baho ;  
void yasash ( );  
void yopish ( );  
void eyish ( );  
}
```

Sinfni e'lon qilishda xotira ajratilmaydi. Sinf e'lon qilinganda kompilyator faqat shunday (non) sinf borligini, hamda unda qanday qiymatlar (ogirlik, baho) saqlanishi mumkinligini, ular yordamida qanday amallarni (yasash, yopish, yeyish) bajarish mumkinligi haqida xabar beradi. Bu sinf Obyekti hammasi bo'lib 4 bayt joy egallaydi (2 ta int).

Obyekt sinfning biror bir nusxasi hisoblanadi.

C++ tilida toifalarga qiymat o'zlashtirilmaydi, balki o'zgaruvchiga o'zlashtiriladi. Shuning uchun to'g'ridan-to'g'ri `int = 55` deb yozib bo'lmaganidek `non.baho=1200` deb ham bo'lmaydi. O'zlashtirishda xatolikka yo'l qo'ymaslik uchun oldin non sinfiga tegishli patir Obyektini hosil qilamiz keyin esa unga kerakli qiymatlarni beramiz.

Masalan:

```
int a; // butun toifali a o'zgaruvchisi, Obyekti  
non patir; //
```

Endi non sinfining real Obyekti aniqlanganidan so'ng uning a'zolariga murojaat

```
patir.baho = 1200;  
patir.ogirlik = 500;  
patir.yasash ( );
```

Sinfni e'lon qilishda quyidagilardan foydalaniladi:

public - ochiq

private – yopiq

Sinfning barcha usul va a'zolari boshlang'ich holda avtomatik ravishda yopiq bo'ladi. Yopiq a'zolariga esa faqat shu sinfning usullari orqaligina murojaat qilish mumkin. Obyektning ochiq a'zolariga esa dasturdagi barcha funksiyalar murojaat qilishi mumkin. Lekin sinf a'zolariga murojaat qilish ancha mushkul ish hisoblanadi. Agar to'g'ridan to'g'ri:

non patir;

patir.baho = 1200;

patir.og'irlik = 500; deb yozsak xato bo'ladi.

A'zolariga murojaat qilishdan oldin uni ochiq deb e'lon qilish kerak:

```
# include < iostream.h >
```

```
class non
```

```
{ public :
```

```
int baho;
```

```
int ogirlik;
```

```
void yasash ( ); };
```

```
int main ( ){
```

```
non patir;
```

```
patir.baho = 1200; patir.ogirlik = 500;
```

```
cout <<"men olgan patir" <<patir.baho <<"so'm"<<endl;
```

```
cout <<"uning og'irligi ="<<patir.og'irlik <<endl ; }
```

3. Obyektlar trassirovkasi

Foydalanuvchi ma'lumotlarni kiritmaguncha menyu kutib turadi. Agarda foydalanuvchi to'g'ri qiymatni kiritmasa, menyu yangilanadi, foydalanuvchi ma'lumotlarni boshqatdan kiritishi mumkin.

1 Qadam. O'z obyektlaringizning majburiyatini belgilovchi ro'yxat tuzing.

Faqat sizning topshirig'ingizni yechish uchun zarur bo'lgan funksiyalarni amalga oshiring. Real narsalar, masalan kassa aparati yoki bank hisob-raqami funksiyasini amalga oshirish uchun o'n ikkilik funksiyasidan foydalaniladi. Biroq, sizning vazifaningiz real dunyoning modelini yaratishdan iborat emas. Sizning topshirig'ingizni yechish uchun zarur bo'ladigan vazifalarni aniqlashtirib olishingiz lozim.

Display the menu.(Menyu kiritish)

Get user input. (Foydalanuvchidan kirish ma'lumotlarini olish)

Tavsif muammosi qismiga kirmaydigan yashirin majburiyatlarga qarang. Obyekt qanday yaratiladi? Qanday oddiy faoliyatlar ro'y berishi kerak, har bir savdoni boshlanishida kassa aparatini tozalashga o'xshash? Menyuni tuzishni menyu yaratish misolida ko'rib chiqing. Programmist menyuning bo'sh obyektini yaratadi va undan so'ng **"Yangi akkaunt ochish"**, **"Yordam"** opsiyasini qo'shadi. Bu yerda yashirin majburiyat bor:

```
Menu main_menu;
```

```
main_menu.add_option("Open new account");
```

```
// Add more options
```

```
int input = main_menu.get_input();
```

```
Endi biz o'ziga xos metodlar ro'yxatiga egamiz • void add_option(string option)
```

```
int get_input() const;
```

Menyu chiqarish masalasichi? Menyuni foydalanuvchidan ma'lumot kiritishni so'ramasdan ko'rsatishning ma'nosi yo'q. Agar foydalanuvchi xato ma'lumot kiritrsa **get_input** menyuni bir

martadan ortiq kiritadi. Shunday qilib **display** xususiy metod uchun yaxshi kandidatdir. Ijtimoiy interfeysni yakunlash uchun siz konstruktorlarni aniqlashingiz kerak. Ozingizdan so‘rang obyekt yaratish uchun sizga nima kerak. Bazan siz 2 ta konstruktorga extiyoj sezasiz: biri hamma elementlar uchun **default** ikkinchisi esa foydalanuvchi tomonidan kiritilgan qiymatlarni o‘rnatadi. Menyu misolida biz bo‘sh menyu yaratuvchi yagona konstruktor bilan kifoyalanamiz.

3. Nusxalash konstruktori

Vektorlar obyektida paralel vektorlarni yaranting. Ba’zida, bir xil uzunlikdagi vektorlarni ishlatyotganingizni anglaysiz, har bir saqlaydigan qismi obyekt hisoblanadi. Bu vaziyatda, dasturingizni qayta yaratish va elementlari obyekt sanaladigan yagona vektordan foydalanish yaxshi fikrdir.

Masalan, faraz qilaylik hisob raqami bir qator tavsif va narxlardan iboratdir. Yagona yechim ikkita vektorni saqlab turishdir:

```
vector<string> descriptions;
```

```
vector<double> prices;
```

Vektorlarning har biri bir xil uzunlik va bo‘lakka ega bo‘lib, *consisting of descriptions[i] tavsifi va and [i]* narxlaridan iborat, birga ishlanadigan ma’lumotlar ham kiradi. Bu vektorlar paralel vektrlar deb atiladi.

Parallel vektorlar katta dasturlarda muammoni keltirib chiqaradi. Dasturchi ishonch hosil qilishi kerak, vektorlar doim o‘sha uzunlikka ega, qaysiki bir-biriga tegishli kesimlar qiymatlar bilan to‘ldirilgan bo‘ladi. Bundan tashqari kesimda ishlaydigan ixtiyoriy funksiya barcha vektorlarni argument sifatida qabul qilishi kerak, ya’ni dastur uchun zerikarli bo‘lganlarini. Buning yo‘li oson. Kesimga qarang va vakil bo‘layotgan tushunchani toping, so‘ng bu tushunchani class ni ichiga kiriting. Bu kesimda har bir sinfnig tavsifi va narx bandidan iborat; buni sinfga kiriting.

```
class Item {  
public:  
...  
private:  
string description;  
double price;  
};
```

Fayldagi manba’

- Komponentlik funksiyalari tavsifi
- Komponentsiz funksiyalar tavsifi.

CashRegister sinfi uchun, siz bir juft faylni yaratasiz: **cashregister.h** va **cashregister.cpp** unda interfeys va qo‘llanilish mavjud.

Nazorat savollari

1. Obyektga yo‘naltirilgan dasturlash tamoyillari.
2. Sinflar va Obyektlar.
3. Joylashtiriladigan (inline) funksiyalar–a’zolar.
4. Inkapsulyasiya tushunchasi.
5. Vorislik tushunchasi.
6. Polimorfizm tushunchasi
7. Inkapsulyasiya nima?
8. Polimorfizm haqida tushuncha.
9. Vorislikning qo‘llanishi.
10. Sinfdagi o‘zgaruvchilar – sinf a’zolarining qo‘llanishi.

10- MA'RUZA

MAVZU: SATRLAR VA KENGAYTIRILGAN BELGILAR

Reja:

1. Satrlar bilan ishlash;
2. String turi bilan ishlash;
3. String Obyekt sifatida.

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().*

C++ tilida standart satr turiga qo'shimcha sifatida string turi kiritilgan va u string sinfi ko'rinishida amalga oshirilgan. Bu turdagi satr uchun '\0' belgisi tugash belgisi hisoblanmaydi va u oddiygina belgilar massivi sifatida qaraladi. string turida satrlar uzunligining bajariladigan amallar natijasida dinamik ravishda o'zgarib turishi, uning tarkibida bir qator funksiyalar aniqlanganligi bu - tur bilan ishlashda ma'lum bir qulayliklar yaratadi.

string turidagi o'zgaruvchilar quyidagicha e'lon qilinishi mumkin:

```
string s1,s2,s3;
```

Bu turdagi satrlar uchun maxsus amallar va funksiyalar aniqlangan. string satrga boshlang'ich qiymatlar har xil usullar orqali berish mumkin:

```
string s1="birinchi usul";
```

```
string s2("ikkinchi usul");
```

```
string s3(s2);
```

```
string s4=s2;
```

Xuddi shunday, string turidagi o'zgaruvchilar ustida qiymat berish amallari ham har xil:

```
string s1,s2,s3; char *str="misol";
```

```
//satrli o'zgarmas qiymati berish
```

```
s1="Qiymat berish 1-usul";
```

```
s2=str; // char turidagi satr yuklanmoqda
```

```
s3='A'; // bitta belgi qiymat sifatida berish
```

```
s3=s3+s1+s2+"0123abc"; //qiymat sifatida satr ifoda
```

8.2-jadvalida string turidagi satrlar ustidan amallar keltirilgan.

Satr elementiga indeks vositasidan tashqari at() funksiyasi orqali murojaat qilish mumkin:

```
string s1="satr misoli";
```

```
cout<<s.at(3) // natijada 'r' belgisi ekranga chiqadi
```

Shuni aytib o'tish kerakki, string sinfda shu turdagi o'zgaruvchilar bilan ishlaydigan funksiyalar aniqlangan. Boshqacha aytganda, string turida e'lon qilingan o'zgaruvchilar (obyektlar) o'z funksiyalariga ega hisoblanadi va ularni chaqirish uchun oldin o'zgaruvchi nomi, keyin '.' (nuqta) va zarur funksiya nomi (argumentlari bilan) yoziladi.

8.2-jadval. string turidagi satrlar ustidan amallar

Amal	Mazmuni	Misol
=, +=	Qiymat berish amali	s="satr01234" s+="2satr000"

+	Satrlar ulash amali (konkantenatsiya)	s1+s2
==, !=, <, <=, >, >=	Satrlarni solishtirish amallari	s1==s2 s1>s2 && s1!=s2
[]	Indeks berish	s[4]
<<	Oqimga chiqarish	sout << s
>>	Oqimdan o'qish	sin >> s (probelgacha)

Satr qismini boshqa satrga nusxalash funksiyasi

Bir satr qismini boshqa satrga yuklash uchun quyidagi funktsiya-larni ishlatish mumkin, ularni prototipi quyidagicha:

```
assign(const string &str);  
assign(const string &str,unsigned int pos,  
    unsigned int n);  
assign(const char *str, int n);
```

Birinchi funktsiya qiymat berish amal bilan ekvivalentdir: string turidagi str satr o'zgaruvchi yoki satr o'zgarmasni amalni chaqiruvchi satrga beradi:

```
string s1,s2;  
s1="birinchi satr";  
s2.assign(s1); // s2=s1 amalga ekvivalent
```

Ikkinchi funktsiya chaqiruvchi satrga argumentdagi str satrning pos o'rnidan n ta belgidan iborat bo'lgan satr qismini nusxalaydi. Agarda pos qiymati str satr uzunligidan katta bo'lsa, xatolik haqida ogohlantiriladi, agar pos + n ifoda qiymati str satr uzunligidan katta bo'lsa, str satrning pos o'rnidan boshlab satr oxirigacha bo'lgan belgilar nusxalanadi. Bu qoida barcha funktsiyalar uchun tegishlidir.

Misol:

```
string s1,s2,s3;  
s1="0123456789";  
s2.assign(s1,4,5);           // s2="45678"  
s3.assign(s1,2,20);          // s3="23456789"
```

Uchinchi funktsiya argumentdagi char turidagi str satrni string turiga aylantirib, funktsiyani chaqiruvchi satrga o'zlashtiradi:

```
char * stroid;  
cin.getline(stroid,100);// "0123456789" kiritiladi  
string s1,s2;  
s2.assign(stroid,6);           // s2="012345"  
s3.assign(stroid,20);          // s3="0123456789"
```

Satr qismini boshqa satrga qo'shish funktsiyasi

Satr qismini boshqa satrga qo'shish funktsiyalari quyidagicha:

```
append(const string &str);  
append(const string & str,unsigned int pos,  
    unsigned int n);  
append(const char *str, int n);
```

Bu funktsiyalarni yuqorida keltirilgan mos assign funktsiya-lardan farqi - funktsiyani chaqiruvchi satr oxiriga str satrni o'zini yoki uning qismini qo'shadi.


```

char * sc;
cin.getline(sc,100);          //"0123456789" kiritiladi
string s1,s2;
s2=sc; s1="misol";
s="aaa";                      //s2="0123456789"
s2.append("abcdef"); //s2+="abcdef" amali
//va s2="0123456789abcdef"
s1.append(s2,4,5); //s1="misol45678"
s.append(ss,5);              // s="aaa012345"

```

Satr qismini boshqa satr ichiga joylashtirish funksiyasi

Bir satrga ikkinchi satr qismini joylashtirish uchun quyidagi funksiyalar ishlatiladi:

```

insert(unsigned int pos1,const string &str);
insert(unsigned int pos1,const string & str,
        unsigned int pos2,unsigned int n);
insert(unsigned int pos1,const char *str, int n);

```

Bu funksiyalar append kabi ishlaydi, farqi shundaki, str satrini yoki uning qismini funksiyani chaqiruvchi satrning ko'rsatilgan pos1 o'rnidan boshlab joylashtiradi. Bunda amal chaqiruvchi satrning pos1 o'ndan keyin joylashgan belgilar o'nga suriladi.

Misol:

```

char * sc;
cin.getline (sc,100); //"0123456789" satri kiritiladi
unsigned int i=3;
string s1,s2;
s2=sc; s1="misollar"; s="xyz"; // s2="0123456789"
s2.insert(i,"abcdef"); // s2="012abcdef3456789"
s1.insert(i-1,s2,4,5); // s1="mi45678sollar"
s.insert(i-2,sc,5); // s="x01234yz"

```

Satr qismini o'chirish funksiyasi

Satr qismini o'chirish uchun quyidagi funksiyani ishlatish mumkin:

```

erase(unsigned int pos=0,unsigned int n=npos);

```

Bu funksiya, uni chaqiruvchi satrning pos o'rnidan boshlab n ta belgini o'chiradi. Agarda pos ko'rsatilmasa, satr boshidan boshlab o'chiriladi. Agar n ko'rsatilmasa, satrni oxirigacha bo'lgan belgilar o'chiriladi:

```

string s1,s2,s3;
s1="0123456789";
s2=s1;s3=s1;
s1.erase(4,5); // s1="01239"
s2.erase(3);    // s2="012"
s3.erase();     // s3=""

```

void clear() funksiyasi, uni chaqiruvchi satrni to'liq tozalaydi.

Masalan:

```

s1.clear(); //satr bo'sh hisoblanadi (s1="")

```

Satr qismini almashtirish funksiyasi

Bir satr qismining o'rniga boshqa satr qismini qo'yish uchun quyidagi funksiyalardan foydalanish mumkin:

```

replace(unsigned int pos1,unsigned int n1,

```

```

const string & str);
replace(unsigned int pos1,unsigned int n1,
const string & str,unsigned int pos2,
unsigned int n2);
replace(unsigned int pos1,unsigned int n1,
const char *str, int n);

```

Bu funksiyalar insert kabi ishlaydi, undan farqli ravishda amal chaqiruvchi satrning ko'rsatilgan o'rnidan (pos1) n1 belgilar o'rniga str satrini yoki uning pos2 o'rindan boshlangan n2 belgidan iborat qismini qo'yadi (almashtiradi).

Misol:

```

char * sc="0123456789";
unsigned int i=3,j=2;
string s1,s2;
s2=sc; s1="misollar"; s="xyz"; // s2="0123456789"
s2.replace(i,j,"abcdef"); // s2="012abcdef56789"
s1.replace(i-1,j+1,s2,4,5); // s1="mi45678lar"
s.replace(i-2,j+2,sc,5); // s="x012345"
swap(string & str) funksiyasi ikkita satrlarni o'zaro almashtirish uchun ishlatiladi. Masalan:
string s1,s2;
s1="01234";
s2="98765432";
s1.swap(s2); // s2="01234" va s1="98765432" bo'ladi.

```

Satr qismini ajratib olish funksiyasi

Funksiya prototipi quyidagicha:

```

string substr(unsigned int pos=0,
unsigned int n=npos)const;

```

Bu funksiya, uni chaqiruvchi satrning pos o'rnidan boshlab n belgini natija sifatida qaytaradi. Agarda pos ko'rsatilmasa, satr boshidan boshlab ajratib olinadi, agar n ko'rsatilmasa, satr oxirigacha bo'lgan belgilar natija sifatida qaytariladi:

```

string s1,s2,s3;
s1="0123456789";
s2=s1; s3=s1;
s2=s1.substr(4,5); // s2="45678"
s3=s1.substr(3); // s3="3456789"
// "30123456789" satr ekranga chiqadi
cout<<s1.substr(1,3)+s1.substr();

```

string turidagi satrni char turiga o'tkazish

string turidagi satrni char turiga o'tkazish uchun

```
const char * c_str()const;
```

funksiyani ishlatish kerak. Bu funksiya char turdagi '\0' belgisi bilan tugaydigan satrga o'zgarma ko'rsatkichni qaytaradi:

```

shar *s1; string s2="0123456789";
s1=s2.c_str();

```

Xuddi shu maqsadda

```
const char * data()const;
```

funksiyasidan ham foydalanish mumkin. Lekin bu funksiya satr oxiriga '\0' belgisini qo'shmaydi.

Satr qismini izlash funksiyalari

string sinfida satr qismini izlash uchun har xil variantdagi funksiyalar aniqlangan. Quyida ulardan asosiylarining tavsifini keltiramiz.

**unsigned int find(const string &str,
unsigned int pos=0) const;**

Funksiya, uni chaqirgan satrning ko'rsatilgan joydan (pos) boshlab str satrni qidiradi va birinchi mos keluvchi satr qismining boshlanish indeksini javob sifatida qaytaradi, aks holda maksimal musbat butun npos sonini qaytaradi (npos=4294967295), agar izlash o'ri (pos) berilmasa, satr boshidan boshlab izlanadi.

unsigned int find(char c, unsigned int pos=0) const;

Bu funksiya oldingidan farqi ravishda satrdan s belgisini izlaydi.

unsigned int rfind(const string &str, unsigned int pos=npow) const;

Funksiya, uni chaqirgan satrning ko'rsatilgan pos o'rnigacha str satrning birinchi uchragan joyini indeksini qaytaradi, aks holda npos qiymatini qaytaradi, agar pos ko'rsatilmasa satr oxirigacha izlaydi.

unsigned int rfind(char c, unsigned int pos=npow) const;

Bu funksiyaning oldingidan farqi - satrdan s belgisi izlanadi.

unsigned int find_first_of(const string &str, unsigned int pos=0) const;

Funksiya, uni chaqirgan satrning ko'rsatilgan (pos) joyidan boshlab str satrning ixtiyoriy birorta belgisini qidiradi va birinchi uchraganining indeksini, aks holda npos sonini qaytaradi.

unsigned int find_first_of(char c, unsigned int pos=0) const;

Bu funksiyaning oldingidan farqi - satrdan s belgisini izlaydi;

unsigned int find_last_of(const string &str, unsigned int pos=npow) const;

Funksiya, uni chaqirgan satrning ko'rsatilgan (pos) joydan boshlab str satrni ixtiyoriy birorta belgisini qidiradi va o'ng tomondan birinchi uchraganining indeksini, aks holda npos sonini qaytaradi.

unsigned int find_last_of(char c, unsigned int pos=npow) const;

Bu funksiya oldingidan farqi - satrdan s belgisini izlaydi;

unsigned int find_first_not_of(const string &str, unsigned int pos=0) const;

Funksiya, uni chaqirgan satrning ko'rsatilgan (pos) joydan boshlab str satrning birorta ham belgisi kirmaydigan satr qismini qidiradi va chap tomondan birinchi uchraganining indeksini, aks holda npos sonini qaytariladi.

unsigned int find_first_not_of(char c, unsigned int pos=0) const;

Bu funksiyaning oldingidan farqi - satrdan s belgisidan farqli birinchi belgini izlaydi;

unsigned int find_last_not_of(const string &str, unsigned int pos=npow) const;

Funksiya, uni chaqiruvchi satrning ko'rsatilgan joydan boshlab str satrini tashkil etuvchi belgilar to'plamiga kirmagan belgini qidi-radi va eng o'ng tomondan birinchi topilgan belgining indeksini, aks holda npos sonini qaytaradi.

unsigned int find_last_not_of(char c, unsigned int pos=npow) const;

Bu funksiyaning oldingidan farqi - satr oxiridan boshlab s belgisiga o'xshamagan belgini izlaydi.

Izlash funksiyalarini qo'llashga misol:

```
#include <iostream.h>
```

```
#include <conio.h>
```

```
int main(){
```

```
string s1="01234567893456ab2csef",
```

```
s2="456",s3="ghk2";
```

```
int i,j;
```

```
i=s1.find(s2);
```

```
j=s1.rfind(s2);
```

```

cout<<i; // i=4
cout<<j; // j=11
cout<<s1.find('3') <<endl; // natija 3
cout<<s1.rfind('3') <<endl; // natija 10
cout<<s1.find_first_of(s3)<<endl; // natija 2
cout<<s1.find_last_of(s3)<<endl; // natija 16
cout<<s1.find_first_not_of(s2)<<endl; // natija 14
cout<<s1.find_last_not_of(s2)<<endl; // natija 20
}

```

Satrlarni solishtirish

Satrlar qismlarini solishtirish uchun compare funksiyasi ishlatiladi:

```

int compare(const string &str)const;
int compare(unsigned int pos1,unsigned int n1, const string & str)const;
int compare(unsigned int pos1,unsigned int n1, const string & str,unsigned int pos2,
            unsigned int n2)const;

```

Funksiyaning birinchi shaklida ikkita satrlar to'la solishtiriladi: funksiya manfiy son qaytaradi, agar funktsiyani chaqiruvchi satr str satrdan kichik bo'lsa, 0 qaytaradi agar ular teng bo'lsa va musbat son qaytaradi, agar funksiya chaqiruvchi satr str satrdan katta bo'lsa.

Ikkinchi shaklda xuddi birinchidek amallar bajariladi, faqat funksiya chaqiruvchi satrning pos1 o'rnidan boshlab n1 ta belgili satr osti str satr bilan solishtiriladi.

Uchinchi ko'rinishda funksiya chaqiruvchi satrning pos1 o'rnidan boshlab n1 ta belgili satr qismi va str satrdan pos2 o'rnidan boshlab n2 ta belgili satr qismlari o'zaro solishtiriladi.

Misol:

```

#include <iostream.h>
int main() {
    String s1="01234567893456ab2csef", s2="456",
    s3="ghk";
    cout<<"s1="<<s1<<endl;
    cout<<"s2="<<s2<<endl;
    cout<<"s3="<<s3<<endl;
    if(s2.compare(s3)>0)cout<<"s2>s3"<<endl;
    if(s2.compare(s3)==0)cout<<"s2=s3"<<endl;
    if(s2.compare(s3)<0)cout<<"s2<s3"<<endl;
    if(s1.compare(4,6,s2)>0)cout<<"s1[4-9]>s2"<<endl;
    if(s1.compare(5,2,s2,1,2)==0)
    cout<<"s1[5-6]=s2[1-2]"<<endl;
}

```

Masala. Familiya, ismi va shariflari bilan talabalar ro'yxati berilgan. Ro'yxat alfavit bo'yicha tartiblansin.

Programma matni:

```

#include <iostream.h>
#include <alloc.h>
int main(int argc, char* argv[]){
    const int FISH_uzunligi=50;
    string * Talaba;
    char * Satr=(char*)malloc(FISH_uzunligi);
    unsigned int talabalar_soni;
    char son[3];
}

```

```

do {
cout<<"Talabalar sonini kiriting: ";
cin>>son; }
while((talabalar_soni=atoi(son))<=0);
Talaba =new string[talabalar_soni];
cin.ignore();
for(int i=0; i<talabalar_soni; i++) {
cout<<i+1<<"-talabaning Familya ismi sharifi: ";
cin.getline(Satr,50);
Talaba[i].assign(Satr); }
bool almashdi=true;
for(int i=0; i<talabalar_soni-1 && almashdi; i++) {
almashdi=false;
for(int j=i; j<talabalar_soni-1; j++)
if(Talaba[j].compare(Talaba[j+1])>0) {
almashdi=true;
strcpy(Satr,Talaba[j].data());
Talaba[j].assign(Talaba[j+1]);
Talaba[j+1].assign(Satr); } }
cout<<"Alfavit bo'yicha tartiblangan ro'yxat:\n";
for(int i=0; i<talabalar_soni; i++)
cout<<Talaba[i]<<endl;
delete [] Talaba; free(Satr); return 0; }

```

Programmada talabalar ro'yxati string turidagi Talaba dinamik massiv ko'rinishida e'lon qilingan va uning o'lchami foydalanuvchi tomonidan kiritilgan talabar_soni bilan aniqlanadi. Talabalar sonini kiritishda nazorat qilinadi: klaviaturadan satr o'qiladi va u atoi() funksiyasi yordamida songa aylantiriladi. Agar hosil bo'lgan son noldan katta son bo'lmasa, sonni kiritish jarayoni takrorlanadi. Talabalar soni aniq bo'lgandan keyin har bir talabaning familiya, ismi va sharifi bitta satr sifatida oqimdan o'qiladi. Keyin, string turida aniqlangan compare() funksiyasi yordamida massivdagi satr-lar o'zaro solishtiriladi va mos o'rindagi belgilar kodlarini o'sishi bo'yicha «pufakchali saralash» orqali tartiblanadi. Programma oxirida hosil bo'lgan massiv chop etiladi, hamda dinamik massivlar yo'qotiladi.

Satr xossalari aniqlash funksiyalari

string sinfida satr uzunligi, uning bo'shligini yoki egallagan xotira hajmini aniqlaydigan funksiyalar bor:

```

unsigned int size()const; // satr o'lchami
unsigned int length()const; // satr elementlar soni
unsigned int max_size()const; // satrning maksimal uzunligi(4294967295)
unsigned int capacity()const; // satr egallagan xotira hajmi
bool empty()const; // true, agar satr bo'sh bo'lsa

```

Nazorat savollari

1. C++ dasturida nechta funksiya bo'ladi?
2. Kutubxonalar nima?
3. String turi qaysi kutubxonada joylashgan?
4. String turi bilovchi qanday funksiyalarni bilasiz?
5. Nomlangan nomlar sohasi nima?
6. O'zgaruvchilar nima uchun qo'llaniladi?

7. Qanday o'zgaruvchi turlari bor?
8. O'zgaruvchilar qanday qiymatlar qabul qiladi?
9. Kiritish operatori qanday ishlaydi?

11- MA'RUZA

MAVZU: MATNLI FAYLLAR BILAN ISHLASH OPERATORLARI

Reja:

1. Fayllar va oqimlar, diskdagi fayllar bilan ishlash;
2. Matnli fayllar;
3. Binar fayllar;
4. C++ tilida fayllar bilan ishlovchi maxsus funksiyalar. Istisno (exception) larni qayta ishlash (throw, try va catch).

Annotatsiya: Dasturchi fayllar ishini tashkil qilar ekan, faqat dastur va uning natijasi haqida qayg'uribgina qolmasdan, balki ko'plab qo'shimcha dasturlar yordamida fayllar yaratish, faylda saqlanayotgan ma'lumotlarni boshqarish, tahlil qilish, tartiblash, ehtiyojga qarab ekran yoki qog'ozda akslantirish kabi masalalarni ham hal qilishi kerak. Yana ilgari ko'zda tutilmagan yangi ehtiyojlar uchun qo'shimcha dasturlar yaratish haqida ham o'ylashi kerak.

Kalit so'zlar: *Fayl, axborotlarni saqlash, matnli ma'lumotlar, nom va kengaytma, tiplashgan fayllar, ofstream, ifstream, fstream, fizik fayllar, faylni ochish, oqim, oqim obykti, mode, open, ikkilik rejim, matnli fayl, binar fayl, faylni yopish, EOF, o'qish, seekg, seekp, tellg, tellp, streamoff, streampos, sinxronizatsiya, buffer, istisno, exception, xatoliklar, try, catch, throw.*

Fayl tushunchasi

C++ tilidagi standart va foydalanuvchi tomonidan aniqlangan turlarning muxim xususiyati shundan iboratki, ular oldindan berilgan chekli komponentalardan iborat yoki dinamik aniqlanganda operativ xotiraning cheklanganligidadir. Ma'lum bir sinf masalalari uchun oldindan komponentalar sonini aniqlash imkoni yo'q, ular masalani yechish jarayonida aniqlanishi va yetarlicha katta hajmda bo'lishi mumkin.

O'tilganlardan bizga ma'lumki, massivlar yuzlab, xatto minglab elementdan iborat bo'lishi mumkin. Buncha ma'lumotni klaviatura orqali kiritish uchun qancha vaqt behuda sarf bo'lishini tushunish qiyin emas. Shuning uchun dasturlashda, odatda, kata hajmdagi ma'lumotlar matnli fayldan o'qib olinadi. Bunday ma'lumotlar matnli ma'lumotlar sifatida turli usullar bilan hosil qilinadi. Masalan, ba'zi qurilmalarni nazorat testidan o'tkazish vaqtida olingan natijalar maxsus qurilmalar yordamida matnli faylga yozib boriladi.

Ma'lumotlarni kompyuter xotira qurilmalaridan birida saqlashning eng qulay shakli fayllar hisoblanadi. Axborotlarni saqlashning boshqa variantlari (masalan, ma'lumotlar bazasi) ham fayllarga asoslanadi.

Fayl — bu kompyuter xotira qurilmalaridan birida saqlanayotgan va o'z nomiga ega bo'lgan ma'lumotlar to'plamidir.

Fayl bo'sh bo'lishi ham mumkin. Fayllar ma'lumot saqlashning eng qulay usuli ekanligining sababi quyidagilardan iborat:

1) odatda dasturi bajarib, olingan natijalar dastur o'z ishini tugatgandan so'ng, EHM xotirasidan o'chib ketadi. Bu ma'lumotlarga yana ehtiyoj paydo bo'lsa, dasturni yangidan ishga tushirishga to'g'ri

keladi. Buning oldini olish uchun hosil qilingan natijalami fayllarga yozib qo'yish mumkin;

2)faylda saqlanayotgan ma'lumotlar ko'plab masala va dasturlar uchun yangi asos bo'lishi mumkin, ya'ni dastur natijalari saqlab qo'yilsa, bu ma'lumotlardan foydalanib boshqa masalalami yechish mumkin;

3)ma'lumotlar soni EHMning operativ xotirasiga sig'maydigan darajada ko'p bo'lishi mumkin. Bunday vaqtda ma'lumotlarning bir qismini biror faylda vaqtincha saqlab qo'yish mumkin;

4)fayllardan ulardagi ma'lumotlar doirasidagi ixtiyoriy maqsad va masalalar uchun foydalanish mumkin.

Fayllar o'zining manzili hamda nomiga ega bo'ladi. Faylning nomi odatda ikkita qismdan iborat bo'lishi mumkin: nom va kengaytma. Masalan:

D: /AkmX /local/ tuit. cpp
yozuvi, tuit.cpp faylni anglatadi. Bu yerda tuit - faylning nomi, .cpp esa uning kengaytmasi. Bu faylning manzili - D diskdagi AkmX papkasi.

Dasturchi fayllar ishini tashkil qilar ekan, faqat dastur va uning natijasi haqida qayg'uribgina qolmasdan, balki ko'plab qo'shimcha dasturlar yordamida fayllar yaratish, faylda saqlanayotgan ma'lumotlarni boshqarish, tahlil qilish, tartiblash, ehtiyojga qarab displey yoki qog'ozda akslantirish kabi masalalarni ham hal qilishi kerak. Yana ilgari ko'zda tutilmagan yangi ehtiyojlar uchun qo'shimcha dasturlar yaratish haqida ham o'ylashi kerak.

C++ tilida fayllar deb, EHMda saqlanayotgan bir xil tipga mansub bo'lgan ma'lumotlar (komponentalar) to'plamiga aytiladi.

Faylda saqlanayotgan ma'lumotlardan foydalanish uchun ularni o'qish va o'zgaruvchilarga qiymat qilib berish talab qilinadi.

Ixtiyoriy vaqtda faylning faqat bitta komponentasi bilan ishlash mumkin, xolos. Bu ma'lumotni ko'rsatkich (kursor) ko'rsatib turadi. Ko'rsatkich birinchi komponentadan boshlab, har bir ma'lumot o'qilgandan keyin, navbatdagi o'qish kerak bo'lgan ma'lumotni ko'rsatib turadi. (Boshlang'ich sinfdagi xatcho'plarni eslab ko'ring.)

Fayldagi ma'lumotlar soni o'zgarib turadi va u dastlab nolga teng bo'ladi. Bu son keyinchalik faylga yangi ma'lumotlar qo'shilganda ortishi yoki o'chirilganda nolgacha kamayishi mumkin. Yangi ma'lumotlar odat bo'yicha doim faylning oxiriga qo'shiladi.

Dastur yordamida qayta ishlashga mo'ljallangan fayllar odatda tiplashgan va tiplashmagan bo'ladi.

Tiplashgan fayllar faqat ma'lum bir tipdagi ma'lumotlarni saqlaydi.

Ma'lum bir tipga mansub bo'lgan va fayllarda saqlanayotgan ma'lumotlar yozuv deb ataladi. Faylning yozuvlari baytlar bilan o'lchanadigan chekli hajmga ega va bu hajm barcha yozuvlar uchun bir xil. Har bir yozuvning faylda turgan o'rnini doimo aniqlash mumkin.

Tiplashmagan fayllar ma'nosi dasturchi tomonidan aniqlanadigan baytlarning chiziqli ketma-ketligini o'z ichiga oladi.

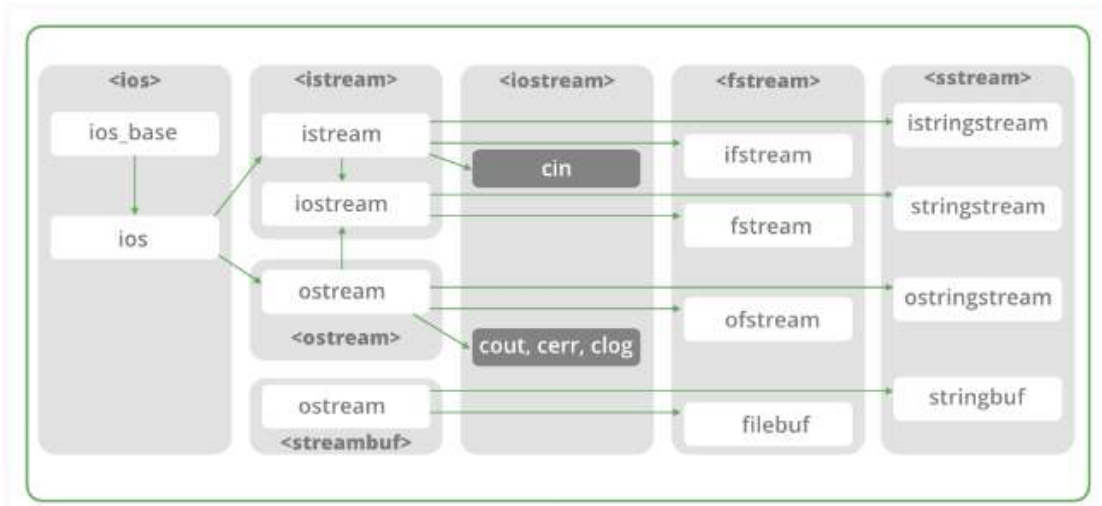
Fayllarni C++ sinflari orqali ishlash

C++ da fayllar, asosan, fstream sarlavha faylida mavjud bo'lgan uchta ofstream, ifstream, fstream oqimlari yordamida ko'rib chiqiladi.

ofstream: fayllarga ma'lumotlarni yozish uchun sinf oqimi;

ifstream: fayllardan ma'lumotlarni o'qish uchun sinf oqimi;

fstream: fayllardan ma'lumotlarni o'qish va fayllarga yozish uchun sinf oqimi.



Bu sinflar to'g'ridan-to'g'ri yoki bilvosita istream va ostream sinflaridan olinadi. Biz allaqachon bu klaslarning turlaridan bo'lgan ya'ni istream sinfining obyektini hisoblagan cin va ostream sinfining obyektini hisoblagan cout lardan foydalanganmiz. Shuning uchun, biz avvaldan fayl oqimlari bilan bog'liq bo'lgan sinflardan foydalanib kelganmiz. Aslida, biz fayl oqimlarimizdan avvalgidek foydalana olamiz, faqat cin va coutdan yagona farqi bu fayllarni fizik fayllar bilan bog'lashimiz kerak. Quyidagi misolni ko'rib chiqaylik:

```
#include <iostream>
```

```
#include <fstream>
```

```
using namespace std;
```

```
int main () {
```

```
    ofstream meningfaylim;
```

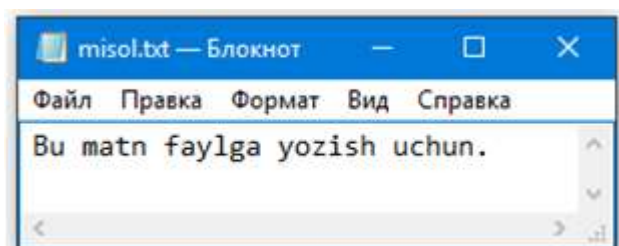
```
    meningfaylim.open ("misol.txt");
```

```
    meningfaylim << "Bu matn faylga yozish uchun.\n";
```

```
    meningfaylim.close();
```

```
    return 0; }
```

Natija:



Ushbu kod *misol.txt* nomli fayl yaratadi va unga biz odatdagi ya'ni coutdan foydalanganimiz kabi, lekin cout o'rniga fayl oqimidan foydalanib *meningfaylim* jumllarini qo'shamiz.

Faylni ochish

Odatda ushbu sinflar orqali hosil qilingan obyekt yordamida bajariladigan birinchi operatsiya uni haqiqiy faylga bog'lashdir. Ushbu jarayon faylni ochish sifatida ma'lum . Ochiq fayl dastur ichida oqim bilan taqdim etiladi (masalan, ushbu sinflardan birining obyekt; oldingi misolda bu meningfaylim deb nomlangan edi) va ushbu oqim obyekt ustida bajarilgan har qanday kirish yoki chiqish operatsiyalari bog'langan fizik faylga qo'llaniladi, ya'ni har qanday bajargan kodlarimizning natijasi biz yaratgan fayl ichida namoyon bo'ladi.

Oqim obyekt yordamida faylni ochish uchun biz uning a'zo funksiyasi bo'lgan open dan foydalanamiz:

open (faylnomi, mode);

Bu yerda *faylnomi* - ochiladigan fayl nomini ko'rsatadigan satr va *mode*(rejim) - bu parametrlar quyidagi jadvalda keltirilgan belgilarning ixtiyoriy bittasi:

ios::in	Kirish operatsiyalari ochish uchun.
ios::out	Chiqish operatsiyalari ochish uchun.
ios::binary	Ikkilik rejimda ochish.
ios::ate	Fayl oxirida boshlang'ich pozitsiyasini o'rnatish. Agar ushbu belgi o'rnatilmagan bo'lsa, boshlang'ich pozitsiya faylning boshidir.
ios::app	Barcha chiqish operatsiyalari kontentni faylning amaldagi tarkibiga qo'shib, fayl oxirida amalga oshiriladi.
ios::trunc	Agar fayl chiqish operatsiyalari uchun ochilgan bo'lsa va u avvaldan mavjud bo'lsa, avvalgi tarkib o'chiriladi va yangisi bilan almashtiriladi.

Ushbu belgilarning barchasini OR (|) buyrug'i yordamida birlashtirish mumkin . Masalan, agar biz misol.bin ma'lumotlarni qo'shish uchun ikkilik rejimda faylni ochishni xohlasak, uni a'zo funksiyasiga quyidagicha murojaat qilish orqali amalga oshirishimiz mumkin open:

```
1 ofstream meningfaylim;
   meningfaylim.open ("misol.bin", ios::out | ios::app | ios::binary);
2
```

Ofstream, ifstream va fstream sinflari har birining open a'zo funksiyalari, agar fayl ikkinchi argumentsiz ochilgan bo'lsa, ishlatiladigan odatiy rejimiga ega:

sinf	odatiy rejim parametri
ofstream	ios :: out
ifstream	ios :: in
fstream	ios :: in ios::out

Ifstream va ofstream sinflari uchun ios :: in va ios :: out avtomatik ravishda va mos ravishda qabul qilinadi, hatto ular mode(rejim)ni o'z ichiga olmasa ham mode **open** a'zo funksiyasining ikkinchi parametri sifatida beriladi (belgilar birlashtirilgan).

Fstream uchun odatiy(default) qiymat faqat funksiya mode(rejim) parametri uchun hech qanday qiymat ko'rsatmasdan murojaat qilinganda qo'llaniladi. Agar funksiya ushbu parametrdan biron bir qiymat bilan chaqirilsa, odatiy mode(rejim) bekor qilinadi, birlashtirilmaydi.

Ikkilik rejimda ochilgan fayl oqimlari kirish va chiqish operatsiyalarini har qanday format nuqtai nazaridan mustaqil ravishda amalga oshiradi. Ikkilik bo'lmagan fayllar *matnli fayllar* deb nomlanadi va ba'zi bir maxsus belgilar (masalan, yangi qator va yetkazadigan qaytaruvchi belgilar)ni formatlash tufayli ba'zi tarjimalar yuzaga kelishi mumkin.

Fayl oqimida bajariladigan birinchi vazifa odatda faylni ochish bo'lganligi sababli, ushbu uchta sinf avtomatik ravishda ochiq a'zo funksiyasini chaqiradigan va ushbu a'zo bilan bir xil parametrlarga ega bo'lgan konstruktorni o'z ichiga oladi. Shuning uchun biz oldin *meningfaylim*

obyektini e'lon qilishimiz va avvalgi misolimizdagi bilan bir xil bo'lgan ochilish operatsiyasini yozish orqali amalga oshirishimiz mumkin edi:

```
ofstream meningfaylim("misol.bin", ios::out | ios::app | ios::binary);
```

Obyektning yaratilishi va oqim ochilishini bitta ifodada birlashtirish. Faylni ochish uchun ikkala holat ham to'g'ri va ikkalasini ham qo'llash mumkin.

Fayl oqimi faylni muvaffaqiyatli ochganligini tekshirish uchun a'zo funktsiya `is_openga` murojaat qilib buni amalga oshirish mumkin. Ushbu funktsiya, agar oqim obyekti ochiq fayl bilan bog'langan bo'lsa yoki aksi bo'lsa, mantiqiy qiymat qaytaradi:

```
if(meningfaylim.is_open()){ /* OK, chiqish bilan davom etish */}
```

Faylni yopish

Faylni kiritish va chiqarish bo'yicha operatsiyalarimizni tugatgandan so'ng, biz uni yopamiz, shunda operatsion tizim xabardor qilinadi va uning manbalari yana mavjud bo'ladi. Buning uchun biz oqimning `close` deb nomlangan a'zo funktsiyasidan foydalanamiz. Ushbu a'zo funktsiya bog'liq buferlarni tozalaydi va faylni yopadi:

```
meningfaylim.close();
```

Ushbu funktsiya chaqirilganda, oqim obyekti boshqa faylni ochishda qayta ishlatilishi mumkin va fayl boshqa jarayonlar uchun yana ochib ishlatilishi mumkin bo'ladi.

Agar obyekt ochiq fayl bilan bog'langan paytda o'chirilsa yoki qaysidir sabab bilan yo'q qilinsa, destruktork avtomatik ravishda a'zo funktsiyasi bo'lgan ***close*** ni chaqiradi .

Matnli fayllar

Matnli fayl oqimlari bu `ios::binary` belgisi ochilish rejimiga kiritilmagan qismlardir. Ushbu fayllar matnni saqlash uchun mo'ljallangan va shuning uchun ularga kirish yoki chiqish qiymatlari bazi formatlash o'zgarishlariga duch kelishi mumkin, lekin bu ularning asl ikkilik qiymatiga to'g'ri kelmaydi. Boshqacha qilib aytganda, matnli fayllar - bu odam o'qiydigan belgilarni matnli hujjat sifatida saqlash uchun foydalaniladigan ikkilik fayllarning maxsus to'plami. Matnli fayllar ham ma'lumotlarni ketma-ket baytlarda saqlaydi, ammo matnli fayllardagi bitlar belgilarni anglatadi.

Matnli fayllar buzilishlarga kamroq moyil bo'ladi, chunki istalmagan o'zgarishlar shunchaki fayl ochilganda paydo bo'lishi mumkin va keyin ularni osongina olib tashlash mumkin.

Matnli fayllar ikki xil bo'ladi:

- Oddiy matnli fayllar: Ushbu fayllar har bir satr oxirida satr uzilishini va fayl oxirida faylning oxiri(End of File-EOF)ni ifodalash uchun End of Line (EOL) markerini saqlaydi.

- Boyitilgan matnli fayllar: Ushbu fayllar oddiy matnli fayllar bilan bir xil sxema bo'yicha ishlaydi, lekin boyitilgan matnli fayllar matn rangi, matn uslubi, shrift uslubi va boshqalar kabi matn bilan bog'liq ma'lumotlarni saqlashi mumkin.

Ma'lumotni saqlash uchun oddiy va standart format tufayli, matnli fayllar matnli ma'lumotlarni saqlash uchun eng ko'p ishlatiladigan fayl formatlaridan biri bo'lib, ko'plab dasturlarda qo'llab-quvvatlanadi.

Matn fayllariga yozish operatsiyalari biz ishlatgan `cout` operatori bilan bir xil usulda bajariladi:

```

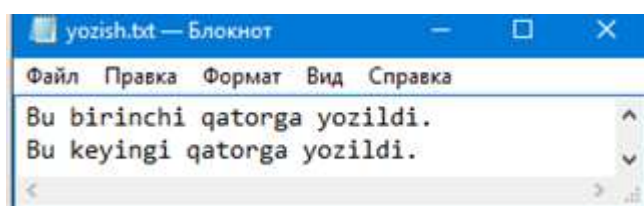
1 // Matnli faylga yozish
  #include <iostream>
  #include <fstream>
2 using namespace std;

  int main () {
3   ofstream meningfaylim ("misol.txt");
    if (meningfaylim.is_open())
    {
4     meningfaylim<< "Bu birinchi qatorga yozildi.\n";
      meningfaylim<< "Bu keyingi qatorga yozildi. \n";
      meningfaylim.close();
5   }
    else cout << "Faylni ochib bo'lmadi!";
    return 0;
6 }

```

[misol.txt fayli]
Bu birinchi qatorga yozildi.
Bu keyingi qatorga yozildi.

Natija:



Fayldan ma'lumotni o'qish, xuddi cin operatori qo'llanilish jarayonidek amalga oshiriladi:

```

1 // Matnli fayldan o'qish
  #include <iostream>
  #include <fstream>
2  #include <string>
  using namespace std;

3  int main () {
    string uq_matn;
    ifstream meningfaylim ("yozish.txt");
    if (meningfaylim.is_open())

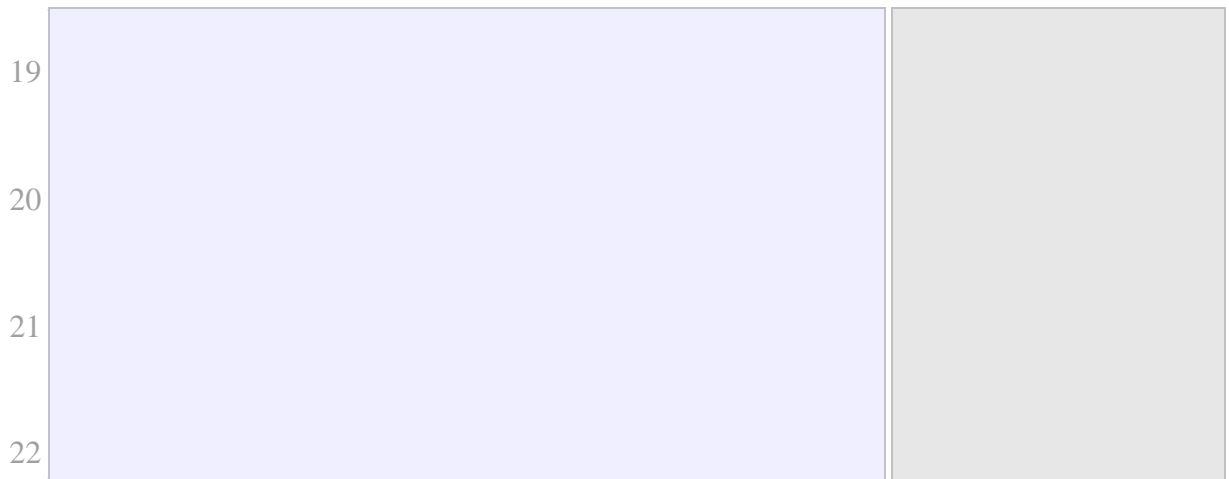
```

Matnli fayldan o'qilgan ma'lumotlar:
Bu birinchi qatorga yozildi.
Bu keyingi qatorga yozildi.

```
4      {
      cout<<"Matnli fayldan o'qilgan ma'lumotlar:"<<endl;
      while ( getline (meningfaylim, uq_matn) )
      {
5      cout << uq_matn << '\n';
      }
      meningfaylim.close();
6      }

      else cout << " Faylni ochib bo'lmadi!";

7      return 0;
8      }
9
10
11
12
13
14
15
16
17
18
```



Ushbu oxirgi misol matn faylini o'qiydi va o'qilgan tarkibini ekranda chiqaradi. Biz getline() funksiyasidan foydalanib fayl satrini ketma-ket o'qiydigan while siklini yaratdik.

Getline tomonidan qaytarilgan qiymat oqim obyektining havolasi bo'lib, agar u mantiqiy ifoda sifatida baholanganda (ushbu while sikli kabi) oqim ko'proq operatsiyalarga tayyor bo'lsa va u faylning oxiriga yetgan bo'lsa yoki boshqa biron bir xato bo'lsa, yolg'on(false) qiymat qaytaradi.

Masala. Binar fayldan haqiqiy sonlarni o'qish (agar fayl mavjud bo'lmasa uni hosil qilish va haqiqiy sonlar bilan to'ldirish) va o'qilgan sonlarning o'rta arifmetigini hisoblash, hamda ushbu sonlar orasidan hisoblangan o'rta arifmetikdan kichiklari miqdorini aniqlash dasturi tuzilsin.

Izoh: Faylni yaratish va undagi o'rta arifmetikdan kichik sonlar miqdorini aniqlash, alohida funksiyalar ko'rinishida tasvirlanishi mumkin.

```
#include <iostream.h>
#include <stdio.h>
# include <string.h>
// Yangi fayl yaratish va unga sonlarni yozish funksiyasi;
int Fayl_tuzish()
{
    FILE *f;
    double x;
    // «f» fayl yangidan hosil qilish uchun ochilmoqda;
    if((f=fopen("Haqiqiy.son", "wb+"))==NULL) return 0;
    char *satr=new char[10];
    int n=1;
    do {
        cout<<"Haqiqiy sonni kiriting: ";
        gets(satr);
        if(strlen(satr))
        { x=atof(satr);
          fwrite (&x,sizeof(double),n,f); } }
    while(strlen(satr));
    // kiritilgan satr bo'sh bo'lmasa, takrorlanish davom etadi;
    fclose(f);
    return 1; }
// O'rta arifmetikdan kichik sonlar miqdorini hisoblash funksiyasi;
int Kichiklar_soni()
{
    FILE*f; double x;
```

```

f=fopen("Haqiqiy.son", "rb+");
double s=0; // s - f fayl elementlari yig'indisi;
while(!feof(f))
{
if (fread(&x,sizeof(double),1,f)) s+=x;
}
long sonlar_miqdori=ftell(f)/sizeof (double);
s/=sonlar_miqdori; // s- o'rta arifmetik;
cout<<"Fayldagi sonlar o'rta arifmetigi="<<s<<'endl';
fseek(f,SEEK_SET,0); // fayl boshiga kelinsin;
int k=0;
while (fread(&x,sizeof(x),1,f))
{
k+=(x<s); //o'rta arifmetikdan kichik elementlar soni;
}
fclose(f);
return k;
}
int main()
{
if(Fayl_tuzish())
{
cout<<"Haqiqiy.son faylidagi \n";
int Kichik=Kichiklar_soni();
cout<<"O'rta arifmetikdan kichik sonlar miqdori="; cout<<Kichik;
}
else // f faylini yaratish muvaffaqiyatsiz bo'ldi.
cout<<"Haqiqiy.son faylini ochish imkoni bo'lmadi!!!"; return 0;
}

```

Dasturda bosh funksiyadan tashqari ikkita funksiya aniqlangan:

Int Fayl_tuzish() - diskda “*Haqiqiy.son*” nomli faylni yaratadi. Agar faylni yaratish muvaffaqiyatli bo'lsa, funksiya *1* qiymatini, aks holda *0* qiymatini qaytaradi. Faylni yaratishda klaviaturadan sonlarning satr ko'rinishi o'qiladi va haqiqiy songa aylantirilib, faylga yoziladi. Agar bo'sh satr kiritilsa, sonlarni kiritish jarayoni to'xtatiladi va fayl yopiladi;

int Kichiklar_soni() funksiyasi diskdagi “*Haqiqiy.son*” nomli faylni o'qish uchun ochadi va fayl elementlarining o'rta arifmetigi *s* hisoblanadi so'ngra o'rta arifmetikdan kichik bo'lgan elementlar miqdori *k* hisoblanib, funksiya natijasi sifatida qaytariladi.

Bosh funksiyada faylning yaratilishi tekshiriladi va unga mos xabar beriladi.

Holat belgilarini tekshirish

Oqimning muayyan holatini tekshirish uchun quyidagi funktsiyalar mavjud (ularning barchasi `bool` qiymatni qaytaradi):

bad()

Agar o'qish yoki yozish jarayoni muvaffaqiyatsiz bo'lsa funksiya `true` qaytaradi . Masalan, biz yozishga tayyor bo'lmagan faylga yozishga harakat qilsak yoki biz yozmoqchi bo'lgan qurilmada bo'sh joy qolmasa.

fail()

`true` ba'zi bir xil hollarda `bad()` qiymat qaytaradi, ammo bu holatda format xatosi yuz berishi ham mumkin, masalan, butun sonni o'qishga harakat qilayotganimizda alfavit belgisi olinganda.

eof()

Agar o'qish uchun ochilgan fayl oxiriga yetgan bo'lsa, `true`(rost) qiymatni qaytaradi.

good()

Bu eng umumiy holat belgisi: oldingi funktsiyalardan birini chaqirgan paytda rostan `true`(rost) bo'lgan holatlarda u `false` qaytaradi. E'tibor bering, yaxshi va yomon aniq bir-biriga zid emas (`good` bir vaqtning o'zida ko'proq holat belgilarini tekshiradi).

`clear()` a'zo funktsiyasi holat belgilarini tiklash uchun ishlatilishi mumkin.

Oqim holatida get(olish) va put(joylashtirish)

Barcha k/ch oqim obyektlari ichkarida saqlanadi – kamida - bitta ichki holat:

`ifstream`, `istream` singari, keyingi kirish jarayonida o'qilishi kerak bo'lgan elementning joylashuvi bilan ichki kirish holatini saqlab turadi.

`ofstream`, `ostream` singari, keyingi element yozilishi kerak bo'lgan joylashuv bilan ichki joylashuv pozitsiyasini saqlab turadi.

Va nihoyat, `fstream`, `iostream` kabi, ikkalasini ham olish va qo'yish pozitsiyasi(holati)ni saqlaydi.

Oqimning ichki pozitsiyalari keyingi o'qish yoki yozish jarayoni amalga oshiriladigan oqim ichidagi joylarga ishora qiladi. Ushbu pozitsiyalar quyidagi a'zo funktsiyalari yordamida kuzatilishi va o'zgartirilishi mumkin:

tellg() and tellp()

Parametrlari bo'lmagan ushbu ikki a'zo funktsiyalari mavjud `get` pozitsiyasini (`tellg` holatida) yoki joylashish pozitsiyasini (`tellp` holatida) ifodalaydigan `streampos` turdagi qiymatini qaytaradi.

seekg() and seekp()

Ushbu funktsiyalar `get(olish)` va `put(joylashtirish)` pozitsiyalarini o'zgartirishga imkon beradi. Ikkala funktsiya ikki xil prototip bilan qayta yuklangan. Birinchi shakl:

`seekg (position);`

Ushbu prototipdan foydalanib, oqim ko'rsatkichi mutlaq pozitsiya holatiga o'zgartiriladi (fayl boshidan hisoblab chiqiladi). Ushbu parametr uchun tip - bu `streampos`, bu esa `tellg` va `tellp` funktsiyalari tomonidan qaytariladigan turga o'xshaydi.

Ushbu funktsiyalar uchun boshqa shakl:

`seekg (offset, direction);`

Ushbu prototipdan foydalanib `get` yoki `put` pozitsiyasi parametr yo'nalishi bilan aniqlangan ba'zi bir aniq nuqtaga nisbatan ofset qiymatiga o'rnatiladi. Offset bu `streamoff`(oqim)ning tipidir. Va yo'nalish `seek`dir turiga kiradi, bu sanab o'tilgan tip bo'lib, u offset hisoblanadigan joyni belgilaydi va quyidagi qiymatlardan birini qabul qilishi mumkin:

<code>ios::beg</code>	oqim boshidan hisoblangan offset
<code>ios::cur</code>	joriy holatdan hisoblangan offset
<code>ios::end</code>	oqim oxiridan hisoblangan offset

Quyidagi misol faylning hajmini olish uchun biz ko'rgan a'zo funktsiyalaridan foydalanadi:

```
// fayl hajmini olish
#include <iostream>
#include <fstream>
using namespace std;

int main () {
```

hajmi: 60 bayt.

```
3      streampos begin,end;
      ifstream meningfaylim ("yozish.txt", ios::binary);
      begin = meningfaylim.tellg();
      meningfaylim.seekg (0, ios::end);
4      end = meningfaylim.tellg();
      meningfaylim.close();
      cout << "hajmi: " << (end-begin) << " bayt.\n";
5      return 0;
      }
```

6
7
8
9
10
11
12
13
14
15

O'zgaruvchilar uchun foydalangan turimizga e'tibor bering begin va end:

streampos size

streampos - bufer va faylni joylashishi uchun ishlatiladigan o'ziga xos tur va file.tellg() tomonidan qaytarilgan tur. Ushbu turdagi qiymatlarni boshqa turdagi qiymatlardan xavfsiz ajratib olish mumkin va shuningdek, fayl hajmini o'z ichiga oladigan butun son turiga o'tkazish mumkin.

Oqimni aniqlashning ushbu funktsiyalari ikkita o'ziga xos turdan foydalanadi: streampos va streamoff. Ushbu turlar, shuningdek, oqim sinfining a'zo turlari sifatida aniqlanadi:

Turi	A'zo turi	Ta'rif
streampos	ios::pos_type	fpos<mbstate_t> sifatida belgilangan. U streamoff-ga / dan o'zgartirilishi mumkin va ushbu turdagi qiymatlarni qo'shish yoki ayirish mumkin.
streamoff	ios::off_type	Bu asosiy integral turlaridan biriga(masalan int yoki long long) biri kabi o'xshashdir .

Yuqoridagi a'zolar turlarining har biri uning a'zosi bo'lmagan ekvivalenti o'xshashidir (ular aynan bir xil). Qaysi biri ishlatilganligi muhim emas. A'zolar turlari ko'proq umumiydir, chunki ular barcha oqim obyektlarida bir xil (hattoki belgilarning ekzotik turlaridan foydalanadigan oqimlarda), lekin a'zo bo'lmagan turlar mavjud kodda tarixiy sabablarga ko'ra keng qo'llaniladi.

Fayldan o'qish va yozishga namuna

Quyidagi C++ dasturi faylni o'qish va yozish rejimida ochiladi. Foydalanuvchi tomonidan kiritilgan ma'lumotni sinov.dat nomli faylga yozgandan so'ng, dastur fayldan ma'lumotlarni o'qiydi va uni ekranga chiqaradi.

<i>Fayldan o'qish va yozishga namuna</i>
<pre> #include <fstream> #include <iostream> using namespace std; int main () { char data[100]; // faylni yozish rejimida ochish. ofstream outfile; outfile.open("sinov.dat"); cout << "Faylga yozish" << endl; cout << "Ismingizni kiriting: "; cin.getline(data, 100); // kiritilgan ma'lumotlarni faylga yozish. outfile << data << endl; cout << "Yoshingizni kiriting: "; cin >> data; cin.ignore(); // faylga yana kiritilgan ma'lumotlarni yozing. outfile << data << endl; // ochilgan faylni yopish. outfile.close(); // faylni o'qish rejimida ochish. ifstream infile; infile.open("sinov.dat"); </pre>

```

cout << "Fayldan o'qish" << endl;
infile >> data;

// ekranda ma'lumotlarni chop qilish.
cout << data << endl;

// fayldan ma'lumotlarni qayta o'qish va uni namoyish etish.
infile >> data;
cout << data << endl;

// ochilgan faylni yopish.
infile.close();

return 0;
}

```

Binar(ikkilik) fayllar

Binar(ikkilik) fayllar—bu oddiygina baytlar ketma-ketligidir. Binar fayllardan berilganlarni foydalanuvchi tomonidan bevosita ko'rish zarur bo'lmagan hollarda foydalaniladi. Binar fayllaridan o'qish-yozishda baytlar ustida hech qanday konvertatsiya amallari bajarilmaydi.

Ikkilik fayl sakkiz yoki ba'zan o'n olti bitga guruhlangan baytlar ketma-ketligi ko'rinishida ma'lumotlarni saqlaydigan tipik fayllar. Ushbu bitlar maxsus ma'lumotlarni anglatadi va bunday fayllar bir nechta ma'lumotlarni (rasmlar, audio, matn va h.k.) bitta fayl ostida saqlashi mumkin.

Ikkilik faylning eng keng tarqalgan namunalaridan biri bu rasmiy fayl .PNG yoki .JPG. Agar kimdir ushbu fayllarni matn muharriri yordamida ochishga harakat qilsa, u tanib bo'lmaydigan belgilarga ega bo'lishi mumkin, ammo qo'llab-quvvatlaydigan rasmlarni ko'rish vositasi yordamida ochilganda, fayl bitta rasm sifatida ko'rsatiladi. Buning sababi, fayl ikkilik formatda va baytlar ketma-ketligi ko'rinishidagi ma'lumotlarni o'z ichiga oladi. Matn muharriri ushbu baytlarni o'qishga harakat qilganda va bitlarni belgilarga aylantirishga harakat qilganda, ular keraksiz maxsus belgilarni olishadi va uni foydalanuvchiga ko'rsatadi.

Ikkilik fayllar, shuningdek, fayl nomi, fayl formati va boshqalar kabi fayl ma'lumotlarini saqlaydi, ular faylga sarlavha sifatida kiritilishi mumkin va hatto fayl matn muharririda ochilganda ham ko'rinadi.

Ikkilik fayllar ma'lumotni ketma-ket baytlarda saqlaganligi sababli, fayldagi kichik o'zgarish faylni buzishi va uni qo'llab-quvvatlaydigan dasturda o'qib bo'lmaydigan holga keltirishi mumkin.

Ikkilik fayllarda ma'lumotlarni o'qish va yozish, olish va kiritish operatorlari (<< va >>) va getline kabi funktsiyalarda samarasiz, chunki biz biron-bir ma'lumotlarni formatlashimiz shart emas va ma'lumotlar satrlarda formatlanmagan bo'lishi mumkin.

Endi fayllar bilan ishlashda kerak bo'ladigan bir qator tushunchalar bilan tanishamiz. Oqim tushunchasi—bu berilganlarni faylga o'qish-yozishda ularni belgilar ketma-ketligi yoki oqimi ko'rinishida tasavvur qilishdan kelib chiqqan. Oqim ustida quyidagi amallarni bajarish mumkin:

- oqimdan berilganlar blokini operativ xotiraga o'qish;
- operativ xotiradagi berilganlar blokini oqimga yozish (chiqarish);
- oqimdagi berilganlar blokini yangilash;
- oqimdan yozuvni o'qish;
- oqimga yozuvni chiqarish.

Oqim bilan ishlaydigan barcha funktsiyalar buferli, formatlashgan yoki formatlashmagan o'qish-yozishni ta'minlaydi.

Dastur ishga tushganda o'qish-yozishning quyidagi standart oqimlari ochiladi:

stdin - o'qishning standart vositasi;

stdout - yozishning standart vositasi;

stderr - xatolik haqida xabar berishning standart vositasi;

stdprn - qog'ozga chop qilishning standart vositasi;

stdaux - standart yordamchi qurilma.

Kelishuv bo'yicha *stdin*-foydalanuvchi klaviaturasi, *stdout* va *stderr* - terminal (monitor), *stdprn* - printer bilan, hamda *stdaux* - kompyuter yordamchi portlariga bog'lanish hisoblanadi. Berilganlarni o'qish-yozishda *stderr* va *stdaux* oqimidan boshqa oqimlar buferlanadi, ya'ni belgilar ketma-ketligi operativ xotiraning bufer deb nomlanuvchi sohasida vaqtincha jamlanadi.

Fayl oqimlari ikkilik(binary) ma'lumotlarini ketma-ket o'qish va yozish uchun maxsus yaratilgan ikkita a'zo funksiyasini o'z ichiga oladi: *write* (yozish) va *read* (o'qish). Birinchisi (yozish) *ostream*ning a'zo funksiyasi (meros orqali meros qilib olingan). Va o'qish *istream* a'zoligi funksiyasi (*ifstream* tomonidan meros qilib olingan). *Fstream* sinf obyektlari ikkalasiga ham ega. Ularning prototiplari:

```
write(xotira_block,size);
```

```
read(xotira_block,size);
```

xotira_block *char ** (pointer to *char*) turiga kiradi va o'qilgan ma'lumotlar elementlari saqlanadigan yoki yoziladigan ma'lumot elementlari olinadigan baytlarning manzilini bildiradi. *size* parametri - bu xotira blokidan o'qiladigan yoki yoziladigan belgilar sonini aniqlaydigan butun son.

```
1 // to'liq ikkilik faylni o'qish
  #include <iostream>
2 #include <fstream>
  using namespace std;
3
4 int main () {
  streampos hajm;
  char* xotirablock;
5
  ifstream fayl ("misol.bin", ios::in|ios::binary|ios::ate);
6  if (fayl.is_open())
  {
7    hajm = fayl.tellg();
    xotirablock = new char [hajm];
8    fayl.seekg (0, ios::beg);
    fayl.read (xotirablock, hajm);
9    fayl.close();

10   cout << " butun fayl tarkibi xotirada";

    delete[]xotirablock;
  }
11  else cout << " Faylni ochib bo'lmadi!";
    return 0;
12 }
```

butun fayl tarkibi xotirada

13

14

15

16

17

18

19

20

21

22

23

24

25

Natija:

```
D:\ab\akmal\progsC++\2019\fayloqimlari\bin\Debu...
butun fayl tarkibi xotirada
Process returned 0 (0x0)   execution time : 0.148 s
Press any key to continue.
```

Ushbu misolda, fayl to'lig'icha o'qiladi va xotira blokida saqlanadi. Buni qanday amalga oshirilishini ko'rib chiqamiz:

Birinchidan, fayl ios::ate belgisi bilan ochilgan, ya'ni get pointer fayl oxirida joylashishini anglatadi. Shu tarzda, tellg() a'zoga murojaat qilganimizda, biz fayl hajmini to'g'ridan-to'g'ri olamiz.

Fayl hajmini olgandan so'ng, butun faylni ushlab turish uchun katta hajmdagi xotira blokini ajratishni so'raymiz:

```
xotirablock = new char[hajm];
```

Shundan so'ng, biz faylning boshida *get* manzilini o'rnatamiz (oxirida faylni ushbu ko'rsatgich bilan ochganimizni eslang), so'ngra faylni to'liq o'qib chiqdik va nihoyat uni yopamiz:

```
file.seekg (0, ios::beg);
```

```
file.read (memblock, size);
```

```
file.close();
```

Bu vaqtda biz fayldan olingan ma'lumotlar bilan ishlashimiz mumkin edi. Ammo bizning dasturimiz shunchaki faylning mazmuni xotirada ekanligini va keyin tugashini e'lon qiladi.

Qisqacha qilib aytganda ma'lumotlar ikkilik formatdagi faylda saqlansa, ma'lumotlarni o'qish va yozish tezroq amalga oshiriladi, chunki ma'lumotlarni bir formatdan boshqa formatga o'tkazish uchun vaqt yo'qotilmaydi. Bunday fayllarga **ikkilik fayllar** deyiladi.

Matnli va binar fayllarni ochish turlari(mode o'rniga yozish mumkin bo'lgan qiymatlar) quyidagilar:

mode	Ma'nosi
r	Faylni o'qish uchun ochiladi(ingl.read – o'qish)
w	Faylni yozish uchun hosil qiladi(ingl.write - yozish)
a	Fayl davomiga qo'shish uchun ochadi(ingl.append – oxiriga qo'shish)
rb	Ikkilik faylni o'qish uchun ochadi
wb	Ikkilik faylni yozish uchun hosil qiladi
ab	Ikkilik faylni oxiriga qo'shish uchun ochadi
r+	Faylni o'qish va yozish uchun ochadi
w+	O'qish va yozish uchun fayl hosil qiladi
a+	Faylni o'qish va davomiga qo'shish uchun ochadi
r+b	Ikkilik faylni o'qish va yozish uchun ochadi
w+b	Ikkilik faylni o'qish va yozish uchun hosil qiladi
a+b	Ikkilik faylni o'qish va oxiriga yozish uchun ochadi

Bufferlar va sinxronizatsiya

Fayl oqimlari bilan ishlaganda, ular streambuf tipidagi ichki bufer obyekt bilan bog'lanadi. Ushbu bufer obyekt oqim va fizik fayl o'rtasida vositachi sifatida ishlaydigan xotira blokini ifodalashi mumkin. Masalan, ofstream bilan har safar a'zo funktsiyasi put (bitta belgi yozadi) chaqirilganda, oqim to'g'ridan-to'g'ri bog'liq bo'lgan fizik faylga yozilmasdan, belgi shu oraliq buferga joylashtirilishi mumkin.

Operatsion tizim fayllarni o'qish va yozish uchun buferlashning boshqa qatlamlarini ham belgilashi mumkin.

Bufer tozalanganda, undagi barcha ma'lumotlar fizik muhitga yoziladi (agar u chiqish oqimi bo'lsa). Ushbu jarayon **sinxronizatsiya** deb ataladi va quyidagi holatlardan birida sodir bo'ladi:

- **Fayl yopilganda:** faylni yopishdan oldin, hali tozalanmagan barcha buferlar sinxronlashtiriladi va barcha kutilayotgan ma'lumotlar fizik vositaga yoziladi yoki o'qiladi.

- Bufer to'lganda : Buferlar ma'lum hajmga ega. Bufer to'lgandan keyin u avtomatik ravishda sinxronlashtiriladi.
- **Aniq, manipulyatorlar bilan:** Ba'zi manipulyatorlardan oqimlarda foydalanilganda, aniq sinxronizatsiya sodir bo'ladi. Ushbu manipulyatorlar quyidagilar: [flush](#) va [endl](#).
- **Shubhasiz, sync() a'zo funktsiyasi bilan:** Oqimning a'zo funktsiyasi `sync()` ni chaqirish darhol sinxronizatsiyaga olib keladi. Agar oqim buferga tegishli bo'lmasa yoki ishlamay qolsa, bu funktsiya -1 ga teng int qiymatini qaytaradi. Aks holda (agar oqim buferi muvaffaqiyatli sinxronlangan bo'lsa) 0 ni qaytaradi.

Matnli Fayl va Binar Fayl o'rtasidagi farqi

Matnli fayl	Ikkilik fayl
Bitlar xarakterini anglatadi.	Bitlar odatiy ma'lumotlarni anglatadi.
Fayl ochilishi bilanoq o'zgartirishlar aks etadi va osongina qaytarib olinishi mumkinligi sababli buzilish xavfi kamroq.	Osonlik bilan buzilib ketishi mumkin, hatto bitta bit o'zgarishi ham faylni buzishi mumkin.
Faqat oddiy matnni faylda saqlash mumkin.	Turli xil ma'lumotlarni (rasm, audio, matn) bitta faylda saqlashi mumkin.
Keng tarqalgan fayl formati va har qanday oddiy matn muharriri yordamida ochilishi mumkin.	Aniq dastur uchun ishlab chiqilgan va uni boshqa ilovalar tushunmasligi mumkin.
Ko'pincha .txt va .rtf matnli fayllarga kengaytma sifatida ishlatiladi.	Ha bir ilova o'zining aniqlangan kengaytmasiga ega bo'lishi mumkin.

C++ da ikkilik fayllar ustida asosiy operatsiyalar

Ushbu dastur ikkilik fayllarni qanday yaratishni va shuningdek, ikkilik fayllardan ma'lumotlarni qanday o'qish, yozish, qidirish, o'chirish va o'zgartirish usullarini tushuntiradi.

```
#include<iostream>
#include<fstream>
#include<cstdio>
using namespace std;

class Talaba
{
    int admno;
    char name[50];
public:
```

```

void setData()
{ cout << "\nKirish raqamini kiriting ";
cin >> admno;
cin.ignore(1000, '\n');
cout << "Talaba ismini kiriting ";
cin.getline(name, 50); }

void showData()
{ cout << "\nKirish raqami : " << admno;
cout << "\nTalaba ismi : " << name; }
int retAdmno()
{ return admno; }
};

// ikkilik faylga yozish funksiyasi.
void write_record()
{ ofstream outFile;
outFile.open("talaba.dat", ios::binary | ios::app);
Talaba obj;
obj.setData();
outFile.write((char*)&obj, sizeof(obj));
outFile.close();
}

// fayl yozuvlarini ko'rsatish funksiyasi
void display()
{ ifstream inFile;
inFile.open("talaba.dat", ios::binary);
Talaba obj;
while(inFile.read((char*)&obj, sizeof(obj)))
{ obj.showData(); }
inFile.close();
}

//ikkilik fayldan qidirish va namoyish qilish funktsiyasi
void search(int n)
{ ifstream inFile;
inFile.open("talaba.dat", ios::binary);
Talaba obj;
while(inFile.read((char*)&obj, sizeof(obj)))
{ if(obj.retAdmno() == n)
{ obj.showData(); }
}
inFile.close();
}

// yozuvni o'chirish funktsiyasi

```

```

void delete_record(int n)
{ Talaba obj;
  ifstream inFile;
  inFile.open("talaba.dat", ios::binary);
  ofstream outFile;
  outFile.open("vaqtincha.dat", ios::out | ios::binary);

  while(inFile.read((char*)&obj, sizeof(obj)))
  { if(obj.retAdmno() != n)
    { outFile.write((char*)&obj, sizeof(obj)); }
  }
  inFile.close();
  outFile.close();
  remove("talaba.dat");
  rename("vaqtincha.dat", "talaba.dat");
}

// yozuvni o'zgartirish uchun funksiya
void modify_record(int n)
{ fstream file;
  file.open("talaba.dat", ios::in | ios::out);
  Talaba obj;
  while(file.read((char*)&obj, sizeof(obj)))
  { if(obj.retAdmno() == n)
    { cout << "\nTalaba haqidagi yangi ma'lumotlarni kiriting";
      obj.setData();
      int pos = -1 * sizeof(obj);
      file.seekp(pos, ios::cur);
      file.write((char*)&obj, sizeof(obj));
    }
  }
  file.close();
}

int main()
{ //4 ta yozuvni faylda saqlash
  for(int i = 1; i <= 4; i++)
    write_record();
  //Barcha yozuvlarni ko'rsatish
  cout << "\nYozuvlar ro'yxati";
  display();
  //Yozuvni qidirish
  cout << "\nQidiruv natijasi";
  search(100);
  //Yozuvni o'chirish
  delete_record(100);
  cout << "\nYozuv o'chirildi";
  //Yozuvni o'zgartirish

```



```
cout << "\n101 yozuvini o'zgartirish ";  
modify_record(101);  
return 0;}
```

Fayl ustida amal bajaruvchi turli misollardan namunalar:

C ++ tilidagi matnli fayllar ustida ishlash uchun misollar

Matn faylidan o'qish va uni namoyish etish uchun dastur

```
//Matn faylidan o'qish va uni namoyish etish uchun dastur  
#include<fstream>  
#include<iostream>  
using namespace std;  
  
int main()  
{ ifstream fin;  
  fin.open("yozish.txt");  
  char ch;  
  while(!fin.eof())  
  { fin.get(ch);  
    cout << ch; }  
  fin.close();  
  return 0;  
}
```

Fayl tarkibini boshqa faylga nusxalash uchun dastur.

```
//Fayl tarkibini boshqa faylga nusxalash uchun dastur.  
#include<iostream>  
#include<fstream>  
using namespace std;  
  
int main()  
{ ifstream fin;  
  fin.open("yozish.txt");  
  ofstream fout;  
  fout.open("nusxafayl.txt");  
  char ch;  
  while(!fin.eof())  
  { fin.get(ch);  
    fout << ch;  
  }  
  fin.close();  
  fout.close();  
  cout<<"Nusxa olish jarayoni tugadi!"<<endl;  
  return 0;  
}
```

ISTISNO QILINADIGAN HOLATLARNI QAYTA ISHLASH

Xatoliklarni qayta ishlashning umumiy mexanizmi

Ayrim hollarda dastur kutilmagan bir qator vaziyatlar ro'y berganda o'z vazifasini bajara olmay qolishi mumkin. Nolga bo'linish, mavjud bo'lmagan xotira qismiga murojaat qilish mumkin bo'lgan ana shunday holatlardan sanaladi.

Istisno qilinadigan holatlar (sodda qilib xatoliklar deb aytish mumkin) – bajarilishi jarayonida oldindan kutilmagan holatlar yuzaga kelganda dastur o'z ishini davom ettirishini ta'minlashdan iborat.

C++ tili istisno qilinadigan holatlar sodir bo'lganda dastur o'zini qanday tutishini belgilab qo'yish uchun dasturchilarga bir qator vositalarni taklif qiladi.

Ma'lumki, *xatoliklar ikki turga* bo'linadi: *kompilyatsiya vaqtidagi* va *dasturni bajarish vaqtidagi* xatoliklar. Odatda 1 -turdagi xatoliklarni kompilyator aniqlab beradi, 2-tur xatoliklarni esa faqat dastur bajarilayotgan vaqtda aniqlash mumkin, xolos. 2-tur xatoliklar yuzaga kelganda dastur o'z ishini to'xtatib qo'yadi. Dasturchi buning oldini olishi, ya'ni dastur har qanday olda ham o'z ishini davom ettirib, kutilgan natijani berishini ta'minlashi lozim. Boshqacha aytganda, bajarish vaqtida yuzaga kelishi mumkin bo'lgan turli xatoliklarni nazarda tutishi va ularning har biri sodir bo'lganda dasturning javob reaksiyasini belgilab qo'yishi talab qilinadi.

Dastur ishlab chiqish jarayonida dasturchi xatoliklar yuzaga kelishi mumkin bo'lgan barcha holatlarni nazorat qilishi lozim. Bunday holatlar *try* bilan boshlanadigan nazorat bloklarida qayta ishlanadi.

Istisno qilinadigan holatlarni qayta ishlash xato yuzaga kelganidan keyin boshlanadi. Bu jarayonni tashkil qilish uchun *throw* operatoridan foydalaniladi.

catch - dasturdagi muammoni hal qilmoqchi bo'lgan joyda dastur istisno tutish vositasi yordamida istisnoni tutadi. Catch kalit so'zi istisnolarni tutishni anglatadi.

Dasturning javob reaksiyasi xatolikni qayta ishlagichlar yordamida ta'minlanadi. Agar dasturning ularga mos javob reaksiyasi belgilanmagan bo'lsa, standart *terminate* funksiyasi ishga tushadi va u hisoblash jarayonini to'xtatish uchun *abort* funksiyasini chaqiradi.

Dasturchi mana shunday hollarda jarayonni to'xtatish uchun shaxsiy funksiyalarini ishlab chiqishi mumkin.

Istisno qilinadigan holatlar sintaksisi

Istisno qilinadigan holatlarni nazorat qiluvchi blok *try* so'zi bilan boshlanadi va figurali qavslar orasida yoziladi.

```
try
{
...
}
```

C++ try/catch

C++ dasturlashda istisnolarni ishlatish *try / catch* ifodasi yordamida amalga oshiriladi. Istisno yuzaga kelishi mumkin bo'lgan kodni joylashtirish uchun C++ *try* blokidan foydalaniladi. Catchblok istisnolarni bajarish uchun ishlatiladi.

C++ try/catch dan foydalanilmagan holatga misol

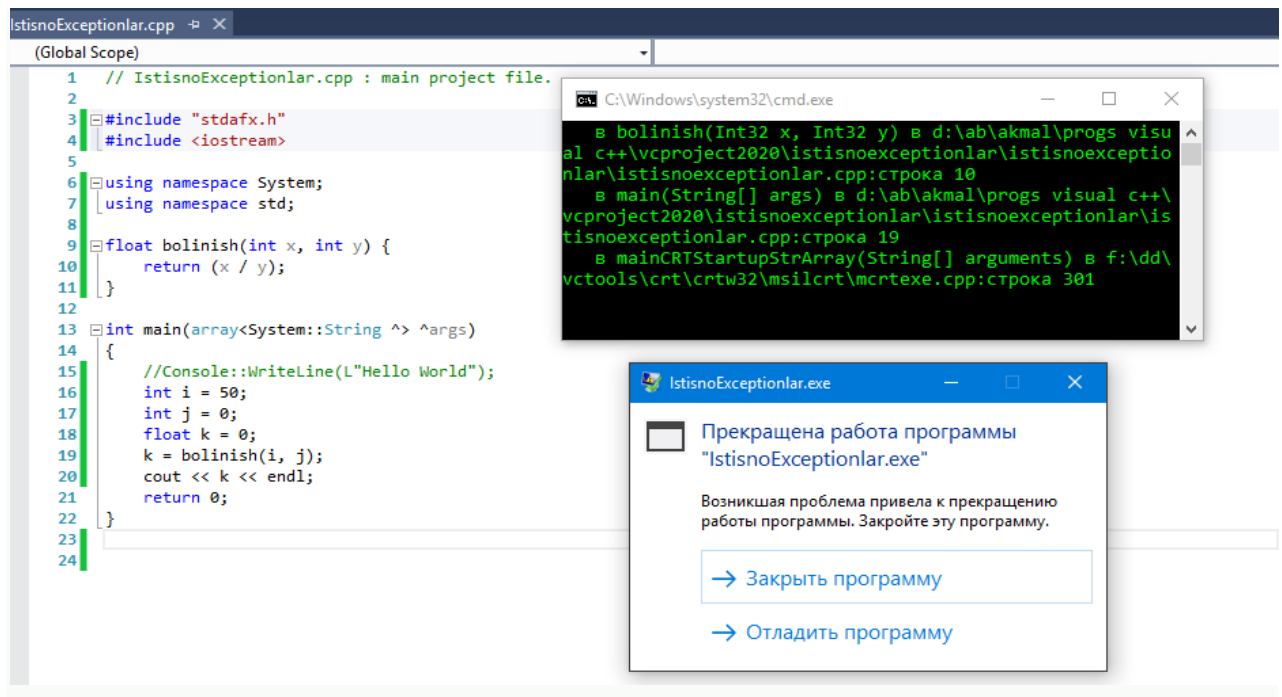
1. `#include <iostream>`
2. `using namespace std;`
3. `float bolinish(int x, int y) {`
4. `return (x/y);`
5. `}`

```

6.  int main () {
7.  int i = 50;
8.  int j = 0;
9.  float k = 0;
10. k = bolinish (i, j);
11. cout << k << endl;
12. return 0;
13. }

```

Chiqish natijasi:



C ++ try/catch misoli

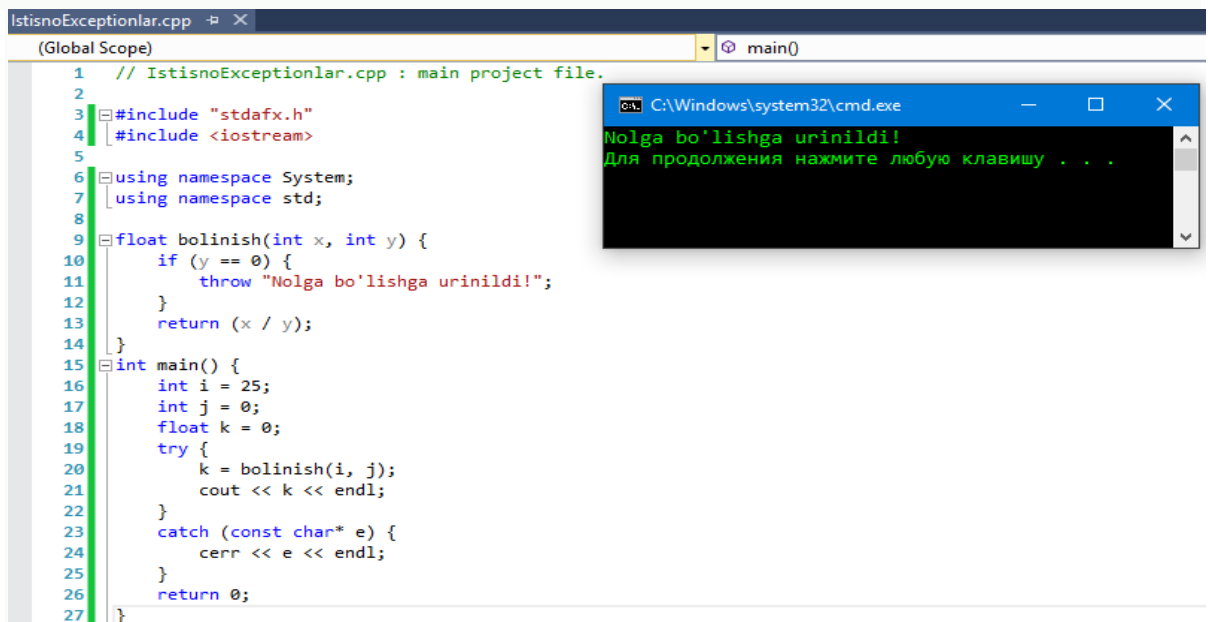
```

1.  #include <iostream>
2.  using namespace std;
3.  float bolinish (int x, int y) {
4.  if( y == 0 ) {
5.  throw "Nolga bo'lishga urinildi!";
6.  return (x/y);
7.  }
8.  int main () {
9.  int i = 25;
10. int j = 0;
11. float k = 0;
12. try {
13. k = bolinish (i, j);
14. cout << k << endl;
15. } catch (const char* e) {
16. cerr << e << endl;
17. }
18. return 0;
19. }

```

Chiqish natijasi:

Nolga bo'lishga urinildi!



```
IstisnoExceptionlar.cpp -# x
(Global Scope) main()
1 // IstisnoExceptionlar.cpp : main project file.
2
3 #include "stdafx.h"
4 #include <iostream>
5
6 using namespace System;
7 using namespace std;
8
9 float bolinish(int x, int y) {
10     if (y == 0) {
11         throw "Nolga bo'lishga urinildi!";
12     }
13     return (x / y);
14 }
15
16 int main() {
17     int i = 25;
18     int j = 0;
19     float k = 0;
20     try {
21         k = bolinish(i, j);
22         cout << k << endl;
23     }
24     catch (const char* e) {
25         cerr << e << endl;
26     }
27     return 0;
28 }
```

```
C:\Windows\system32\cmd.exe
Nolga bo'lishga urinildi!
Для продолжения нажмите любую клавишу . . .
```

Istisno qilinadigan holatlarni aniqlash *throw* xizmatchi so'zi yordamida amalga oshiriladi:

throw [ifoda];

throw dan keyin ko'rsatilgan ifodaning tipi xatolik tipini aniqlab beradi. Xatolikni qayd etish vaqtida joriy blokni bajarish jarayoni to'xtatiladi va boshqaruv unga mos qayta ishlagichga uzatiladi.

Yuzaga kelgan xatolikni to'g'ri qayta ishlash har doim ham mumkin bo'lavermaydi. Ayrim hollarda bir nazorat blogi ikkinchisining ichida kelishi mumkin. Bunday holda boshqaruv parametrsiz *throw* yordamida tashqi qayta ishlagichlarga uzatiladi.

Xatoliklarni qayta ishlagichlar *catch* xizmatchi so'zi bilan boshlanib, qavslar ichida istisno qilinadigan holatlar tipi ko'rsatiladi. Ular bevosita *try* blogidan keyin yoziladi. Qayta ishlanayotgan xatoliklarning tipiga mos ravishda bir yoki bir nechta qayta ishlagichlardan foydalanish mumkin.

Qayta ishlagichlarni umumiy holda quyidagi ko'rinishlardan birida yozish mumkin:

catch (tip nom){ ... /* qayta ishlagich */ }

catch (tip){ ... /* qayta ishlagich */ }

catch (...){ ... /* qayta ishlagich */ }

1-shakl parametr nomi qayta ishlagichda qandaydir amallarni bajarishni tashkil qilishga to'g'ri kelganda (masalan, xatolik haqida axborotni ekranga chiqarilganda) qo'llanadi. Ikkinchi shakl esa xatolik haqida axborot berishni nazarda tutmaydi. Uchinchi shakldagi uch nuqta qayta ishlagich hamma xatoliklarni tutib qolishini anglatadi. Masalan:

catch (int i)

{

... // int tipidagi xatolikni qayta ishlash

}

catch (const char *)

{

... // const char* tipidagi xatolikni qayta ishlash

}

catch (overflow)

{

```

... // Overflow klassidagi xatoliklarni qayta ishlash
}
{
catch(...)
{
... // ko'zda tutilmagan barcha xatoliklarni qayta ishlash
}
}

```

Qayta ishlash tugaganidan so'ng boshqaruv bevosita qayta ishlagichdan keyin ko'rsatilgan birinchi operatorga uzatiladi. *Try* blogida ko'rsatilgan xatoliklar ro'y bermaganda ham boshqaruv aynan shu operatorga o'tadi.

Xatoliklarni tutib qolish

Throw operatori yordamida istisno qilinadigan holatlar qayd qilinganda C++ quyidagi amallarni bajaradi:

1) *throw* parametrini statik obyekt sifatida nusxasini oladi va istisno qilinadigan holatlar qayta ishlanmaguncha saqlab turadi;

2) mos qayta ishlagichni qidirib, steklarni aylantirib ko'rib chiqadi va amal qilish doirasidan chetga chiqqan lokal obyektlarning destruktorelarini chaqiradi;

3) boshqaruvni shu obyekt bilan tipi bir xil bo'lgan parametrli qayta ishlagichga uzatadi.

Qayta ishlagich topilgan hisoblanadi. agar *throw* dan keyin ko'rsatilgan obyektning tipi:

a) *catch* ning parametrda ko'rsatilgan bo'lsa (parametr T. const T, T& yoki const T& shaklida yozilgan bo'lishi mumkin. Bu yerda T - xatolik tipi);

b) *catch* parametrining hosilasi bo'lsa (agar vorislik public kaliti bilan hosil qilingan bo'lsa);

c) tipini standart tiplarni almashtirish qoidalari yordamida *catch* parametridagi ko'rsatkich tipiga keltirish mumkin bo'lgan ko'rsatkichlar.

Ko'rinib turibdiki, klass hosilalari qayta ishlagichlarini bazaviy qayta ishlagichlardan oldinroq joylashtirish lozim, aks holda boshqaruv hech qachon ularga o'tmaydi.

Void tipidagi ko'rsatkichlarni qayta ishlagichlar to'g'ridan-to'g'ri boshqa tipdagi ko'rsatkichlarni to'sib qo'yadi va shu sababli uni barcha konkret tipli qayta ishlagichlardan keyin ko'rsatish lozim.

Quyidagi dasturga e'tibor bering.

```

#include <iostream.h>
class Hello
{
// o 'zining yo 'qotilgani haqida axborot beruvchi klass
public:
Hello(){cout << "Hello!" << endl;}
~Hello(){cout << "Bye!" << endl;}
};
void f1()
{
ifstream ifs("\\INVALID\\ FILE 'NAME '"); //Faylni ochamiz
if (!ifs) {
cout << "xatolikni qayd etamiz << endl;
throw "Fayli ochishdagi xatolik ";}
}
void f2()
{

```

```

        Hello H; // Lokal obyekt yaratilmoqda
    fl(); //xatolikni yuzaga keltiruvchi funksiya chaqirilmoqda
}

int ra in ()
{
    try
    {
        cout<< " try-blokka kirish " << endl;
        f2();
        cout << "try-blokdan chiqish " << endl;
    }
    catch(int i)
    {
        cout<< "int istisnoni qayta ishlagich chaqirildi- " <<i<< endl;
return -1;
    }
    catch (const char * p)
    {
        cout <<"const char* istisnoni qayta ishlagich chaqirildi-" << p << endl;
        return -1;
    }
    catch(...)
    {
        cout << "Barcha istisnolarni qayta ishlagich chaqirildi —" << endl;
return -1;
    }
    return 0; // Hammasi yaxshilik bilan tugadi
}
Ushbu dastur quyidagi natijani beradi:
Try - blokka kirish

```

Hello!

Istisnoni qayd qilamiz

Bye!

**Const char* istisnoni qayta ishlagich chaqirildi Faylni
ochishdagi xatolik**

E'tibor bering, xatolik yuz berganidan keyin lokal obyektning destruktori chaqirildi, ammo bu vaqtda boshqaruv fl dan main funksiyasida turgan qayta ishlagichga uzatildi. 'Try-blokdan chiqish' axboroti ekranga chiqarilmadi. Dasturda fayllar bilan ishlash uchun oqimlardan foydalanildi.

Shunday qilib, istisnolarni qayta ishlash mexanizmi xatoliklar yuz berganda obyektlarni yo'qotishi mumkin. Shuning uchun resurslarni ajratish va bo'shatish amalini klasslar ko'rinishida (konstruktor tashkil qiladi, destruktor esa bo'shatadi) tashkil qilish maqsadga muvofiq hisoblanadi. Misol tariqasida fayllar bilan ishlash uchun klassni keltirish mumkin. Bu klassning konstruktori faylni ochadi, destruktor esa yopadi. Albatta bu holda xatolik yuz berganda faylni yopish to'g'ri tashkil qilinadi va undagi ma'lumotlar yo'qolmaydi.

Ta'kidlab o'tilganidek, istisno qilinadigan holatlar standart tipda ham, foydalanuvchi aniqlagan tipda ham bo'lishi mumkin. Bunday holatlarda bu tipni global e'lon qilish shart emas va xatolikni qayd qilish hamda ularni qayta ishlash vaqtida ma'lum bo'lsa yetarli.

Istisno qilinadigan holatlarni ifodalovchi klasslarni istisnolarni qayta ishlash vaqtida yuzaga kelishi mumkin bo'lgan klasslar ichida e'lon qilish mumkin. Bu klassning ko'chirish konstruktori public tarzida e'lon qilinishi shart, aks holda xatolikni qayd qilish vaqtida obyektning nusxasini yaratish mumkin bo'lmay qoladi.

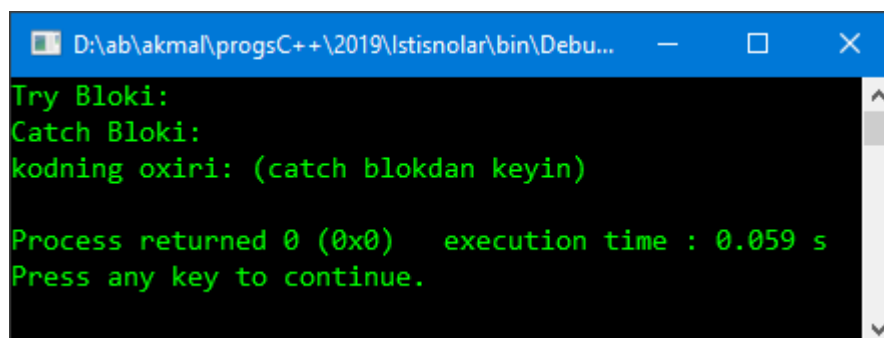
Sintaksis:

Try bloki ichidagi kod bajariladi. Agar biron bir xato yuzaga kelsa, unda **throw** kalit so'zi istisnoni qayta ishlov beruvchiga, ya'ni **catch** blokiga tashlaydi. Keyinchalik **catch** bloki blokni ichida joylashgan kodni bajaradi va shu bilan istisnolarni ko'rib chiqadi.

C++ da exception ishlashi uchun namunaviy kodni ko'rib chiqamiz:

```
#include <iostream>
using namespace std;
int main()
{
    int x = 1;
    try{
        cout << "Try Bloki: "<<endl;
        if(x < 10)
        {
            throw x;
        }
    }
    catch (int x ) {
        cout << "Catch Bloki: "<<endl;
    }
    cout<<"kodning oxiri: (catch blokdan keyin) "<<endl;
    return 0;
}
```

Chiqish natijasi:



```
D:\ab\akmal\progsC++\2019\Istisnolar\bin\Debu...
Try Bloki:
Catch Bloki:
kodning oxiri: (catch blokdan keyin)

Process returned 0 (0x0)   execution time : 0.059 s
Press any key to continue.
```

Izoh

Ushbu dasturda istisno bilan ishlashni ko'rsatilgan. Bizda x o'zgaruvchisi mavjud bo'lib, unga 1 qiymati berilgan, keying qatorda **try** bloki boshlangan. Ushbu blokda x < 10 shartni tekshiruvchi if operatorimiz bor.

Bizning holatimizda shart rost, chunki x=1. Keyin dastur istisno qaytaradi va boshqaruv catch blokiga o'tadi. Biz shartni catch qismida bajaramiz va blokdan chiqamiz.

```
1 catch (...) {
```

```
2 cout << "Odatiy Istisno"<<endl;
3 }
```

Mumkin bo'lgan istisnolar soniga qarab, catchning bir nechta versiyasi bo'lishi mumkin. C++ tilida ushbu istisnolarga oid holatlarni ko'rib chiqamiz.

catch blokining bajarilmay qolishi

Oldingi dasturni ko'rib chiqamiz, agar x qiymati o'rniga "ABC" so'zi qo'yilsa, catch funksiyasi uni bajara olmaydi. Bunday holda ekranda xato xabari ko'rsatilishi mumkin.

Bunday muammolarni hal qilish uchun biz kodga *odatiy(default) catch* funksiyasini qo'shishimiz kerak.

```
#include <iostream>
using namespace std;
int main()
{
    int x = 1;
    try {
        cout << "Try Bloki: "<<endl;
        if(x < 10)
        {
            throw "ABC";
        }
    }
    catch (int x ) {
        cout << "Catch Bloki: n"<<endl;
    }
    catch(...){//Odatiy (default) catch
        cout << "Odatiy Istisno"<<endl;
    }
    return 0;
}
```

Natija:

Izoh:

Ushbu kod avvalgisiga o'xshash. Faqatgina o'zgarish shundaki, istisnolar char tipiga tegishli. Bu bizning catch funksiyamiz foydasiz bo'lishiga olib keladi. Shunday qilib, biz odatiy catch funksiyasini qo'shdik.

Agar catch iboralarining hech biri mos kelmasa, u holda odatiy catch bajariladi.

Bir nechta catch bloklari

Bitta try blokining bir nechta catch bloklari bo'lishi mumkin.

Mana, misol,

```
#include <iostream>
```

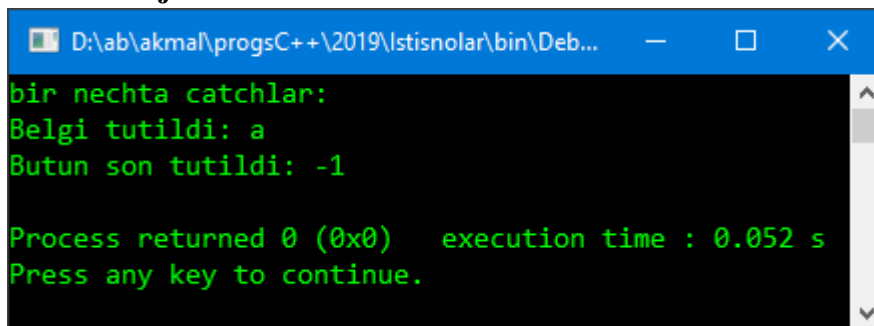


```

using namespace std;
int test(int a) {
    try{
        if(a < 0)
            throw a;
        else
            throw 'a';
    }catch(int a){
        cout<<"Butun son tutildi: " << a<<endl;
    }catch (char a){
        cout<<"Belgi tutildi: " << a<<endl;
    }
    return 0;
}
int main() {
    cout<<"bir nechta catchlar:"<<endl;
    test(10);
    test(-1);
    return 0;
}

```

Natija:



```

D:\ab\akmal\progsC++\2019\Istisnolar\bin\Deb...
bir nechta catchlar:
Belgi tutildi: a
Butun son tutildi: -1

Process returned 0 (0x0)   execution time : 0.052 s
Press any key to continue.

```

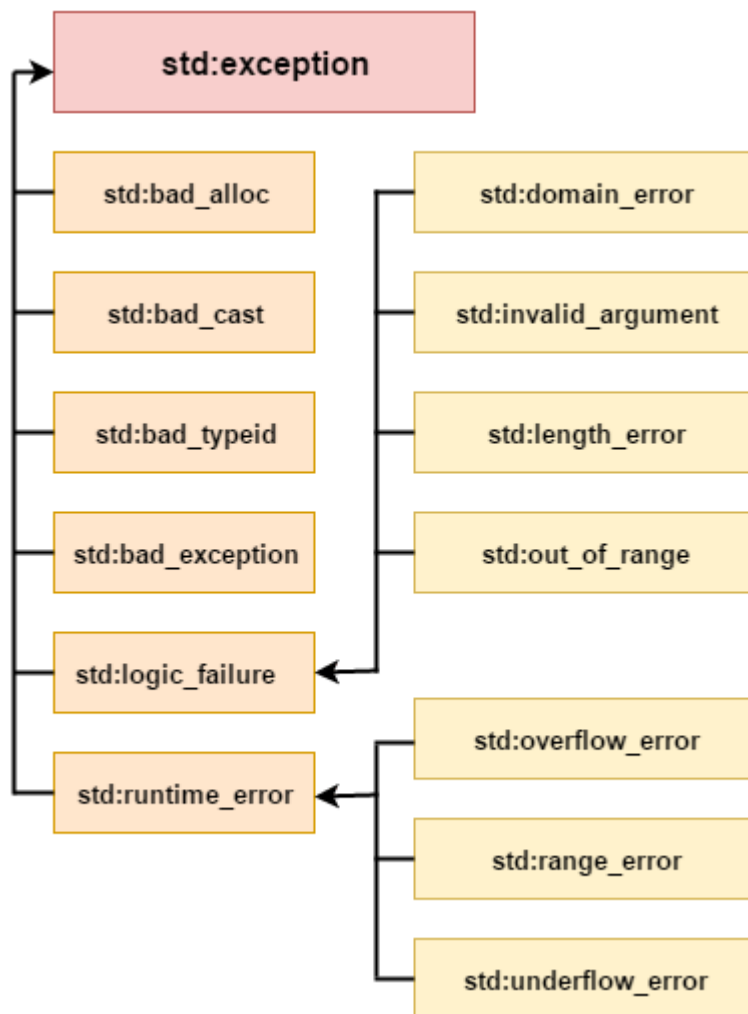
Izoh:

Yuqoridagi kodda biz *catch*ning bir nechta variantlaridan foydalanamiz. Bizda istisnoni keltirib chiqaradigan test funksiyasi mavjud. Birinchi test holatida qiymat 10 ga teng, 'a' belgi tashlanadi, chunki 10 noldan katta va ikkinchi *catch* funksiyasi bilan ushlanadi.

Ikkinchi holda, qiymat 0 dan kichik, shuning uchun -1 qiymati tashlanadi va u butun istisno bilan tutiladi.

C ++ istisno(exception) sinflar

C ++ tilida standart istisnolar <exception> sinfida aniqlanadi, biz ularni o'z dasturlarimiz ichida ishlatishimiz mumkin. Asosiy(parent)-avlod(child) sinflari ierarxiyasining tartibi quyida ko'rsatilgan:



C ++ tilidagi barcha istisnolar sinflari `std :: exception` sinfidan meros qilib olingan. Keling, C ++ istisno sinflarining umumiy ro'yxatini ko'rib chiqamiz.

Istisno	Ta'rif
<code>std :: exception</code>	Bu barcha standart C ++ istisnolarining istisno va asosiy sinfidir.
<code>std :: logic_failure</code>	Kodni o'qish orqali aniqlanishi mumkin bo'lgan istisno.
<code>std :: runtime_error</code>	Kodni o'qish orqali aniqlab bo'lmaydigan istisno.
<code>std :: bad_exception</code>	U c ++ dasturida kutilmagan istisnolarni qayta ishlash uchun ishlatiladi.
<code>std :: bad_cast</code>	Ushbu istisno odatda dynamic_cast tomonidan sodir bo'ladi .
<code>std :: bad_typeid</code>	Ushbu istisno odatda typeid tomonidan sodir bo'ladi .
<code>std :: bad_alloc</code>	Ushbu istisno odatda new tomonidan sodir bo'ladi .

Konstruktor va destruktordagi istisnolar

C++ tili konstruktor va destruktordan qiymat qaytarishda foydalanishga ruxsat bermaydi. Istisnolarni qayta ishlash mexanizmi obyektning konstruktori yoki destruktorida yuzaga kelgan xatolik haqida axborot berishi mumkin. Bu fikrni namoyish qilish uchun Vector klassini tashkil qilamiz. Unda soʻraladigan xotira hajmi cheklanadi.

```
class Vector{
public:
    class Size{};          // istisno klassi
    enum {max = 32000} : // vektorning maksimal uzunligi
    Vector(int n)          //konstruktor
    {if (n<0 || n>max) throw Size():... }
};
Vector klassidan foydalanganda Size tipidagi xatoliklarni kuzatish mumkin:
try{
    Vector *p = new Vector(i);
}
Catch( Vector: :Size)(
... // vector o 'lchami bilan bog 'liq xatolikni qayta ishlash
}
```

Qayta ishlagichda xatolik haqida axborot berish va qayta tiklashning asosiy usullaridan foydalanish mumkin. Istisnoni aniqlovchi klass ichida qayta ishlagichga uzatiladigan istisno haqidagi axborotni ham saqlashga ruxsat beriladi. Buning maʼnosi istisno qilinadigan holat aniqlangan nuqtadan xatolik haqidagi axborotni qayta ishlagich yetarlicha imkoniyatga ega boʻlgan joyga uzatishni taʼminlashdan iborat.

Agar obyekt konstruktorida istisno qayd qilinsa, avtomatik tarzda joriy vaqtgacha shu blokda toʻla yaratilgan obyektlar hamda joriy obyektning maydonlari uchun destruktur chaqiriladi. Masalan, agar istisno obyektlar massivini yaratishda yuzaga kelsa, destruktorga faqat muvaffaqiyatli yaratilgan elementlar uchun ishga tushadi.

Agar obyekt dinamik xotirada *new* yordamida yaratilayotgan boʻlib, konstruktorda xatolik roʻy bersa, bu obyekt band qilgan xotira qismi boʻshatiladi.

Standart tipdagi istisnolarga qaraganda shaxsiy tipdagi istisnolarni qayta ishlash afzal sanaladi. Istisnolarni qayta ishlashga qaraganda klasslar yordamida istisnolar haqidagi axborotlarni uzatish osonroq. Bundan tashqari, klasslar shajarasidan foydalanishga imkon paydo boʻladi.

Istisnolarni boshqarish mexanizmi bazaviy klasslar uchun shaxsiy qayta ishlagichlarni yaratishga imkon beradi, qardosh (bir-biriga yaqin) istisnolarni koʻpincha shajaralar koʻrinishida yaratish mumkin boʻladi. Umumiy bazaviy klassdan istisnolarni keltirib chiqarishda polimorfizm prinsipidan foydalanib qayta ishlagichda bazaviy klassga koʻrsatkichlarni tutib qolish mumkin boʻladi. Masalan, matematik kutubxonada klasslarni quyidagicha tashkil qilish mumkin:

```
class Matherr{};
class Overflow: public Matherr{}; //xotiraning to 'lib ketishi
```

```
class Underflow: public Matherr{}; // tartibning yo 'qolishi
```

```
class ZeroDivide: public Matherr{}; // nolga bo 'linish
```

Ma'lumotlarni kiritish va chiqarish bilan bog'liq xatoliklarni ifodalash uchun quyidagi klasslardan foydalanish mumkin:

```
class Ioegg{}:
```

```
class Readerr: public Ioegg{}; //o'qishdagi xatolik
```

```
class Writerr: public Ioegg{}; //yozishdagi xatolik
```

```
class Seekerr: public Ioegg{}; // qidirishdagi xatolik
```

Vaziyatga bog'liq ravishda yoki hosila istisnolarni ham tutib qola oladigan bazaviy klass qayta ishlagichidan yoki shaxsiy hosila klass qayta ishlagichlaridan foydalanish mumkin.

Afzalligi

Ilovaning normal oqimini saqlaydi. Bunday holda, kodning qolgan qismi istisnolardan keyin ham bajariladi.

C++ foydalanuvchi belgilaydigan istisnolar

Istisno klasi funktsional imkoniyatlarini bekor qilish va meros qilib olish bilan yangi istisno aniqlanishi mumkin .

C++ foydalanuvchi tomonidan belgilangan istisno misoli

Istisnoni aniqlash uchun `std::exception` klasi foydalanadigan foydalanuvchi tomonidan belgilangan istisnoning oddiy misolini ko'rib chiqamiz .

```
#include <iostream>
#include <exception>
using namespace std;
class UzIstisno : public exception{
public:
    const char * what() const throw()
    {
        return "Nolga bo'lishga urinish!\n";
    }
};
int main()
{
    try
    {
        int x, y;
        cout << "Ikkita raqam kiriting : \n";
        cin >> x >> y;
        if (y == 0)
        {
            UzIstisno z;
            throw z;
        }
        else
        {
            cout << "x / y = " << x/y << endl;
        }
    }
    catch(exception& e)
    {

```

```
cout << e.what();
}
}
```

Chiqish natijasi:

Ikkita raqam kiriting:

10

2

$x / y = 5$

Chiqish natijasi:

Ikkita raqam kiriting:

10

0

Nolga bo'lishga urinish!

```
(Global Scope)
3 #include "stdafx.h"
4 #include <iostream>
5 #include <exception>
6
7 using namespace System;
8 using namespace std;
9
10 class Uzistisno : public exception{
11 public:
12     const char *what() const throw()
13     {
14         return " Nolga bo'lishga urinish!\n";
15     }
16 };
17
18 int main()
19 {
20     try
21     {
22         int x, y;
23         cout << "Ikkita raqam kiriting : \n";
24         cin >> x >> y;
25         if (y == 0)
26         {
27             Uzistisno z;
28             throw z;
29         }
30         else
31         {
32             cout << "x / y = " << x / y << endl;
33         }
34     }
35     catch (exception& e)
36     {
37         cout << e.what();
38     }
39     system("PAUSE");
40 }
```

```
D:\ab\akmal\progs visual c++\VCproject2020\Isti...
Ikkita raqam kiriting :
5
0
Nolga bo'lishga urinish!
Для продолжения нажмите любую клавишу . . .
```

Eslatma: Yuqoridagi misolda what() exception klasi tomonidan taqdim etilgan public metod. Istisno sababini qaytarish uchun ishlatiladi.

Nazorat savollari

1. Fayl nima?
2. Faylning qanday turlarini bilasiz?
3. Fayllar qanday maqsadlarda ochiladi?
4. Matnli fayllar qanday fayllar?
5. Binar fayllar qanday fayllar?
6. Ma'lumotlar oqimini qanday tashkil qilish mumkin?
7. Faylga ma'lumotlar oqimi qanday ko'rsatma bilan yoziladi?
8. Faylning oxirini aniqlash mumkinmi?
9. Fayl bilan ishlovchi qanday sinflarni bilasiz?
10. fstream sinfi qanday sinf?
11. ostream sinfi qanday sinf?
12. ifstream sinfi qanday sinf?
13. Oqim argumentlari haqida nimalarni bilasiz?
14. ios::ate ofstream uchun argument sifatida nimani anglatadi?
15. eof()funksiyasi qanday funksiya?

16. bad()funksiyasi qanday funksiya?
17. ios::beg qiymati nima?
18. ios::cur qiymati nima?
19. ios::end qiymati nima?
20. Istisnolar nima?

12- MA'RUZA

MAVZU: SINIF XUSUSIYATI VA VORISLIK

Reja:

1. Meros ierarxiyasi;
2. Mavjud sinflarni joriy etish;
3. Komponentlik funksiya;
4. Virtual funksiyalar va polimorfizm.

Annotatsiya: Mazkur ma'ruzada obyektga yo'naltirilgan dasturlashning asosiy tushunchalariga qisqacha to'xtalib o'tilgan bo'lib, asosiy urg'u inkapsulyatsiya va merosho'rlik (vorislik) mavzulariga qaratilgan. Inkapsulyatsiya – bu berilganlar va ularni qayta ishlovchi kodni birlashtirish mexanizmi hisoblanib, u berilganlar va kodni tashqi ta'sirdan saqlash imkonini berishini nazariy ma'lumotlar va amaliy misollar yordamida yoritilgan. Vorislik – ham OYD ning asosiy ustunlaridan biri ekanligi va vorislik sinflarda ierarxik ko'rinishdagi sinflanishni ta'minlashi nazariy ma'lumotlar va amaliy misollar bilan asoslangan.

Kalit so'zlar: Inkapsulyatsiya, merosxo'rlik, vorislik, public, private, protected, ma'lumotlarni abstraksiyalash, konstruktor, destruktor, asos sinf, voris sinf, yakka vorislik, to'plamli vorislik

Inkapsulyatsiya

Agarda muhandis ishlab chiqarish jarayonida rezistorni qo'llasa, u buni yangidan ixtiro qilmaydi, omborga (magazinga) borib, mos parametrlarga muvofiq kerakli detalni tanlaydi. Bu holda muhandis joriy rezistor qanday tuzilganligiga e'tiborini qaratmaydi, rezistor faqatgina zavod xarakteristikalariga muvofiq ishlasa yetarlidir. Aynan, shu tashqi konstruksiyada qo'llaniladigan yashirinlik yoki obyektning yashirinligi yoki avtonomligi xossasi inkapsulyatsiya deyiladi. Inkapsulyatsiya yordamida berilganlarni yashirish ta'minlanadi. Bu juda yaxshi xarakteristika bo'lib foydalanuvchi o'zi ishlatayotgan obyektning ichki ishlari haqida umuman o'ylamaydi. Haqiqatan ham, xolodilnikni ishlatishda refrijeratorni ishlash tamoyilini bilish shart emas. Yaxshi ishlab chiqilgan dastur obyektini qo'llashda uning ichki o'zgaruvchilarining o'zaro munosabati haqida qayg'urish zarur emas.

Yana bir marta takrorlash joizki, rezistorni samarali qo'llash uchun uning ishlash tamoyili va ichki qurilmalari haqidagi ma'lumotlarni bilish umuman shart emas. Rezistorning barcha xususiyatlari inkapsulyatsiya qilingan, ya'ni yashirilgan. Rezistor faqatgina o'z funksiyasini bajarishi yetarlidir.

C++ tilida inkapsulyatsiya tamoyili sinf deb ataluvchi nostandart tiplarni(foydalanuvchi tiplarini) hosil qilish orqali himoya qilinadi.

To'g'ri aniqlangan sinf obyektini butun dasturiy modul sifatida ishlatish mumkin. Haqiqiy sinfning barcha ichki ishlari yashirin bo'lishi lozim. To'g'ri aniqlangan sinfning foydalanuvchilari uning qanday ishlashini bilishi shart emas, ular sinf qanday vazifani bajarishini bilsalar yetarlidir.

Sinf elementini e'lon qilishda bir nechta kalit so'zlardan foydalaniladi: **public, private, protected.**

Umumiy (**public**) komponentalar dasturni ixtiyoriy qismida murojaat xuquqiga ega. Ulardan ixtiyoriy funksiya ushbu sinf ichida va sinf tashqarida foydalansa ham bo'ladi.

Xususiy (**private**) komponentalar sinf ichida murojaat xuquqiga ega, lekin sinf tashqarisidan esa murojaat qilish mumkin emas. Komponentalardan ushbu ular tavsiflangan sinfdagi funksiya - a`zolari yoki "do'stona"- funksiyalar orqali foydalanish mumkin.

Ximoyalangan (**protected**) komponentalar sinf ichida va hosila sinflarda murojaat xuquqiga ega.

Ulardan eng muhimlari **public** (ochiq) va **private** (yopiq) kalit so'zlari bo'lib, ular orqali obyektning a`zolariga murojaat qilish imkoniyati chegaralanadi. Sinfning barcha usullari va xossalari boshlang'ich holda yopiq deb e`lon qilinadi. Yopiq a`zolariga faqatgina shu sinfning usullari orqaligina murojaat qilish mumkin. Obyektning ochiq a`zolariga esa dasturdagi barcha funksiyalar murojaat qilishlari mumkin. Sinf a`zolariga murojaat qilish imkonini belgilash juda muhim xususiyat bo'lib, bu masalani yechishda uncha katta tajribaga ega bo'lmagan dasturlarchilar ko'pincha qiyinchiliklarga duch keladilar. Bu holatni batafsilroq tushuntirish uchun mavzuni boshida keltirilgan masalamizga qaytamiz.

```
class Mushuk {  
    unsigned int itsYosh;  
    unsigned int itsOgirlik;  
    void Miyovlash(); }
```

Bu tarzda sinfni e`lon qilishda itsYosh va itsOgirlik maydonlari ham, Miyovlash () usuli ham yopiq a`zo sifatida aniqlanadi. Dasturda yuqoridagi tartibda Mushuk sinfi e`lon qilingan bo'lsa va bu sinf ekzemplyari bo'lgan obyektning itsYosh a`zosiga main() funksiyaci tanasidan turib murojaat qilsak, kompilyator xatolik ro'y berganligi haqida xabar beradi.

```
Mushuk Baroq;  
Baroq.itsYosh = 5 // Xatolik!  
// Yopiq a`zoga murojaat qilish mumkin emas.
```

Statik elementlar hamda funksiyalar

Shu paytgacha, har bir yaratilgan element o'zining xususiy ma'lumotlar elementiga ega bo'lar edi. Lekin, shunday holat bo'ladiki, bitta sinf doirasidagi obyektlarning ba'zi elementlari o'zaro bog'langan bo'ladi. Masalan, ish vaqti bir xil bo'lgan 1000 ta ishchining oylik maoshini hisoblaydigan dastur tuzish taklif qilinayotgan bo'lsin. Soliq stavkasini aniqlash uchun dastur har bir ishchining sharoitini bilishi kerak. Buning uchun aytaylik, state_of_employee nomli sinfdan foydalanamiz. Agar, ishchilar bir xil sharoitda ishlasa, demak, dastur barcha employee tipidagi obyektlar uchun (barcha ishchilar uchun) ushbu elementlardan o'zaro moslikda foydalanadi. Ushbu holatda dastur, bitta axborotning 999 ta nusxasidan foydalanish bilan xotiradan foydalanish hajmini kamaytiradi.

Sinfning elementidan o'zaro moslikda foydalanish uchun, ushbu element **static** (statik) deb e`lon qilinishi zarur. Agar, dastur ushbu elementga yangi qiymat o'zlashtirsa, hamma obyekt elementi ushbu yangi qiymatni qabul qiladi. Sinf elementi statik deb e`lon qilinganidan so'ng, u umumiy (global) o'zgaruvchi sifatida e`lon qilinishi zarur.

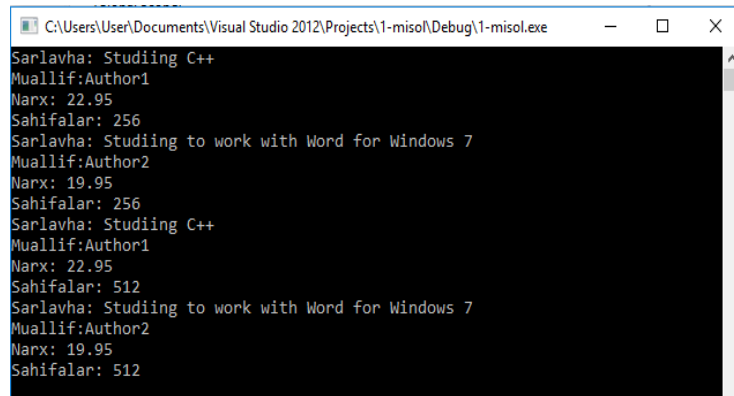
1. **#include "stdafx.h"**
2. **#include <string.h> //strcpy() uchun**
3. **#include <stdio.h> //printf() uchun**
4. **#include <conio.h> //_getch() uchun**
5. **using namespace std;**

```

6.  class book_series{
7.      public:
8.      book_series(char *, char *, float);
9.      void show_book(void); void set_pages(int);
10.     private:
11.     static int page_count; /*bu umumiy element hisoblanadi*/
12.     char title[64]; char author[64];
13.     float price; };
14.     int book_series::page_count; /*sinfdan tashqirda umumiy o'zgaruvchini e'lon qilish*/
15.     void book_series::set_pages(int pages){
16.     page_count = pages; }
17.     book_series::book_series(char *title, char *author, float price){ /*Sinfning
    konstruktori*/
18.     strcpy(book_series::title, title); /*string sinfiga ulanish uchun zarur bo'lgan, strcpy()
    funksiyasi*/
19.     strcpy(book_series::author, author);
20.     book_series::price = price; }
21.     void book_series:: show_book (void){
22.     printf("Sarlavha: %s\n",title); printf("Muallif:%s\n",author);
23.     printf("Narx: %.2f\n",price);
24.     printf("Sahifalar: %d\n",page_count); }
25.     void main(){
26.     book_series programming("Studiing C++", "Author1", 22.95);
/*programming obyektini konstruktor yordamida yaratish*/
27.     book_series word( "Studiing to work with Word for Windows 7", "Author2", 19.95);
/*word obyektini konstruktor yordamida yaratish*/
28.     word.set_pages(256); /*Word o'ektining sahifalari soni beriladi, bu programmingga
    ham ta'sir qiladi */
29.     programming.show_book ();
30.     word.show_book() ;
31.     programming.set_pages(512); /*page_countni o'zgartirish*/
32.     programming.show_book(); /*obyekt ma'lumotlarini ekranga chiqarish*/
33.     word.show_book(); /*obyekt ma'lumotlarini ekranga chiqarish*/
34.     _getch(); }

```

Natijasi:



```
C:\Users\User\Documents\Visual Studio 2012\Projects\1-misol\Debug\1-misol.exe
Sarlavha: Studiing C++
Muallif:Author1
Narx: 22.95
Sahifalar: 256
Sarlavha: Studiing to work with Word for Windows 7
Muallif:Author2
Narx: 19.95
Sahifalar: 256
Sarlavha: Studiing C++
Muallif:Author1
Narx: 22.95
Sahifalar: 512
Sarlavha: Studiing to work with Word for Windows 7
Muallif:Author2
Narx: 19.95
Sahifalar: 512
```

Obyekt mavjud bo'lmaganda, public static atributli elementlardan foydalanish

Sinfning barcha obyektlarida o'zaro moslikda foydalaniladigan, elementi **static** sifatida e'lon qilinishi tushunarli bo'ldi, lekin, shunday holat bo'lishi mumkin: hech qanday obyekt yaratilmagan, ammo, ushbu elementdan foydalanish zarur. Dasturda bu elementdan foydalanish uchun, uni **public** hamda **static** deb e'lon qilish zarur. Ushbu dasturda xuddi shu holatga e'tibor qaratilgan.

Bu holatni ifodalaydigan dasturning kodi quyida ifodalangan:

1. **#include "stdafx.h"**
2. **#include <string.h> //strcpy() uchun**
3. **#include <stdio.h> //printf() uchun**
4. **#include <conio.h> //_getch() uchun**
5. **using namespace std;**
6. **class book_series{**
7. **book_series();**
8. **public:**
9. **static void show_book(void); /*Funksiyani statis elementini chop etish uchun, ushbu atribut qo'shiladi*/**
10. **static int page_count;**
11. **private:**
12. **char title [64];**
13. **char author[64];**
14. **float price; };**
15. **int book_series::page_count; /*O'zgaruvchini global o'zgaruvchi sifatida e'lon qilish*/**
16. **void book_series::show_book (void){**
17. **printf("Sahifalar soni=%d\n",page_count); }**
18. **int main(void){**
19. **book_series::page_count = 256;**
20. **book_series::show_book();_getch(); }**

Nazorat savollari

1. Sinf ichidagi ma'lumotlarni himoyalashning nechta xil usuli bor?
2. Sinfning **static** elementi qanday e'lon qilinadi?
3. Sinf elementini qachon **private static** ko'rinishida e'lon qilishga zarurat tug'iladi?
4. Sinf elementini qachon **public va static** ko'rinishida e'lon qilishga zarurat tug'iladi?
5. **private** ko'rinishida himoyalangan elementga **int main()** funksiya orqali qiymat o'zlashtirib bo'ladimi? **Static** atributi bilan e'lon qilingan elementgachi?

Merosxo'rlik.

Vorislikda murojaat xuquqlarini boshqarish

Vorislik o'zining barcha ajdodlarining xususiyatlari, ma'lumotlari, metodlari va voqealarini meros qilib oladigan hosila sinfini e'lon qilish imkoniyatini beradi, shuningdek yangi tavsiflarni e'lon qilishi xamda meros sifatida olinayotgan ayrim funksiyalarni ortiqcha yuklashi mumkin. Bazaviy sinfnings ko'rsatib o'tilgan tavsiflarini meros qilib olib, yangi tug'ilgan sinfni ushbu tavsiflarni kengaytirish, toraytirish, o'zgartirish, yo'q qilish yoki o'zgarishsiz qoldirishga majburlash mumkin.

Hosila sinfni e'lon qilishning umumlashgan sintaksisi:

class <sinf nomi>: [<kirish xuquqini beruvchi sertifikat >] <ajdod sinf nomi> {...}

Sinf o'zining bazaviy sinfidan yuzaga kelayotganida, uning barcha nomlari hosila sinfda avtomatik tarzda yashirin **private** bo'lib qoladi. Ammo uni, bazaviy sinfnings quyidagi kirish spertifikatorlarini ko'rsatgan holda osongina o'zgartirish mumkin:

private. Bazaviy sinfnings meros bo'lib o'tayotgan (ya'ni ximoyalangan va ommaviy) nomlari hosila sinf nushalarida kirib bo'lmaydigan bo'lib qoladi.

public. Bazaviy sinf va uning ajdodlarining nomlari hosila sinf nushalarida kirib bo'ladigan bo'ladi, barcha himoyalangan nomlar esa himoyalangan bo'lib qolaveradi.

Agarda yangi sinf **class** kalitli so'z yordamida aniqlangan bo'lsa unda hosila sinfdagi meros komponentalar **private** kirish statusiga ega bo'ladi, **struct** yordamida esa **public** statusiga.

Me'roslikda ko'rsatilmagan kirish statusini asosiy(bazaviy) sinf ismini oldidan ko'rsatilgan **private**, **protected** va **public** kirish atributlari yordamida o'zgartirish mumkin. Agarda V sinf quyidagicha aniqlangan bo'lsa:

```
class B { protected: int t;
```

```
public: char u; };
```

unda quyidagi hosila sinflarni kiritish mumkin:

```
class M: protected B { ... }; //t va u protected sifatida merosxo'r.
```

```
class P: public B { ... }; // protected, va u- public sifatida merosxo'r.
```

class D: private B { ... }; //t va u private sifatida merosxo‘r.

struct F: private B { ... }; //t i u private sifatida merosxo‘r.

struct G: public B { ... }; t - **protected** va u – public sifatida merosxo‘r.

Konstruktor va destruktorlarda vorislik

Konstruktorlar meros bo‘lmagani uchun, hosila sinfni yaratishda undan meros bo‘lgan ma`lumot – a`zolari asosiy (bazaviy) sinf konstruktori orqali inisializasiyalanishi lozim. Asosiy sinf konstruktori avtomatik ravishda chaqiriladi va hosila sinfni konstruktoridan oldin bajariladi. Asosiy (bazaviy) sinfni konstruktorining parametrlari hosila sinfni konstruktorini aniqlashda ko‘rsatiladi. Shunday qilib argumentlarni hosila sinfni konstruktoridan asosiy (bazaviy) sinfni konstruktoriga uzatish vazifasi bajariladi.

Masalan.

```
class Basis{  
    int a,b;  
public: Basis(int x,int y){aqx;bqy;} };  
class Inherit:public Basis  
{int sum;  
public:  
Inherit(int x,int y, int s):Basis(x,y){sum=s;} };
```

Sinf obyektlari pastdan tepaga qarab konstruktorlanadi: avvalo asosiy(bazaviy), keyin esa komponent – obyektlar (agarda ular mavjud bo‘lsa), undan keyin esa hosila sinfning o‘zi. SHunday qilib, hosila sinfning obyekti quyi obyekt sifatida asosiy (bazaviy) sinf obyektini o‘z ichiga oladi. Obyektlar teskari tartibda o‘chiriladi: avvalo hosila, keyin uning komponent – obyektlari, undan keyin esa asosiy(bazaviy) obyekt.

Shunday qilib, obyektning o‘chirish tartibi uning konstruktorlash tartibiga nisbatan teskari bo‘ladi.

Ko‘plikdagi vorislik va virtual sinflar

Bu sinf ketma-ket (to‘g‘ri-to‘g‘ri) baza sinfidir, agar u boshqa sinflarni aniqlashda ishlatilsa, baza ro‘yxatidan chiqariladi. Ba`zi hollarda A sinf B sinfning bazasini ifodalasa va C uchun B baza bor bo‘lsa, u holda B sinf C uchun to‘g‘ridan-to‘g‘ri baza hisoblanadi, natijada A sinf C sinf uchun to‘g‘ri bo‘lmagan baza bo‘lib hisoblanadi. Quyida keltirilgan sinflarni tasvirlashda bazalar ishlab chiqilgan. Xuddi shu tartibda yangi baza sinflarini kompilyator e`lon qiladi.

Sinflar bir nechta ketma-ket sinflardan tashkil topishi mumkin, sinf bazasida ixtiyoriy son yo‘qolishi mumkin, misol uchun,

```
class X1 { ... };
```

```

class X2 { ... };
class X3 { ... };
class Y1: public X1, public X2, public X3 { ... };

```

Bir necha to'g'ri baza sinflari mavjud bo'lib, ular ko'plik vorislari deb nomlanadi. Ko'plik vorislarida ketma-ket bazada hech qanday sinf bittadan ortiq ishlatilishi mumkin emas. Bitta sinf to'g'ri bo'lmagan sinfda bir necha marta ishlatilishi mumkin:

```

class X { ...; f () ; ... };
class Y: public X { ... };
class Z: public X { ... };
class D: public Y, public Z { ... };

```

Bu misolda X va Z sinflari D sinfiga voris bo'ladi. Bir xil nomdagi obyektlarni bartaraf qilishda to'g'ri bo'lmagan sinf bazalarining ko'plik vorislari virtual deb e'lon qilinadi. Buning uchun sinf bazalari ro'yxatida oldingi sinf nomini **virtual** kalit so'zini ishlatish kerak. Misol uchun X sinfi virtual baza sinfi bo'l ko'rinishda quyidagicha yoziladi:

```

class X { ... f() ; ... };
class Y: virtual public X { ... };
class Z: virtual public X { ... };
class D: public Y, public Z { ... };

```

Abstrakt sinflar

Xech bo'lmasa bitta sof (bo'sh) virtual funksiyaga ega bo'lgan sinf abstrakt sinf deyiladi. Quyidagi tavsifga ega bo'lgan komponentali funksiya sof virtual funksiya deyiladi:

```

virtual <tip> <funksiya_nomi>(<formal_parametrlar_ro'yxati>) = 0;

```

Abstrakt sinf hosila sinf uchun asosiy (bazaviy) sinf sifatida ishlatilishi mumkin. Abstrakt sinflarning mexanizmi keyinchalik konkretizasiyalanadigan umumiy tushunchalarni tavsiflash uchun ishlab chiqilgan. Bu holda, sinflar ierarxiyasini yaratish quyidagi sxema bo'yicha bajariladi. Ierarxiya asosida abstrakt bazaviy sinf turadi. U interfeysni meros qilib olish uchun foydalaniladi. Hosila sinflar bu interfeysni konkretizasiyalaydi va amalga oshiradi. Abstrakt sinfda sof virtual funksiyalar e'lon etilgan, ular aslida **abstrakt usullar**.

Ba'zi sinflar masalan shape sinfi, abstrakt tushunchalarni ifodalaydi va ular uchun obyekt yaratib bo'lmaydi. Bunday sinflar biror hosila sinfda ma'noga ega bo'ladi:

```

class shape {
//...
public:
    virtual void rotate(int) q =0; //sof virtual funksiya
    virtual void draw() = 0; // sof virtual funksiya
};

```

Abstrakt sinfni faqat boshqa sinf ajdodi sifatida ishlatish mumkin:

```
class circle : public shape {  
    int radius;  
public:  
    void rotate(int) { }  
    //qayta ta`riflash shape::rotate  
    void draw();  
    //qayta ta`riflash shape::draw  
    circle(point p, int r); };
```

Agar sof virtual funksiya hosila sinfda to'liq ta'riflanmasa, u hosila sinfda ham sof virtual bo'lib qoladi, natijada hosila sinf ham abstrakt sinf bo'ladi.

Abstrakt sinflar realizatsiya detallarini aniqlashtirmasdan faqat interfeysni ko'rsatish uchun ishlatiladi. Masalan operasion tizimda qurilma drayveri abstrakt sinf sifatida berilishi mumkin:

```
class character_device { public:  
    virtual int open() = 0;  
    virtual int close(const char*) = 0;  
    virtual int read(const char*, int) = 0;  
    virtual int write(const char*, int) = 0;  
        virtual int ioctl(int ...) = 0; };
```

Drayverlar character_device sinfining ajdodlari sifatida kiritilishi mumkin.

Nazorat savollari

1. Konstruktordan voris olish nima uchun kerak?
2. Destruktordan qanday voris olinadi?
3. Ko'plikdagi vorislik qanaqa bo'ladi?
4. Abstrakt sinflar nima uchun ishlatiladi?

**MAVZU: POLIMORFIZM VA UNING TURLARI.
VIRTUAL FUNKSIYA. ABSTRAKT SINFLAR VA FUNKSIYALAR**

Reja :

1. **Vaqtli va kechiktirilgan bog'lanishlar;**
2. **Virtual funksiyalar;**
3. **Virtual va novirtual funksiyalar;**
4. **Dinamik polimorfizmni qo'llash;**
5. **Virtual destructorlar;**
6. **Abstrakt sinflar va sof virtual funksiyalar.**

Annotatsiya: Ushbu ma'ruzada obyektga yo'naltirilgan dasturlash tamoyillaridan polimorfizm va uning turlarini qo'llanilish usullari haqida ma'lumotlar keltirilgan.

Kalit so'zlar: sinf, polimorfizm, virtual funksiyalar, novirtual funksiyalar, abstrakt sinflar, dinamik polimorfizm, virtual destructor.

1. Vaqtli va kechiktirilgan bog'lanishlar.

C++tilida polimorfizm ikki usulda qo'llab-quvvatlanadi. Birinchisi, funksiya va operatorlarni qayta yuklash vositasi bilan kompilyatsiya paytida. Polimorfizmning bu ko'rinishiga *statik polimorfizm* deyiladi, chunki u dastur bajarilishidan oldin, ya'ni kompilyatsiya va jamlash (komponovka) paytida funksiya identifikatorlarini fizik adreslar bilan *vaqtli bog'lash* orqali amalga oshiriladi. Ikkinchisida, dastur bajarilishida virtual funksiyalar vositasida. Dastur kodida virtual funksiyaga murojaatni uchratgan kompilyator, bu chaqirishni faqat belgilab qo'yadi, funksiya identifikatorini adres bilan bog'lashni dasturni bajarish bosqichiga qoldiradi. Bu jarayonga *kechiktirilgan bog'lanish* deyiladi. *Virtual funksiya* - bu shunday funksiyaki, uni chaqirish va mos amallarni bajarish, uni chaqirgan obyekt turiga bog'liq bo'ladi. Obyekt dastur bajarilish jarayonida qaysi funksiyani chaqirish kerakligini aniqlaydi. Polimorfizmning bu ko'rinishiga *dinamik polimorfizm* deyiladi. Dinamik polimorfizmni amalga oshirishning asosi sifatida C++ tilidagi tayanch sinfga ko'rsatkichni aniqlanishidir. O'z navbatida, bu ko'rsatkich nafaqat tayanch sinfga, balki shu sinfnin vorisi bo'lgan ixtiyoriy sinf obyektiga ko'rsatishi mumkin. Sinflarning bu xossasi vorislikdan kelib chiqadi, chunki har qanday voris sinf obyektini tayanch sinf turida bo'ladi. Dasturni yig'ish paytida (komponovka paytida) tayanch sinfga ko'rsatkich egasi bo'lgan foydalanuvchi tomonidan qaysi sinf obyektini yaratilishi noma'lum bo'ladi. Shu sababli, ko'rsatkich o'z obyektini bilan faqat dastur ishlashi paytidagini, ya'ni dinamik ravishda bog'lanishi mumkin. Tarkibida hech bo'lmaganda bitta virtual funksiyasi bo'lgan sinf *polimorf sinf* deyiladi. Har bir polimorf turdagi berilganlar uchun kompilyator *virtual funksiyalar jadvalini* yaratadi va shu jadvalga sinfnin har bir obyektiga yashiringan ko'rsatkichni joylashtiradi. Kompilyator virtual funksiyalar jadvaliga ko'rsatkichni initsializatsiya qiluvchi kod bo'lagini polimorf sinf konstruktori boshlanishiga joylashtiradi. Virtual funksiya chaqirilganda ushbu kod virtual funksiyalar jadvaliga ko'rsatkichni

topadi, keyin jadvaldan mos funksiya adresini oladi. Keyin ko'rsatilgan adresga o'tish bilan funksiya chaqirishi ro'y beradi.

Jadval 1. Sinf virtual funksiyalar jadvali

Sinf shajarasi va ko'rsatgichlar	Ko'rsatgich qiymatining turi	Sinflardagi virtual funksiyalarning adreslari		
		f1()	f2()	...
Tayanch * tayanch;	Tayanch	Adres ₁₁	Adres ₂₁	...
Voris1 * voris1; // <i>Tayanch vorisi</i>	Voris1	Adres ₁₂	Adres ₂₂	...
Voris2 * voris2; // <i>Tayanch vorisi</i>	Voris2	Adres ₁₃	Adres ₂₃	...
...

Ma'lumki, hosilaviy sinf obykti yaratilishida tayanch sinf konstruktori chaqiriladi. Ayni shu bosqichda virtual funksiyalar jadvali va unga ko'rsatkich hosil bo'ladi. Hosilaviy sinf konstruktori chaqirilgandan keyin virtual funksiyalar jadvaliga ko'rsatkich, shu sinf obykti uchun qayta aniqlangan virtual funksiya variantini ko'rsatish uchun moslanadi (agar u mavjud bo'lsa).

Ko'rinib turibdiki, kechiktirilgan bog'lanishni amalga oshirish muayyan bir resurslar sarflashni taqoza etadi va undan oqilona foydalanish zarur bo'ladi.

Virtual funksiyalar

Mazmunidan kelib chiqqan holda virtual funksiyaga boshqa tavsiflarni berish mumkin:

1) chaqirish interfeysi (prototipi) ma'lum, amalga oshirilishi umumiy ko'rinishda berilishi mumkin bo'lmasdan, faqat konkret holatlardagini aniqlanadigan funksiyalarga *virtual funksiyalar* deyiladi;

2) *virtual funksiya* - bu chaqirilishi uchun qanday ifoda ishlatilishidan qat'iy nazar obyekt uchun to'g'ri (mos) funksiya chaqirilishini kafolatlaydigan funksiyadir.

Faraz qilaylik, tayanch sinfda funksiyaning virtual e'loni, hosilaviy sinfda ham xuddi shu funksiya e'loni bo'lsin. U holda hosilaviy sinf obyektlari uchun hosilaviy sinf funksiyasi chaqiriladi, agar ular chaqirilishida tayanch sinfga ko'rsatkich yoki murojaat ishlatilgan bo'lsa ham.

```
class Tayanch
{
public:
    Tayanch(int _x) {x=_x;}
    virtual int Qiymat_X(){return x;}
    virtual void Chop_X();
private:
    int x;};
void Tayanch:: Chop_X() {cout<<"Tayanch::x="<<Qiymat_X()<<"\n"; }
class Hosila1: public Tayanch
{
public:
    Hosila1(int _x): Tayanch(_x){}
    void Chop_X();
```

```

};
void Hosila1:: Chop_X() {cout<<"Hosila1::x="<<Qiymat_X()<<"\n";}
class Hosila2: public Tayanch
{
public:
Hosila2(int _x): Tayanch(_x){}
void Chop_X();
};
void Hosila2:: Chop_X() {cout<<"Hosila2::x="<<Qiymat_X()<<"\n"; }
int main(int argc, char* argv[])
{
Tayanch * tayanch=new Tayanch(10);
Hosila1 * hos1=new Hosila1(20);
Hosila2 * hos2=new Hosila2(30);
tayanch->Chop_X();
tayanch=hos1;
tayanch->Chop_X();
tayanch=hos2;
tayanch->Chop_X();
return 0;
}

```

Hosilaviy sinflardagi Chop_X() funksiyalari virtual hisoblanadi, chunki u Tayanch tayanch sinfida virtual deb e'lon qilingan. Virtual funksiyalarni chaqirish uchun quyidagi kodlar ishlatilgan:

```

tayanch=hos1;
tayanch->Chop_X();
tayanch=hos2;
tayanch->Chop_X();

```

Sinflardagi Chop_X() funksiyalari virtual bo'lganligi uchun har bir obyektning o'z funksiyasi chaqiriladi. Hosila1 va Hosila2 sinflarida Chop_X() funksiyalar Tayanch tayanch sinfidagi Chop_X() funksiyasini qayta aniqlaydi. Agar hosilaviy sinfda Chop_X() funksiyasi qayta aniqlanmasa, kelishuv bo'yicha tayanch sinfdagi Chop_X() funksiyasi amal qiladi.

Yuqoridagi dastur ishlashi natijasida ekranga

Tayanch::x=10

Hosila1::x=20

Hosila2::x=30

natijalar chop etiladi:

Funksiyalarni ko'rsatkich va adresni olish amallari yordamida chaqirishda quyidagi qoidalarga amal qilinadi:

- virtual funksiyani chaqirish uni chaqirayotgan obyekt turiga mos ravishda hal qilinadi;
- virtual bo'lmagan funksiyalarni chaqirish ko'rsatkich turiga mos ravishda amalga oshiriladi.

Virtual funksiyalar faqat qandaydir sinfga tegishli obyektlar uchun chaqirilishini inobatga oladigan bo'lsak, global yoki statik funksiyalarni virtual deb e'lon qilish mumkin emas. virtual kalit so'zi hosilaviy sinfda funksiyani qayta aniqlashda ishlatilishi mumkin, lekin majburiy emas.

Tayanch sinfdagi virtual funksiyalar shu sinfda aniqlanishi kerak, agar ular sof virtual deb e'lon qilingan bo'lmasa. Hosilaviy sinfda e'lon qilingan funksiya tayanch sinfdagi virtual funksiyani qayta aniqlaydi, agar uning nomi virtual funksiya nomi bilan mos tushsa, ular bir xil miqdordagi va turlari mos kelgan parametrlarga ega bo'lsa. Agar funksiya virtual funksiyadan hattoki bitta parametri bilan farq qilsa, u holda hosilaviy sinfdagi bu funksiya yangi hisoblanadi va

qayta aniqlash ro'y bermaydi. Hosilaviy sinfdagi funksiya virtual funksiya bilan faqat qaytaruvchi qiymati bilan farqlanmaydi, ularning parametrlar ro'yxati turlicha bo'lishi kerak.

Quyidagi misolda virtual funksiya tayanch sinfdagi o'zi bilan bir xil prototipga ega virtual funksiyaning qayta aniqlanishi.

```
#include <iostream.h>
class Tayanch
{
    int x;
public:
    virtual void Qiymat(int _x) { x=_x; cout<<"Tayanch::x = "<<x<<"\n"; }
    virtual void Chop_Qilish(Tayanch * pOb) { Qiymat(10); }
};
class Hosila: public Tayanch
{
    int x,y;
public:
    virtual void Qiymat(int _x,int _y)
    {
        x=_x; y=_y;
        cout<<"Hosila::x = "<<x<<" Hosila::y = "<<y<<"\n";
    }
    virtual void Chop_Qilish(Tayanch * pOb) { Qiymat(15,20); }
};
int main()
{
    Tayanch * pOb1=new Tayanch;
    Tayanch * pOb2=new Hosila;
    // Tayanch sinfidan virtual Chop_Qilish() funksiyasini chaqirish
    pOb1->Chop_Qilish(pOb1);
    // Hosila sinfidan virtual Chop_Qilish() funksiyasini chaqirish
    pOb2->Chop_Qilish(pOb2);
    // Hosila sinfidan virtual Chop_Qilish() funksiyasini chaqirish
    pOb2->Chop_Qilish(pOb2);
    return 0;
}
```

Dastur ishlashi natijasida ekranga quyidagilarni chop etadi:

```
Tayanch::x = 10
Hosila::x =15 Hosila::y = 20
Hosila::x =15 Hosila::y = 20
```

Keltirilgan misolda tayanch va hosilaviy sinflar ikkita bir xil nomdagi virtual funksiya bilan ega. Lekin kompilyator ularni turlicha talqin qiladi. Qiymat() funksiyasining prototipi hosilaviy sinfdagi o'zgarganligi sababli, u mutlaqo boshqa virtual funksiya deb qaraladi. Ikkinchi tomondan, hosilaviy sinfdagi Chop_Qilish() funksiyasi tayanch sinfdagi mos virtual funksiyaning qayta aniqlanishi deb qaraladi.

Virtual va novirtual funksiyalar

Quyidagi misol, ko'rsatkich orqali chaqirilganda virtual va novirtual funksiyalar o'zini qanday tutishi ko'rsatilgan:

```
#include <iostream.h>
class Tayanch
{
public:
```

```

virtual void Virtual_Fun(){cout<<"Tayanch::Virtual_Fun()\n"; }
void NoVirtual_Fun(){cout<<"Tayanch::NoVirtual_Fun()\n";}
};
class Hosila: public Tayanch
{
public:
virtual void Virtual_Fun(){ cout<<"Hosila::Virtual_Fun()\n";}
void NoVirtual_Fun(){cout<<"Hosila::NoVirtual_Fun()\n";}
};
int main(){
Hosila hosila;
Hosila * pHosila = &hosila;
Tayanch * pTayanch = &hosila;
// Hosila sinfidan Virtual_Fun() funksiyasini chaqirish
pTayanch->Virtual_Fun();
//Tayanch sinfidan NoVirtual_Fun() funksiyasini chaqirish
pTayanch->NoVirtual_Fun();
// Hosila sinfidan Virtual_Fun() funksiyasini chaqirish
pHosila->Virtual_Fun();
// Hosila sinfidan NoVirtual_Fun() funksiyasini chaqirish
pHosila->NoVirtual_Fun();
return 0;
}

```

Dastur bajarilishi natijasida ekranga

```

Hosila::Virtual_Fun()
Tayanch::NoVirtual_Fun()
Hosila::Virtual_Fun()
Hosila::NoVirtual_Fun()

```

Shunga e'tibor berish kerakki, Virtual_Fun() funksiyasi qaysi sinfga -Tayanch yoki Hosila ko'rsatkich orqali chaqirilishidan qat'iy nazar Hosila sinfidan Virtual_Fun() funksiyasi chaqiriladi. Bunga sabab -Virtual_Fun() funksiyasi virtual va pTayanch hamda pHosila ko'rsatkichlari Hosila turidagi obyektga ko'rsatadi. Ikkinchi tomondan, tayanch sinfga ko'rsatkichi pTayanch garchi novirtual funksiyalarga ega hosilaviy sinf obyektiga ko'rsatsa ham, tayanch sinfdagi mos funksiyani chaqiradi.

Ko'rish sohasiga ruxsat berish operatori vositasida kechiktirilgan bog'lanishni man qilish mumkin:

```

#include <iostream.h>
class Tayanch
{
public:
virtual void Virtual_Fun(){cout<<"Tayanch::Virtual_Fun()\n";}
};
class Hosila: public Tayanch
{
public:
virtual void Virtual_Fun(){cout<<"Hosila::Virtual_Fun()\n";}
};
int main()
{
Tayanch * pTayanch =new Hosila;

```

```

// Hosila sinfidan Virtual_Fun() funksiyasini chaqirish
pTayanch->Virtual_Fun();
//Tayanch sinfidan Virtual_Fun() funksiyasini chaqirish
pTayanch->Tayanch::Virtual_Fun();
return 0;

```

```

}

```

Dastur ekranga

```

Hosila::Virtual_Fun()

```

```

Tayanch::Virtual_Fun()

```

xabarlarini chop etadi.

Ko‘rinibturibdiki,

```

pTayanch->Tayanch::Virtual_Fun();

```

ko‘rsatmasi kechiktirilgan bog‘lanishni yo‘qqa chiqaradi.

Sinflar shajarasida virtual funksiyalar nomi bilan bir xil qayta yuklanuvchi funksiyalarni e‘lon qilish mumkin. Lekin, bunday novirtual funksiyalar yashiringan bo‘ladi.

Yuqoridagi fikrlarni tasdiqlaydigan misol keltiramiz:

```

#include <iostream.h>

```

```

#include <string.h>

```

```

class Tayanch

```

```

{

```

```

public:

```

```

Tayanch(char * nom){strcpy(Nom,nom);}

```

```

virtual void Fun(char c)

```

```

{

```

```

cout<<"Virtual "<<Nom<< "::Fun "<<c<<" parametrini qabul qildi.\n";

```

```

}

```

```

protected:

```

```

char Nom[20];

```

```

};

```

```

class Hosila1: public Tayanch

```

```

{

```

```

public:

```

```

Hosila1(char * nom):Tayanch(nom){ }

```

```

void Fun(const char * s){cout<<Nom<< "::Fun "<<s<<" parametrini qabul qildi.\n";}

```

```

void Fun(int n){cout<<Nom<< "::Fun "<<n<<" parametrini qabul qildi.\n";}

```

```

virtual void Fun(char c)

```

```

{

```

```

cout<<"Virtual "<<Nom<< "::Fun "<<c<<" parametrini qabul qildi.\n";

```

```

}

```

```

};

```

```

class Hosila11: public Hosila1

```

```

{

```

```

public:

```

```

Hosila11(char * nom):Hosila1(nom){ }

```

```

void Fun(const char * s){cout<<Nom<< "::Fun "<<s<<" parametrini qabul qildi.\n";}

```

```

void Fun(double d){cout<<Nom<< "::Fun "<<d<<" parametrini qabul qildi.\n";}

```

```

virtual void Fun(char c)

```

```

{

```

```

cout<<"Virtual "<<Nom<< "::Fun "<<c<<" parametrini qabul qildi.\n";

```

```

}

```

```

};
int main()
{
    Tayanch tayanch("Tayanch");
    Hosila1 hosila1("Hosila1");
    Hosila11 hosila11("Hosila11");
    tayanch.Fun('X');
    hosila1.Fun('Y');
    hosila1.Fun(10);
    hosila1.Fun("Hos1");
    hosila11.Fun("Z");
    hosila11.Fun(10.1234);
    hosila11.Fun("Hos11");
    return 0;
}

```

Dastur bajarilgandan keyin ekranda quyidagi satrlar paydo bo'ladi:

Virtual Tayanch::Fun X parametrini qabul qildi.

Virtual Hosila1::Fun Y parametrini qabul qildi.

Hosila1::Fun 10 parametrini qabul qildi.

Hosila1::Fun Hos1 parametrini qabul qildi.

Virtual Hosila11::Fun Z parametrini qabul qildi.

Hosila11::Fun 10.1234 parametrini qabul qildi.

Hosila11::Fun Hos11 parametrini qabul qildi.

Keltirilgan misolda chiziqli vorislikni hosil qiluvchi uchta sinf aniqlangan. Tayanch sinfida Fun(char) virtual funksiya e'lon qilingan. Hosila1 sinfi Fun(char) virtual funksiya sinfining o'z variantini va ikkita qayta yuklanuvchi novirtual Fun(const char*) va Fun(int) funksiyalarni e'lon qilgan. O'z navbatida, Hosila11 sinfi Fun(char) virtual funksiya sinfining o'z variantini va ikkita qayta yuklanuvchi novirtual Fun(const char*) va Fun(double) funksiyalarni e'lon qilgan. Garchi, Fun(const char*) funksiyasi Hosila1 sinfidagi analogi bilan to'la ustma-ust tushsa ham, u Hosila11 sinfida qayta e'lon qilingan. Chunki, Hosila1 sinfida xuddi shu nomdagi virtual va novirtual funksiyalar mavjudligi sababli, Fun(const char*) funksiyasi yashiringan bo'ladi. Xuddi shunday, Fun(char) virtual funksiyasi Hosila1 va Hosila11 sinflarida qaytadan e'lon qilishga to'g'ri keladi, chunki ular ham sinflarda xuddi shu nomdagi qayta yuklanuvchi funksiyalarni mavjudligi sababli yashiringan bo'ladi.

Agar voris sinflardagi virtual funksiyalar e'lonlari o'chirilsa, funksiyaning belgi argumentli chaqirishda, amalda Hosila1 sinfidagi Fun(int) funksiyasini chaqirish ro'y beradi. Tayanch sinfidagi virtual funksiyani chaqirish zarur bo'lsa, ko'rish sohasiga ruxsat berish amalidan foydalanish mumkin:

```
Hosila1.Tayanch::Fun('Y');
```

Dinamik polimorfizmni qo'llash

Dinamik polimorfizm vositasida dastur bajarilishini boshqarishning moslanuvchan boshqarishni amalga oshirish mumkin. Quyida, butun sonlarning bog'langan ro'yxati ko'rinishida amalga oshirilgan *stek* va *navbat* tuzilmalari ustida ishlash qaralgan. Ma'lumki, navbat - "*birinchi kelgan - birinchi ketadi*", stek - "*oxirda kelgan - birinchi ketadi*" tamoyili bo'yicha berilganlarni saqlash va qayta ishlashni amalga oshiruvchi tuzilmalar hisoblanadi. Dasturda bog'langan ro'yxatni yaratish, unga qiymat joylashtirish va o'chirishni amalga oshiruvchi Ruysat tayanch sinfi va uning vorislari sifatida navbat hosil qiluvchi mos ravishda Navbat va Stek sinflari yaratiladi. Garchi bu tuzilmalar bilan ishlash turlicha amalga oshirilsa ham, ularni ishlatishda yagona interfeysdan foydalaniladi.

```
#include <iostream.h>
```

```
#include <stdlib.h>
```

```

#include <ctype.h>
class Ruyxat
{
public:
    Ruyxat(){Boshi=Oxiri=Keyingi=0;}
    virtual void Joylash(int n)=0;
    virtual bool Olish(int& n)=0;
    void Qiymat_n(int n){Son=n;}
    int n_Qiymat(){return Son;}
    Ruyxat * Boshi;
    Ruyxat * Oxiri;
    Ruyxat * Keyingi;
private:
    int Son;
};
class Navbat: public Ruyxat
{
public:
    void Joylash(int n);
    bool Olish(int& n);
};
void Navbat::Joylash(int n)
{
    Ruyxat * Yangi;
    Yangi=new Navbat;           // Navbatning yangi elementini yaratish
    Yangi->Qiymat_n(n);
    Yangi->Keyingi=NULL;
    if(Oxiri) Oxiri->Keyingi=Yangi; // Elementni navbatning oxiriga joylash
    Oxiri=Yangi;
    if(!Boshi) Boshi=Yangi;
}
bool Navbat::Olish(int& n)
{
    Ruyxat * Element;
    if(!Boshi){n=0; return false;}
    n=Boshi->n_Qiymat();
    Element=Boshi;
    Boshi=Boshi->Keyingi;
    delete Element;
    return true;
}
class Stek: public Ruyxat
{
public:
    void Joylash(int n);
    bool Olish(int& n);
};
void Stek::Joylash(int n)
{
    Ruyxat * Yangi;
    Yangi=new Stek;           // Stekning yangi elementini yaratish

```

```

Yangi->Qiymat_n(n);
Yangi->Keyingi=NULL;
if(Boshi) Yangi->Keyingi=Boshi;    // Elementni stek boshiga joylash
Boshi=Yangi;
if(!Oxiri) Oxiri=Yangi;
}
bool Stek::Olish(int& n)
{
    Ruyxat * Element;
    if(!Boshi){n=0; return false;}
    n=Boshi->n_Qiymat();
    Element=Boshi;
    Boshi=Boshi->Keyingi;
    delete Element;
    return true;
}
int main()
{
    Ruyxat * ruyxat;
    Navbat navbat;
    Stek stek;
    int son;
    char stek_navbat;
    cout<<"Sonlarni navbat va stekka joylash:\n";
    do
    {
        cout<<"Sonni kiriting(0-tamom): ";
        cin>>son;
        if(son)
        {
            do
            {
                cout<<"Joylshtirish? Stekka(S) yoki Navbatga(N):";
                cin>>stek_navbat;
            }
            while (stek_navbat!='S' && stek_navbat!='s'
                && stek_navbat!='N' && stek_navbat!='n');
            switch(stek_navbat)
            {
                case 'S': case 's': ruyxat=&stek; break;
                case 'N': case 'n': ruyxat=&navbat;
            }
            ruyxat->Joylash(son);
        }
    }
    while (son);
    for(;;)
    {
        do
        {
            cout<<"O'qish? Stekdan(S) yoki Navbatdan(N):\n";

```

```

cout<<"Dasturdan chiqish (Q):\n";
cin>>stek_navbat;
}
while(stek_navbat!='S' && stek_navbat!='s' &&
stek_navbat!='N' && stek_navbat!='n' &&
stek_navbat!='Q' && stek_navbat!='q');
switch(stek_navbat)
{
case 'S':
case 's': ruyxat=&stek; break;
case 'N': case 'n': ruyxat=&navbat; break;
case 'Q':case 'q': return 0;
}
if(ruyxat->Olish(son)) cout<<son<<endl;
else cout<<"Ro'yxat bo'sh!"<<endl;
}
}

```

Dastur kirish oqimidan butun sonlarni o'qiydi va foydalanuvchi tanlagan ro'yxatga - navbat yoki stekka joylashtiradi. Sonlarni kiritish jarayoni navbatdagi son tariqasida 0 soni kiritilganda to'xtaydi. Keyinchalik, foydalanuvchi ko'rsatgan ro'yxatdan son qiymatlari o'qiladi va ekranga chop qilinadi. Dinamik polimorfizm ruyxat ko'rsatkichi navbat yoki stek obyektlariga ko'rsatishiga mos ravishda virtual Olish() va Joylash() funksiyalarini chaqirishida namoyon bo'ladi.

Virtual destruktorglar

Konstruktorlar virtual bo'lmaydi, lekin destruktorglar virtual bo'lishi mumkin va aksariyat holatlarda shunday bo'ladi. Tayanch sinfiga ko'rsatkich hosilaviy sinf obyektiga ko'rsatib turganda, agar destruktorg virtual qilib e'lon qilingan bo'lsa, hosilaviy sinf destruktorgi chaqiriladi. Hosilaviy sinf destruktorgi o'z navbatida tayanch sinf destruktorgini chaqiradi va obyekt to'g'ri (to'laligicha) o'chiriladi. Aks holda ko'rsatkich turiga mos ravishda tayanch sinf destruktorgi chaqiriladi, hosilaviy sinf uchun ajratilgan xotira bo'shatilmay qoladi - xotirada band qilingan, lekin qayta ishlatish imkoni bo'lmagan xotira bo'lagi - "*xotira axlati*" paydo bo'ladi.

Misol ko'raylik:

```

#include <iostream.h>
class Tayanch
{
int * px;
public:
Tayanch(int _x) {px=new int; *px=_x; }
/*virtual*/~Tayanch() { cout<<"Tayanch sinf destruktorgi ishladi!\n"; delete px; }
};
class Hosila: public Tayanch
{
int * pxx;
public:
Hosila (int n):Tayanch(n) { pxx=new int; *pxx=n*n; }
~Hosila() { cout<<"Hosila sinf destruktorgi ishladi!\n"; delete pxx; }
};
int main()
{
Tayanch * pTayanch=new Hosila(5);
delete pTayanch;
return 0;
}

```

}

Hosilaviy sinf - Hosila sinfida destruktore virtual deb e'lon qilinmagan va delete pTayanch;

til ko'rsatmasi bajarilishi natijasida ekranga

Tayanch sinf destruktore ishladi!

xabari chiqadi. Bu holat xotiradagi Hosila sinf obyektini uchun ajratilgan xotira bo'shatilmay qolganligini bildiradi. Agar tayanch sinf destruktoreni virtual deb e'lon qilinsa, obyektini o'chirish to'g'ri ro'y beradi - oldin hosilaviy sinf destruktore, keyin tayanch sinf destruktore bajariladi.

Dasturning ekranga chiqaradigan

Hosila sinf destruktore ishladi!

Tayanch sinf destruktore ishladi!

xabarlari buni isbotlaydi.

Abstrakt sinflar va sof virtual funksiyalar

Sinflar, shu turga tegishli bo'lgan obyektlarning o'zaro bajaradigan amallari qoidalarini oldindan aniqlab berish uchun yaratilishi mumkin. Bunday sinflarga *abstrakt sinflar* deyiladi. Abstrakt sinflarning obyektlarini yaratib bo'lmaydi. Ular faqat hosilaviy sinflarni yaratish uchun xizmat qiladi.

Abstrakt sinf kamida bitta virtual funksiyaga ega bo'lishi kerak. Tayanch sinfning sof virtual funksiyalari hosilaviy sinflarda albatta aniqlanishi kerak, aks holda hosilaviy sinf ham virtual hisoblanadi.

Sof virtual funksiya quyidagi sintaksis bilan e'lon qilinadi:

virtual <funksiya nomi>(<parametrlar ro'yxati>)=0;

Misol ko'raylik. Faraz qilaylik, sinflar shajarisini yaratish zarur bo'lsin va tayanch sinf umumiy funksional imkoniyatlarni ta'minlashi kerak. Lekin, tayanch sinfi shu darajada umumlashgan bo'lib, natijada undagi ayrim funksiyalarni konkretlashtirish imkoni bo'lmashligi mumkin. Bunday tayanch sinfi abstrakt sinf uchun eng yaxshi nomzod hisoblanadi:

Misol uchun jonivorlar shajarasini tavsiflovchi Jonivor abstrakt tayanch sinf va uning vorislari Kuchuk va Mushuk sinflarini e'lon qilishni ko'raylik.

```
class Jonivor
```

```
{
```

```
public:
```

```
Jonivor(char * nomi)
```

```
{
```

```
Nomi=new char[15];
```

```
strcpy(Nomi,nomi);
```

```
};
```

```
virtual void Ovozi()=0;
```

```
virtual void Ozuqasi()=0;
```

```
protected:
```

```
char * Nomi;
```

```
};
```

```
class Kuchuk : public Jonivor
```

```
{
```

```
public:
```

```
Kuchuk(char * nomi):Jonivor(nomi){};
```

```
void Ovozi(){cout<<Nomi<<" ovozi: Vov"<<endl;}
```

```
void Ozuqasi(){cout<<Nomi<<" ozuqasi: Go'sht"<<endl;}
```

```
};
```

```
class Mushuk: public Jonivor
```

```
{
```

```
public:
```



```

Mushuk(char * nomi):Jonivor(nomi){};
void Ovozi(){cout<<Nomi<<" ovozi: Miyov"<<endl;}
void Ozuqasi(){cout<< Nomi<<" ozuqasi: Sut"<<endl;}
};
int main()
{
Mushuk mushuk("Baroq");
Kuchuk kuchuk("Tuzik");
mushuk.Ovozi();
mushuk.Ozuqasi();
kuchuk.Ovozi();
kuchuk.Ozuqasi();
}

```

Bu misolning e'tiborli tomoni shundaki, Jonivor sinfida e'lon qilingan ovozi() va ozuqasi() funksiya-a'zolar abstrakt funksiyalardir. Bu funksiyalarni konkretlashtirishning imkoni yo'q, chunki Jonivor sinfi hayvonlar shajarasini aniqlab beruvchi, umumlashtiruvchi sinf va konkret hayvon aniqlanmaguncha uning ovozinig qanday bo'lishi va nima bilan oziqlanishini bilib bo'lmaydi. Lekin, aksariyat hayvonlar ovoz chiqaradi va albatta oziqlanadi. Shu sababli, ovozi() va oziqasi() funksiyalar umumiy bo'lib, u Jonivor sinfida mavhum holda e'lon qilingan. Bu funksiyalar Kuchuk va Mushuk sinflarida konkretlashtirilgan (majburiy ravishda).

Dastur ishlashi natijasida ekranga quyidagi xabarlar chiqadi:

Baroq ovozi: Miyov

Baroq ozuqasi: Sut

Tuzik ovozi: Vov

Tuzik ozuqasi: Go'sht

Abstrakt sinf bilan bog'liq yana bir o'ziga xos holat shundan iboratki, agar abstrakt sinf konstruktori bevosita yoki bilvosita sof abstrakt funksiyani chaqirsa, nima ro'y berishini oldindan aytishning iloji yo'q.

Sof abstrakt funksiyalar tavsifiga "zid" ravishda bunday funksiyalar abstrakt sinfda nafaqat e'lon qilinishi, balkim aniqlanishi ham mumkin. Ular quyidagi sintaksis asosida bevosita chaqirilishi mumkin:

<abstrakt sinf nomi>::<abstrakt funksiya nomi> (<parametrlar ro'yxati>)

Odatda bu sintaksisdan sof virtual destruktorga ega sinflar shajarasini yaratishda foydalaniladi:

```

#include <iostream.h>
class Tayanch
{
public:
Tayanch(){};
virtual ~Tayanch()=0; // Sof virtual destruktorkor
};
Tayanch::~~Tayanch(){} // Destruktorni aniqlash
class Hosila: public Tayanch
{
public:
Hosila(){};
~Hosila(){};
};
void main()
{
Hosila * pHosila = new Hosila;

```

```
delete pHosila;
}
```

Ma'lumki, destruktorki virtual bo'lganda, oldin hosilaviy sinf destruktorki, keyin tayanch sinf destruktorki bajariladi. Sof virtual destruktorki aynan tayanch sinfda aniqlanishi, uning qandaydir amalga oshirilgan variantini yaratadiki, u destruktorki ketma-ketligini to'g'ri bajarilishini ta'minlaydi.

Xulosa sifatida abstrakt sinflarga qo'llaniladigan qoidalarni keltiramiz:

- abstrakt sinfni funksiyaga uzatiladigan argumentning turi sifatida ishlatib bo'lmaydi;
- abstrakt sinfni funksiya qaytaradigan qiymatning turi sifatida ishlatib bo'lmaydi;
- obyekt turini oshkor ravishda abstrakt sinf turiga keltirish mumkin emas;
- abstrakt sinf obyektini yaratib bo'lmaydi;
- abstrakt sinfga ko'rsatkich yoki adres olish amalini e'lon qilish mumkin.

Nazorat savollari

1. Kompilyatsiya jarayonidagi vaqtli va kechiktirilgan bog'lanishlar tushunchalarini izohlab bering.
2. C++ tilida dinamik polimorfizmni amalga oshirish mexanizmi qanday?
3. Sinflar shajarasida virtual va novirtual funksiyalar amal qilishini tushuntiring.
4. Sinf destruktorki virtual qilib aniqlanishiga sabab nima?
5. Sof virtual funksiya vazifasi nimadan iborat?

14- MA'RUZA

MAVZU: OPERATORLARNI QAYTA YUKLASH

Reja:

1. Operatorlarni qayta yuklash;
2. Funksiyalarni qayta yuklash.

Nazariy qism

C++ tilida o'rnatilgan operatorlarni qayta yuklash imkoniyati mavjud. Operatorlar global ravishda yoki sinf chegarasida qayta yukla-nishi mumkin. Qayta yuklangan operatorlar operator kalit so'zi yordamida funksiya ko'rinishida amalga oshiriladi. Qayta yuklanuvchi funksiya *operator funksiya* nomlanadi va nomi operatorX ko'rinishida bo'lishi kerak, bu erda X – qayta yuklanuvchi operator. C++ tilida qayta yukla-nishi mumkin bo'lgan operatorlar ro'yxati 13.1-jadvalida keltirilgan. Masalan, qo'shish operatorini qayta yuklash uchun operator+ nomli funksiyani aniqlash kerak bo'ladi. Agar qo'shish qiymat berish amali bilan kelgan holini qayta yuklash uchun operator+= ko'rinishida funksiya aniqlash zarur bo'ladi. Odatda kompilyator programma kodida qayta yuklangan operatorlar uchraganda ularni oshkormas ravishda qo'llaydi. Zarur bo'lganda ularni oshkor chaqirish mumkin:

```
Nuqta nuqta1, nuqta2, nuqta3;
```

```
// Qayta yuklangan qo'shish operatorini oshkor chaqirish
```

```
nuqta3=nuqta1.operator+(nuqta2);
```

13.1-jadval. Qayta yuklanuvchi operatorlar

Operator	Tavsifi	Toifasi
,	Vergul	Binar
!	Mantiqiy inkor	Unar
!=	Teng emas	Binar
%	Bo'lish qoldig'i	Binar
%=	Modulli bo'lish qiymat berish bilan	Binar

&	Razryadli VA	Binar
&	Adresni olish	Unar
&&	Mantiqiy VA	Binar
&=	Razryadli VA qiymat berish bilan	Binar
()	Funksiyani chaqirish	–
*	Ko‘paytirish	Binar
*	Vositali murojaat	Binar
*=	Ko‘paytirish qiymat berish bilan	Binar
+	Qo‘shish	Binar
+	Unar plyus	Unar
++	Inkrement	Unar
+=	Qo‘shish qiymat berish bilan	Binar
–	Ayirish	Binar
–	Unar minus	Unar
--	Dekrement	Unar
–=	Ayirish qiymat berish bilan	Binar
->	Elementini tanlash	Binar
->*	Elementini ko‘rsatkich orqali tanlash	Binar
/	Bo‘lish	Binar
/=	Bo‘lish qiymat berish bilan	Binar
<	Kichik	Binar
<=	Kichik yoki teng	Binar
<<	Razryad bo‘yicha chapga surish	Binar
<<=	CHapga surish qiymat berish bilan	Binar
=	Qiymat berish	Binar
= =	Teng	Binar
>	Katta	Binar
>=	Katta yoki teng	Binar
>>	Razryad bo‘yicha o‘ngga surish	Binar
>>=	O‘ngga surish qiymat berish bilan	Binar
[]	Massiv indeksi	–
^	Razryadli istisno qiluvchi YOKI	Binar
^=	Razryadli istisno qiluvchi YOKI qiymat berish bilan	Binar
	Razryadli YOKI	Binar
=	Razryadli YOKI qiymat berish bilan	Binar
	Mantiqiy YOKI	Binar
~	Bitli mantiqiy INKOR	Binar
delete	Dinamik ob’ektni yo‘qotish	–
new	Dinamik ob’ektni yaratish	–

13.2–jadvalda keltirilgan operatorlar qayta yuklanmaydigan operatorlar hisoblanadi.

13.2–jadval. Qayta yuklanmaydigan operatorlar

Operator	Tavsifi
.	A’zoni tanlash
::	Ko‘rinish sohasiga ruxsat berish operatori
.*	Ko‘rsatkich bo‘yicha a’zoni tanlash
?:	SHart amali
#	Preprotssessor belgilari
##	Preprotssessor belgilari

Qayta yuklanadigan operatorlarning operator funksiyalari, new va delete operatorlaridan tashqari, quyidagi qoidalar bo'yinishi kerak:

1) operator funksiya sinfning nostatik funksiya–a'zosi bo'lishi kerak yoki operator funksiya sinf yoki sanab o'tiladigan turdagi argument qabul qilishi kerak yoki operator funksiya sinf yoki sanab o'tiladigan turga ko'rsatkich yoki murojaat bo'lgan argumentlarni qabul qilishi kerak.

Masalan,

```
class Nuqta
{
public:
    //«kichik» operatori uchun operator funksiya-a'zoni
    // e'lon qilish
    Nuqta operator<(Point&);
    ...
    // Qo'shish operatorlarini e'lon qilish
    friend Nuqta operator+(Point&, int);
    friend Nuqta operator+(int, Point&);
};
```

Bu misolda «kichik» operatori sinfning funksiya–a'zosi sifatida e'lon qilingan, qo'shish operatori esa sinfning do'sti sifatida e'lon qilingan va u bitta operatorni qayta yuklashning bir nechta varianti bo'lishi mumkinligini ko'rsatadi;

2) operator funksiya operatorning argumentlar (operandlar) sonini, ularning ustunligi va bajarilish tartibini o'zgartira olmaydi;

3) sinf funksiya a'zosi sifatida e'lon qilingan unar operatorning operator funksiyasi parametrغا ega bo'lmasligi kerak. Agar operator funksiya global funksiya bo'lsa, u faqat bitta parametrغا ega bo'ladi;

4) sinf funksiya a'zosi sifatida e'lon qilingan binar operatorning operator funksiyasi bitta parametrغا ega bo'lishi kerak. Agar operator funksiya global funksiya bo'lsa, u faqat ikkita parametrغا ega bo'ladi;

5) operator funksiya kelishuv bo'yicha parametrlarga ega bo'lmasligi kerak;

6) sinf funksiya a'zosi sifatida e'lon qilingan operator funksiyaning birinchi parametri (agar u bo'lsa) sinf turida bo'lishi kerak. Chunki aynan shu sinf ob'ekti uchun mazkur operator chaqiriladi. Birinchi argument ustida hech qanday turga keltirish amali bajaril–masligi kerak;

7) qiymat berish operatorining operator funksiyasidan tashqari barcha operator funksiyalar vorislik bilan o'tadi;

8) =, (), [] va -> operatorlarning operator funksiyalari sinfning nostatik funksiya a'zolari bo'lishi kerak (va ular global funksiya bo'la olmaydi).

Operatorlarni qayta yuklash orqali, sinf chegarasida operatorning mohiyatini tubdan o'zgartirib yuborish mumkin. Lekin bu ishni zarurat bo'lgandagina amalga oshirgan ma'qul. Aks holda bajariladigan amal–larda mazmuniy xatolar yuzaga kelishi mumkin.

Binar operatorlarni qayta yuklash

Binar operatorning operator funksiyasi sinfning nostatik funksiya–a'zosi sifatida e'lon qilinganda u quyidagi sintaksisga ega bo'lishi kerak:

<qaytariladigan qiymat turi>operatorX(<parametr turi><parametr>);

Bu erda <qaytariladigan qiymat turi> – funksiya qaytaradigan qiymat turi, X– qayta yuklanadigan operator, <parametr turi> –parametr turi va <parametr> – funksiya parametri.

Funksiya parametrغا operatorning o'ng tomonidagi ob'ekt uzatiladi, operatorning chap tomonidagi ob'ekt esa nooshkor ravishda this ko'rsatkichi bilan uzatiladi.

Agar operator funksiya global deb e'lon qilinsa, u quyidagi ko'rinishga ega bo'ladi:

<qaytariladigan qiymat turi>operatorX(<parametr turi₁><parametr₁>,
<parametr turi₂><parametr₂>);

Bu erda funksiya parametrlarining kamida bittasi operator qayta yuklanayotgan sinf turida bo'lishi kerak.

Garchi operator funksiya qaytaradigan qiymat turiga hech qanday cheklov bo'lmasa ham, u sinf turida yoki sinfga ko'rsatkich bo'ladi.

Operator funksiyalarni yozishning bir nechta misollarini keltiramiz. Bu misollar operatorlarni qayta yuklashning to'liq imkoniyatlarini ochib bermasa ham, uning muhim qirralarini ko'rsatadi.

Birinchi navbatda operator funksiyaning sinfnining funksiya–a'zosi ko'rinishida aniqlashni ko'ramiz.

Quyidagi programmada Nuqta sinfi uchun qo'shish va ayirish operatorlarini qayta yuklash amalga oshirilgan.

```
#include <iostream.h>
class Nuqta{
    int x,y;
    public:
    Nuqta(){x=0; y=0;}
    Nuqta(int _x,int _y){x=_x; y=_y;}
    void Nuqta_Qiymati(int & _x,int & _y){_x=x; _y=y;}
    Nuqta operator+(Nuqta& ob);
    Nuqta operator-(Nuqta& ob);
};
Nuqta Nuqta::operator+(Nuqta& ob){
    Nuqta OraliqOb;
    OraliqOb.x=x+ob.x;
    OraliqOb.y=y+ob.y;
    return OraliqOb;
}
Nuqta Nuqta::operator-(Nuqta& ob){
    Nuqta OraliqOb;
    OraliqOb.x=x-ob.x;
    OraliqOb.y=y-ob.y;
    return OraliqOb;
}
int main(){
    int x,y;
    Nuqta A(100,200), B(50,100),C;
    C=A+B; // qayta yuklangan qo'shish operatori amal qiladi
    C.Nuqta_Qiymati(x,y);
    cout<<" C=A+B: "<<"C.x="<<x<<" C.y="<<y<<endl;
    A=A-B; // qayta yuklangan ayirish operatori amal qiladi
    A.Nuqta_Qiymati(x,y);
    cout<<" A=A-B: "<<"A.x="<<x<<" A.y="<<y<<endl;
    return 0;
}
```

Programma ishlashi natijasida ekranga quyidagi ko'rinishidagi natijalar chop etiladi:

C=A+B amali natijasi: C.x=150 C.y=300

A=A-B amali natijasi: A.x=50 A.y=100

Programmada shu narsaga e'tibor berish kerakki, operator funksiya parametri sinf ob'ektga murojaat ko'rinishida aniqlangan. Umuman olganda argument sifatida ob'ektni o'zini ham chaqirish mumkin, lekin funksiyadan chiqishda bu ob'ekt destruktord yordamida yo'qotiladi. Funksiya parametri sinf ob'ektga murojaat ko'rinishida bo'lishining afzalligi shundaki, funksiya chaqirilganda unga ob'ekt emas, balki ob'ektga ko'rsatkich uzatiladi va sinf nusxasi uchun chaqiriladigan destruktorni ishlatilmaydi. Operator funksiyalarning qaytaruvchi qiymati ayni shu sinf turida va hol ob'ektlarni nisbatan murakkab ifodalarda qo'llash imkonini beradi. Masalan, quyidagi amallar programma uchun ruxsat etilgan til ko'rsatmasi hisoblanadi:

C=A+B-C;

Ikkinchi tomondan, quyidagi ifoda ham o'rinli:

(A+B).Nuqta_Qiymati(x,y);

Bu ifodada qo'shish operatoring operator funksiyasidagi vaqtincha (OraliqOb) ob'ektning Nuqta_Qiymati() funksiyasi ishlatiladi.

Keyingi misol operator funksiya parametri sifatida sanab o'tiladigan turdagi berilgan kelgan holatini namoyon qiladi. Bu berilgan operatorning o'ng tomonida kelishiga e'tibor berish kerak.

```
#include <iostream.h>

class Nuqta{
    int x,y;
public:
    Nuqta(){x=0; y=0;}
    Nuqta(int _x,int _y){x=_x; y=_y;}
    void Nuqta_Qiymati(int & _x,int & _y){_x=x; _y=y;}
    Nuqta operator+(Nuqta& ob);
    Nuqta operator+(int n);
};

Nuqta Nuqta::operator+(Nuqta& ob){
    Nuqta OraliqOb;
    OraliqOb.x=x+ob.x;
    OraliqOb.y=y+ob.y;
    return OraliqOb;
}

Nuqta Nuqta::operator+(int n){
    Nuqta OraliqOb;
    OraliqOb.x=x+n;
    OraliqOb.y=y+n;
    return OraliqOb;
}

int main(){
    int x,y;
    Nuqta A(100,200), B(50,100),C;
    C=A+B; // parametri sinf turidagi ob'ekt bo'lgan
    // qayta yuklangan qo'shish operatori amal qiladi
    C.Nuqta_Qiymati(x,y);
    cout<<" C=A+B: "<<"C.x="<<x<<" C.y="<<y<<endl;
    C=A+30; // parametri sanab o'tiladigan turidagi ob'ekt
    //bo'lgan qayta yuklangan qo'shish operatori amal qiladi
    C.Nuqta_Qiymati(x,y);
    cout<<" C=A+30: "<<"C.x="<<x<<" C.y="<<y<<endl;
```

```

    return 0;
}

```

Programma ishlashi natijasida ekranga quyidagi ko‘rinishidagi natijalar chop etiladi:

C=A+B amali natijasi: C.x=150 C.y=300

C=A+30 amali natijasi: C.x=130 C.y=230

Operator funksiya parametri operatorning o‘ng tomonidagi operand ekanligi sababli kompilyator quyidagi ko‘rsatmalarni to‘g‘ri «tushunadi»:

C=A+30;

Lekin kompilyator

C=30+A;

ko‘rsatmasini qabul qilmaydi.

Bu muammoni operator funksiyaning «ichki» imkoniyatlari bilan hal qilib bo‘lmaydi. Muammoni do‘st operator funksiyalardan foydalanish orqali echish mumkin. Ma’lumki, do‘st funksiyalarga yashiringan this ko‘rsatkichi uzatilmaydi. SHuning uchun binar operator funksiyasi ikkita argumentga ega bo‘lishi kerak – birinchisi chap operand uchun, ikkinchisi o‘ng operand uchun.

```

#include <iostream.h>
class Nuqta{
    int x,y;
    public:
    Nuqta(){x=0; y=0;}
    Nuqta(int _x,int _y){x=_x; y=_y;}
    void Nuqta_Qiymati(int & _x,int & _y){_x=x; _y=y;}
    friend class Nuqta operator+(Nuqta& ob1, Nuqta& ob2);
    friend class Nuqta operator+(Nuqta& ob,int n);
    friend class Nuqta operator+(int n, Nuqta& ob);
};
Nuqta operator+(Nuqta& ob1,Nuqta& ob2){
    Nuqta OraliqOb;
    OraliqOb.x=ob1.x+ob2.x;
    OraliqOb.y=ob1.y+ob2.y;
    return OraliqOb;
}
Nuqta operator+(Nuqta& ob,int n){
    Nuqta OraliqOb;
    OraliqOb.x=ob.x+n;
    OraliqOb.y=ob.y+n;
    return OraliqOb;
}
Nuqta operator+(int n, Nuqta& ob){
    Nuqta OraliqOb;
    OraliqOb.x=ob.x+n;
    OraliqOb.y=ob.y+n;
    return OraliqOb;
}
int main(){
    int x,y;
    Nuqta A(100,200), B(50,100),C;

```

```

C=A+B;
C.Nuqta_Qiymati(x,y);
cout<<" C=A+B: "<<"C.x="<<x<<" C.y="<<y<<endl;
C=A+30;
C.Nuqta_Qiymati(x,y);
cout<<" C=A+30: "<<"C.x="<<x<<" C.y="<<y<<endl;
C=30+A;
C.Nuqta_Qiymati(x,y);
cout<<" C=30+A: "<<"C.x="<<x<<" C.y="<<y<<endl;
return 0;
}

```

Do'st funksiyalarni qayta yuklash hisobiga

```
C=A+30;
```

```
C=30+A;
```

til ko'rsatmalarini bajarish imkoniyati yuzaga keldi.

Taqqoslash va mantiqiy operatorlarni qayta yuklash

Taqqoslash va mantiqiy operatorlari, garchi binar operatorlar bo'lsa ham ular alohida qaraladi. Chunki ularga mos keluvchi operator-funksiyalar o'zlari aniqlangan sinf turini emas, balki mantiqiy qiymatlarni qaytarishi kerak (yoki true va false sifatida qabul qilinuvchi butun son qiymatini). Misol tariqasida, == va && operatorlarini qayta yuklashni ko'raylik.

```

#include <iostream.h>
class Nuqta{
    int x,y;
public:
    Nuqta(int _x,int _y){x=_x; y=_y;}
    Nuqta(){x=0; y=0;}
    Qiymat_xy(int & _x, int & _y){_x=x; _y=y;}
    bool operator==(Nuqta ob);
    bool operator&&(Nuqta ob);
};
bool Nuqta::operator==(Nuqta ob){
    return (x==ob.x && y==ob.y);
}
bool Nuqta::operator&&(Nuqta ob){
    return (x && ob.x) && (y && ob.y);
}
int main(){
    Nuqta Nuqta1(10,20), Nuqta2(10,25),
    Nuqta3(10,20),Nuqta4;
    if(Nuqta1==Nuqta2)
        cout<<"Nuqta1 va Nuqta2 o'zaro teng.\n";
    else cout<<"Nuqta1 va Nuqta2 o'zaro teng emas.\n";
    if(Nuqta1==Nuqta3)
        cout<<"Nuqta1 va Nuqta3 o'zaro teng.\n";
    else cout<<"Nuqta1 va Nuqta3 o'zaro teng emas.\n";
    if(Nuqta1 && Nuqta2) cout<<"Nuqta1 && Nuqta2 rost.\n";
    else cout<<"Nuqta1 && Nuqta2 yolg'on.\n";
}

```



```

if(Nuqta1 && Nuqta4) cout<<"Nuqta1 && Nuqta4 rost.\n";
else cout<<"Nuqta1 && Nuqta4 yolg'on.\n";
return 0;
}

```

Programma ishlashi natijasida ekranga quyidagilar chop etiladi:

Nuqta1 va Nuqta2 o'zaro teng emas.

Nuqta1 va Nuqta3 o'zaro teng.

Nuqta1 && Nuqta2 rost.

Nuqta1 && Nuqta4 yolg'on.

Operatorlarni qayta yuklash orqali koordinata nuqtalari orasi–dagi yangi mazmundagi munosabatlar aniqlandi.

Qiymat berish operatorini qayta yuklash

Qiymat berish operatori ham binar operator hisoblanadi, lekin uni qayta yuklash bir qator o'ziga xosliklarga ega:

- qiymat berish operatorining operator funksiyasi global ravishda e'lon qilinishi mumkin emas, ya'ni u faqat sinfning funksiya–a'zosi bo'lishi kerak;
- qiymat berish operatorining operator funksiyasi hosilaviy sinfga vorislik bilan o'tmaydi;
- kompilyator qiymat berish operatorining operator funksiyasi hosil qilishi mumkin, agar u sinfda aniqlanmagan bo'lsa.

Kompilyator tomonidan kelishuv bo'yicha hosil qilingan qiymat berish operatori sinfning har bir statik bo'lmagan a'zolariga qiymat berish amalini bajaradi. Lekin, agar sinfda ko'rsatkichlar bo'ladigan bo'lsa, bunday qiymat berish operatori ishlamaydi.

Qayd etish kerakki, qiymat berish operatori bajarilgandaen keyin chap tomondagi operand o'zgaradi, chunki unga yangi qiymat beriladi. SHuning uchun qiymat berish operatorining operator funksiyasi uni chaqirilgan ob'ektga ko'rsatkichni qaytarishi shart. Buning uchun funksiya nooshkor ravishda sinf funksiyalariga birinchi parametrlar sifatida uzatiladigan this ko'rsatkichini qaytarishi etarli. O'z navbatida funksiyaning this ko'rsatkichini qaytarishi quyidagi ko'rinishdagi qiymat berish amallarini «tushunish» imkonini beradi:

Nuqta1=Nuqta2=Nuqta3;

Quyidagi misol qiymat berish operatorini qayta yuklashni namoyon qiladi:

```

#include <iostream.h>
class Nuqta{
    int x,y;
    public:
    Nuqta(int _x,int _y){x=_x; y=_y;}
    Nuqta(){x=0; y=0;}
    Qiymat_xy(int & _x,int & _y){_x=x; _y=y;}
    bool operator==(Nuqta ob);
    Nuqta & operator=(Nuqta & ob);
};
bool Nuqta::operator==(Nuqta ob){
    return (x==ob.x && y==ob.y);
}
Nuqta & Nuqta::operator=(Nuqta & ob){
    if (this==&ob) return *this;
    x=ob.x;
    y=ob.y;
    return *this;
}

```

```

}
int main(){
    int a,b;
    Nuqta Nuqta1(10,20), Nuqta2(20,25), Nuqta3;
    Nuqta3=Nuqta2;
    if(Nuqta2==Nuqta3)
        cout<<"Nuqta2 va Nuqta3 o'zaro teng.\n";
    else cout<<"Nuqta2 va Nuqta3 o'zaro teng emas.\n";
    Nuqta3=Nuqta2=Nuqta1;
    if(Nuqta1==Nuqta3)
        cout<<"Nuqta1 va Nuqta3 o'zaro teng.\n";
    else cout<<"Nuqta1 va Nuqta3 o'zaro teng emas.\n";
    Nuqta3.Qiymat_xy(a,b);
    cout<<"Nuqta3.x="<<a<<" Nuqta3.y="<<b<<endl;
    return 0;
}

```

Programma ishlashi natijasida ekranga

Nuqta2 va Nuqta3 o'zaro teng.

Nuqta1 va Nuqta3 o'zaro teng.

Nuqta3.x=10 Nuqta3.y=20

xabarlari chop etiladi.

Qiymat berish amalini qayta yuklashda mazmunan xatoga olib keladigan

Nuqta1=Nuqta1;

ko‘rinishdagi o‘zini o‘ziga yuklash holati alohida nazorat qilinishi kerak bo‘ladi. CHunki qiymat berish operatori bajarilishida oldin chap tarafdagi operand xotirasi tozalanadi va keyinchalik shu joyga o‘ng tomondagi operandning haqiqatga to‘g‘ri kelmaydigan qiymatini joylashtiradi. SHu sababli, yuqoridagi misolda operator=() funksiyasi

if (this==&ob) return *this;

nazorat ko‘rsatmasiga ega va u xatolik ro‘y berishiga yo‘l qo‘ymaydi.

Unar operatorlarni qayta yuklash

Unar operatorlar uchun faqat bitta operand kerak bo‘ladi. Unar operatorni sinfning funksiya–a’zosi ko‘rinishida qayta yuklashda bu yagona operand bu amalni chaqirgan ob’ektning o‘zi hisoblanadi. SHu sababli, unar operatorning operator funksiyasi nostatik funksiya-a’zo sifatida e’lon qilinadi va u quyidagi ko‘rinishga ega bo‘ladi:

<qaytaruvchi qiymat turi > operatorX();

bu erda <qaytaruvchi qiymat turi > funksiya qaytaradigan qiymat turi, X– qayta yuklanayotgan unar operator.

Agar operator funksiya global ravishda e’lon qilinganda, u quyidagi sintaksisga javob berishi kerak:

<qaytaruvchi qiymat turi > operatorX(<parametr turi> <parametr>);

bu erda <parametr turi> – parametr turi va <parametr> – funksiya parametri.

Plyus va minus unar operatorlarni qayta yuklashga misol ko‘raylik.

#include <iostream.h>

class Nuqta{

int x,y;

public:

Nuqta(int _x,int _y){x=_x; y=_y;}

Nuqta(){x=0; y=0;}

```

    Qiymat_xy(int & _x,int & _y){_x=x; _y=y;}
    Nuqta operator+();
    Nuqta operator-();
};
Nuqta & Nuqta::operator+(){
    x+=x;
    y+=y;
    return *this;
}
Nuqta & Nuqta::operator-()
{
    x=-x;
    y=-y;
    return *this;
}
int main(){
    int a,b;
    Nuqta N1(-10,20);
    N1=+N1;           //qayta yuklangan plyus operatorini chaqirish
    N1.Qiymat_xy(a,b);
    cout<<"N1.x="<<a<<" N1.y="<<b<<endl;
    N1=-N1;           //qayta yuklangan minus operatorini chaqirish
    N1.Qiymat_xy(a,b);
    cout<<"N1.x="<<a<<" N.y="<<b<<endl;
    return 0;
}

```

Programma ishlashi natijasida ekranda

N1.x=-10 N1.y=20

N1.x=10 N1.y=-20

satrlari paydo bo'ladi.

Xuddi shu natijalarga global operator funksiyalarni sinfning do'st funksiyalari ko'rinishida e'lon qilish orqali erishish mumkin:

```

friend Nuqta operator+(Nuqta & ob);
friend Nuqta operator-(Nuqta & ob);
Bu funksiyalar aniqlanishi
friend Nuqta operator+(Nuqta & ob){
    ob.x+=ob.x;
    ob.y+=ob.y;
    return ob;
}
friend Nuqta operator-(Nuqta & ob){
    ob.x=-ob.x;
    ob.y=-ob.y;
    return ob;
}

```

Ushbu funksiyalarni chaqirish natijalari yuqoridagi funksiyalar bilan bir xil bo'ladi.

Inkrement va dekrement operatorlarini qayta yuklash

Inkrement va dekrement operatorlari, ularning prefiks va postfiks ko'rinishlari bo'lishi hisobiga qayta yuklash nuqtai-nazaridan alohida kategoriyaga tushadi. Prefiks va postfiks ko'rinishlarni farqlash uchun quyidagi qoidalarga amal qilinadi: prefiks ko'rinishni qayta yuklash, odatdagi unar operatorni qayta yuklash bilan bir xil; postfiks ko'rinish uchun operator funksiya qo'shimcha int turidagi parametrga ega bo'ladi. Amalda bu parametr ishlatilmaydi va funksiyaning chaqirishda uning qiymati 0 bo'ladi (zarurat bo'lganda ishlatilishi mumkin). Bu parametrning vazifasi – kompilyatorga operatorning postfiks ko'rinishi ishlatilayotganligini bildirishdir. Quyidagi misolda Nuqta sinfi uchun inkrement va dekrement operatorlarini qayta yuklash ko'rsatilgan:

```
#include <iostream.h>
class Nuqta{
    int x,y;
public:
    Nuqta(int _x,int _y){x=_x; y=_y;}
    Nuqta(){x=0; y=0;}
    Qiymat_xy(int & _x,int & _y){_x=x; _y=y;}
    Nuqta & operator++(); // prefiks inkrement uchun
    Nuqta operator++(int); // postfiks inkrement uchun
    Nuqta & operator--(); // prefiks dekrement uchun
    Nuqta operator--(int); // postfiks dekrement uchun
};
Nuqta & Nuqta::operator++(){
    x++;
    y++;
    return *this;
}
Nuqta Nuqta::operator++(int){
    Nuqta Oraliq=*this;
    ++*this;
    return Oraliq;
}
Nuqta & Nuqta::operator--(){
    x--;
    y--;
    return *this;
}
Nuqta Nuqta::operator--(int){
    Nuqta Oraliq=*this;
    --*this;
    return Oraliq;
}

int main(){
    int a,b;
    Nuqta N1(-10,20);N2(15,25),N3;
    ++N1;      //prefiks inkrement operatorini chaqirish
    N1.Qiymat_xy(a,b);
    cout<<"(++N1).x="<<a<<" (++N1).y="<<b<<endl;
```

```

N1++;      //postfiks inkrement operatorini chaqirish
N1.Qiymat_xy(a,b);
cout<<"(N1++).x="<<a<<" (N1++).y="<<b<<endl;
N3=--N2; //prefiks dekrement operatorini chaqirish
N3.Qiymat_xy(a,b);
cout<<"N3=--N2; => N3.x="<<a<<" N3.y="<<b<<endl;
//postfiks dekrement operatorini chaqirish
(N3=N1--).Qiymat_xy(a,b);
cout<<"(N3=N1--).x="<<a<<" (N3=N1--).y="<<b<<endl;
N1.Qiymat_xy(a,b);
cout<<"N1.x="<<a<<" N1.y="<<b<<endl;
N2.Qiymat_xy(a,b);
cout<<"N2.x="<<a<<" N2.y="<<b<<endl;
return 0;
}

```

Programma ishlashi natijasida ekranga

```

(++N1).x=11 (++N1).y=21
(N1++).x=12 (N1++).y=22
N3=--N2; => N3.x=14 N3.y=24
(N3=N1--).x=12 (N3=N1--).y=22
N1.x=11 N1.y=22
N2.x=14 N2.y=24

```

xabarlari chiqadi.

Programmada inkrement va dekrement operatorlarining prefiks va postfiks ko‘rinishlarini qayta yuklashni amalga oshirishda o‘ziga xos yo‘l tanlangan. Masalan, inkrement operatorining prefiks ko‘rinishi uchun aniqlangan operator funksiyaning qaytaradigan qiymati sinf ob‘ektiga murojaat, chunki inkrement operatorining postfiks ko‘rinish uchun aniqlangan operator funksiya shu funktsiyani chaqiradi va o‘zgargan ob‘ektni qaytarib olishi kerak. Umuman olganda, bu funktsiyalarni bir–biriga bog‘liqmas ravishda aniqlash mumkin:

```

Nuqta Nuqta::operator++(){
    x++;
    y++;
    return *this;}
Nuqta Nuqta::operator++(int){
    x++;
    y++;
    return *this; }

```

Lekin bu variantda bir xil amallar ketma-ketligini takror yoziladi va inkrement operatorini turlicha talqin qilish bilan bog‘liq xatolarni yuzaga kelishiga zamin bo‘ladi. Ma’qul variant– bu operatorning prefiks ko‘rinishining operator funksiyasida operator mazmuni aniqlanadi va postfiks ko‘rinishni qayta yuklash unga tayanadi.

Endi inkrement va dekrement operatorlarini do‘st funktsiyalar orqali qayta yuklashni ko‘ramiz. SHunga e’tibor berish kerakki, do‘st funktsiyaga murojaat ko‘rinishidagi argument uzatilishi va u o‘zgartirilib funksiya tomonidan qaytarilishi kerak bo‘ladi.

```

#include <iostream.h>
class Nuqta{
    int x,y;
public:
    Nuqta(int _x,int _y){x=_x; y=_y;}

```

```

Nuqta(){x=0; y=0;}
Qiymat_xy(int & _x,int & _y){_x=x; _y=y;}
friend Nuqta & operator++(Nuqta &);// prefiks inkrement
//postfiks inkrement
friend Nuqta operator++(Nuqta&,int);
friend Nuqta & operator--(Nuqta&); // prefiks dekrement
// postfiks dekrement
friend Nuqta operator--(Nuqta&int);
};
Nuqta & operator++(Nuqta & ob){
    ob.x++;
    ob.y++;
    return ob;
}
Nuqta operator++(Nuqta & ob,int){
    Nuqta Oraliq=ob;
    ++ob;
    return Oraliq;
}
Nuqta & operator--(Nuqta &){
    ob.x--;
    ob.y--;
    return ob;
}
Nuqta operator--(Nuqta &,int){
    Nuqta Oraliq=ob;
    --ob;
    return Oraliq;}
int main(){
    int a,b;
    Nuqta N1(-10,20);N2(15,25),N3;
    ++N1;      //prefiks inkrement operatorini chaqirish
    N1.Qiymat_xy(a,b);
    cout<<"(++N1).x="<<a<<" (++N1).y="<<b<<endl;
    N1++;      //postfiks inkrement operatorini chaqirish
    N1.Qiymat_xy(a,b);
    cout<<"(N1++).x="<<a<<" (N1++).y="<<b<<endl;
    N3--N2; //prefiks dekrement operatorini chaqirish
    N3.Qiymat_xy(a,b);
    cout<<"N3--N2; => N3.x="<<a<<" N3.y="<<b<<endl;
    //postfiks dekrement operatorini chaqirish
    (N3=N1--).Qiymat_xy(a,b);
    cout<<"(N3=N1--).x="<<a<<" (N3=N1--).y="<<b<<endl;
    N1.Qiymat_xy(a,b);
    cout<<"N1.x="<<a<<" N1.y="<<b<<endl;
    N2.Qiymat_xy(a,b);
    cout<<"N2.x="<<a<<" N2.y="<<b<<endl;
    return 0;}

```

Programma ishlashi natijasida ekranga xuddi oldingi misoldagidek xabarlar chop etiladi.

Yuqorida qayd qilingandek, int turidagi argument odatda ishlatishmaydi, lekin zarur bo'lganda ishlatishi mumkin. Bu argumentni ishlatishga misol:

```
#include <iostream.h>
class Nuqta{
    int x,y;
    public:
    Nuqta(int _x,int _y){x=_x; y=_y;}
    Nuqta(){x=0; y=0;}
    Qiymat_xy(int & _x,int & _y){_x=x; _y=y;}
    Nuqta & operator++(int);
};
Nuqta & Nuqta::operator++(int n){
    if (n!=0) {
        x+=n;
        y+=n; }
    else {
        x++;
        y++; }
    return *this;}
int main(){
    Nuqta N1(10,20);
    N1.operator++(100); //100 soniga inkrement
    return 0;}
```

Bu holatning o'ziga xosligi shundaki, postfix inkrement operatorining operator funksiyasini oshkor ravishda chaqirishga to'g'ri keladi. Chunki kompilyator

```
N1++(100);
```

ifodasini operatorgacha bo'lgan qismini alohida ajratib

```
N1++;
```

ko'rsatma deb tushunadi.

Indekslash operatorini qayta yuklash

Kvadrat qavslar ('[' , ']') bilan yoziladigan indekslash operatori binar operator hisoblanadi va u qayta yuklanishida operator funksiya sinfning bitta argumentli nostatik funksiya–a'zosi sifatida aniqlanishi kerak. Funksiya argumenti ixtiyoriy turda bo'lishi va u sinf ob'ektlari massivining indeksi deb qabul qilinadi. Quyidagi misol buni namoyon qiladi:

```
#include <iostream.h>
class BS_Massiv{
    int MaxIndex;
    int * kButun;
    public:
    BS_Massiv(int Elem_Soni)
    ~BS_Massiv(){delete kButun;}
    int & operator[](int index);
};
BS_Massiv::BS_Massiv(int Elem_Soni){
```

```

    kInt=new int(Elem_Soni);
    MaxIndex=Elem_Soni;
}
int & BS_Massiv::operator[](int index){
    static int iXato=-1;
    if(index>=0 && index<MaxIndex) return kInt[index];
    else {
        cout<<"Xato: Massiv chegarasidan chiqish ro'y berdi!";
        cout<<endl;
        return iXato; }
}
main(){
    BS_Massiv vector(5);
    for(int i=0; i<5; i++)
        vector[i]=i;
    for(int i=0; i<=5; i++)
        cout<<" vector["<<i<<"]="<<vector[i]<<endl;
    return 0;
}

```

Programmada 5 ta, butun son turidagi elementlardan tashkil topgan vector massivi BS_Massiv sinfining ob'ekti sifatida e'lon qilingan va unga qiymatlar berilib, keyin chop qilingan. Indeks argumenti i=5 bo'lganda xatolik haqida xabar beriladi. Programma ishlashi natijasida ekranga quyidagilar chiqadi:

```

vector[0]=0
vector[1]=1
vector[2]=2
vector[3]=3
vector[4]=4
Xato: Massiv chegarasidan chiqish ro'y berdi!
vector[5]=-1

```

SHunga e'tibor berish kerakki, operator[] funksiyasi murojaat qaytaradi (son qiymatini emas) va shu sababli bu funktsiyani qiymat berish operatorining ikki tomonida ham qo'llash imkoniyatiga ega bo'lgan binar operator hisoblanadi.

14.2- mavzu: Funktsiyalarni qayta yuklash

Ishdan maqsad: Funktsiyalarni qayta yuklash ko'nikmalarini egallash.

Nazariy qism

Qavslar vositasida amalga oshiriladigan funktsiyani chaqirish operatori quyidagi sintaksisga ega bo'lgan binar operator hisoblanadi:

```
<ifoda>(<ifodalar ro'yxati>)
```

Bu erda <ifoda> – birinchi operand, hamda <ifodalar ro'yxati> – majburiy bo'lmagan ikkinchi operanddir. Funktsiyani chaqirish operatorining operator funktsiyasi sinfining nostatik funktsiya – a'zo ko'rinishida e'lon qilinishi kerak. Funktsiyani chaqirish operatorini qayta yuklashga zarurat, odatda ko'p parametrlarni talab qiladigan amallarni bajarishda yuzaga keladi.

Funksiyani qayta yuklash operatori qayta yuklanganda, u faqat qavs ichidagi o'zi e'lon qilingan sinf ob'ektlariga bo'lgan murojaatni o'zgartiradi, lekin funksiya chaqirilish jarayoniga ta'sir qilmaydi.

Funksiyani chaqirish operatorini qayta yuklashga misol ko'raylik:

```
#include <iostream.h>
class Nuqta{
    int x,y;
    public:
    Nuqta(int _x,int _y){x=_x; y=_y;}
    Nuqta(){x=0; y=0;}
    Qiymat_xy(int & _x,int & _y){_x=x; _y=y;}
    Nuqta & operator()(int dx, int dy)
    { x+=dx; y+=dy; return *this;};
};
int main(){
    int x,y;
    Nuqta N1,N2;
    // Nuqta sinfining qayta yuklangan funksiyaning chaqirish
    // operatorining operator funksiyasini N1 ob'ekt uchun
    // chaqirish.
    N2=N1(5,10);
    N2.Qiymat_xy(x,y);
    cout<<"1-chaqirishda: N2.x="<<x<<" N2.y="<<y<<endl;
    N2=N2(1,2);//qayta yuklangan funksiyaning chaqirish operatori
    N2.Qiymat_xy(x,y);
    cout<<"2-chaqirishda: N2.x="<<x<<" N2.y="<<y<<endl;
    return 0;
}
```

Programma ishlashi natijasida ekranga quyidagi satrlar chiqadi:

1-chaqirishda: N2.x=5 N2.y=10

2-chaqirishda: N2.x=6 N2.y=12

Keltirilgan misol uchun

Nuqta N1(5,10);

va

N1(5,10);

ifodalarini chalkashtirmaslik kerak. Birinchi ifoda sinfning parametrli konstruktoriga murojaatni anglatadi, ikkinchisi – sinfdagi funksiyaning chaqirish operatori qayta yuklangan operator funksiyaning chaqirish kerakligini bildiradi. Misol shuni ko'rsatadiki, funksiyaning chaqirish operatori qayta yuklangan sinf ob'ektiga funksiya vositasida murojaat qilish mumkin.

Sinf a'zolariga murojaat operatorlarini qayta yuklash

Sinf a'zolariga murojaat operatorlarini qayta yuklash a'zolari tanlash operatorini (->) qayta yuklash orqali amalga oshiriladi («.» operatori qayta yuklanmaydi). Bu operator unar hisoblanadi va uning operator funksiyasi sinfning nostatik a'zosi qilib e'lon qilinishi kerak. Mos operator funksiyaning ko'rinishi quyidagicha:

<sinf nomi>* operator->() {<til ko'rsatmalari>;

Bu erda <sinf nomi> – operator tegishli bo‘lgan sinf nomi. Sinf a’zolarini tanlash operatorini qayta yuklash, odatda qo‘llanishi ayni paytda o‘rinlimi yoki yo‘qligini nazorat qiluvchi «aqli» ko‘rsatkichlarni amalga oshirishda ishlatiladi.

Sinf a’zolarini tanlash operatorini qayta yuklashga misol keltiramiz:

```
#include <iostream.h>
class Nuqta{
    int x,y;
public:
    Nuqta(int _x,int _y){x=_x; y=_y;}
    Nuqta(){x=0; y=0;}
    Qiymat_X(){return x;}
    Qiymat_Y(){return y;}
    Nuqta & operator->();
};
Nuqta * Nuqta::operator->(){
    cout<<"Ob'ect elementiga murojaat: ";
    return this;
}
int main(){
    Nuqta N(5,10);
    // Nuqta sinfining qayta yuklangan a'zolarini tanlash
    // operatorining operator funksiyasini N ob'ekt uchun
    // chaqirish.
    cout<<"N->x = "<<N->Qiymat_X()<<endl;
    cout<<"N->y = "<<N->Qiymat_Y()<<endl;
    return 0;
}
```

Programma bajarilganda, ekranga quyidagi satrlar chop etiladi:

Ob'ect elementiga murojaat: N->x = 5

Ob'ect elementiga murojaat: N->y = 10

Kompilyator tomonidan "N->Qiymat_X()" ko‘rsatmasini

(N.operator->())->Qiymat_X()

ko‘rinishida talqin qilinishi operator funksiyani qanday bajarili-shining ichki mohiyatini ochib beradi.

new va delete operatorlarini qayta yuklash

Xotirani dinamik ravishda ajratish va tozalash vazifasini bajaruvchi new va delete operatorlari bajarilganda mos ravishda standart aniqlangan operator new() (yoki operator new []() - massiv uchun qo‘llanilganda) va operator delete() (yoki operator delete []() - massiv uchun qo‘llanilganda) maxsus funksiyalari chaqiriladi.

Bundan keyin alohida zarurat bo‘lmasa, bu funksiyalarning massiv varianti qaralmaydi va bittasi uchun aytilgan fikrlar ikkinchisi uchun ham o‘rinli deb hisoblanadi.

Umuman olganda, new va delete operatorlari ikki xil variantda qayta yuklanishi mumkin:

::operator new()	- global (standart);
::operator delete()	- global (standart);
<sinf nomi>:: operator new()	- sinf funksiyasi;
<sinf nomi>:: operator delete()	- sinf funksiyasi.

Sinfda aniqlangan operator `new()` operator funksiya sinfnining statik a'zosi bo'lib, u sinf ob'ektlari uchun global `::operator new()` funksiyasini yashiradi (berkitadi).

Global `::operator new()` – operator funksiyaning o'zi ham qayta yuklanishi (sinfdan tashqarida) va qayta yuklanuvchi funksiyalarning turli prototipga ega bir nechta variantlari bo'lishi mumkin.

Foydalanuvchi tomonidan aniqlanadigan `new` operatorning operator funksiyasi `void*` qiymatini qaytarishi kerak va birinchi parametr sifatida `size_t` turidagi parametrga ega bo'lishi kerak. Oxirgi parametr «`stddef.h`» sarlavha faylida `unsigned int` ko'rinishida aniqlangan. `new` operatorini qayta yuklash uchun quyidagi prototipdan foydalaniladi:

`void*operator new(size_t size); // ob'ektlar uchun`

`void*operator new[](size_t size); //ob'ektlar massivi uchun`

Bu prototiplar «`new.h`» sarlavha faylida joylashgan. SHu sababli, agar `new` va `delete` operatorlarni qayta yuklash zarur bo'lganda bu faylni programmaga qo'shish kerak.

SHunga e'tibor berish kerakki, `new` operator funksiyasidagi `size` parametrining baytlardagi o'lchamini (`sizeof(size)`), ya'ni xotiradan ajratilishi kerak sohaning baytlardagi o'lchamini operator funksiya uchun kompilyatorning o'zi hisoblab beradi. Operator funksiyaga bu parametrning qo'yilishiga sabab shundaki, hosilaviy sinflar operator `new()` va operator `new[]()` funksiyalarini vorislik bilan oladi va hosilaviy sinf ob'ektlarining o'lchami tayanch sinf ob'ektlari o'lchamidan farq qilishi mumkin.

Agar xotiradagi oldin ajratilgan joyni qaytadan «taqsimlash» zarur bo'lsa, `new` operatorining qo'shimcha parametrga ega bo'lgan *joylashuvchi shaklidan* foydalanishi mumkin. Mos operator funksiya sintaksisi quyidagi ko'rinishiga ega:

`// ob'ektlar uchun`

`void* operator new(size_t size, void* p);`

`// ob'ektlar massivi uchun`

`void* operator new(size_t Type_size, void* p);`

Odatda `new` operatorining joylashuvchi shaklidan global ob'ektlar uchun, turga keltirish amali bajarilgan holda qo'llaniladi. Bu variantda absolyut adres bo'yicha oldindan ajratilgan joyga ob'ektni joylashtirish amalga oshiriladi. Ko'rsatilgan adres uyumdan bo'lishi shart emas. Joylashadigan ob'ekt o'lchami ko'rsatilgan sohada o'lchamidan kichik bo'lgan hollarda ajratilgan sohani korrekt ravishda tozalash foydalanuvchi zimmasiga yuklatiladi, chunki `delete` operatori to'g'ri ishlashiga kafolat yo'q.

Qayta yuklangan `new` operatorini chaqirish sintaksisi quyidagicha:

`<::> new <tur uzunligi> <tur nomi> <(<initsializator>)> yoki`

`<::> new <tur uzunligi> (<tur nomi>) <(<initsializator>)>`

Bu erda

`<::>` - shart bo'lmagan, ko'rinish sohasiga ruxsat berish operatori;

<tur uzunligi> - operator new() funksiyasining size parametri, ko'rsatilmasligi mumkin;

<tur nomi> - xotira ajratiladigan berilganning turi;

<(<initsializator>)> - <tur nomi> turining konstruktori uchun uzatiladigan boshlang'ich qiymatlar ro'yxati. Ro'yxat ko'rsatilmasligi mumkin.

Sintaksis shuni ko'rsatadiki, new operatorini ob'ektga boshlang'ich qiymat berish bilan chaqirish mumkin. Lekin, bu operatorni ob'ektlar massivi uchun chaqirilganda initsializatsiyani ishlatib bo'lmaydi va ob'ektlarning boshlang'ich qiymatlari noaniq bo'ladi.

Global new va delete operatorlarini qayta aniqlash va qayta yuklashga misol keltiramiz:

```
#include <iostream.h>
#include <stdio.h>
// Global new operatorini qayta aniqlash
void* operator new(size_t size)    {
    cout<<"Xotiradan "<<size;
    cout<<" bayt ajratishga so'rov bo'ldi\n";
    return malloc(size);
}
void operator delete(void *p)      {
    cout<<"Xotirani bo'shatish!\n";
    free(p);
}
// Global new operatorini qayta yuklash
void* operator new(size_t size, char * fname, int line){
    cout<<fname<<" faylining "<<line<<"-qatorida \n";
    cout<<size<<" bayt ajratishga so'rov bo'ldi!\n";
    return malloc(size);
}
int main(){
    char fayl_nomi[]="new_quyk.cpp";
    int qator=5;
    // Global new operatorini chaqirish
    long * pInt = new long;
    // Global delete operatorini chaqirish
    delete pInt;
    // Qayta yuklangan new operatorini chaqirish
    pInt = new(fayl_nomi, qator) long;
    // Global delete operatorini chaqirish
    delete pInt;
    return 0;
}
```

Programma ishlashi natijasida ekranga quyidagi satrlar chop etiladi:

Xotiradan 4 bayt ajratishga so'rov bo'ldi

Xotira bo'shatildi!

new_quyk.cpp faylining 5-qatorida

4 bayt ajratishga so'rov bo'ldi!

Xotira bo'shatildi!

Navbatdagi misolda new operatorining joylashuvchi shaklidan foydalanish ko'rsatilgan.

```
#include <iostream.h>
// Global new operatorini qayta aniqlash
void* operator new(size_t sizeb void * krst) {
    cout<<"Ob'ekt ko'rstatilgan adresga joylandi\n";
    return krst;
}
void Tizimni_tekshirish() {
    cout<<"Tizim normal ishlaypti!\n";
}
class Nuqta{
    int x,y;
    public:
    Nuqta(int _x,int _y){
        x=_x; y=_y;
        cout<<" Ob'ektlar berilgan qiymatlar bilan";
        cout<<" initsializatsiyalandi.\n";
        cout<<" x="<<x<<" y="<<y<<"\n";
    }
    Nuqta(){
        x=0; y=0;
        cout<<" Ob'ektlarni kelishuv bo'yicha";
        cout<<" initsializatsiyalandi.\n";
        cout<<" x="<<x<<" y="<<y<<"\n";
    }
    ~Nuqta() {cout<<"Nuqta::~~Nuqta() ishladi\n";}
}
// Global ob'ekt
Nuqta nuqta;
int main(){
    // Qandaydir ishni bajarish
    Tizimni_tekshirish();
    // Endi ob'ektni initsializatsiyalash mumkin!
    Nuqta * krst=new(nuqta) Nuqta(10,20);
    krst->Nuqta::~~Nuqta();
    return 0;
}
```

Programma natijasi quyidagi xabarlar bo'ladi:

```
Ob'ektlarni kelishuv bo'yicha initsializatsiyalandi.
x=0 y=0
Tizim normal ishlaypti!
Ob'ekt ko'rstatilgan adresga joylandi
Ob'ektlar berilgan qiymatlar bilan initsializatsiyalandi.
x=10 y=20
Nuqta::~~Nuqta() ishladi
Nuqta::~~Nuqta() ishladi
```

Keyingi programma new operatorini qayta yuklashni sinf ichida joylashuvchi shaklidan foydalangan holda amalga oshirishga misol:

```
#include <iostream.h>
class Satr {
    union {
        char ch;
        char buf[81];
    };
public:
    Satr(char c='\0'): ch(c) {
        cout<<"Satr sinfining belgili konstruktori\n";
    }
    Satr(char * s){
        cout<<"Satr sinfining satrli konstruktori\n";
        strcpy(buf,s);
    }
    ~Satr(){cout<<"Satr::~~Satr()"<<endl;}
    // new operatorining joylashuvchi sintaksisi
    void* operator new(size_t, void * buffer)
    { return buffer; }
};
char satr_buffer[sizeof(Satr)]; // xotira buferi
int main(){
    // Satrni buferga joylashtirish
    Satr * krst=new(satr_buffer) Satr("C++");
    cout<<"satr_buffer"<<satr_buffer<<endl;
    // Destruktorni oshkor ravishda chaqirish
    krst->Satr::~~Satr();
    // 'c' belgisini satr boshiga joylashtirish
    krst=new(satr_buffer)Satr('c');
    cout<<"satr_buffer[0]="<<satr_buffer[0]<<endl;
    cout<<"Buferning yangi qiymati"<<endl;
    cout<<"satr_buffer="<<satr_buffer<<endl;
    // Destruktorni oshkor ravishda chaqirish
    krst->Satr::~~Satr();
    return 0;
}
```

Programmada global ob'ekt satr_buffer satri yaratiladi va shu xotira sohasiga Satr sinfi ob'ekti "C++" qiymati bilan joylashtiriladi. Oshkor ravishda sinf destruktorni chaqirish orkali sinf ob'ekti yo'qotiladi. Keyin, xuddi shu adresda Satr sinfining ikkinchi ob'ektni 'c' belgisi bilan yaratiladi, xotiraning satr_buffer adresli sohasida satr qiymat chop etiladi va sinf ob'ekti yo'qotiladi.

Nazorat savollari:

1. Operator nima?
2. Operatorni qayta yuklash nimaga kerak?
3. Funksiyalarni qayta yuklash nima uchun kerak?

4. Qayta yuklash jarayoni nimadan tashkil topadi?
5. Qanday operator turlarini bilasiz?
6. Clasdand olingan obyektga qayta yuklashni nima dahli bor?
7. Kiritish chiqarish operatorlarni ham qayta yuklab bo'ladi?
8. Qanday operatorlarni qayta yuklab bo'lmaydi?
9. Eng ko'p qaysi operator qayta yuklanadi?
10. Bitta classdand olingan boyektlar uchun operatorlarni qayta yuklash shartmi?

15 – MA'RUZA

MAVZU: FUNKSIYA VA SINIF SHABLONLARI

Reja:

Kirish;

1. Shablonlar nazariyasi;
2. Misol: *Umumiy toifa*;
3. Funksiya shablonlari (function template);
4. Sinf shablonlari (class template);
5. Foydalanilgan adabiyotlar;
6. Nazorat savollari (40 ta);
7. Test savollari (Darajalarga bo'lingan 60 ta);

Annatatsiya:

Ushubu ma'ruz materialida C++ da shablonlar yaratish usullariga bag'ishlangan. Ma'ruzada shablonlar va ularning qo'llanilishi, funksiya shablona va class shablona hamda ularning qo'llanilish usullari, muammolari, murakkabliklari batavfsil yoritilgan. Funksiya va class shablonlari ko'pgina holatlarni inobatga olgan holda amaliy masalalar yordamida aniq va to'liq tushuntirilgan.

Kalit so'zlar:

✓ *template*

✓ *template class*

✓ *template function*

✓ *late prefix*

✓ *type paramete*

Mavzuga kirish

C++ da umumiy turlardan foydalangan holda, shablon funksiyalar va sinflarni aniqlashimiz mumkin.

C++ tili qayta foydalaniluvchi dasturiy ta'minotni ishlab chiqish uchun shablon funksiyalar va sinflar bilan ta'minlaydi. Shablonlar funksiyalar va sinflarda turlarni muvofiqlashtirish (sozlash) qobiliyatini taqdim etadi. Bunday qobiliyat bilan, kompilyator aniq bir tur o'rnida qabul qila oladigan umumiy turga sifatida bitta funksiya yoki bitta sinfni aniqlashimiz mumkin. Masalan, biz umumiy turdagi ikkita son dan kattasini topish uchun bitta funksiyani aniqlashimiz mumkin. Agar bu funksiyani ikkita **int** argumentlar orqali chaqirsak, umumiy tur **int** turi bilan almashadi. Agar bu funksiyani ikkita **double** argumentlar orqali chaqirsak, umumiy tur **double** turi bilan almashadi.

Mazkur va bundan keying ma'ruzada shablonlar tushunchasi yoritib beriladi va siz qanday qilib funksiya shablonlari yoki sinf shablonlarini aniqlashni hamda ularni aniq turlar bilan ishlatishni o'rganib olishingiz mumkin. Shuningdek, ko'p qo'llaniluvchi, massivlarni almashtirishda qo'llashingiz mumkin bo'lgan umumiy **vector** shablonlarini ham o'rganishingiz mumkin.

1. Shablonlar nazariyasi

Shablonlar sinflar va funksiyalarda turlarni muvofiqlashtirish imkonini taqdim etadi. Biz funksiyalarni yoki sinflarni kompilyator tomonidan aniq bir tur o'rnida qabul qilinuvchi umumiy tur bilan aniqlashimiz mumkin.

Keling, shablonlarga bo'lgan ehtiyojni ko'rsatib berish uchun, oddiy bir misoldan boshlaymiz. Faraz qilaylik, biz ikkita butun sonlar, ikkita dubl sonlar, ikkita belgilar va ikkita satrlardan kattasini topmoqchimiz. Shu kungacha o'rgangan bilimlarimiz asosida, biz quyidagicha ko'rinishdagi to'rtta ko'p yuklanuvchi funksiyalarni aniqlashimiz mumkin:

```
1.      int maxValue(int value1, int value2)
2.      {
3.      if (value1 > value2)
4.      return value1;
5.      else
6.      return value2;
7.      }
8.      double maxValue(double value1, double value2)
9.      {
10.     if (value1 > value2)
11.     return value1;
12.     else
13.     return value2;
14.     }
15.     char maxValue(char value1, char value2)
16.     {
17.     if (value1 > value2)
18.     return value1;
19.     else
20.     return value2;
21.     }
22.     string maxValue(string value1, string value2)
```



```

23.     {
24.     if (value1 > value2)
25.     return value1;
26.     else
27.     return value2;
28.     }

```

Bu funksiyalarning to'rtalasi ham ulardagi qo'llanilgan turlarning har xil ekanliklari inobatga olinmasa, deyarli bir xil. Birinchi funksiya **int** turini, ikkinchi funksiya **double** turini, uchinchi funksiya **char** turini va to'rtinchi funksiya **string** turini qo'llaydi. Agar biz quyidagicha ko'rinishda, umumiy tur bilan, bittagina, oddiy funksiyani aniqlasak, bir funksiyaning o'zida barcha turlarning saqlanib qolishi, ortiqcha bo'sh joyning paydo bo'lishi va dasturning osonlik bilan qayta sozlanishiga erishishimiz mumkin:

```

1. GenericType maxValue(GenericType value1, GenericType value2)
2. {
3. if (value1 > value2)
4. return value1;
5. else
6. return value2;
7. }

```

Mazkur **GenericType** (UmumiyTur) **int**, **double**, **char**, **string** kabi turlarning barchasini qo'llay oladi. C++ funksiya shablonlarini umumiy tur bilan aniqlash imkonini beradi. 6.1-kodli ro'yxat umumiy turdagi ikkita qiymatning kattasini topish uchun funksiya shablonini aniqlaydi.

1.1-kodli ro'yxat. GenericMaxValue.cpp

```

1. #include <iostream>
2. #include <string>
3. using namespace std;
4. template <typename T>
5. T maxValue(T value1, T value2)
6. {
7. if (value1 > value2)
8. return value1;
9. else
10. return value2;
11. }

12. int main()
13. {
14. cout << "1 va 3 ning kattasi: " << maxValue(1, 3) << endl;
15. cout << "1.5 va 0.3 ning kattasi: " <<

```

```

16. << maxValue (1.5, 0.3) << endl;
17. cout << "‘A’ va ‘N’ ning kattasi: "
18. << maxValue ('A', 'N') << endl;
19. cout << " \"NBC\" va \"ABC\" ning kattasi: "
20. << maxValue (string("NBC"), string("ABC")) << endl;
21. return 0;
22. }

```

Natija:

```

1 va 3 ning kattasi: 3
1.5 va 0.3 ning kattasi: 1.5
‘A’ va ‘N’ ning kattasi: N
"NBC" va "ABC" ning kattasi: NBC

```

Funksiya shablonining aniqlanishi parametrlar roʻyxati tomonidan berilgan **template** – kalit soʻzi bilan boshlanadi. Har bir parametr dastlab oʻzaro teng kuchli boʻlgan **typename** yoki **class** kalit soʻzi orqali, **<typename typeParameter>** yoki **<class typeParameter>** koʻrinishida beriladi. Masalan, 5-satrdagi

template<typename T>

maxValue uchun funksiya shablonining aniqlanishini boshlaydi. Shuningdek, bu satr *prefiks shablon* deb ham yuritiladi. Bu yerda **T** – parametr turi. Katta **T** harfining faqat parametr turini ifodalashda ishlatilishi kelishilgan.

maxValue funksiyasi 6-12 qatorlarda aniqlangan. Undan funksiya qaytaruvchi qiymat turi, funksiya parametrlari yoki funksiyada eʼlon qilingan oʻzgaruvchilarning turlarini aniqlashda foydalanish mumkin. Kodning 16-22-qatorlarida **int**, **double**, **char** va **string** turlari boʻyicha katta qiymatlilarni qaytarish uchun **maxValue** funksiyasi chaqirilgan. Funksiya **maxValue(1, 3)** koʻrinishda chaqirilganda, kompilyator argument turi **int** ekanligini aniqlaydi va funksiyani aniq **int** turida chaqirish uchun, **T** – parametr turini **int** ga oʻzgartiradi. Funksiya **maxValue(string("NBC"), string("ABC"))** koʻrinishda chaqirilganda, kompilyator argument turi **string** ekanligini aniqlaydi va funksiyani aniq **string** turida chaqirish uchun, **T** – parametr turini **string** ga oʻzgartiradi.

Agar 22-qatorda berilgan **maxValue(string("NBC"), string("ABC"))** ni **maxValue("NBC", "ABC")** deb oʻzgartirsak nima boʻladi? Unda biz funksiya **ABC** ni qaytarishini koʻrish uchun “syurpriz” tayyorlagan boʻlamiz. Nima uchun? Chunki “NBC” va “ABC” lar – C-satrlardir. **maxValue("NBC", "ABC")** ning chaqirilishi “NBC” va “ABC” larning manzillarini funksiya parametriga uzatadi. **value1 > value2** taqqoslash vaqtida ikkita massivning manzillari taqqoslanadi, tarkiblari emas.

Ogohlantirish. Umumiy **maxValue** funksiyasi ixtiyoriy turdagi ikkita qiymatning kattasini topish uchun moʻljallangan boʻlib, quyidagicha shartlar asosida ishlaydi:

- The two values have the same type;
- The two values can be compared using the >operator.

Masalan, agar siz bir qiymatni **int** turida, ikkinchisini esa, **double** turida bersangiz, kompilyatsion xatolik yuz beradi. Chunki kompilyator funksiyani chaqirishda mos turni aniqlay olmaydi. Agar siz funksiyani **maxValue(Circle(1), Circle(2))** koʻrinishda chaqirsangiz, kompilyatsion xatolik yuz beradi. Chunki **Circle** sinfida > operatori aniqlanmagan.

Maslahat. Parametr turini belgilashda `<typename T>` yoki `<class T>` dan foydalanishimiz mumkin. `<typename T>` dan foydalangan ma’qulroq, chunki `<typename T>` – tasviriyydir. `<class T>` ni esa, sinf aniqlanishi bilan adashtirib yuborish mumkin.

Eslatma. Ba’zi hollarda funksiya shabloni bittadan ko’p parametrlarga ega bo’lishi mumkin. Bunday vaziyatda parametrlarni `<typename T1, typename T2, typename T3>` kabi, barchasini bitta uchburchak qavslar oralig’iga, vergullar bilan ajratilgan holda joylashtiriladi.

1.1-kodli ro’yxatdagi asosiy funksiya parametrlari qiymat qabul qilib oluvchi sifatida aniqlangan. Uni havola qabul qilib oladigan qilib, 1.2-kodli ro’yxatdagi kabi o’zgartirishimiz mumkin.

1.2-kodli ro’yxat. *GenericMaxValuePassByReference.cpp*

```
1. #include <iostream>
2. #include <string>
3. using namespace std;
4. template <typename T>
5. T maxValue(const T& value1, const T& value2)
6. {
7.     if (value1 > value2)
8.         return value1;
9.     else
10.        return value2;
11. }

12. int main()
13. {
14.     cout << "1 va 3 ning kattasi: " << maxValue(1, 3) << endl;
15.     cout << "1.5 va 0.3 ning kattasi: "
16.     << maxValue(1.5, 0.3) << endl;
17.     cout << "'A' va 'N' ning kattasi: "
18.     << maxValue('A', 'N') << endl;
19.     cout << "\"NBC\" va \"ABC\" ning kattasi: "
20.     << maxValue(string("NBC"), string("ABC")) << endl;

21.     return 0;
22. }
```

Natija:

```
1 va 3 ning kattasi: 3
1.5 va 0.3 ning kattasi: 1.5
'A' va 'N' ning kattasi: N
"NBC" va "ABC" ning kattasi: NBC
```

Misol: Umumiy toifa

Qismda umumiy toifali funksiya aniqlanadi.

O`tgan semestrda ko`rib chiqilgan 12.8-kodli ro`yxatdagi *TanlabSarlash.cpp* dasturida **double** turidagi elementlardan tashkil topgan massivni saralovchi funksiya berilgan edi. Bu yerda o`sha funksiya nusxasi keltirilgan:

```
1. void tanlabSarlash(double list[], int listHajm)
2. {
3.     for (int i = 0; i < listHajm - 1; i++)
4.     {
5.         // list[i..listHajm-1] dagi minimumni topish
6.         double joriyMinimum = list[i];
7.         int joriyMinimumIndeks = i;
8.         for (int j = i + 1; j < listHajm; j++)
9.         {
10.            if (joriyMinimum > list[j])
11.            {
12.                joriyMinimum = list[j];
13.                joriyMinimumIndeks = j;
14.            }
15.        }
16.        // list[i] ni list[joriyMinimumIndeks] bilan almashtirish, agar zarur bo`lsa;
17.        if (joriyMinimumIndeks != i) {
18.            list[joriyMinimumIndeks] = list[i];
19.            list[i] = joriyMinimum; }
20.    }
21. }
```

Bu funksiyani **int**, **char**, **string** va hokazo turlardagi qiymatlardan iborat massivlarni saralashga mo`ljallangan yangi funksiya hosil qilish uchun qayta sozlash oson. Bu turlarning har biri uchun saralashni bajarish uchun koddagi **double** kalit so`zini **int**, **char** yoki **string** kalit so`zlari bilan almashtirish kerak (1- va 6-qator).

Bir qancha saralash funksiyalarini yozish o`rniga, shunchaki, barcha turlar uchun o`rinli bo`lgan, bir dona funksiya shablonini yozishimiz mumkin. 1.3-kodli ro`yxatda massiv elementlarini saralash funksiyasi aniqlangan.

1.3-kodli ro`yxat. GenericSort.cpp

```
1. #include <iostream>
2. #include <string>
3. using namespace std;
4. template <typename T>
5. void sort(T list[], int listSize)
6. {
7.     for (int i = 0; i < listSize; i++)
8.     {
9.         // list[i..listHajm-1] dagi minimumni topish
10.        T currentMin = list[i];
11.        int currentMinIndex = i;
12.        for(int j = i + 1; j < listSize; j++)
13.        {
```

```

14. if (currentMin > list[j])
15. {
16.     currentMin = list[j];
17.     currentMinIndex = j;
18. }
19. }
20. // list[i] ni list[joriyMinimumIndeks] bilan almashtirish, agar zarur bo`lsa;
21. if(currentMinIndex != i)
22. {
23.     list[currentMinIndex] = list[i];
24.     list[i] = currentMin;
25. }
26. }
27. }
28. template <typename T>
29. void printArray(const T list[], int listSize)
30. {
31.     for (int i = 0; i < listSize; i++)
32.     {
33.         cout << list[i] << " ";
34.     }
35.     cout << endl;
36. }
37. int main()
38. {
39.     int list1[] = {3, 5, 1, 0, 2, 8, 7};
40.     sort(list1, 7);
41.     printArray(list1, 7);
42.     double list2[] = {3.5, 0.5, 1.4, 0.4, 2.5, 1.8, 4.7};
43.     sort(list2, 7);
44.     printArray(list2, 7);
45.     string list3[]={"Atlanta", "Denver", "Chicago", "Dallas"};
46.     sort(list3, 4);
47.     printArray(list3, 4);
48.     return 0;}

```

Natija:

0 1 2 3 5 7 8

0.4 0.5 1.4 1.8 2.5 3.5 4.7

Atlanta Chicago Dallas Denver

Bu dasturda ikkita funksiya shaboni aniqlangan. **sort** funksiya shablони (5-30-qatorlar) massiv elementlari turini belgilab olish uchun **T** parametridan foydalanadi. **double** parametri umumiy **T** parametri bilan almashtirilganligi hisobga olinmaganda, bu funksiya *tanlabSaralash* funksiyasining o'zginasi.

printArray funksiya shablони (32-40-qatorlar) massiv elementlari turini aniqlash uchun **T** – tur parametridan foydalanadi. Bu funksiya massivdagi barcha elementlarni konsol oynada chop etadi.

main funksiya **int**, **double** va **string** qiymatlardagi massivlarni saralash uchun **sort** funksiyasini chaqiradi (45, 49-, 53-qatorlar) va bu massivlarni chop etish uchun **printArray** funksiyasini chaqiradi (46-, 50-, 54-qatorlar).

Maslahat. Umumiy funksiyani aniqlaganingizdan avval, unga mos bo'lgan oddiy funksiyani yozishdan boshlab, keyin uni umumiy funksiyaga o'tkazsangiz yaxshiroq bo'ladi.

Funksiya shablonlari (function template).

Shablonlar yordamida umumiy funksiyalar va umumiy classlar yaratish imkoniyati mavjud. Umumiy funksiya va umumiy classlar har xil ma'lumot toifalaridan ularni **overload** qilmasdan (ko'p kod yozmasdan) foydalanish imkoniyatini beradi. Ya'ni bunda biz har bir toifa uchun alohida funksiya yozishimiz shart bo'lmaydi.

Shablonlar ikki xil bo'ladi:

- **Funksiya shablони (function template)**
- **Sinf shablони (class template)**

Funksiya shablonini (function template) yaratish.

Funksiya shablони template kalit so'zidan foydalangan holda amalga oshiriladi. Quyida funksiya shablonini yaratish formasi keltirilgan:

```
template <class TOIFA> qaytarish-tipi funk-nomi (arg-lar)
{
    // funksiya tanasi
}
```

Bu yerda **TOIFA** funksiya tomonidan joriy holda ishlatiladigan ma'lumot tipi. Ushbu toifani kompilyator avtomatik ravishda funksiyaga kelayotgan ma'lumot tipi bilan almashtirib qo'yadi. Bu yerda **class** va **template** lar funksiya shablonini yaratish uchun ishlatiladigan kalit so'zlardir. Lekin ba'zi hollarda **class** kalit so'zi o'rniga **typename** kalit so'zini ishlatishimiz mumkin. Quyidagi misol xohlagan tipda berilgan ikkita o'zgaruvchi o'rnini almashtirib berish uchun xizmat qiladi va bunda har bir toifa uchun alohida funksiya yozishimizga zaruriyat qolmaydi.

Funksiya shabloniga misol

1. `#include <iostream>`
2. `using namespace std;`
3. `// Funksiya shablони e'lon qilinishi...`
4. `template <class X> void swapargs(X &a, X &b)`
5. `{`
6. `X temp;`
7. `temp = a;`
8. `a = b;`
9. `b = temp;`

```

10. }
11. int main()
12. {
13. int i=10, j=20;
14. double x=10.1, y=23.3;
15. char a='x', b='z';
16. cout << "Original i, j: " << i << ' ' << j << '\n';
17. cout << "Original x, y: " << x << ' ' << y << '\n';
18. cout << "Original a, b: " << a << ' ' << b << '\n';
19. swapargs(i, j); // swap funksiyasi butun toifa uchun (int)
20. swapargs(x, y); // swap funksiyasi haqiqiy toifa uchun (float)
21. swapargs(a, b); // swap funksiyasi simvol toifa uchun (char)
22. cout << "Swapped i, j: " << i << ' ' << j << '\n';
23. cout << "Swapped x, y: " << x << ' ' << y << '\n';
24. cout << "Swapped a, b: " << a << ' ' << b << '\n';
25. return 0;
26. }

```

Dastur izohi:

Umumiy funksiyaning boshqacha ko'rinishi.

Quyidagi misolda **swapargs()** funksiyasi boshqacharoq ko'rinishda e'lon qilingan. Ya'ni shablon birinchi satrda funksiya esa alohida satrda joylashgan.

```

1. template <class X>
2. void swapargs(X &a, X &b)
3. {
4. X temp;
5. temp = a;
6. a = b;
7. b = temp;
8. }

```

Lekin bu ko'rinishda birinchi va ikkinchi satr o'rniga bironta kod yozilsa xatolik beradi.

```

1. template <class X>
2. int c // ERROR
3. void swapargs(X &a, X &b)
4. {
5. X temp;
6. temp = a;
7. a = b;
8. b = temp;
9. }

```

Funksiya shablonini override (qayta yozish) qilish.

```

1. template <class X> void swapargs(X &a, X &b)
2. {
3. X temp;
4. temp = a;
5. a = b;

```

```

6.  b = temp;
7.  cout << "swapargs funksiya shablone chaqirildi.\n";
8.  }
9.  // Bunda swapargs() funksiyasi faqatgina int tipi uchun ishlaydi.
10. void swapargs(int &a, int &b)
11. {
12. int temp;
13. temp = a;
14. a = b;
15. b = temp;
16. cout << " int tipi uchun maxsus swapargs funksiyasi.\n";
17. }
18. int main()
19. {
20. int i=10, j=20;
21. double x=10.1, y=23.3;
22. char a='x', b='z';
23. cout << "Original i, j: " << i << ' ' << j << '\n';
24. cout << "Original x, y: " << x << ' ' << y << '\n';
25. cout << "Original a, b: " << a << ' ' << b << '\n';
26. swapargs(i, j); // calls explicitly overloaded swapargs()
27. swapargs(x, y); // calls generic swapargs()
28. swapargs(a, b); // calls generic swapargs()
29. cout << "Swapped i, j: " << i << ' ' << j << '\n';
30. cout << "Swapped x, y: " << x << ' ' << y << '\n';
31. cout << "Swapped a, b: " << a << ' ' << b << '\n';
32. return 0;
33. }

```

Dastur natijasi:

```

Original i, j: 10 20
Original x, y: 10.1 23.3
Original a, b: x z
int tipi uchun maxsus swapargs funksiyasi.
swapargs funksiya shablone chaqirildi.
swapargs funksiya shablone chaqirildi.
Swapped i, j: 20 10
Swapped x, y: 23.3 10.1
Swapped a, b: z x

```

Funksiya shablonini Override qilish yangi usuli:

```

1.  template<> void swapargs<int>(int &a, int &b)
2.  {
3.  int temp;
4.  temp = a;
5.  a = b;
6.  b = temp;
7.  cout << " int tipi uchun maxsus swapargs funksiyasi.\n";

```


8. }

Funksiya shablonini overload qilish:

// Oddiy funksiyalardek, funksiya shablonini ham overload qilish mumkin.

```
1. #include <iostream>
2. using namespace std;
3. // f() funksiya shablonining birinchi turi.
4. template <class X> void f(X a)
5. {
6.     cout << "Inside f(X a)\n";
7. }
8. // f() funksiya shablonining ikkinchi turi.
9. template <class X, class Y> void f(X a, Y b)
10. {
11.     cout << "Inside f(X a, Y b)\n";
12. }
13. int main()
14. Original i, j: 10 20
15. Original x, y: 10.1 23.3
16. Original a, b: x z
17. int tipi uchun maxsus swapargs funksiyasi.
18. swapargs funksiya shabloni chaqirildi.
19. swapargs funksiya shabloni chaqirildi.
20. Swapped i, j: 20 10
21. Swapped x, y: 23.3 10.1
22. Swapped a, b: z x
23. {
24. f(10); // calls f(X)
25. f(10, 20); // calls f(X, Y)
26. return 0;
27. }
```

Funksiya shablonining kamchiligi:

- Umumiy funksiyalar funksiya overloadining o'rnini bosishi mumkin. Lekin bu yerda bitta kamchilik mavjud. Biz oddiy funksiyani overload qilganimizda, har xil ma'lumotlar tipi uchun funksiya tanasini har xil qilib yozishimiz mumkin. Lekin umumiy funksiyada har xil tip qabul qila olgani bilan funksiya tanasi har doim bir xil bo'ladi, chunki bitta funksiyaga murojaat bo'ladi.

- Faqatgina ma'lumotlar tipi har xil bo'la oladi.

Umumiy sinflar (sinf shablони)

Sinf shablonini e'lon qilishning umumiy formasi:

```
template <class TOIFA> class sinf_nomi{
...
}
```

Yoki quyidagi ko'rinishda e'lon qilish mumkin

```
template <class TOIFA>
class sinf_nomi {
...
}
```

Sinf shablonini ishlatish
sinf_nomi <tip> obyekt;

Sinf shablони uchun oddiy misol.

```
1. #include <iostream>
2. using namespace std;
3. template <class T>
4. class mypair {
5.     T a, b;
6. public:
7.     mypair (T first, T second)
8.     {a=first; b=second;}
9.     T getmax ();
10. };
11. template <class T>
12. T mypair<T>::getmax ()
13. {
14.     T retval;
15.     retval = a>b? a : b;
16.     return retval;
17. }
18. int main () {
19.     mypair <int> myobject (100, 75);
20.     cout << myobject.getmax();
21.     return 0;
22. }
```

Sinf shablonida ikki xil toifadan foydalanish

```
1. #include <iostream>
2. using namespace std;
3. template <class Type1, class Type2> class myclass
4. {
5.     Type1 i;
6.     Type2 j;
7. public:
8.     myclass(Type1 a, Type2 b) { i = a; j = b; }
9.     void show() { cout << i << ' ' << j << '\n'; }
10. };
11. int main()
12. {
13.     myclass<int, double> ob1(10, 0.23);
14.     myclass<char, char *> ob2('X', "Templates add power.");
15.     ob1.show(); // show int, double
16.     ob2.show(); // show char, char *
17.     return 0;
18. }
```

Dastur natijasi:

10 0.23

X Templates add power.

Maxsuslashtirilgan sinf shablони

template<> konstruktori maxsuslashtirilgan sinf shablони uchun ishlatiladi.

```
1. template <class T> class myclass {
2.   T x;
3.   public:
4.   myclass(T a) {
5.     cout << "Inside generic myclass\n";
6.     x = a;
7.   }
8.   T getx() { return x; }
9. };
10. // int toifasi uchun maxsuslashtirilgan sinf shablони.
11. template <> class myclass<int> {
12.   int x;
13.   public:
14.   myclass(int a) {
15.     cout << "Inside myclass<int> specialization\n";
16.     x = a * a;
17.   }
18.   int getx() { return x; }
19. };
```

Shablonning asosiy xususiyatlari:

- reusable kod yozish imkonini beradi.
- Shablonlar yordamida framework lar yaratish mumkin

X Templates add power.

- Dastur kodi egiluvchanlik xususiyatiga ega bo'ladi.
- Turli xil tipdagi ma'lumotlar ustida ishlash uchun kod yozishda vaqtni tejash.
- C++ dagi STL lar (Standard Shablon Kutubxonalar), nomidan ko'rinib turibdiki, shablonlar yordamida yaratilgan

3. Sinf shablonlari (class template)

Sinf uchun umumiy turni aniqlash mumkin.

Oldingi qismda funksiya uchun mo'ljallangan tur parametrغا ega funksiya shablони aniqlandi. Biz sinf uchun mo'ljallangan tur parametrغا ega sinfni ham aniqlashimiz mumkin. Tur parametrlar sinfning tur shakllantiriladigan ixtiyoriy qismida foydalanilishi mumkin.

int qiymatlar uchun stek hosil qilish mumkin. Quyida, 1.1a-rasmdagi kabi, o'sha sinf nusxasi va uning UML sinf diagrammasi berilgan.

StackOfIntegers	StackOfIntegers
-elements[100]: int -size: int	-elements[100]: T -size: int
+StackOfIntegers() +empty(): bool const +peek(): int const +push(value: int): void +pop(): int +getSize(): int const	+Stack() +empty(): bool const +peek(): T const +push(value: T): void +pop(): T +getSize(): int const

(a)

(b)

1.1-rasm. *Stack<T>* – *Stack* sinfining umumiy versiyasi

```

1. #ifndef STACK_H
2. #define STACK_H
3. class StackOfIntegers
4. {
5. public:
6. StackOfIntegers();
7. bool empty() const;
8. int peek() const;
9. void push(int value);
10.    int pop();
11.    int getSize() const;
12. private:
13.    int elements[100];
14.    int size;
15. };
16. StackOfIntegers::StackOfIntegers()
17. {
18.     size = 0;
19. }
20. bool StackOfIntegers::empty() const
21. {
22.     return size == 0;
23. }
24. int StackOfIntegers::peek() const
25. {
26.     return elements[size - 1];
27. }
28. void StackOfIntegers::push(int value)
29. {
30.     elements[size++] = value;
31. }
32. int StackOfIntegers::pop()

```

```

33.     {
34.     return elements[--size];
35.     }
36.     int StackOfIntegers::getSize() const
37.     {
38.     return size;
39.     }
40.     #endif

```

Boyalgan **int** kalit soʻzlarini **double**, **char**, yoki **string** bilan almashtirib, **double**, **char** va **string** qiymatlar uchun **StackOfDouble**, **StackOfChar** va **StackOfString** kabi sinflarni aniqlash uchun ushbu sinfni osongina oʻzgartirishlar kiritishimiz mumkin. Lekin, deyarli bir xil boʻlgan bir nechta sinflarni aniqlamasdan, shunchaki ixtiyoriy turdagi elemen uchun ishlaydigan bitta shablon sinfni aniqlashimiz mumkin. Yangi umumiy **Stack** sinfi uchun UML diagramma 1.1b-rasmda berilgan. 1.4-kodli roʻyxatda umumiy tur elementlarini saqlashga moʻljallangan umumiy stek sinfi aniqlangan.

1.4-kodli roʻyxat. GenericStack.h

```

1.  #ifndef STACK_H
2.  #define STACK_H
3.  template <typename T>
4.  class Stack
5.  {
6.  public:
7.      Stack();
8.      bool empty() const;
9.      T peek() const;
10.     void push(T value);
11.     T pop();
12.     int getSize() const;
13.     private:
14.     T elements[100];
15.     int size;
16. };
17. template <typename T>
18. Stack<T>::Stack()
19. {
20.     size = 0;
21. }
22. template <typename T>
23. bool Stack<T>::empty() const
24. {
25.     return size == 0;
26. }
27. template <typename T>
28. T Stack<T>::peek() const
29. {
30.     return elements[size - 1];
31. }

```

```

32. template <typename T>
33. void Stack<T>::push(T value)
34. {
35.     elements[size++] = value;
36. }
37. template <typename T>
38. T Stack<T>::pop()
39. {
40.     return elements[--size];
41. }
42. template <typename T>
43. int Stack<T>::getSize() const
44. {
45.     return size;
46. }
47. #endif

```

Sinf shablonlari sintaksisi asosan funksiya shablonlari bilan bir xil. Xuddi funksiya shablonidagi kabi sinf aniqlanishidan oldin *template* qo`shimchasi yozilishi kerak (47-qator).

```
template <typename T>
```

Xuddi oddiy ma'lumot turiga o`xshab, sinfda tur parametri qo`llanilishi mumkin. Bu yerda **T** tur **peek()**(10-qator), **push(T value)** (11-qator), va **pop()** (12-qator) funksiyani aniqlash uchun ishlatilgan. Shuningdek, **T** tur 16-qatorda, massiv elementlarini e`lon qilishda ham qo`llanilgan.

Konstruktorlar va funksiyalar ularning o`zlari shablonlar ekanliklari inobatga olinmaganda, oddiy sinflardagi kabi, bir xil yo`l bilan aniqlanadi. Buni amalga oshirish uchun, *template* qo`shimchasini konstruktor yoki funksiyaning bosh qismidan avval joylashtirish kerak bo`ladi. Masalan:

```

template <typename T>
Stack<T>::Stack()
{
    size = 0;
}
template <typename T>
bool Stack<T>::empty()
{
    return size == 0;
}
template <typename T>
T Stack<T>::peek()
{
    return elements[size - 1];
}

```

Shu o`rinda e`tibor qaratishimiz lozimki, **::** qarashlilik operatoridan oldin keladigan sinf nomi **Stack<T>**, **Stack** emas.

Maslahat. GenericStack.h sinf aniqlanishini va tadbiqini bitta faylga o`tkazadi. Sinf aniqlanishini va tadbiqini ikkita turli fayllarga joylashtirish yaxshi ish albatta. Biroq, shunday bo`lsa ham, sinf shablonlari uchun ularni birlashtirish bir muncha xavfsizroq, chunki, ba`zi kompilyatorlar ularni tarqoq holda kompilyatsiya qilmaydi.

1.5-kodli ro`yxatda, uning qatorida **int** qiymatlar uchun va 18-qatorda satrlar uchun stek hosil qiluvchi sinovchi dastur berilgan.

1.5-kodli ro`yxat. *TestGenericStack.cpp*

```
1.  #include <iostream>
2.  #include <string>
3.  #include "GenericStack.h"
4.  using namespace std;
5.  int main()
6.  {
7.      // int qiymatlar uchun yangi stek yaratish
8.      Stack<int> intStack;
9.      for (int i = 0; i < 10; i++)
10.         intStack.push(i);
11.     while (!intStack.empty())
12.         cout << intStack.pop() << " ";
13.     cout << endl;
14.     // Satrlar uchun yangi stek yaratish
15.     Stack<string> stringStack;
16.     stringStack.push("Chicago");
17.     stringStack.push("Denver");
18.     stringStack.push("London");
19.     while(!stringStack.empty())
20.         cout << stringStack.pop() << " ";
21.     cout << endl;
22.     return 0;
23. }
```

Natija:

9 8 7 6 5 4 3 2 1 0 London Denver Chicago
--

Shablon sinfda ob'yekt yaratishda **T** - tur parametri uchun aniq bir turni belgilab olishimiz zarur. Masalan:

Stack<int> intStack;

Bu e'lon qilinish **T** - tur parametrini **int** bilan almashtiradi. Shu sababli ham **intStack** steki **int** qiymatlar steki hisoblanadi. **intStack** ob'yekti ham xuddi ixtiyoriy boshqa ob'yektlarga o'xshaydi. Dastur **intStack** stekka **int** qiymatlarni qo'shish uchun **push** funksiyasini chaqiradi (11-qator) va stekdan elementlarni chop etadi (13-14-qatorlar).

Dastur 18-qatorda satrlarni yozish uchun stek ob'yektni e'lon qiladi, stekda uchta satrni yozadi (19-21-qatorlar) va stekdagi satrlarni chop etadi (24-qator).

Quyidagi kodli qismlarga e'tibor beraylik:

9-11-qatorlar.

```
while (!intStack.empty())
    cout << intStack.pop() << " ";
cout << endl;
```

23-25-qatorlar

```

while (!stringStack.empty())
    cout << stringStack.pop() << " ";
cout << endl;

```

Bu kodli qismlar deyarli bir xil. Ular o`rtasidagi farq **intStack** va **stringStack** yozuvlaridagi shakllantirish operatsiyalarida. Stekdagi elementlarni chop etish uchun stek parametrli funksiya aniqlashimiz mumkin. Yangi dastur 1.6-kodli ro`yxatda keltirilgan.

1.6-kodli ro`yxat. TestGenericStackWithTemplateFunction.cpp

```

1. #include <iostream>
2. #include <string>
3. #include "GenericStack.h"
4. using namespace std;
5.
6. template <typename T>
7. void printStack(Stack<T>& stack)
8. {
9.     while(!stack.empty())
10.    cout << stack.pop() << " ";
11.    cout << endl;
12. }
13.
14. int main()
15. {
16.    // int qiymatlar uchun yangi stek yaratish
17.    Stack<int> intStack;
18.    for (int i = 0; i < 10; i++)
19.        intStack.push(i);
20.    printStack(intStack);
21.
22.    // Satrlar uchun yangi stek yaratish
23.    Stack<string> stringStack;
24.    stringStack.push("Chicago");
25.    stringStack.push("Denver");
26.    stringStack.push("London");
27.    printStack(stringStack);
28.
29.    return 0;
30. }

```

Shablon funksiyada umumiy sinf nomi **Stack<T>** tur parametr sifatida qo`llanilgan (7-qator).

Eslatma. C++ sinf shablonida tur parametr uchun *jimlik qoidasiga ko`ra* tur ni ta`minlashga ruxsat beradi. Masalan, jimlik qoidasiga ko`ra tur sifatida, umumiy **Stack** sinfida quyidagicha ta`minlash mumkin:

```

template<typename T = int>
class Stack
{
...
};

```


Endi sinf shablonida jimlik qoidasi turidan foydalanishimiz mumkin, lekin funksiya shablonida emas.

Eslatma. Biz shuningdek, *template* qo`shimchasi tarkibida tur parametri bilan *tursiz parametr* ni ham qo`llashimiz mumkin. Masalan, **Stack** sinfda parametr sifatida, massiv o`lchamini quyidagicha e`lon qilish mumkin:

```
template<typename T, int capacity>
class Stack
{
    ...
private:
    T elements[capacity];
    int size;};
```

Shunday qilib, biz stek hosil qilganimizda, massiv o`lchamini belgilashimiz mumkin. Masalan, `Stack<string, 500> stack;`

500 tagacha satrlarni saqlay oluvchi stekni e`lon qiladi.

Eslatma. Sinf shablonida statik a`zolari aniqlash mumkin. Har bir shablon belgilanishi o`zining statik ma`lumotlar maydonidagi nusxasiga ega.

Nazorat savollari

1. Shablon nima va uning qanday turlari mavjud?
2. Funksiya shabloni qanday yaratiladi?
3. Funksiya shablonida turli xil tiplardan foydalanish mumkinmi? Agar mumkin bo'lsa bunday turdagi funksiya shablonlari qanday ko'rinishda yaratiladi?
4. Funksiya shablonini override qilish formasi qanday? Misol keltiring?
5. Funksiya shablonini override qilishning yangi formasi qanday ko'rinishda?
6. Funksiya shabloni qanday overload qilinadi?
7. Sinf shabloni qanday yaratiladi?
8. Sinf shablonida ikki va undan ortiq toifalardan foydalanish qanday amalga oshiriladi. Misol keltiring?
9. Shablonning kamchiliklari?
10. Shablonning avzalliklari?
11. Funksiya shablonida parametrlar qanday turda bo'ladi?
12. Sinf shablonida parametrlar bo'ladimi? Bo'lsa qanday?
13. Funksiya shablonining parametrini turlari har-hil bo'lishi mumkinmi?
14. Funksiya shabloni orqali to'plamlarni qanday qayta ishlash mumkin?
15. Sinf shabloni orqali to'plamlar qiymatlari qanday qayta ishlanadi?
16. Funksiya shablonlaridan friend funksiya sifatida foydalanilsa bo'ladimi?
17. Funksiya shabloni ko'rsatgich bo'la oladimi?
18. Sinf shablonida funksiya shablonini yartish mumkinmi?
19. Sinf shablonida konstruktorni asossiy vasifasi nimadan iborat?
20. Funksiya shabloni obyekt qaytarish mumkinmi? Mumkin bo'lsa qanday?
21. Funksiya parametrda funksiya shablonini ishlatish mumkinmi?
22. Funksiya parametrda class shablonini ishlatish mumkinmi?
23. Funksiya parametrda funksiya shabloni birinchi keladimi yoki class shablonimi?
24. Funksiya parametrda funksiya shabloni birinchi keladimi yoki class shablonimi?
25. Funksiya shabloni foydalanuvchi yaratgan class uchun ham ishlaydimi?
26. Funksiya shablonini massiv sifatida ishlatish mumkinmi?
27. Funksiya shabloni konstruktor bo'lishi mumkinmi?

28. Funksiya shabloni obyektlar qabul qilishi mumkinmi?
29. Funksiya shabloni qaysi turda bo'lishi mumkinmi?
30. Funksiya shabloni rekursiv bo'lishi mumkinmi?
31. Shablon class dan voris olish mumkinmi?
32. Shablon class dan obyekt olish mumkinmi?
33. Shablon classda virtual funksiya yaratish mumkinmi?
34. Shablon class obyekt qabul qiladimi?
35. Shablon class ni massiv sifatida ishlatish mumkinmi?
36. Shablon class da modifikatorlar ishlaydimi?
37. Shablon class ning zamonaviy nomi nima?
38. Shablon class da shablon funksiyalar ishlaydimi?
39. Shablon class da shablon class yaratish mumkinmi?

AMALIYOT MASHG'ULOTLARI MATERIALLARI

Amaliy ish №1-2

Mavzu: Dasturlasning asosiy tushunchalari: Algoritm tuzish, Murakkab algoritm tuzish

Ishdan maqsad. Algoritm va algoritmning berilishi usullari to'g'risida mukammal tasavvurga ega bo'lish. Algoritmning turlari va chiziqli algoritm tuzish bo'yicha ko'nikmalarni shakllantirish.


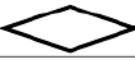



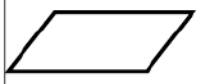


Nazariy qism

Algoritmning tasvirlash usullari haqida gapirganda algoritmning berilish usullari xilma-xilligi va ular orasida eng ko'p uchraydiganlari quyidagilar ekanligini ko'rsatib o'tish joiz:

1. Algoritmning so'zlar orqali ifodalanishi.
2. Algoritmning formulalar yordamida berilishi.
3. Algoritmning jadval ko'rinishida berilishi, masalan, turli matematik jadvallar, loteriya yutuqlari jadvali, funksiyalar qiymatlari jadvalari bunga misol bo'ladi.
4. Algoritmning dastur shaklida ifodalanishi, ya'ni algoritm kompyuter ijrochisiga tushunarli bo'lgan dastur shaklida beriladi.
5. Algoritmning algoritmik tilda tasvirlanishi, ya'ni algoritm bir xil va aniq ifodalash, bajarish uchun qo'llanadigan belgilash va qoidalar majmui algoritmik til orqali ifodalashdir. Ulardan o'quv o'rganish tili sifatida foydalanilmoqda. Bu-lardan Ye-praktikum yoki Ye-tili algoritm ijrochisi algoritmik tili ham mavjud.
6. Algoritmning grafik shaklda tasvirlanishi. Masalan, grafiklar, sxemalar ya'ni blok - sxema bunga misol bo'la oladi. Blok sxemaning asosiy elementlari quyidagilar: oval (ellips shakli)-algoritm boshlanishi va tugallanishi, to'g'ri bur-chakli to'rtburchak-qiymat berish yoki tegishli ko'rsatmalarni bajarish. Romb -shart tekshirishni belgilaydi. Uning yo'naltiruvchilari tarmoqlar bo'yicha biri ha ikkinchisi yo'q yo'nalishlarni beradi, parallelogramm- ma'lumotlarni kiritish yoki chiqarish, yordamchi algoritmga murojaat - parallelogramm ikki tomoni chiziq, yo'naltiruvchi chiziq - blok-sxemadagi harakat boshqaruvi, nuqta-to'g'ri chiziq (ikkita parallel) - qiymat berish.

Algoritm bajarilishi tugallangan amallar ketma-ketligi algoritm qadami deb yuritiladi. Har bir alhoxida qadamni ijro etish uchun bajarilishi kerak bo'lgan amallar haqidagi ko'rsatma buyruq deb aytiladi.

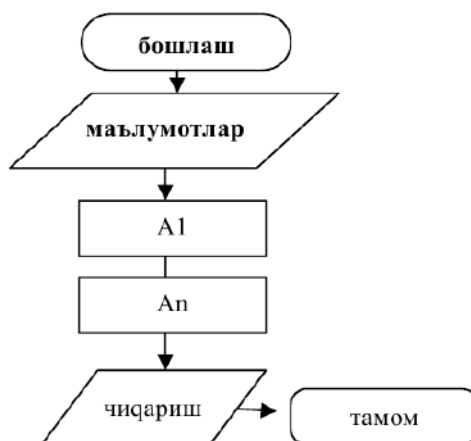
Algoritmni ko'rgazmaliroq qilib tasvirlash uchun blok-sxema, ya'ni geometrik usul ko'proq qo'llaniladi. Algoritmning blok-sxemasi algoritmning asosiy tuzilishining yaqqol geometrik tasviri: algoritm bloklari, ya'ni geometrik shakllar ko'rinishida, bloklar orasidagi aloqa esa yunaltirilgan chiziqlar bilan ko'rsatiladi. Chiziqlarning yunalishi bir blokdan so'ng qaysi blok bajarilishini bildiradi. Algoritmni ushbu usulda ifodalashda vazifasi, tutgan o'rniga qarab quyidagi geometrik shakl(blok) lardan foydalaniladi.

Blokning atalishi	Belgilanishi	Tushunilishi
Hisoblashlar bloki (to'g'ri-to'rtburchak)		Hisoblash amali yoki hisoblash amallari ketma-ketligi
shartli blok (romb)		Shartlarni tekshirish
siklik jarayon (oltiburchak)		Siklning boshlanishi
qism dastur		qism dastur bo'yicha hisoblash, standart qism dasturi
birlashtirish (aylana)		Yo'nalish chizig'ini o'zgartirish
Ma'lumotlarni kiritish va chiqarish (parallelogramm)		Ma'lumotlarni kiritish va natijalarni chiqarish
Algoritmnining boshi va oxiri (oval)		Boshlash, tamom, to'xtash
Chiqarish bloki		Ma'lumotlarni qog'ozga chiqarish

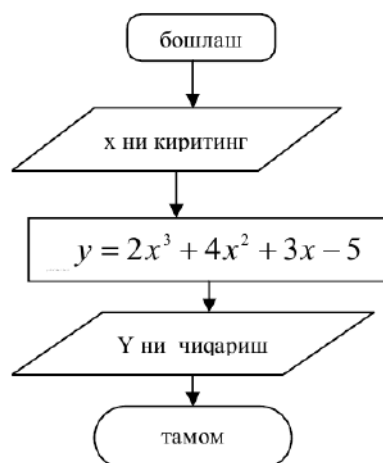
Algoritmlar berilishi va ifodalanishiga qarab: chiziqli, tarmoqlanuvchi va takrorlanuvchi turlarga bo'linadi. Algoritmnining turlari bilan tanishtirganda, avvalo hiech qanday shart tekshirilmaydigan va tartib bilan faqat ketma-ket bajariladigan jarayonlarni ifodalaydigan chiziqli algoritmlar aytib o'tiladi.

Chiziqli algoritmlar. Chiziqli algoritmlar algoritmlarning eng sodda va oddiy ko'rinishi hisoblanadi. Unida bajariladigan amallar ham buyruqlar ham buyruqlar ham qanday tartibda berilgan bo'lsa shunday tartibda ketma - ket bajariladi, ya'ni hiech qanday shart tekshirilmasdan chiziqli algoritmlarda buyruqlar ketma- ket tartib bilan bajariladi.

Chiziqli algoritmlarni quyidagi ko'rinishda ifodalash mumkin. Bu yerda A_1, \dots, A_n lar chiziqli algoritmlarda bajarilishi kerak bo'lgan buyruqlar ketma- ketligidir.



2- misol. $y=2x^3 + 4x^2 + 3x - 5$ funksiyani x ning ixtiyoriy qiymatlarida xisoblash algoritmini tuzing. Yechish. Algoritmnining blok sxema ko'rinishda ifodalaymiz.



C++ da matematik funksiyalar.

Matematik funksiyalardan foydalanish uchun math.h jutibxonasini e'lon qilish lozim.

Funksiya	Tavsifi	Misol
abs(a)	a ning moduli	abs(-3)= 3 abs(5)= 5
sqrt(a)	a ning kvadrat ildizi	sqrt(9)=3.0
pow(a, b)	a ni b darajaga ko'tarish	pow(2,3)=8
ceil(a)	a ni o'zidan kichik bo'lmagan eng kichik butun songa yaxlitlash	ceil(2.3)=3.0 ceil(-2.3)=-2.0
floor(a)	a ni o'zidan katta bo'lmagan eng kichik butun songa yaxlitlash	floor(12.4)=12 floor(-2.9)=-3
fmod(a, b)	a/b ni hisoblashdagi qoldiqni olish	fmod(4.4, 7.5) = 4.4 fmod(7.5, 4.4) = 3.1
exp(a)	e ^a ni hisoblash	exp(0)=1
sin(a)	sina, a radiyanda beriladi.	
cos(a)	cosa, a radiyanda beriladi.	
log(a)	a natular logarifmi	log(1.0)=0.0
log10(a)	a ning o'nlik logarifmi	Log10(10)=1
asin(a)	arcsina, bunda -1.0 < a < 1.0. Natija radiyanda xosil bo'ladi	asin(1)=1.5708

Bo'linmaning haqiqiy qismi kerak bo'lga, agar o'zgaruvchilar butun son bo'lsa bo'lish amaliga e'tibor qaratish lozim.

Misol. Asosining uzunligi a va balandligi h ga teng bo'lgan uchburchakning yuzasini hisoblovchi dastur tuzing.

Yechimi.

Kiruvchi ma'lumot a va h butun sonlari. Uchburchak yuzasi formulasi: $s = \frac{ah}{2}$.

a va b sonlari butun, lekin s soni haqiqiy son.

```
#include <iostream>
using namespace std;
```

```
int main() {
    int a, h;
    cin>>a>>h;
    double s = a * h / 2;
    cout<<s;
}
```

Dasturda hatolik mavjud. Bu hatolik shundan iboratki, butun sonlarni bo'lganda bo'linmaning butun qiymati hisoblanadi. Bo'linmaning haqiqiy qiymatini hisoblash uchun bo'linuvchilardan birining qiymati haqiqiy bo'lishi kerak. Yuqoridagi masalada buni

```
double s = a * h / 2.0;
```

yoki

```
double s = 1.0 * a * h / 2;
```

ko'rinishida yozish orqali to'g'irlash kiritishimiz mumkin.

Murakkab topshiriq bo'yicha na'muna:

$$AF = 2^{-x} \cdot \sqrt{x + \sqrt[4]{|y|} + 2} \cdot \sqrt[3]{e^{x-1} / \sin(z+2) + 2};$$

Bunda kiruvchi ma'lumotlar x, y, z haqiqiy sonlari.

Chiquvchi ma'lumot AF.

```
#include <iostream>
```

```
#include <math.h>
```

```
#include <stdio.h>
```

```
using namespace std;
```

```
int main() {
    double x, y, z;
    cin>>x>>y>>z;
    double AF = pow(2, -x) * sqrt(x + sqrt(sqrt(fabs(y)+2))) * pow(exp(x-1) / sin(z+2) + 2, 1. / 3);
    printf("%.2f", AF);
}
```

printf() funksiyasi xaqiqiy sonni nuqtadan so'ng biror xona aniqlikda chiqarish uchun xizmat qiladi. Agar sonning qiymati 3.5689 ga teng bo'lsa yaxlitlab chiqarilganda 3.57 soni chiqariladi.

Katta ifodani yozishni o'rniga uni qismlarga ajratishimiz ham mumkin:

The diagram shows the mathematical expression $AF = 2^{-x} \cdot \sqrt{x + \sqrt[4]{|y|} + 2} \cdot \sqrt[3]{e^{x-1} / \sin(z+2) + 2}$. The parts are highlighted as follows:
 A (red circle): 2^{-x}
 B (blue circle): $\sqrt{x + \sqrt[4]{|y|} + 2}$
 C (yellow circle): $\sqrt[3]{e^{x-1} / \sin(z+2) + 2}$

```
#include <iostream>
```

```
#include <math.h>
```

```
#include <stdio.h>
```

```
using namespace std;
```

```
int main() {
    double x, y, z;
```

```

cin>>x>>y>>z;
double A = pow(2, -x);
double B = sqrt(x + sqrt(sqrt(fabs(y)+2)));
double C = pow(exp(x-1) / sin(z+2) + 2, 1. / 3);
double AF = A * B * C;
printf("%.2f", AF);
}

```

Nazorat savollari:

1. Algoritm nima va unga misollar keltiring?
2. Algoritmning asosiy xossalari.
3. Algoritmning tasvirlashning asosiy usullari.
4. Bu tasvirlash usullarining har biriga misollar keltiring.
5. Blok-sxema nima? Asosiy elementlarini ayting.
6. Chiziqli algoritmlar qanday ifodalanadi?
7. Algoritmning turlari.

Topshiriqlar.

1-Topshiriq

1. O'lchami x bo'lgan qubik berilgan. Uning hajmini toping.
2. Radyuslari r1, r2, r3 bolgan 3 to doira radiuslari berilgan. Doiralarni yuzini hisoblang.
3. Yuzasi s va balandligi h bo'lgan uch burchag berilgan. Uni asosini toping.
4. Radyusi r bo'lgan sharing yuzini toping.
5. a,b va c tamonli uch burchag berilgan. Uch burchagning pirimetiri topilsin.
6. Asoslari a va b, balandligi h bo'lgan o'layuzini toping.
7. Qo'nisni balandligi h va radyusi r bo'lsa uni hajmi nimaga teng bo'ladi.
8. Tezligi v bo'lgan avtomobil s masofani qancha vaqtda bosib o'tadi.
9. h balandlikqan erkin tushayotgan jism qancha vaqtdan keyin erga uriladi.
10. Jo'mrakdan 1 s da 1 milli litr suv tomsa x yilda necha litr suv tomadi.
11. 1 dan n gacha sonlar berilgan. Berilgan sonlarni yig'indisini toping.
12. Massasi m bo'lgan jismni og'irligini toping.
13. m massali jismga a tezlanish berilganda unga qanchali kuch tasir qiladi.

2-Topshiriq

$$1. Z = \frac{ax}{by^2} + \frac{a}{b^2} + c^3 + \frac{a^2 + 6a^2b^3 + 2c^2}{\sqrt[3]{a^3 + b^3 - 2ab}}$$

$$2. Q = \frac{Xa \sin^2 y^2 + ab \cos^2 x^3}{by^2} + \frac{a^2 + 2b^2 + 3c^3}{a + 2c^2 + \sqrt[3]{7a}}$$

$$3. y = 2x^2 + b^2x^3 + \sqrt[3]{a+b^2} - \frac{a^3 + b^3 + 3cd}{\sqrt[5]{2|a-b| + c^2}}$$

$$4. Z = \frac{a + b^3 + c^2}{2abc} + \sin^2 x^3 + (a + b)^{2\sin^2 x} + b \cos^2 y^3$$

$$5. y = \frac{2a^2 + bc^3 + 2d}{\sqrt[3]{a^3 - b^3} 3abc} - \frac{(a + b)^2 + 2ac^2}{a^2 + \sqrt[3]{b^2} + \ln x}$$

$$6. Q = \frac{2^{a+b} + c^2 + \sqrt[3]{x^2}}{(a - b)^2} - \frac{2x^2 + 7a^2b^3 + c^2}{a^2 + 8b^3 + c^2}$$

$$7. y = \frac{y^5 \sqrt{7,2631} + x^3 \sqrt{71,8672}}{\sqrt[3]{a} + \sqrt[5]{b}} - \frac{2a + 2b^2}{c^2 + d^2}$$

$$8. y = \frac{a + bx + c^2}{a^2 - b^2} + \frac{\ln x^2 - \ln ax}{2ab + c^2}$$

$$9. Z = \sqrt[5]{\frac{a}{x^2}} + \sqrt[3]{\frac{c^2}{b^2}} - \frac{2ab + c^2}{\sqrt[3]{a + b} - 2ab^2}$$

$$10. Z = \frac{2x^3 + \sqrt[3]{a + x^2}}{a^2 + bc} + \frac{\lg b + \ln c^2}{\cos^2 ax^3}$$

$$11. y = ax^2 + b + \frac{\sin x^2 + c}{\sqrt[3]{a + b^2}}$$

$$12. y = \frac{ax^3 + 7bx^2 + 8c^3}{x^2 + 3a^3} + \frac{a + 2ab + c^2}{\sqrt{a^2 - b^2}}$$

$$13. Q = \frac{ax^2 + \sqrt[3]{bc}}{2ab} + \frac{\sin^2 x + b^3}{\ln ax^2}$$

3-Topshiriq

$$1. Q = \frac{2x^2 + 7a^2b^3 + c^2}{a^2 + 8b^3 + c^2}$$

$$2. y = \sqrt[3]{a + b^2} - \frac{a^3 + b^3 + 3cd}{\sqrt[5]{2|a - b| + c^2}}$$

$$3. Q = \frac{Xa \sin^2 y^2 + ab \cos^2 x^3}{by^2}$$

$$4. y = \frac{a + bx + c^2}{a^2 - b^2}$$

$$5. y = \frac{2a^2 + bc^3 + 2d}{\sqrt[3]{a^3 - b^3} 3abc}$$

$$6. Z = \sin^2 x^3 + (a+b)^{2\sin^2 x} + b \cos^2 y^3$$

$$7. Z = \sqrt[3]{\frac{c^2}{b^2}} - \frac{2ab + c^2}{\sqrt[3]{a+b} - 2ab^2}$$

$$8. Z = \frac{a}{b^2} + \frac{a^2 + 6a^2b^3 + 2c^2}{\sqrt[3]{a^3 + b^3} - 2ab}$$

$$9. y = \frac{y\sqrt[5]{7,2631} + x\sqrt[3]{71,8672}}{\sqrt[3]{a} + \sqrt[5]{b}}$$

$$10. y = \frac{ax^3 + 7}{x^2 + 3a^3} + \frac{ab + c^2}{\sqrt{a^2 - b^2}}$$

$$11. Q = \frac{ax^2}{2ab} + \frac{\sin^2 x}{\ln ax^2}$$

$$12. Z = \frac{2x^3 + \sqrt[3]{a+x^2}}{a^2 + bc}$$

$$13. y = b + \frac{\sin x^2 + c}{\sqrt[3]{a+b^2}}$$

4-Topshiriq

$$1. c1 = \frac{x+y}{y^2 + \left| \frac{y^2+2}{x+x^3/5} \right|} + e^{y+2} \quad x - \text{butun, } y - \text{haqiqiy.}$$

$$2. f1 = \frac{2tg(x + \pi/6)}{1/3 + \cos^2(y+x^2)} + \log_2^{(x^2+2)} \quad x, y - \text{haqiqiy}$$

$$3. f2 = \frac{1/(x+2/x^2+3/x^3) + e^{x^2+3x}}{\arctg(x+y) + |5+x|^2} - \cos^2(y^2 + x^2/2);$$

$$4. \quad z = \ln \left| (x+y)^2 + \sqrt{|y|+2} - \left(x - \frac{xy}{x^2/2 - 5} \right) \right| + \frac{\cos^2(x+y)}{(x+y)^{1/3}}; \quad x, y - \text{butun}$$

$$5. \quad T11 = \frac{x^2 + 1}{x^2 + \frac{xy + y^2}{y^2 + \frac{y + xy}{|xy| + 5}}} + \frac{1}{1 + \cos x + \frac{1}{\sin |x|}} \quad x, y - \text{haqiqiy son}$$

$$6. \quad T = \sqrt[5]{a} + \sqrt[4]{b \cdot \frac{ax^2 + b}{2 \cdot b + a \cdot b}} \cdot (a^2 + x^2 + b^2 + 2) \quad a, b - \text{haqiqiy, } x - \text{butun son}$$

$$7. \quad F = \frac{|\sin^2 |cx_2^3 + dx_1^3 - cd||}{\sqrt{(cx_1^2 + dx_2^2 + 5) + 2}} + tg(x_1 \cdot x_2^2 + d^3); \quad x_1, x_2 - \text{haqiqiy, } s, d - \text{butun.}$$

$$8. \quad y2 = \frac{ax^2 + bx + c}{xa^3 + a^2 + a^{b-c}} + \cos \left| \frac{ax + b}{cx + d + 2^c} \right| \quad a, b, c, d - \text{butun, } x - \text{haqiqiy.}$$

$$9. \quad W1 = 0.75 + \frac{8.2x^2 + \sqrt{|x^3 + 3x|} + \cos(x-2)}{a/4 + b/3 + c/2 + 1} \quad a, b, c - \text{butun, } x - \text{haqiqiy.}$$

$$10. \quad TT = \frac{\sqrt{x-1} + \sqrt{x+2} + \lg(\sqrt{ax^2 + 2})}{\sqrt[2]{\sqrt{x+2} + \sqrt{x+24} + x^5}} \quad x - \text{haqiqiy, } a - \text{butun.}$$

$$11. \quad W2 = \sqrt{e^{xy} - x \cdot \sin(ax) - \frac{x^2 + 2}{|x| + 5}} + \sqrt{\ln(x^2 + 2) + 5} \quad a - \text{butun, } x, y - \text{haqiqiy.}$$

$$12. \quad AA = \sqrt{\frac{2tg(x+2) - \cos(x+2^x)}{1 + \cos^2(x+2)}} + \frac{\sin x^2}{x^2 + 3} \quad x - \text{haqiqiy son.}$$

$$13. \quad BB1 = x \cdot \sin(x/2 + x/3 + x/4) + \frac{\lg(x^2 - 2) + 3^a}{\cos(x+3) \cdot \sin(x+3) + 8} \quad a - \text{butun, } x - \text{haqiqiy.}$$

$$14. \quad TT = \sqrt{y^2 + e^x} + \sqrt{e^x + \frac{a}{x^2 + 2} + \frac{\cos^2 x}{\sin x^2}} + \cos^3 x; \quad a - \text{butun, } x - \text{haqiqiy.}$$

Amaliy ish №3

Mavzu: Shartli va shartsiz o'tish operatorlari. Tanlash operatori

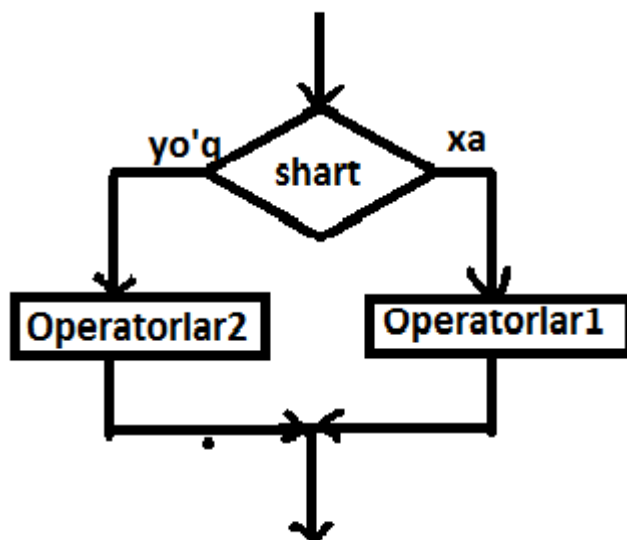
Ishdan maqsad. if shart, case tanlash operatorlarini tog'ri qo'llashni o'rganish, tarmoqlanuvchi algoritmgaga doir masalalarni dasturini tuzishni o'rganish.

Nazariy qism.

Agar algoritm qadamlari ketma-ket bajarilish jarayonida qandaydir shartga bo'g'liq ravishda o'zgarsa, bunday algoritm tarmoqlanuvchi algoritmgaga deb nomlanadi. Shart bu mantiqiy ifoda bo'lib, faqat rost yoki yolg'on qiymatni qabul qiladi. Agar shart rost bo'lsa Xa, yolg'on bo'lsa Yo'q tarmog'i bo'yicha algoritmgaga qadami davom etadi.

Tarmoqlanuvchi algoritmgaga to'liq tarmoqlanuvchi va to'liqmas tarmoqlanuvchi turlariga bo'linadi.

To'liq tarmoqlanuvchi algoritmgada shart bajarilganda va bajarilmaganda ikkalasida ham amallar bajariladi.



Agar shart bajarilsa Operatorlar1 bajariladi, aks holda Operatorlar2 bajariladi.

Tarmoqlanish shart asosida bo'ladi. Shart mantiqiy ifoda bo'ladi. Mantiqiy ifoda mantiqiy o'zgaruvchi, taqqoslash amallari yoki ularning inkor, konyunksiya, dizyunksiya amallaridan iborat bo'lishi mumkin. Shart operatori C++ da shart operatori quyidagicha yoziladi:

```
if (shart) {  
    Operatorlar1;
```

```

}
else {
    Operatorlar2;
}

```

C++ da taqqoslash amallari:

№	Matematika	C++
1	>	>
2	<	<
3	≥	>=
4	≤	<=
5	=	==
6	≠	!=

Misol1. $y = \begin{cases} x^2 & \text{agar } x \geq 0 \\ 2x & \text{agar } x < 0 \end{cases}$

Yechimi: y ning qiymati x ga bog'liq ravishda yoki x^2 formula, yoki $2x$ formula bo'yicha hisoblanadi. Tekshirilishi kerak bo'lgan shart $x \geq 0$.

```
#include <iostream>
```

```
using namespace std;
```

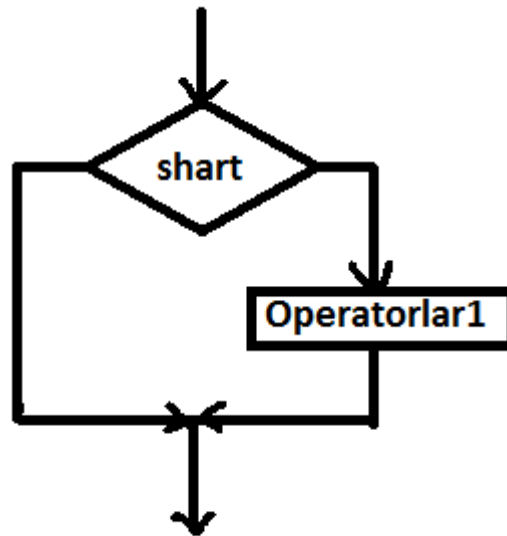
```

int main() {
    double x, y;
    cout<<"x=";
    cin>>x;
    if (x >= 0) {
        y = x * x;
    }
    else {
        y = 2 * x;
    }
    cout<<"y="<<y;
}

```

To'liqmas tarmoqlanuvchi algoritmda shart bajarilganda bu shartga bog'liq amallar bajariladi, bajarilmagan holatda hech qanday amal bajarish shart emas.

C++ da to'liqmas tarmoqlanuvchida faqat **if** operatori ishlatiladi, **else** ishlatilmaydi.



Misol2. a va b sonlari berilgan. Ulardan kattasini topuvchi dastur tuzing.

Yechimi: Dastavval a sonni maksimal deb tasavvur qilamiz. Agar b soni undan katta bo'lsa u holda b soni maksimal bo'ladi.

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
    double a, b;
    cout<<"Birinchi sonni kiriting: ";
    cin>>a;
    cout<<"Ikkinchi sonni kiriting: ";
    cin>>b;
    double max = a;
    if (b > max)
        max = b;
    cout<<a<<" va "<<b<<" sonlarining maksimali "<<max<<" ga teng";
}
```

if else ning boshqacha shaklda yozilishi.

if va else operatorlarini qisqacha shaklda ? va : belgilari orqali yozish mumkin.

Misol3. n natural soni berilgan. Agar u toq bo'lsa "odd", juft bo'lsa "even" so'zini chiqaruvchi dastur tuzing.

Yechimi: n natural soni toq bo'lishi uchun uni ikkiga bo'lganda qoldiq 1 ga teng bo'lishi kerak, aks holda juft bo'ladi.

```
#include <iostream>

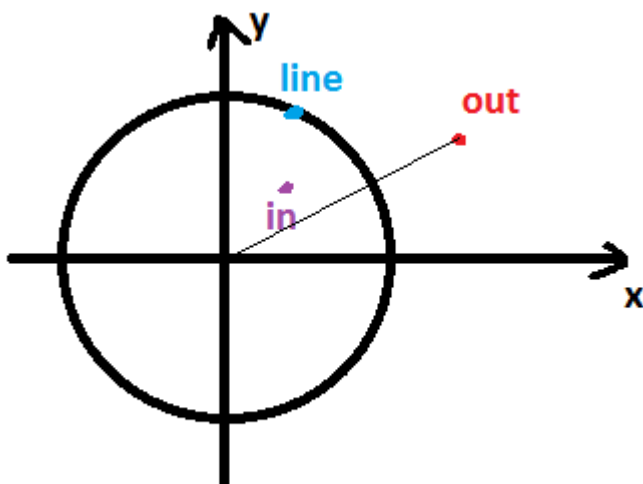
using namespace std;

int main() {
    int n;
    cin >> n;
    n % 2 == 1 ? cout<<"odd" : cout<<"even";
}
```

Murakkab tarmoqlanuvchi.

Agar biror shart asosida tarmoqlangandan so'ng yana shart asosida tarmoqlansa(ya'ni **else if**), bunday tarmoqlanish murakkab tarmoqlanish deyiladi.

Misol4. Markazi koordinatalar boshida va radiyusi R ga teng bo'lgan aylana berilgan. Tekislikdagi (x,y) nuqta bu aylanaga tegishliligini aniqlang. Agar aylana tashqarisida yotsa "out", chizig'ida yotsa "line", ichida yotsa "in" so'zini chiqaring.



Yechimi: Berilgan nuqtdan koordinata boshigacha masofani topamiz. Qaysi holat bo'lishi bu masofaga bo'g'liq. **Masofa** $d = \sqrt{x^2 + y^2}$ ga teng. Agar $d > R$ bo'lsa u holda nuqta aylanadan tashqarda, aks holda agar $d = R$ bo'lsa u holda nuqta aylanaga tegishli, aks holda(ya'ni bu holatda faqat $d < R$ shart qoldi) nuqta aylana ichkarisida yotadi. Bunga mos C++ da gi yechimi quyidagicha bo'ladi.

```
#include <iostream>

using namespace std;

int main() {
    int x, y, R;
    cout<<"x=";
    cin>>x;
    cout<<"y=";
```

```

cin>>y;
cout<<"R=";
cin>>R;
if (x*x+y*y > R*R) {
    cout<<"out";
}
else if (x*x+y*y==R*R) {
    cout<<"line";
}
else {
    cout<<"in";
}
}

```

C++ da murakkabroq shartlarni yozish.

Murakkab shart sodda shartlarning konyuksiya, dizyunksiya va inkorlaridan tashkil topadi.

Berilgan sonning $[a, b]$ intervalga tegishli ekanligini aniqlash uchun, $x \geq a$ va $x \leq b$ shartlari bir vaqtning o'zida o'rinli bo'lishi kerak. Shartlarning ikkalasi ham bajarilish shartini $\&\&$ (va - and) amali orqali yozamiz:

```

if (x >= a && x <= b)
    cout<<"Tegishli";
else
    cout<<"Tegishli emas";

```

Berilgan sonning $[a, b]$ intervalga tegishli emas ekanligini aniqlash uchun, $x < a$ yoki $x > b$ shartlari istalgan biri bajarilishi kerak. Shartlarning istalgan biri bajarilishi yetarliligi shartini $\|$ (yoki - or) amali orqali yozamiz:

```

if (x < a || x > b)
    cout<<"Tegishli emas";
else
    cout<<"Tegishli";

```

Tanlash operatori.

Tanlash operatori switch tanlanuvchi ifoda qiymatini birnechtakonstantalar bilan taqqoslab chiqadi. switch case ko'plik tanlov operatori hisoblanadi. switch da ko'rsatilgan ifosa qiymati case so'zidan keyin yozilgan har bir qiymat bilan taqqoslab chiqiladi. Taqqoslanuvchi qiymat qaysidir qatordagi case operatoridan yozilgan qiymatga teng u holda uning davomida yozilgan amallar bajariladi.

Misol5. Hafta kuni raqamda barilgan. Uni so'zda chiqaruvchi dastur tuzing.

Yechimi:

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {  
    int n;  
    cout<<"Hafta kunini raqamda kiriting: ";  
    cin>>n;  
    switch (n) {  
        case 1: cout<<"Dushanba"; break;  
        case 2: cout<<"Seshanba"; break;  
        case 3: cout<<"Chorshanba"; break;  
        case 4: cout<<"Payshanba"; break;  
        case 5: cout<<"Juma"; break;  
        case 6: cout<<"Shanba"; break;  
        case 7: cout<<"Yakshanba"; break;  
        default: cout<<"Hato kiritildi"; break;  
    }  
}
```

Agar har bir qatordan so'ng **break** yozilmasa u holda qaysidir shart bajarililadigan bo'lsa keying break operatori kelgunga qadar barcha holatdagi amallar bajariladi.

Masalan quyidagi dasturda

```
switch (n) {  
    case 1: cout<<"Dushanba";  
    case 2: cout<<"Seshanba";  
    case 3: cout<<"Chorshanba";  
    case 4: cout<<"Payshanba";  
    case 5: cout<<"Juma"; break;  
    case 6: cout<<"Shanba"; break;  
    case 7: cout<<"Yakshanba"; break;  
    default: cout<<"Hato kiritildi"; break;  
}
```

agar n=2 bo'lsa u holda ekranga SeshanbaChorshanbaPayshanbaJuma lar chiqadi.

1-topshiriq

1. x va y haqiqiy son berilgan. Xisoblang: $\max(x, y)$ va $\min(x, y)$
2. x , y va z haqiqiy son berilgan. Xisoblang: $\max(x, y, z)$ va $\min(x, y, z)$
3. x , y va z haqiqiy son berilgan. Xisoblang: $\max(x+y+z, x, y, z)$ va $\min^2(x+y/2, x, y, z)$
4. a , b va c haqiqiy son berilgan. Tekshiring: $a < b < c$ tengsizlik bajariladimi?
5. a , b va c haqiqiy son berilgan. Agar $a \geq b \geq c$ tengsizlik bajarilsa, u xolda haqiqiy sonlarni ikkilantiring, aks xolda ularni modullari bilan almashtiring.
6. Ikkita haqiqiy son berilgan. Birinchi sonni chiqaring, agar u ikkinchisidan katta bo'lsa, aks xolda ikkalasini ham chiqaring.
7. Ikkita haqiqiy son berilgan. Birinchi sonni no'l bilan almashtiring, agar u ikkinchisidan kichik yoki teng bo'lsa, aks xolda o'zgartirishsiz koldiring.

8. Uchta haqiqiy son berilgan. Ulardan $[1,3]$ intervalga tegishlilarini tanlang.
9. x, y (x va y teng emas) haqiqiy son berilgan. Ularning kichigini ularning yarim yig'indisi bilan, kattasini ularning ikkilangan ko'paytmasi bilan almashtiring.
10. Uchta haqiqiy son berilgan. Ularning musbatini kvadrati bilan almashtiring.
11. Agar uzaro farqli x, y, z haqiqiy sonlar birdan kichik bo'lsa, u xolda bu uchta sonidan eng kichigini boshka ikkitasining yarim yig'indisi bilan almashtiring. Aks xolda o'zgarishsiz qoldiring.
12. a, b, c va d haqiqiy sonlar berilgan. Agar $a \leq b \leq c \leq d$ tengsizlik bajarilsa, u xolda ularning har birini ularning kattasi bilan almashtiring.
13. x, y haqiqiy sonlar berilgan. Agar x va y manfiy bo'lsa, ularning har birini modullari bilan almashtiring; agar fakat bittasi manfiy bo'lsa ikkala sonning har birini 0.5 ga oshiring.
14. x, y, z haqiqiy musbat sonlar berilgan. x, y, z uzunlikka ega tomonli uchburchak mavjudmi?
15. a, b, c haqiqiy sonlar berilgan ($a \neq 0$). Aniqlanki, $ax^2+bx+c=0$ kvadrat tenglama haqiqiy echimga egami. Agar haqiqiy echimlari mavjud bo'lsa, u xolda ularni toping. Aks xolda haqiqiy echimlar mavjud emasligi xaqida xabar bering.

2-topshiriq

$$1 \quad Q = \begin{cases} \frac{ax^2 + \sqrt[3]{bc}}{2ab}, & \text{a} \text{ gap } ab \neq 0 \\ \frac{\sin^2 x + b^3}{\ln ax^2}, & \text{a} \text{ gap } a > 0 \end{cases}$$

$$2 \quad y = \begin{cases} \frac{\sin^2 ax + \cos^3 ax^2}{(a+b)^2 + c}, & \text{a} \text{ gap } a < 0 \\ \frac{\sqrt{a^2 - b^3} + 2ac}{\sin^2 a + bx^2}, & \text{a} \text{ gap } a^2 \geq b^3 \end{cases}$$

$$3 \quad y = \begin{cases} \frac{a + b^2 + 2abx}{\sqrt[3]{c^2 + b^3}}, & \text{a} \text{ gap } x > 0 \\ \frac{a^2 + 2ab + c^2}{7a^2 + 8b^3}, & \text{a} \text{ gap } a \neq 0 \end{cases}$$

$$4 \quad y = \begin{cases} \frac{a \sin x + b \cos x^2}{a^2 + b^2}, & \text{a} \text{ gap } ab \neq 0 \\ \frac{a^3 + 2ab^2 + c^3}{a + b^2}, & \text{a} \text{ gap } b > 0 \end{cases}$$

$$5 \quad Z = \begin{cases} \frac{2x^3 + ax^2 + c^3}{a^2 - c^2}, & \text{a} \text{ gap } a > c \\ \frac{2ab + \sin ax^2}{\sqrt{a^2 + 2q^2}}, & \text{a} \text{ gap } q \neq 0 \end{cases}$$

$$6 \ y = \begin{cases} \frac{2c^2 + abc \cos x^2}{a \sin^2 x^2 + b^3}, & \text{a} \geq 0 \\ \frac{a^2 + 2bc^3}{\sqrt{a^3 + 2c^2 + d^3}}, & \text{a} \neq 0 \end{cases}$$

$$7 \ Q = \begin{cases} \frac{a + b^2 + 2c^3}{a \cdot \sin^2 x^2}, & \text{a} > |b| \\ \frac{a^2 + 2ab + c^2}{a + 2c^2 + xy}, & xy > a \end{cases}$$

$$8 \ Z = \begin{cases} \sqrt[3]{cy^2 + a^3} - \ln x, & x > 0 \\ \frac{a^2 + bx + c^2}{\sqrt{a^3 - b^3}}, & a^3 > b^3 \end{cases}$$

$$9 \ y = \begin{cases} \frac{ax^2 + cx^3}{\sqrt{a^2 - b} + c}, & a^2 > b \\ \frac{\sin^2 x^3}{\cos x^2 + ab}, & x \neq 0 \end{cases}$$

$$10 \ Q = \begin{cases} \frac{2x + ax^2 + c^3}{a^2 + b^2}, & x > c \\ \frac{a(x^2 + y) + 2xy^2}{a^2 - b^2}, & a \neq b \end{cases}$$

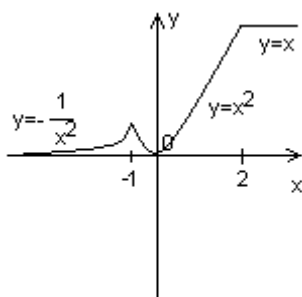
$$11 \ y = \begin{cases} \frac{ax^2 + b + 2c^3}{x^2 + g^2}, & x \neq g \\ \frac{2c^2 + \sqrt[5]{a^2}}{c^3 + ax^2 + d}, & x < 0 \end{cases}$$

$$12 \ Z = \begin{cases} \frac{2x^2 + ab^3 + c^2}{2a + c^2}, & x \neq 0 \\ \frac{2a^2 + 7b^3 + 12c}{2x + 7ax^2 + c^3}, & c > 0 \end{cases}$$

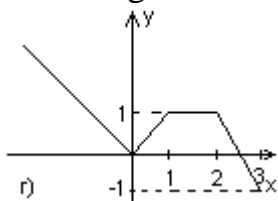
$$13 \ U = \begin{cases} \frac{a^2 + 4abc + c^2}{a - b^2}, & a > b^2 \\ \sqrt[3]{\frac{2a}{c^2}} + \sqrt[5]{\frac{c^2}{x^2}}, & x \neq 0 \end{cases}$$

3-topshiriq

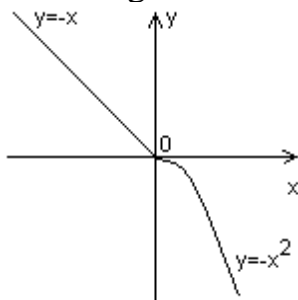
1. a xaqiqiys onberilgan. Quydagi rasmlardatasvirlangany(x)funktsiyauchuny(a)ni xisoblang.



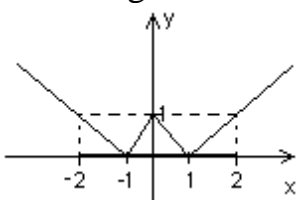
2. axaqiqiysonberilgan. Quydagirasmlardatasvirlanganf(x) funktsiyauchunf(a)ni xisoblang.



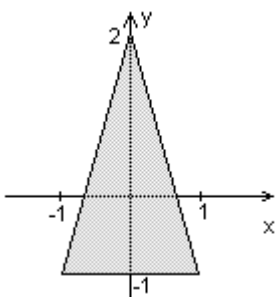
3. axaqiqiysonberilgan. Quydagi rasmlardatasvirlanganf(x) funktsiyauchunf(a)ni xisoblang.



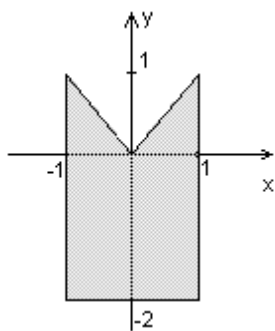
4. axaqiqiysonberilgan. Quydagi rasmlardatasvirlanganf(x) funktsiyauchunf(a)ni xisoblang.



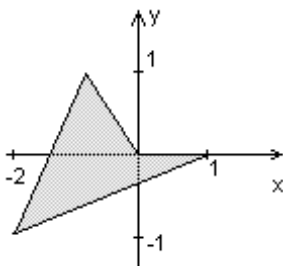
5. x, y xaqiqiysonlarberilgan. Koordinatalari (x, y) bo`lgannuqtaquyda keltirilgan rasmdagi tekislikningshtrixlanganqismigategishlimi? (ha/yo`q)



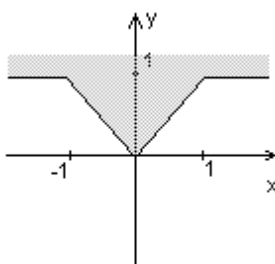
6. x, y xaqiqiysonlarberilgan. Koordinatalari (x, y) bo`lgannuqtaquyda keltirilgan rasmdagi tekislikningshtrixlanganqismigategishlimi? (ha/yo`q)



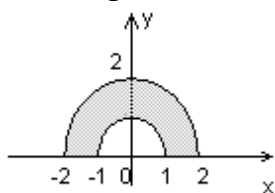
7. x, y xaqiqiysonlar berilgan. Koordinatalari (x, y) bo'lgan nuqta quyda keltirilgan rasmdagi tekislikning shtrixlangan qismiga tegishlimi? (ha/yo'q)



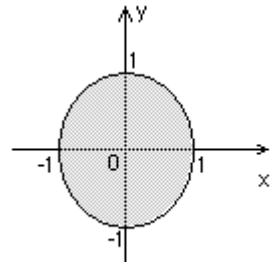
8. x, y xaqiqiysonlar berilgan. Koordinatalari (x, y) bo'lgan nuqta quyda keltirilgan rasmdagi tekislikning shtrixlangan qismiga tegishlimi? (ha/yo'q)



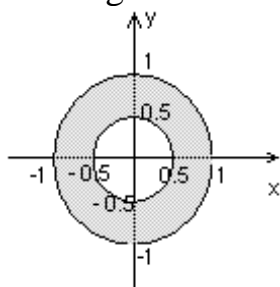
9. x, y xaqiqiysonlar berilgan. Koordinatalari (x, y) bo'lgan nuqta quyda keltirilgan rasmdagi tekislikning shtrixlangan qismiga tegishlimi? (ha/yo'q)



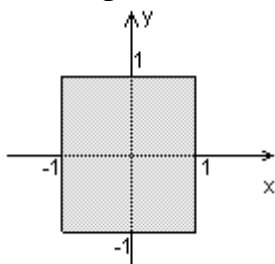
10. x, y xaqiqiysonlar berilgan. Koordinatalari (x, y) bo'lgan nuqta quyda keltirilgan rasmdagi tekislikning shtrixlangan qismiga tegishlimi? (ha/yo'q)



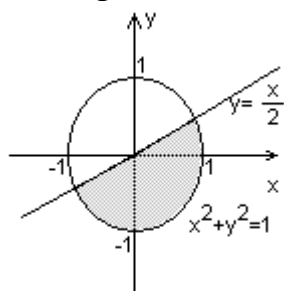
11. x, y haqiqiy sonlar berilgan. Koordinatalari (x, y) bo'lgan nuqta quyida keltirilgan rasmdagi tekislikning shtrixlangan qismiga tegishlimi? (ha/yo'q)



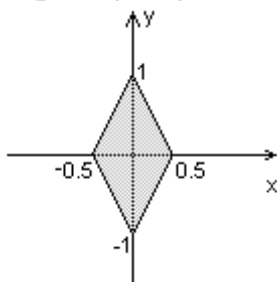
12. x, y haqiqiy sonlar berilgan. Koordinatalari (x, y) bo'lgan nuqta quyida keltirilgan rasmdagi tekislikning shtrixlangan qismiga tegishlimi? (ha/yo'q)



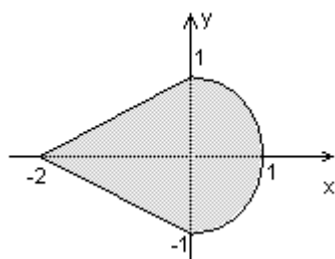
13. x, y haqiqiy sonlar berilgan. Koordinatalari (x, y) bo'lgan nuqta quyida keltirilgan rasmdagi tekislikning shtrixlangan qismiga tegishlimi? (ha/yo'q)



14. x, y haqiqiy sonlar berilgan. Koordinatalari (x, y) bo'lgan nuqta quyida keltirilgan rasmdagi tekislikning shtrixlangan qismiga tegishlimi? (ha/yo'q)



15. x, y haqiqiy sonlar berilgan. Koordinatalari (x, y) bo'lgan nuqta quyida keltirilgan rasmdagi tekislikning shtrixlangan qismiga tegishlimi? (ha/yo'q)



Amaliy ish № 4

Mavzu: Takrorlanish operatorlari(while, do while, for).

Ishtdan maqsad: C++ dasturlash tilining sikl operatorlari bilan tanishish. Sharti oldindan, shart oxiridan beriladigan takrorlanish va parametrik sikl operatorlarini o'rganish.

Nazariy qism.

Dastur kodining biror qismining ko'p marta bajarilishi sikl hisoblanadi. Dastur kodining qandaydir qismini qandaydir shart asosida birnecha marta bajartirish uchun dasturlashda sikldan foydalaniladi. Agar shart rost bo'lsa sikl davom qiladi. Aks holda to'xtatiladi. Agar shart hamisha rost bo'lsa bunday sikl cheksiz sikl deb ataladi.

C++ da siklni tashkil qilish uchun *while*, *do while* va *for* operatorlari mavjud.

Siklni o'rganish uchun eng oson misol bu 1 dan n gacha natural sonlarning yig'indisini($1+2+3+\dots+n$) topish dasturini tuzish. Bu yig'indini takrorlanish jarayoni orqali hisoblash uchun 1 dan n gacha sonlarni birma-bir qo'shib chiqish lozim. Yig'indining dastlabki qiymatini 0 ga tenglaymiz. Siklning har bir qadamida quyidagi amallar bajariladi:

```
s=0;
1-qadam.  $s=s+1=0+1=1$ ;
2-qadam.  $s=s+2=1+2=3$ ;
3-qadam.  $s=s+3=3+3=6$ ;
4-qadam.  $s=s+4=6+4=10$ ;
5-qadam.  $s=s+5=10+5=15$ ;
.....
i-qadam.  $s=s+i$ ;
.....
n-qadam.  $s=s+n$ ;
```

Har bir qadamda bir xil amal bajariladi, ya'ni yog'indining yangi qiymatini hosil qilish uchun uning avvalgi qadamdagi qiymatiga navbatdagi natural son qo'shiladi.

1) *while* sikli.

Bu siklda shart oldindan qo'yiladi. Agar shart rost bo'lsa sikl tanasi bajariladi. Aks holda sikl to'xtab undan keyingi qadamga o'tiladi.

```
while (shart) {

    sikl tanasi

}
```

1 dan n gacha sonlar yig'indisini topish uchun har bir qadamda navbatdagi sonni qo'shib borish uchun i o'zgaruvchi e'lon qilamiz.

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
    int s = 0, i = 1, n;
    cout<<"n=";
    cin>>n;
    while (i <= n) {
        s += i;
        i++;
    }
    cout<<"s="<<s;
}
```

Dastur kodini bir boshdan qarab chiqamiz. Bizga uchta o'zgaruvchi kerak. Birinchi o'zgaruvchi n soni, ikkinchi o'zgaruvchi sanab borish uchun ishlatiladigan i o'zgaruvchisi, uchinchi yig'indining qiymatini saqlash uchun s o'zgaruvchi. Siklni boshlashdan oldin yig'indining qiymatini nolga tenglaymiz, shunda unga qandaydir sonni birinchi marta qo'shganimizda uning o'zi hosil bo'ladi. i o'zgaruvchining dastlabki qiymatini 1 ga tenglaymiz, chunki 1 dan boshlab yig'indiga qo'shib borishimiz lozim. Agar $i \leq n$ shart bajarilsa u holda i ni yig'indiga qo'shamiz ($s+=i$ bu $s=s+i$ ning qisqacha yozilishi) va i ning qiymatini orqali birga oshiramiz ($i++$ bu inkrement).

Cheksiz sikl.

while yordamida cheksiz sikl hosil qilish uchun shart ifodaga hamisha rost qiymat qabul qiladigan mantiqiy ifoda, o'zgaruvchi yoki rost konstanta qiymatini yozishimiz mumkin.

```
while (1) {
    cout<<"Cheksiz sikl\n";
}
```

2) do while sikli.

do while sikli while sikliga o'xshash, farqi shart sikl oxirida tekshiriladi va shart bajarilsin yoki bajarilmasin kamida bir marta (1-sikl) sikl bajariladi.

1 dan n gacha sonlar yig'indisi quyidagicha yoziladi:

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
    int s = 0, i = 1, n;
    cout<<"n=";
```

```

cin>>n;
do {
    s += i;
    i++;
} while (i <= n);
cout<<"s="<<s;
}

```

Bu siklda i o'zgaruvchining qiymati qanday bo'lishidan qat'iy nazar sikl bir marta aylanadi. Bu siklni sonni kiritishda unig tog'riligini tekshirish va toki to'g'ri kiritilmaguncha kiritishni davom qildirish uchun foydalanishimiz mumkin. Masalan yuqoridagi masalamizda n soni natural bo'lishi kerak, agar natural son kiritilmasa yana kiritishni so'rash lozim:

```

do {
    cout<<"n=";
    cin>>n;
} while (n < 1);

```

3) *for* sikli.

for sikli sintaksisi quyidagicha:

```

for(sikl boshlanishidan oldingi amallar; sikl davom etish sharti; siklning har bir iteratsiyasi oxiridagi amallar) {
    sikl tanasi;
}

```

Iteratsiya deb siklning bir marta bajarilishiga aytiladi. Agar ma'lum qadam bilan bitta o'zgaruvchining qiymatini o'zgartirib takrorlanuvchi jarayon amalga oshirish lozim bo'lsa, u holda uni quyidagicha xususiy holda yozishimiz mumkin:

```

for(<o'zgaruvchi tipi> o'zgaruvchi =boshlang'ich qiymat; o'zgaruvchi <=oxirgi qiymat; o'zgaruvchi +=sikl qadami) {
    sikl tanasi;
}

```

1 dan n gacha sonlar yig'indisini topish uchun quyidagicha sikl amalga oshirishimiz mumkin:

```

#include <iostream>
using namespace std;
int main() {
    int s = 0, n;
    cout<<"n=";
    cin>>n;
    for (int i = 1; i <= n; i++) {
        s += i;
    }
    cout<<s;
}

```

Bu siklda i ning qiymati sikl boshlanishidan avval 1 ga teng qiymatni qabul qiladi. Yana bitta iteratsiya qilish uchun bajarilishi kerak bo'lgan shart $i \leq n$, agar shart rost bo'lsa, yana bitta iteratsiya bajariladi, iteratsiya oxirida i ning qiymati birga oshiriladi ($i++$). Keyingi har bir iteratsiyada *for* siklining ikkinchi va uchinchi qismlari bajariladi, 1-qismi boshqa bajarilmaydi. Eng oxirgi iteratsiyadan oxirida i ning qiymati oshirilgach $n+1$ ga teng bo'ladi va keyingi iteratsiyada shart yolg'on qiymat qabul qilganligi sababli ($n+1 \leq n$ yolg'on qiymat qabul qiladi) sikl aylanishi tugaydi.

Sikl o'zgaruvchisi i haqiqiy son ham bo'la oladi. Masalan 1 dan 10 gacha sonlarni 0.01 qadam bilan chiqarish uchun ya'ni 1, 1.01, 1.02, 1.03, ..., 10 sonlarini chiqarish uchun quyidagicha sikl yoziladi.

```

for (double x = 1; x <= 10; x += 0.01) {

```



```
        cout<<x<<" ";
    }
```

x sikl parametri bu safar haqiqiy qiymatni qabul qiladi va har bir iteratsiya oxirida qiymati 0.01 ga oshiriladi.

for siklining uchta qismidan istalgan qismini yozmaslik mumkin:

```
double x = 1;
for (; x <= 10; x += 0.01) {
    cout<<x<<" ";
}
```

bu kod avvalgi yozilgani bilan bir xil, faqat $x=1$ dastlabki qiymatni o'zlashtirish *for* ichida yozilmadi.

```
double x = 1;
for (; x += 0.01) {
    cout<<x<<" ";
}
```

Bu kod qismida x ning qiymati 1 dan boshlab 0.01 qadam bilan oshirib boriladi, lekin to'xtash sharti yozilmadi, shuning uchun cheksiz sikl hosil bo'ladi.

```
double x = 1;
for (; ) {
    cout<<x<<" ";
}
```

Bu holatda esa x ning qiymati iteratsiya oxirida o'zgartirilmadi shuning uchun cheksiz ko'p marta x ning dastlabki qiymati 1 chirariladi.

break operatori.

break operatori siklni uning bajarilish sharti rost qiymat qabul qilishiga qaramasdan to'xtatish uchun qo'llaniladi. Yuqoridagi x ning qiymati 1 dan 100 gacha 0.01 qadam bilan oshirib boradigan misolda

```
double x = 1;
for (; ) {
    if (x > 100.000001)
        break;
    cout<<x<<" ";
    x += 0.01;
}
```

break operatorining ishlatishga misollardan biri berilgan sonning tub yoki tub emasligini aniqlaydigan dastur yozish.

Sonning tub ekanligini aniqlash uchun uni 2 dan $\lfloor \sqrt{n} \rfloor$ gacha bo'lgan sonlarga bo'linishini tekshiramiz. Agar ulardan biriga qoldiqsiz bo'linadigan bo'lsa, u holda bu son tub emas. 103 sonining tub ekanligini aniqlash uchun 2,3,4,5,6,7,8,9 va 10 sonlariga bo'linishini tekshiramiz.

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
    int n;
    cin>>n;
    bool is_prime = true;
    for (int i = 2; i*i <= n; i++) {
        if (n % i==0) {
            is_prime = false;
            break;
        }
    }
    if (n==1)
        is_prime = false;
    if (is_prime)
```

```

        cout<<"Tub";
    else
        cout<<"Tub emas";
}

```

Dastur kodini taxlil qilib chiqamiz. $\text{cin}>>n$ – n sonini kiritish. `is_prime` o'zgaruvchisi - berilgan sonning tub ekanligining rost yoki yolg'onligini saqlovchi qiymat. Dastlab sonni tub deb tasavvur qilamiz(`is_prime = true`). 2 dan $\lfloor \sqrt{n} \rfloor$ gacha sonlarni ko'rib chiqish uchun `for (int i=2; i<=sqrt(n); i++)` ko'rinishida siklni amalga oshirish lozim. `i<=sqrt(n)` shartning ikkala tamoni kvadratga ko'tarib, uning o'rniga `i*i <= n` shartni yozish mumkin.

n soni i ga qoldiqsiz bo'linishi uchun n ni i ga bo'lgandagi qoldiq qiymati nolga teng bo'lishi kerak(`if (n % i==0)`). Agar bunday shart bajarilsa, u holda tekshiilayotgan son tub emas degan xulosaga kelinadi, ya'ni uning 1 dan kata va o'ziga teng bo'lmagan birorta bo'luvchisi bor. Endi qolgan sonlarga bo'linishini tekshirishning zaruriyati yo'q, siklni to'xtatish mumkin. Berilgan son tub emas degan xulosaga kelamiz(`is_prime = false`) va siklni to'xtatamiz(`break`).

Agar $n=1$ bo'lsa n soni 2 dan boshlab hech bir songa bo'linmaydi va `is_prime` true qiymatini saqlab qoladi. Buni alohida tekshirish lozim: agar n birga teng bo'lsa u holda u tub emas. Agar berilgan son tub bo'lsa `is_prime` o'zgaruvchisi **true** qiymatni saqlab qoladi.

continue operatori.

continue operatori siklni to'xtatmasdan, uni keyingi iteratsiyadan davom qildirib ketish uchun ishlatiladi. Masalan a dan b gacha sonlar yig'indisi va ular ichidan n ga qoldiqsiz bo'linmaydigan sonlar sonini topish dasturini `for` sikli yordamida quyidagicha yozish mumkin:

```

#include <iostream>
using namespace std;
int main() {
    int a, b, n;
    cin>>a>>b>>n;
    int sum = 0, cnt = 0;
    for (int i = a; i <= b; i++) {
        sum += i;
        if (i % n != 0)
            cnt++;
    }
    cout<<a<<" dan " <<b<<" gacha sonlar yig'indisi: " <<sum<<endl;
    cout<<n<<" ga bo'linmaydigan sonlar soni: " <<cnt<<endl;
}

```

a dan b gacha barcha sonlarni ko'rib chiqamiz, `sum += i` summaga barcha i larni qo'shib boramiz, agar navbatdagi son i ga qoldiqsiz bo'linsa `if (i % n != 0)`, i ga bo'linadigan sonlar sonini birga oshiramiz(`cnt++`). Siklni **continue** operatori bilan quyidagi shaklda ham yozish mumkin:

```

for (int i = a; i <= b; i++) {
    sum += i;
    if (i % n == 0)
        continue;
    cnt++;
}

```

Bu shaklda yozilganda `sum += i` hamisha bajariladi. Agar `n % i == 0` shart bajarilsa u holda siklning navbatdagi iteratsiyasiga o'tiladi. Ya'ni bizga n ga bo'linmaydigan sonlar soni kerak. Agar `n % i == 0` shart bajarilmasa, u holda sikl tanasining navbatdagi amali ya'ni `cnt++` bajarilib bo'linmaydigan sonlar soni birga oshiriladi.

Topshiriqlar

Topshiqnlarni unda ko'rsatilgan sikldan foydalanib yozing.

1-Topshiriq. *while* sikli

1. $S = \frac{x-1}{1} - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \frac{(x-1)^4}{4} + \dots + \frac{(-1)^{n-1}(x-1)^n}{n}$
2. $P = 1 - \frac{x^2}{1} + \frac{x^4}{2} - \frac{x^6}{3} + \dots + \frac{(-1)^n x^{2n}}{n}$
3. $S = 1 - x^2 + x^4 - x^6 + \dots + (-1)^n \cdot x^{2n}$
4. $SS = x - \frac{x^2}{2} + \frac{x^3}{3} - \dots + \frac{(-1)^{n-1} x^{2n-1}}{n}$
5. $S = x + \frac{x^2}{2} + \frac{x^3}{3} + \dots + \frac{x^{2n-1}}{n}$
6. $P = x - \frac{x^3}{3} + \frac{x^5}{5} - \dots + \frac{(-1)^{n+1} x^{2n-1}}{2n-1}$
7. $S = x + \frac{x^3}{3} + \frac{x^5}{5} + \dots + \frac{x^{2n-1}}{(2n-1)}$
8. $PP = 1 + \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$
9. $S = k + \frac{k^2}{2} + \frac{k^3}{3} + \dots + \frac{k^n}{n}$
10. $S = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^n}{n!}$
11. $S = \frac{x}{1!} + \frac{x}{2!} + \frac{x}{3!} + \dots + \frac{x}{n!}$
12. $S = \frac{\pi}{10} + \frac{\pi^3}{10^3} + \frac{\pi^5}{10^5} + \dots + \frac{\pi^{2n+1}}{10^{2n+1}}$
13. $S = \frac{\pi}{10} + \frac{\pi}{10^3} + \frac{\pi}{10^5} + \dots + \frac{\pi}{10^{2n+1}}$

2-Topshiriq. *do while* sikli

1. $y = \sqrt[3]{\frac{\sin ax + b^{2c}}{b^2 + \cos^2 x}} - \frac{\sin x^2}{ab}, \quad c \leq x \leq d, n = 25$
2. $y = \sqrt[3]{\frac{ax + b}{b^2 + \cos^2 x}} - \frac{\sin x^2}{ab}, \quad a \leq x \leq c, h = 0.3$
3. $y = \sqrt[3]{a^a + x^2 \cos ax}, \quad -\frac{\pi}{2} \leq x \leq \pi; h = \frac{\pi}{19}, a = 3.26$
4. $y = \frac{a^2 + bx + x^c}{a^2 + b^2 + x^2}, \quad 5 \leq x \leq 10, h = 0.4$
5. $y = a^2 \cos x + \frac{\sin x}{2} + bx^2, \quad c \leq x \leq e, h = 0.2$
6. $y = \sqrt[3]{\frac{\sin ax + b^{2c}}{b^2 + \cos^2 x}} - \frac{\sin x^2}{ab}, \quad -1 < x < 1, n = 50$
7. $y = a^2 + \sqrt[5]{\frac{b + \sin x}{a^3 + \cos^2 x^3}}, \quad 1 \leq a \leq 12; n = 20$

$$8. \quad y = \frac{ax^2}{b} + \frac{x}{c}, \quad 1 \leq x \leq 10, h=8, a=3, b=12, c=6$$

$$9. \quad y = a \cos x - \sin x^2, \quad 0 \leq x \leq 10, h=0,5$$

$$10. \quad y = \sqrt[k]{\frac{ax+b}{b^2 + \cos^2 x}} - \frac{\sin x^2}{ab}, \quad d \leq x \leq c, n=15$$

$$11. \quad y = \sqrt{\frac{\sin ax + b^{2c}}{b^2 + \cos^2 x}} - \frac{\sin x^2}{ab}, \quad 0 \leq x \leq 1, n=50$$

$$12. \quad y = \frac{\log a^{2\sin x} + e^{2x}}{\arctan x + 2}, \quad -\pi \leq x \leq \pi, n=10$$

$$13. \quad y = \frac{a^b + b^x + c^a}{2x^2 + 3a^{x+c}}, \quad 3 \leq x \leq 5, h=0.2$$

$$14. \quad y = \frac{ax^2 + bx + 4}{a^2 + b^2 + x^2}, \quad 1 \leq x \leq 20, n=100$$

$$15. \quad y = 2\sqrt[3]{a^{\sin 2x}} + x^2 \cos ax, \quad -\frac{\pi}{2} \leq x \leq \pi; n=10$$

3-Topshiriq. for sikli

$$1. \quad S = \frac{\sin 1}{2^1} + \frac{\sin 2}{2^2} + \dots + \frac{\sin n}{2^n}$$

$$2. \quad S = \frac{\sin 1^1}{2^1} - \frac{\sin 2^2}{2^2} + \dots + (-1)^{2n} \frac{\sin n^n}{2^n}$$

$$3. \quad S = \frac{1}{1!} - \frac{1}{3!} + \frac{1}{5!} - \frac{1}{7!} + \dots + (-1)^{n-1} \frac{1}{(2n-1)!}$$

$$4. \quad S = \frac{1}{x^2} - \frac{1}{x^4} + \frac{1}{x^6} - \dots + (-1)^{2n} \frac{1}{x^{2n}}$$

$$5. \quad S = \frac{1}{x^2} + \frac{2}{x^4} + \frac{3}{x^6} + \dots + \frac{n}{x^{2n}}$$

$$6. \quad S = \sin x - \frac{1}{2} \sin 2x + \frac{1}{3} \sin 3x - \frac{1}{4} \sin 4x + \dots + (-1)^{2n} \frac{1}{n} \sin nx$$

$$7. \quad S = \frac{x^1}{\sqrt{1}} + \frac{x^2}{\sqrt{2}} + \dots + \frac{x^n}{\sqrt{n}}$$

$$8. \quad S = 1 + \frac{x^1}{1!} + \frac{x^2}{2!} + \dots + \frac{x^n}{n!}$$

$$9. \quad S = 1 - \frac{x^1}{1!} + \frac{x^2}{2!} - \dots + (-1)^{2n-1} \frac{x^n}{n!}$$

$$10. \quad S = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots + \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!}$$

$$11. \quad S = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots + \frac{(-1)^{n-1} x^{2n-2}}{(2n-2)!}$$

$$12. \quad S = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \dots + \frac{x^{2n-2}}{(2n-2)!}$$

$$13. S = x + \frac{x^2}{2} + \frac{x^3}{3} + \dots + \frac{x^{2n-1}}{2n-1}$$

$$14. S = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \dots + \frac{x^{2n-1}}{(2n-1)!}$$

$$15. S = 1 - \frac{k^1}{1!} + \frac{k^2}{2!} - \frac{k^3}{3!} + \dots + \frac{(-1)^{2n-1} k^n}{n!}$$

4-Topshiriq.

1. Variant

$$a. S = \sum_{m=1}^{19} \frac{3m^3 + 4m + 5}{m^3 + \ln(m-3)}$$

$$b. P = \prod_{k=1}^{46} \frac{k}{k^3 + 7k + 5}.$$

$$c. S = \sum_{i=1}^{32} \prod_{m=1}^{14} \frac{\ln i + m^i}{m^i + n^{2i}}.$$

2. Variant

$$a. S = \sum_{a=1}^{27} \frac{a^2 + 2a}{a^3 + a \cos^2 a + 1}.$$

$$b. P = \prod_{i=1}^{20} \frac{i^2 + 1}{i\sqrt[3]{i^3} + 2}.$$

$$c. S = \sum_{i=1}^{17} \prod_{k>12}^{28} \ln \frac{k^i + i\sqrt{k}}{k^3 + \sqrt[k]{i}}.$$

3. Variant

$$a. S = \sum_{x=5}^{10} (ax + b)^2.$$

$$b. P = \prod_{a=10}^{15} \frac{a+b}{\sqrt{a^2 + x^2}}.$$

$$c. S = \sum_{k=1}^5 \sum_{y=2}^7 \frac{ak + by}{\sqrt{k^2 + y^2}}.$$

4. Variant

$$a) P = \sum_{i=1}^{19} \frac{i^4 + i^2 + 3}{\sqrt{i^1 + e^i}}.$$

$$b) S = \sum_{k=1}^{20} \frac{k+1}{k^3 + 5k + 7}.$$

$$c) S = \sum_{m=1}^{25} \prod_{n=1}^{30} \sqrt{\frac{m^n - n^m}{m^n + n^m}}.$$

5. Variant

$$a) \quad P = \prod_{i=1}^{34} \frac{i^3 + |i-9|}{\ln i + 7i}.$$

$$b) \quad S = \sum_{k=1}^{10} \frac{(-1)^k \cdot (k+1)}{k^3 + k^2 + 1}.$$

$$c) \quad P = \prod_{n=1}^{34} \sum_{m=1}^{35} (-1)^m \frac{\lg(m+5)}{m^{n+3} + n \cdot m}.$$

6. Variant

$$a. \quad S = \sum_{n=1}^{10} \frac{1}{5 - 17n + n^3}.$$

$$b. \quad P = \prod_{m=0}^{12} \frac{\sqrt[2]{|m-5|+1}}{m^3 + 4m + (-1)^3}.$$

$$c. \quad S = \sum_{ii=1}^{33} \prod_{k=1}^{15} (-1)^i \frac{\sqrt[7]{\sin k + e^k}}{|4i^3 - k^4|}.$$

7. Variant

$$a. \quad S = \sum_{a=5}^8 \frac{ax + bc}{a^2 + x}.$$

$$b. \quad P = \prod_{x=2}^5 \frac{a + bx}{a^2 + x^2}.$$

$$c. \quad SP = \sum_{k=1}^4 \prod_{a=2}^6 \frac{ak + bx}{k^2 + x^2}.$$

8. Variant

$$a. \quad S = \sum_{k=2}^8 (ak + bx).$$

$$b. \quad P = \prod_{a=3}^7 \frac{ax + b}{\sqrt{a^2 + b^2}}.$$

$$c. \quad SP = \sum_{x=1}^5 \prod_{a=2}^4 \frac{ax + b}{\sqrt{a^2 + bx}}.$$

9. Variant

$$a. \quad S = \sum_{a=1}^7 \frac{ax + b}{\sqrt{a + b}}.$$

$$b. \quad P = \prod_{x=2}^6 \frac{ax^2 + b}{\sin(ax)}.$$

$$c. \quad PP = \prod_{x=2}^5 \prod_{i=1}^4 \frac{(a \cdot i + bx)}{\sqrt{(ax + b)^i}}.$$

10. Variant

$$a. S = \sum_{i=10}^{30} \sqrt{(ax+b)^i}.$$

$$b. P = \prod_{k=1}^{15} \frac{\sin^k(a+b) + 3a}{\cos(ak) + 2,78b}.$$

$$c. SP = \sum_{i=1}^{10} \prod_{k=1}^5 \frac{ax^k + i \cdot b}{a \cdot i + b \cdot k}.$$

11. Variant

$$a. S = \sum_{x=5}^{10} \frac{kx+b^2}{\sqrt{x^2+ab}}.$$

$$b. P = \prod_{a=1}^6 \frac{ax+c}{\sin^2 ax}.$$

$$c. PS = \prod_{k=1}^4 \sum_{x=2}^5 \frac{ax+kb}{\sqrt{x^2+k^2}}.$$

12. Variant

$$a. S = \sum_{k=1}^5 (ax^2 + bk).$$

$$b. P = \prod_{a=3}^6 (ax + \cos^2 ab).$$

$$c. PS = \prod_{x=1}^4 \sum_{a=3}^6 \frac{ax+bk^2}{a^2+x^2}.$$

13. Variant

$$a. S = \sum_{x=2}^{12} \frac{ax^2+b}{\cos^2(a+bx)}.$$

$$b. P = \prod_{a=3}^9 \sqrt[3]{\frac{ax^2+b}{2 \cdot a}}.$$

$$c. S = \sum_{a=1}^5 \sum_{x=2}^7 \frac{a \cdot \cos^2(2x)}{b+c \cdot \sin(ax)}.$$

14. Variant

$$a. S = \sum_{i=1}^{81} \frac{i+2i+7}{i^2-1+\cos^2 i}.$$

$$b. P = \prod_{k=1}^{28} \sin \frac{k+1}{k+\sqrt{k+1}}.$$

$$c. S = \sum_{k=1}^{15} \prod_{n=8}^{21} \frac{n^k+k^n}{\sqrt[n]{R^k}+(n \cdot k)^5}.$$

Amaliy ish №5

Mavzu: Funktsiyalar yaratish

Ishdan maqsad: C++ dasturlash tilining sikl operatorlari bilan tanishish. Sharti oldindan, shart oxiridan beriladigan takrorlanish va parametrik sikl operatorlarini o'rganish orqali funktsiyalarni tashkil etish.

Nazariy qism.

Funksiya va uning turlari

Funksiya – yordamchi qism dastur bo‘lib, maxsus biror-bir jarayonni amalga oshirishga mo‘ljallangan bo‘ladi. Quyida funktsiyani tuzilishi keltirilgan:

Type nomi (parameter 1, parameter 2,) {amallar}

- type – e‘lon qilinayotgan funktsiyani toyifasi;
- nomi – e‘lon qilinayotgan funktsiyani nomi;
- parameter – funktsiyaga qo‘yilgan vazifani amalga oshirishda qatnashuvchi o‘zgaruvchilarni toyifasi bilan birga e‘lon qilishni ta‘minlaydi;
- amallar – bu qism funktsiyaning tanasi bo‘lib, funktsiyaga qo‘yilgan vazifani bajaruvchi amallardan iborat.

Funksiyalar xususiyatlari asosan ikki turga bo‘linadi.

Funksiya 1.1

Derektivalar va asosiy funktsiya **main()** tarkibida ishlatiladi hamda quyidagicha:

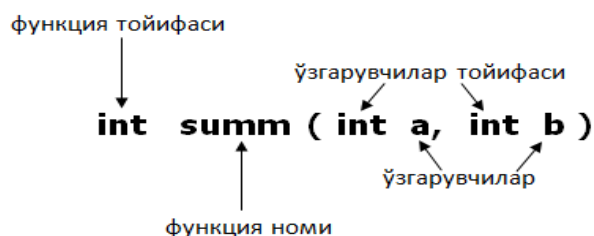
```
#include<iostream>
#include<conio.h>
using namespace std;
    int summ(int a, int b) // funktsiyani e‘lon qilish
{
    // funktsiyani boshlanishi
    ..... // funktsiya tanasi
}; // funktsiyani yopilishi
main ()
{
    .....
    summ(a,b); // funktsiyani chaqirish
    .....
}
```

Quyida funktsiyani shakllantirishning ikki xil varianti keltirilgan:

<i>1-variant</i>	<i>Natija:</i>
-------------------------	-----------------------

<pre> #include<iostream> #include<conio.h> using namespace std; int summ(int a, int b) // funksiyani e'lon qilish { int r; r = a + b; return (r); }; int main() { int s; s = summ(5, 6); // funksiyani chaqirib, ishga tushirish cout<<"Summ = "<<s; // Natijani ekranga chiqarish getch(); return 0; } </pre>	<p>$a = 5$</p> <p>$b = 6$</p> <p>$Summ = 11$</p>
---	---

SHu o'rinda funksiyani e'lon qilishdagi uning tarkibiy qismlariga to'xtalib o'tsak:



YUqorida keltirilgan misoldan ko'rinib turibdiki, *summ* deb ataluvchi funksiyani e'lon qilindi va dasturning asosiy funksiyasi bo'lgan *main* tarkibida ishga tushirilib natijada hisoblashni oshirish orqali qiymatni ekranga chiqardi. Guvohi bo'lganimizdek, *summ* funksiya *int* – toyifasidagi parametrlari ustida qo'shish amalini bajardi. Quyida funksiya o'zi va uni dastur tarkida *s* orqali hisoblashni amalga oshirgan satrlarni keltiramiz:

```

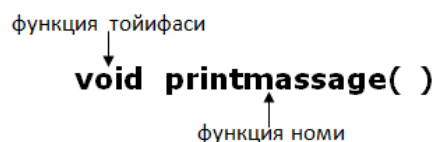
int summ (int a, int b)

s = summ ( 5, 6 )

```

Funksiya 1.2

Bu turdagi funksiyalar toyifasi bo'sh bo'lib, toyifa o'rnida *void* – dan foydalanadi. SHu bilan birga funksiya 1 kabi qiymat qaytarmaydi. Funksiya 1.2 ning vazifasi jarayonni borishini ta'minlaydi. Ushbu funksiya 2 e'lon qilish quyida keltirilgan:



Misol:	Natija:
<pre>#include<iostream> #include<conio.h> using namespace std; void printmessage () { cout<<"Bu void bilan ishlovchi funksiya!"; } int main() { printmessage(); getch(); return 0; }</pre>	<p>Bu void bilan ishlovchi funksiya!</p>

Albatta, dasturni ishlab chiqish mobaynida dasturchi bir nechta funksiyalardan foydalanishimiz mumkin. Yuqorida eng sodda funksiyaning tuzilishini foydalanuvchilar e'tiboriga havola qildik. Keltirilganlardan foydalanib, dasturlar tarkibida foydalanilishi lozim bo'lgan funksiyalarni ishlab chiqish maqsadga muvofiq bo'ladi.

Funksiya 2

Yuqorida funksiyaning umummiy ko'rinishi bilan tanishib chiqdik. Ularda qiymat kiritilib, funksiya qanday amal yuklangan bo'lsa, shu amalni bajaradi. Buni yuqoridagi misollardagi *a* va *b* o'zgaruvchilarga mos ravishda 5 va 6 qiymatlarni berish orqali funksiya o'z funksiyasini bajardi. Biroq *a* va *b* o'zgaruvchilarni boshqa modifikatsiyalarini yuqorida keltirilgan funksiyalar orqali bajarib bo'lmadi. SHuning uchun funksiya ichida o'zgaruvchilar manipulyasiyasi amalga oshirilib, tashqaridan kiritilgan qiymatlar o'zgargan holda ekranga chiqadi. Bunda eng avvalo funksiya nomi e'lon qilinadi, so'ngra har parametrlarini toyifasi bilan birga ampersand (&) belgisi qo'yiladi. &-belgini vazifasi argumentni mos ravishda ko'rsatilgan yoki tavsiya etilgan o'zgaruvchi bilan o'rnini almashtirishdan iborat. O'zgaruvchilarni funksiya nusxasini emas, balki, natijada ularni boshqa modifikatsiyasini beradi. Bu holda dasturni asosiy tarkibida funksiya chaqirilganda foydalanuvchi tomonidan kiritilgan o'zgaruvchilar o'ziga o'xshash kabi argumentni natija sifatida chiqarib beradi. Ushbu jarayon quyidagi misolda keltirilgan.

Misol:	Natija:
<pre>#include<iostream> #include<conio.h> using namespace std;</pre>	<p>X = 2 Y = 6</p>

<pre> void nusxakochirish (int& a, int& b, int& c) { a* = 2; b* = 2; c* = 2; } int main() { int x=1, y=3, z=7; nusxakochirish(x, y, z); cout<<"\n X = "<<x; cout<<"\n Y = "<<y; cout<<"\n Z = "<<z; getch(); return 0; } </pre>	<p>Z = 14</p>
---	---------------

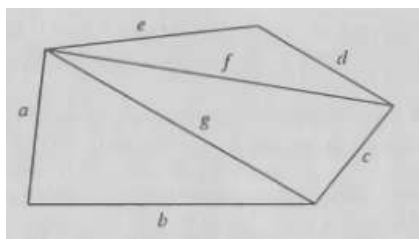
Yana bir shunday misol keltiramiz, bu misol & belgisining vazifasini yanada yaqqolroq ko‘rish mumkin:

Misol:	Natija:
<pre> #include<iostream> #include<conio.h> using namespace std; void prev_next (int x, int& prev, int& next) { prev = x - 1; next = x + 1; } int main() { </pre>	<p>Y = 99</p> <p>X = 100</p> <p>Z = 101</p>

<pre> int x=100, y, z; prev_next (x, y, z); cout<<"\n Y = "<<y; cout<<"\n X = "<<x; cout<<"\n Z = "<<z; getch(); return 0; } </pre>	
--	--

2.2-variant. Funksiyaga doir variantlar

1. Sonni o'nlik sanoq sistemasidan o'n oltilik sanoq sistemasiga (hamda teskarisiga) o'tkazuvchi funksiya tuzing.
2. Sonni o'nlik sanoq sistemasidan sakkizlik oltilik sanoq sistemasiga o'tkazuvchi funksiya tuzing.
3. Sonni o'nlik sanoq sistemasidan ikkilik sanoq sistemasiga o'tkazuvchi funksiya tuzing.
4. Sonni o'n oltilik sanoq sistemasidan ikkilik sanoq sistemasiga o'tkazuvchi funksiya tuzing.
5. Sonni o'n oltilik sanoq sistemasidan sakkizlik sanoq sistemasiga o'tkazuvchi funksiya tuzing.
6. Sonni sakkizlik sanoq sistemasidan ikkilik sanoq sistemasiga o'tkazuvchi funksiya tuzing.
7. Q sonini P darajasini topuvchi funksiya tuzing.
8. P sonini oxiridan L sonini qo'shuvchi funksiya tuzing.
9. Berilgan to'g'ri burchakli uchburchakning katetlari yordamida gipotenuzasini topuvchi funksiya tuzing
10. Berilgan ikki nuqtaning koordinatasi asosida ular orasidagi masofani topuvchi funksiya tuzing.
11. Berilgan sondagi qo'shni raqamlarining raqamlarining yig'indisiga teng bo'lgan raqamni o'chiruvchi funksiya tuzing.
12. Berilgan N soni ikkita tub sonni yig'indisi bo'lishini tekshiruvchi funksiya tuzing.
13. Berilgan son 11 ga bo'linsa 1 aks holda o'chiruvchi funksiya tuzing.
14. Berilgan sonni K o'rindagi raqamini N o'rindagi raqami bilan almashtiruvchi funksiya tuzing.
15. Sonni raqamlarini o'sish (kamayish) tartibida saralovchi funksiya tuzing.
16. Uchburchakni uchta uchining koordinatalari berilgan. Uning yuzasini topuvchi funksiya tuzing.
17. Tomonlari a, b, c, d va e haqiqiy sonlardan iborat bo'lgan beshburchak berilgan (2.1-rasm). Beshburchakni va undagi uchta uchburchaklarni yuzasini topuvchi alohida funksiyalar tuzing.



2.1-rasm.

18. Besh burchakning uchlarining koordinatalari $x_1, y_1, x_2, y_2, \dots, x_5, y_5$ berilgan (2.1-rasm). Beshburchakdagi uchburchakni uchta uchining koordinatalari berilgan. Uning yuzasini topuvchi funksiya tuzing.
19. Sonni tublikka tekshiruvchi funksiya tuzing va barcha uch xonali tub sonlarni chiqaring.
20. Barcha tub sonlar ichidan “egizak”larini topuvchi funksiya tuzing. Tub sonlar “egizak” deyiladi, agarda ular 3 ga farq qilsa. Masalan, 41 va 43. Uch xonali sonlar ichidan barcha “egizak”larni chiqaring.
21. Ikkita son berilgan. Ularni raqamlari yig‘indisi kattasini toping. Sonni raqamlari yig‘indisini topuvchi funksiya tuzing.
22. Ikkita son berilgan. Ularning raqamlaridan soni ko‘pini toping. Sonni raqamlari sonini topuvchi funksiya tuzing.
23. 6 xonali barcha baxtli sonni toping. Agarda 6 xonali sonni dastlabki uchta raqamini yig‘indisi oxirgi uchta raqamini yig‘indisiga teng bo‘lsa baxtli son deyiladi. 6 xonali sonni raqamlarini uchta yig‘indisini topuvchi funksiya tuzing.
24. Berilgan sonni palindromlikka tekshiruvchi funksiya tuzing. Masalan, 1221. Son palindrom bo‘lsa “Palindrom”, aks holda “Palindrom emas” so‘zini chiqaring.
25. Berilgan a va b sonlarini EKUBini topuvchi funksiya tuzing.
26. Berilgan a va b sonlarini EKUKini topuvchi funksiya tuzing.
27. Berilgan a, b va c sonlarini EKUBini topuvchi funksiya tuzing.
28. So‘z berilgan. Shu so‘zni palindromlikka tekshiruvchi funksiya tuzing. Masalan, kiyik. So‘z palindrom bo‘lsa “Palindrom” aks holda “Palindrom emas” so‘zini chiqaring.
29. Berilgan so‘zning harflari sonini topuvchi funksiya tuzing.
30. Berilgan y (yil), o (oy) va k (kun) sonlari berilgan. Shu sonlarni kiritgan holda qaysi kunga to‘g‘ri kelishini aniqlovchi dastur tuzing. Masalan, 2014 2 7 uchun “Juma” chiqishi lozim.

Rekursiv funksiyalar

Funksiya tanasida o‘zini o‘zi chaqirsa **rekursiya** deyiladi. Rekursiya ikki xil bo‘ladi:

- **Oddiy** – agar funksiya o‘z tanasida o‘zini chaqirsa;

- **Vositali** – agar birinchi funksiya ikkinchi funksiyani chaqirsa, ikkinchisi esa o‘z navbatida birinchi funksiyani chaqirsa.

Masalan: Faktorialni hisoblash funksiyasini olamiz. U o‘zini ichida oldingilarini chaqiradi.

Dasturi	Matematik ifodasi
<pre>long Faktorial(int n) { if (!n) return 1; else return n * Faktorial (n - 1); }</pre>	$n! = \begin{cases} 1, & \text{agar } n = 0; \\ n * (n-1)!, & \text{agar } n > 0, \end{cases}$

Xuddi shunday darajani hisoblash funksiyasini ham misol keltirish mumkin.

Dasturi	Matematik ifodasi
<pre>double Daraja(double x, int n) { if (!n) return 1; else return x * Daraja(x, n - 1); }</pre>	$x^n = \begin{cases} 1, & \text{agar } n = 0; \\ x * x^{(n-1)}, & \text{agar } n > 0, \end{cases}$

Namuna. Rekursiv funksiya dan foydalangan holda ikkita sondan raqamlari yig‘indisi katta bo‘lgan sonni topuvchi dastur tuzing.

```
int sum, sum_1, sum_2 ;
int raqam(int son)
{
    sum += son % 10;
    son = son / 10;
    if (son == 0) return sum;
    raqam (son);
}
int main()
{
    int sum_1 = 0, sum_2 = 0;
    int son_1, son_2;
    cin >> son_1 >> son_2;
    sum_1 = raqam(son_1);
    sum_2 = raqam(son_2);
    if (sum_1 > sum_2) cout << son_1; else cout << son_2;
    getch();
    return 0;
}
```

2.3-variant. Rekursiyaga doir variantlar

1. Nyuton Binom koeffitsiyentlarini hisoblovchi rekursiv funksiyali dastur tuzing (M004).

Kiruvchi ma’lumotlar: Nyuton Binom koeffitsiyenti n berilgan ($1 \leq n \leq 100$).

Chiquvchi ma'lumotlar: Nyuton Binom koeffitsiyentlarini probel bilan ajratgan holda tartib bilan chiqaring.

Kiritishga misol	Chiqarishga misol
1	1 1
2	1 2 1
3	1 3 3 1

2. n natural sonini a -darajasini aniqlovchi rekursiv funksiya tuzing.
3. Sonni raqamlar yig'indisini topuvchi rekursiv funksiya tuzing.
4. Sonni raqamlar sonini topuvchi rekursiv funksiya tuzing.
5. Berilgan sonning raqamli ildizini topuvchi rekursiv funksiya tuzing. Sonning raqamlar yig'indisini topamiz va bu yig'indini ham raqamlar yig'indisi ustma-ust tushsa sonni raqamli ildizi deb ataladi.
6. Arifmetik progressiyani birinchi hadi va ayirmasi berilgan. N ta hadini aniqlovchi rekursiv funksiya tuzing.
7. Arifmetik progressiyani birinchi hadi va ayirmasi berilgan. N ta hadini yig'indisini hisoblovchi rekursiv funksiya tuzing.
8. Fibonachi sonini k -hadini topuvchi rekursiv funksiya tuzing.
9. Fibonachi sonini k ta hadini yig'indisini hisoblovchi rekursiv funksiya tuzing.
10. Massivning eng katta elementini topuvchi rekursiv funksiya tuzing.
11. Massivning eng katta elementini indeksini topuvchi rekursiv funksiya tuzing.
12. Manfiy bo'lmagan n va m sonlari uchun Akkerman funksiya'sini hisoblovchi rekursiv funksiya tuzing. Akkerman funksiya'si quyidagicha aniqlanadi:
 $m+1$, agar $n = 0$;
 $A(n, m) = A(n - 1, 1)$, agar $n \neq 0, m = 0$;
 $A(n-1, A(n, m-1))$, agar $n > 0, m > 0$.
13. Berilgan a va b sonlarining EKUBini topuvchi rekursiv funksiya tuzing.
14. Berilgan a va b sonlarining EKUKini topuvchi rekursiv funksiya tuzing.
15. Berilgan a, b va c sonlarining EKUBini topuvchi rekursiv funksiya tuzing.
16. Berilgan sonning raqamlarini teskarisiga yozuvchi rekursiv funksiya tuzing.
17. Soni noma'lum bo'lgan sonlar ketma-ketligini massiv ishlatmagan holda teskarisiga yozuvchi rekursiv funksiya tuzing.
18. N -Fibonachi sonini oxirgi 17 xonasini aniqlovchi rekursiv dastur tuzing.
19. Berilgan S satrning i -elementidan j -elementigacha bo'lgan elementlarini simmetrikligini aniqlovchi rekursiv dastur tuzing.

20. Maxraji n bo'lgan $[0;1]$ orasidagi barcha qisqarmas kasrlarni topuvchi rekursiv dastur tuzing.
21. Berilgan o'nlik natural sonni N ($2 \leq N \leq 16$) sanoq sistemasiga o'tkazuvchi rekursiv funksiya tuzing.
22. Berilgan sonni necha N faktorialga tengligini aniqlovchi rekursiv funksiya tuzing. Masalan, $6 = 3!$.
23. Sonni o'nlik sanoq sistemasidan o'n oltilik sanoq sistemasiga (hamda teskarisi) o'tkazuvchi rekursiv funksiya tuzing.
24. Sonni o'nlik sanoq sistemasidan sakkizlik oltilik sanoq sistemasiga o'tkazuvchi rekursiv funksiya tuzing.
25. Sonni o'nlik sanoq sistemasidan ikkilik sanoq sistemasiga o'tkazuvchi rekursiv funksiya tuzing.
26. Sonni o'n oltilik sanoq sistemasidan ikkilik sanoq sistemasiga o'tkazuvchi rekursiv funksiya tuzing.
27. Sonni o'n oltilik sanoq sistemasidan sakkizlik sanoq sistemasiga o'tkazuvchi rekursiv funksiya tuzing.
28. Berilgan massiv elementlarini saralovchi rekursiv funksiya tuzing. Elementlar soni $[1; 10000]$ bo'lishi mumkin.
29. 0110100110010110 tartibda berilgan ketma – ketlikning n – o'rnida necha soni turganligini topuvchi dastur tuzing. Bu ketma – ketlik birinchi elementi 0 ga teng. Keyingi elementlari esa berilgan satrni 0 ni 1 ga 1 ni esa 0 ga o'g'irgan holatga ko'chirilgan ya'ni 0 1 10 1001 10010110
30. 2^{2^n} ning 10^9 ga bo'lgandagi qoldiqni hisoblovchi rekursiv funksiya tuzing. n soni $[1; 1000000]$ oraliqda bo'lishi mumkin.

Nazorat uchun savollar

1. Ko'p o'lchovli massivlar. Funksiya nima? Rekursiv funksiya nima? Rekursiv funksiya va funksiyaning farqi nima?

Massivlar tasodifiy sonlar bilan qanday to'ldiriladi?

Amaliy ish № 6

Mavzu: Bir o'lchovli massivlar va Ko'p o'lchovli massivlar

Ishdan maqsad. C++ dasturlashda bir va Ko'p o'lchovli massivlar bilan ishlash. Ular ustida amallar bajarishni o'rganish

Nazariy qism.

Massiv – bu bir toifali, chekli qiymatlarning tartiblangan to'plamidir. Massivlarga misol qilib matematika kursidan ma'lum bo'lgan vektorlar, Massivlarni ko'rsatish mumkin.

Massivlar odatda bir o'lchovli va ko'p o'lchovli turlarga bo'linadi.

Massiv bir o'lchamli deyiladi, agar uning elementiga bir indeks orqali murojat qilish mumkin bo'lsa.

C\C++ dasturlash tillaridagi massiv elementlar indeksleri har doim noldan boshlanadi (birdan emas). Bizga char tipidagi m nomli massiv berilgan bo'lsin. Va u 3 ta elementdan tashkil topgan bo'lsin.

$m[0] \rightarrow -9$;

$m[1] \rightarrow 15$;

$m[2] \rightarrow 3$;

Demak, elementga murojat qilish uchun massiv nomi va [] qavslar ichida element indeksi yoziladi. Bu yerda birinchi element qiymati -9, ikkinchi element – 1 nomerli indeksda -15 qiymati bor ekan. Oxirgi element indeksi n-1 bo'ladi (n-massiv elementlari soni). [] qavs ichidagi indeks butun son yoki butun songa olib keluvchi ifoda bo'lmog'i lozim. Masalan:

int n=6, m=4;

L[n-m]=33; // L[2]=33;

Cout<<m[2]; // ekranda : 3;

Massiv elementlariga murojaat qilish oddiy o'zgaruvchilarga murojat qilishdan biroz farq qiladi. Massiv elementiga murojat qilish indeksi orqali bo'ladi.

$a[1] = 5$; a massivning indeksi 1 bo'lgan elementi 5 qiymat o'zlashtirilsin.

cin>>a[2]; a massivning elementi 2 bo'lgan elementi kiritilsin;

cout<<a[3]; a massivning indeksi 3 bo'lgan elementi ekranga chiqarilsin;

Bir o'lchamli massivlarni e'lon quyidagicha bo'ladi :

<Toifa> <massiv_nomi> [elementlar_soni] = { boshlang'ich qiymatlar };

1)float a[5], 2) int b[6], 3) boll c[7];

1) a elementi haqiqiy sondan iborat bo'lgan , 4 ta elementdan tashkil topgan massiv. Indeksleri esa 0 dan 3 gacha bo'lgan sonlar.

Float a[5]					
Massiv elementlari	a [0]	a [1]	a [2]	a [3]	a [4]
qiymati	4	11	-8	12	122

2) b elementi butun sondan iborat bo'lgan , 6 ta elementdan tashkil topgan massiv. Indeksleri esa 0 dan 5 gacha bo'lgan sonlar.

int a[6]						
Massiv elementlari	a [0]	a [1]	a [2]	a [3]	a [4]	a [5]
qiymati	2	99	-5	28	112	54

3) c elementlari mantiqiy qiymatlardan (true, false) iborat bo`lgan 7 ta elementdan tashkil topgan massiv. Indeksleri esa 0 dan 6 gacha bo`lgan sonlardir.

Massivni e`lon qilishda uning elementlariga boshlang`ich qiymat berish mumkin va buning bir necha usuli mavjud.

1) O`lchami ko`ratilgan massivni to`liq initsializatsiyalash.

```
int k[5] = {2, 15 , -9, 45, 3 , 7};
```

Bu yerda 5 ta elementdan iborat k massivi e`lon qilingan va massivning barcha elementlariga boshlang`ich qiymat berilgan.

2) O`lchami ko`rsatilgan massivni to`liqmas to`liqmas initsializatsiyalash.

```
int k[5] = {2, 15, -9 };
```

Bu yerda 5 ta elementdan iborat bo`lgan k massivi e`lon qilingan va dastlabki 3 ta elementlariga boshlang`ich qiymat berilgan.

3) O`lchami ko`rsatilmagan massivni to`liq initsializatsiyalash.

```
int k[] = {2, 15 , -9, 45, 3 , 7};
```

Shuni takidlash lozimki , agar massiv o`lchami ko`rsatilmasa , uni to`liq initsializatsiyalash shart. Bu xolda massiv o`lchami kompilyatsiya jarayonida massiv elementlar soniga qarab aniqlanadi. Bu yerda massiv o`lchami 5 ga teng.

4) O`lchami ko`rsatilgan massivning barcha elementlariga boshlang`ich qiymat 0 berish.

```
int k[5] = {0};
```

Masalan:

1-misol. O`lchami ko`rsatilgan massivning barcha elementlariga boshlang`ich qiymat 0 berish.

```
#include<iostream.h>
```

```
int main ()
```

```
{
```

```
    int k[5]={0}; // massivning barcha elementlariga 0 qiymat berish.
```

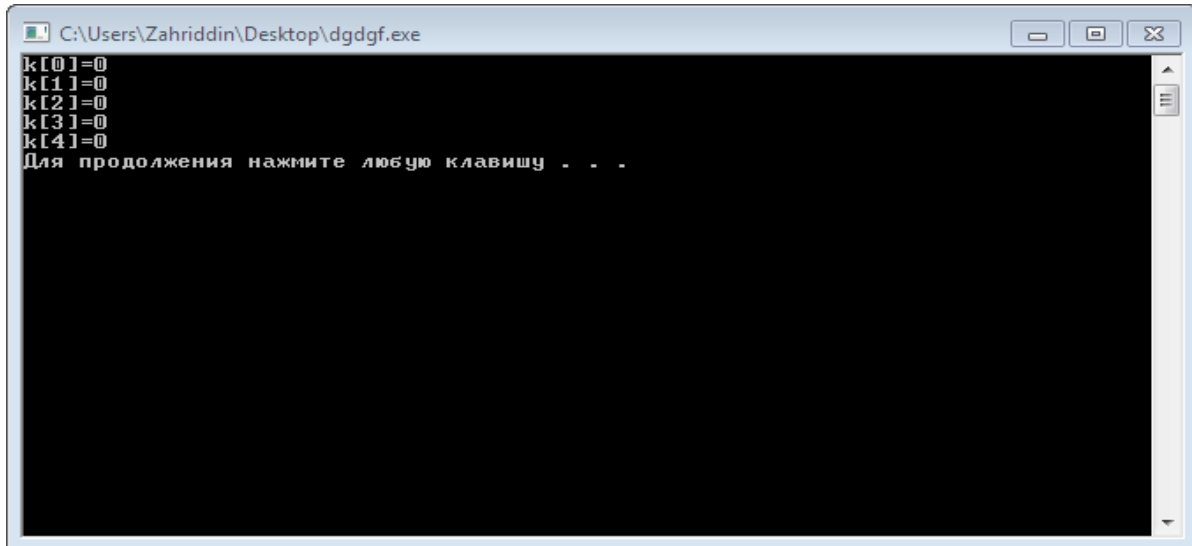
```
    for (int i=0; i<5; i++ )
```

```

    cout<<"k["<<i<<"]= "<<k[i]<<endl;
    return 0;
}

```

Ekranga quyidagicha natija chiqadi:



```

C:\Users\Zahriddin\Desktop\dgdgf.exe
k[0]=0
k[1]=0
k[2]=0
k[3]=0
k[4]=0
Для продолжения нажмите любую клавишу . . .

```

2-misol. O'lchami ko'rsatilgan massivni to'liq initsializatsiyalash.

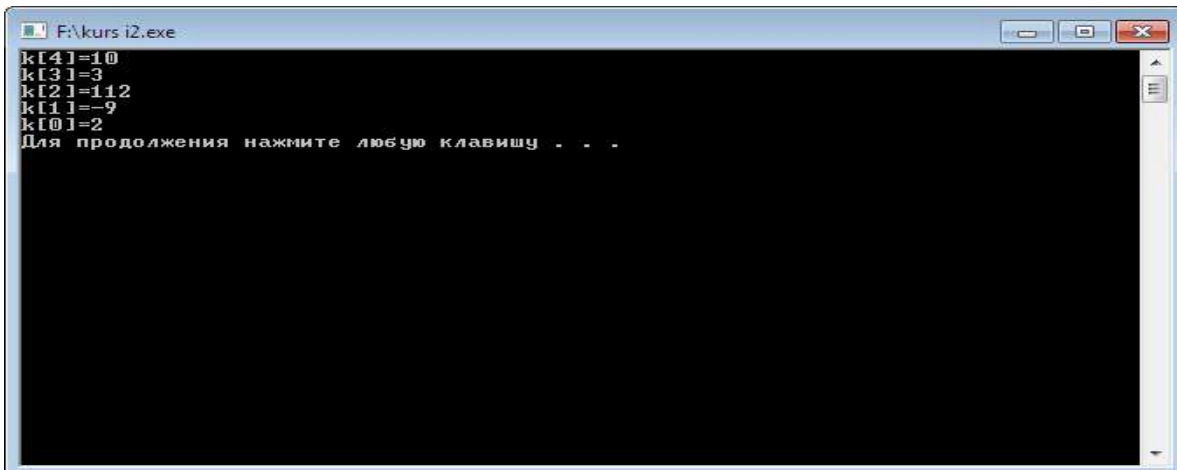
```

#include<iostream.h>

int main ()
{
    int k[5] = { 2, -9, 112, 3, 8 };
    for (int i=4; i>=0; i-- ) // indekslarini teskari tartibda chop etish.
        cout<<"k["<<i<<"]= "<<k[i]<<endl;
    return 0; }

```

Ekranga quyidagicha natija chiqadi:



```

F:\kurs i2.exe
k[4]=10
k[3]=3
k[2]=112
k[1]=-9
k[0]=2
Для продолжения нажмите любую клавишу . . .

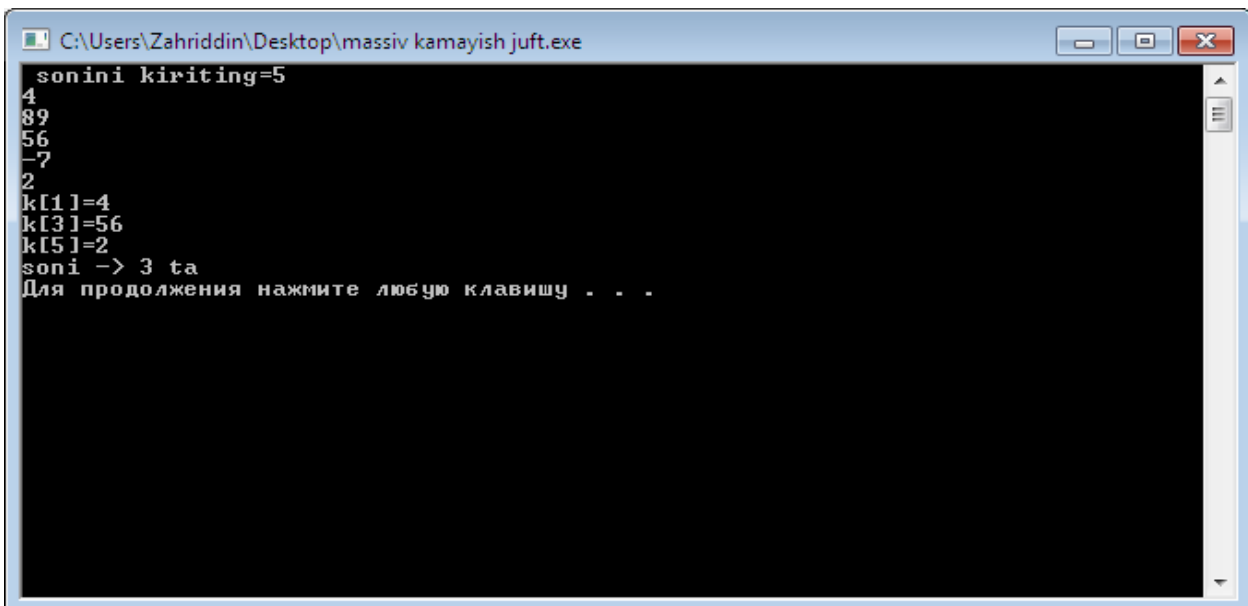
```

3-misol. n oʻlchamli butun sonlardan iborat massiv berilgan . Bu massivning toq elementlarini indekslarini oʻsib borish tartibida chop etish va toq elementlar sonini hisoblash dasturi tuzilsin.

```
#include<iostream>

int main ()
{
    int k[100];
    int i,n,s;
    cout<<" sonini kiriting=";
    cin>>n;
    for ( i=1; i<=n; i++)
        cin>>k[i];
    s=0;
    for (i=1; i<=n; i+=2)
    {
        cout<<"k["<<i<<"]= "<<k[i]<<endl;
        s++;
    }
    cout<<"soni"<<" " "<<"->" "<<" "<<s<<" "<<"ta"<<endl;
    system("pause");
    return 0;
}
```

Ekranga quyidagicha natija chiqadi:



```
C:\Users\Zahriddin\Desktop\massiv kamayish juft.exe
sonini kiriting=5
4
89
56
-7
2
k[1]=4
k[3]=56
k[5]=2
soni -> 3 ta
Для продолжения нажмите любую клавишу . . .
```

Ikki o'lchamli massivda birinchi indeks satrlar sonini , ikkinchisi esa ustunlar sonini bildiradi.

Birinchi satrning dastlabki elementi a_{10} – a biri nol element deb o'qiladi . a o'n deyilmaydi.

M ta satr n ta ustunga ega bo'lgan massivga (mxn)o'lchamli massiv deyiladi. Agar $m=n$ (satrlar va ustunlar soni teng) bo'lsa **kvadrat massiv** deyiladi .

Ikki o'lchamli massivning sintaksi quyidagi ko'rinishda bo'ladi:

<tur><nom>[<uzunlik>][<uzunlik>]

Masalan, 10X20 o'lchamli xaqiqiy sonlar massivning e'loni:

Float a[10][20];

E'lon qilingan a Massiv ko'rinishi quyidagicha ko'rinishda bo'ladi.

J

$a_{[0]}: (a_{[0][0]}, a_{[0][2]}, \dots, \dots a_{[0][18]}, a_{[0][19]},)$

$a_{[1]}: (a_{[1][0]}, a_{[1][1]}, \dots, \dots a_{[1][18]}, a_{[1][19]},)$

....

$i \ a_{[i]}: (\dots, \dots , \dots , \dots a_{[i][j]} \dots , \dots \dots)$

....

$a_{[9]}: (a_{[9][0]}, a_{[9][1]}, \dots, \dots a_{[9][18]}, a_{[9][19]},)$.

Ikki o'lchamli massivning hotirada joylashuvi

Endi adres nuqtayi - nazaridan ko'p o'lchamli massiv elementlariga murojat qilishni ko'raylik.

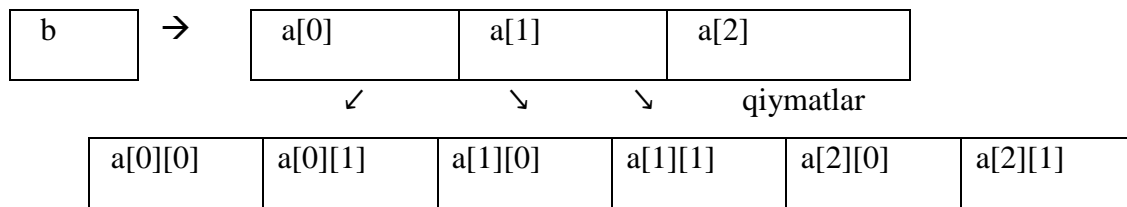
Quyidagi elonlar berilgan bo'lsin:

Int a[3][2];

Float b[2][2][2];

Birinchi elonda ikki o'lchamli massiv, yani 2 ta satr va 3 ustundan iborat Massiv e'lon qilingan , ikkinchisida uch o'lchamli - 3 ta 2x2 Massivdan iborat bo'lgan massiv e'lon qilingan . Uning elementlariga murojat sxemasi:

Adres ko'rsatkichlar massivi

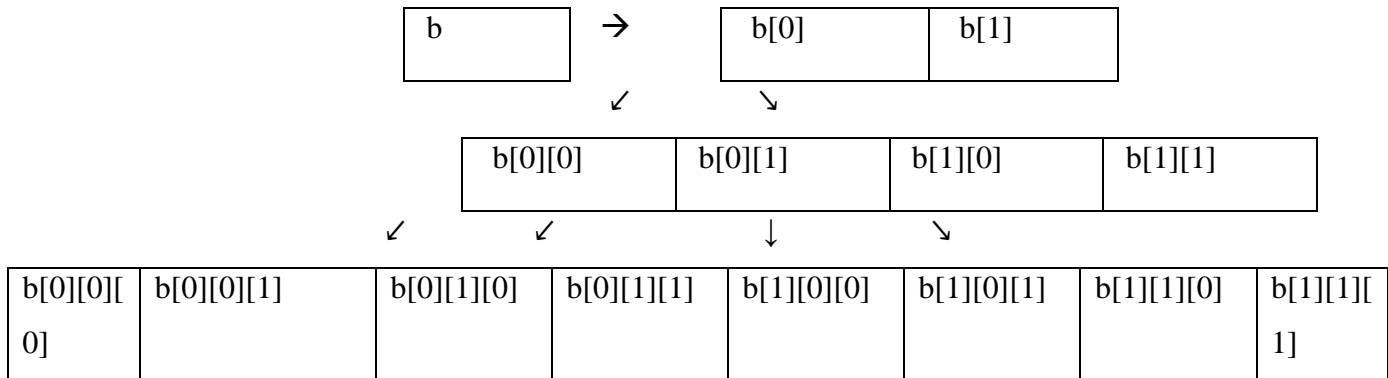


Ikki olchamli massiv elementlariga murojat ;

Bu yerda $a[i]$ ko'rsatkichida i-chi satrning boshlang'ich adresi joylashadi, massiv elementiga $a[i][j]$ ko'rinishidagi asosiy murojatdan tashqari vositali murojat qilish mumkin: $*(a+i)+j$ yoki $*(a[i]+j)$.

Uch o`lchamli massivning xotirada tashkil bo`lishi:

Adres ko`rsatkichlar massivi



Massiv elementlariga murojat qilish uchun nomdan keyin kvadrat qavsda har bir o`lcham uchun indeks yozilishi kerak , masalan `b[i][j][k]`. Bu elementga vositali murojat xam qilish mumkin va uning variantlari:

`*(*(b+i)+j)+k` yoki `*(b[i]+j)+k` yoki `*(b[i][j]+k)`;

Ko`p o`lchovli massivlarni initsializatsiyalash

```
Int a[2][3] = {2, 6, 8, 7, 12, 5};
```

```
Int b[3][3] = { {2, 6, 8}, {7, 12, 5}, {20, 21, 22} }
```

Birinchi operatororda boshlang`ich qiymatlar ketma – ket yozilgan, Ikkinchi operatororda qiymatlar guruhlangan.

Misollar:

1-misol.

M o`lchamli kvadrat matrisa berilgan . Bu massivning elementlarini spiral shaklida chop etish dasturi tuzilsin : avval oxirgi ustun , keyin oxirgi qator teskari tartibda , keyin birinchi ustun teskari tartibda, keyin birinchi qator. Ichki elementlar ham shu tartibda chop etiladi. Eng oxirida matrisaning markaziy elementi chop etiladi.

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    short k,i,j,m,x,y,z,w;
```

```
    float a[100][100];
```

```
    cin>>m;
```

```

for(i=1;i<=m;i++)
for(j=1;j<=m;j++)
    cin>>a[i][j];
x=m; y=m; z=1; w=1;
for(k=1;k<=m/2;k++)
{
for(i=z;i<=x;i++)
    cout<<"a["<<i<<"]["<<x<<"]="<<a[i][x]<<endl;
for(j=y-1;j>=w;j--)
    cout<<"a["<<y<<"]["<<j<<"]="<<a[y][j]<<endl;
for(i=x-1;i>=z;i--)
    cout<<"a["<<i<<"]["<<z<<"]="<<a[i][z]<<endl;
for(j=w+1;j<=y-1;j++)
    cout<<"a["<<w<<"]["<<j<<"]="<<a[w][j]<<endl;
x--;y--;z++;w++;
}
// bu dastur toq sonlar uchun ham o`rinli
if(m%2==1)
    cout<<"a["<<m/2+1<<"]["<<m/2+1<<"]="<<a[m/2+1][m/2+1]<<endl;
    system("pause");
return 0; }

```

Ekranga quyidagicha natija chiqadi:

```

F:\1111111111\N9\2.Avazov_Z_9.1.exe
4
11 12 13 14
15 16 17 18
19 20 21 22
23 24 25 26
a[1][4]=14
a[2][4]=18
a[3][4]=22
a[4][4]=26
a[4][3]=25
a[4][2]=24
a[4][1]=23
a[3][1]=19
a[2][1]=15
a[1][1]=11
a[1][2]=12
a[1][3]=13
a[2][3]=17
a[3][3]=21
a[3][2]=20
a[2][2]=16
Для продолжения нажмите любую клавишу . . .

```

2-misol.

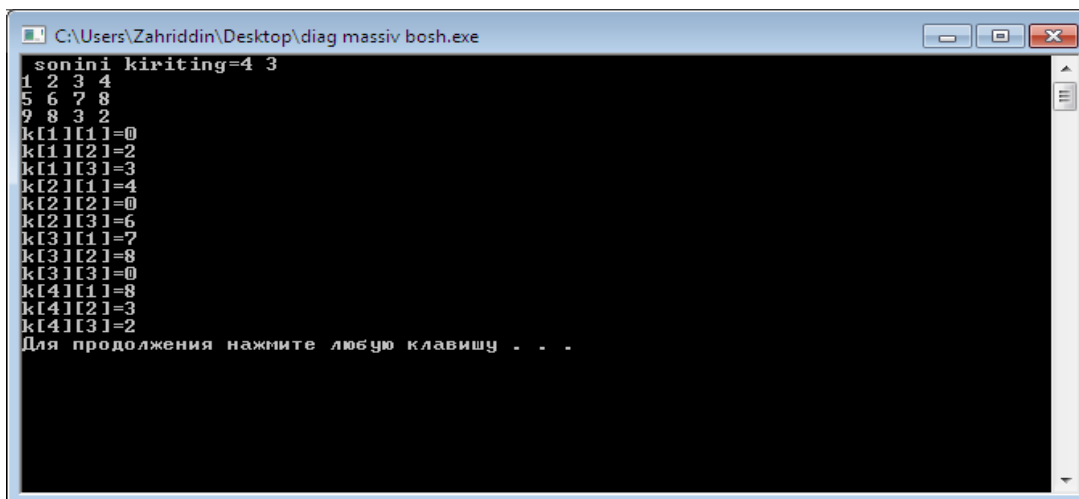
Berilgan $m \times n$ o'lchamli matrisaning bosh diagonal elementlarini nollarga aylantirish dasturi tuzilsin.

```
#include<iostream.h>

int main ()

{
    int k[100][100];
    int i,j,n,m;
    cout<<" sonini kiriting=";
    cin>>n>>m;
    for ( i=1; i<=n; i++)
    for ( j=1; j<=m; j++)
    cin>>k[i][j];
    for ( i=1; i<=n; i++)
    for ( j=1; j<=m; j++)
    {    if(i==j)
        k[i][j]=0;
        cout<<"k["<<i<<"]["<<j<<"]="<<k[i][j]<<endl;    }
    system("pause");
    return 0;    }
```

Ekranga quyidagicha natija chiqadi:



```
C:\Users\Zahridin\Desktop\diag massiv bosh.exe
sonini kiriting=4 3
1 2 3 4
5 6 7 8
9 8 3 2
k[1][1]=0
k[1][2]=2
k[1][3]=3
k[2][1]=4
k[2][2]=0
k[2][3]=6
k[3][1]=7
k[3][2]=8
k[3][3]=0
k[4][1]=8
k[4][2]=3
k[4][3]=2
Для продолжения нажмите любую клавишу . . .
```


Amaliy ish №7

Mavzu: Ko`rsatkich

Ishdan maqsad: C++ dasturlash tilida ko'rsatkichlar bilan ishlash usullarni o'rganish.

Nazariy qism

Программа матнида ўзгарувчи эълон қилинганда, компилятор ўзгарувчига хотирадан жой ажратади. Бошқача айтганда, программа коди хотирага юкланганда берилганлар учун, улар жойлашадиган сегментнинг бошига нисбатан силжишини, яъни нисбий адресини аниқлайди ва объект код ҳосил қилишда ўзгарувчи учраган жойга унинг адресини жойлаштиради.

Умуман олганда, программадаги ўзгармаслар, ўзгарувчилар, функциялар ва синф объектлар адресларини хотиранинг алоҳида жойида сақлаш ва улар устидан амаллар бажариш мумкин. Қиймат-лари адрес бўлган ўзгарувчиларга *кўрсаткич ўзгарувчилар* дейилади.

Кўрсаткич уч хил турда бўлиши мумкин:

- бирорта объектга, хусусан ўзгарувчига кўрсаткич;
- функцияга кўрсаткич;
- void кўрсаткич.

Кўрсаткичнинг бу хусусиятлари унинг қабул қилиши мумкин бўлган қийматларида фарқланади.

Кўрсаткич албатта бирорта турга боғланган бўлиши керак, яъни у кўрсатган адресда қандайдир қиймат жойланиши мумкин ва бу қийматнинг хотирада қанча жой эгаллаши олдиндан маълум бўлиши шарт.

Эълон қилиш:

char *p; // ихтиёрий символ ёки сатрни адреси

int *pI; // бутун сонни адреси

float *pF; // ҳақиқий сонни адреси

Бутун ўзгарувчилар ва массивлар:

int n = 6, A[5] = {0, 1, 2, 3, 4};

int *p; // бутун сонга кўрсаткич

p = &n; // n манзилни ёзиш

*p = 20; // n = 20

p = A + 2; // A[2] (&A[2]) адресни ёзиш

*p = 99; // A[2] ўзгартириш

p++; // A[3] га ўтиш

printf("Adres: %p, qiymat %d", p, *p);

Кўрсаткичлар ва символли сатрлар:

char str[10] = "0123456";

```

char *p;
p = str;
*p = 'A';
p ++;
*p = 'B';
p ++;
strcpy ( p, "CD" );
strcat ( p, "qqq" );
puts ( p );

```

Кўрсаткичга бошланғич қиймат бериш

Кўрсаткичлар кўпинча динамик хотира (бошқача номи «уюм» ёки «heap») билан боғлиқ ҳолда ишлатилади. Хотиранинг динамик дейилишига сабаб, бу соҳадаги бўш хотира программа ишлаш жараёнида, керакли пайтида ажратиб олинади ва зарурат қолмаганида қайтарилади (бўшатилади). Кейинчалик, бу хотира бўлаги программа томонидан бошқа мақсадда яна ишлатилиши мумкин. Динамик хотирага фақат кўрсаткичлар ёрдамида мурожаат қилиш мумкин. Бундай ўзгарувчилар *динамик ўзгарувчилар* дейилади ва уларни яшаш вақти яратилган нуқтадан бошлаб программа охиригача ёки ошкор равишда йўқотилган (боғланган хотира бўшатиладиган) жойгача бўлади.

Кўрсаткичларни эълон қилишда унга бошланғич қийматлар бериш мумкин. Бошланғич қиймат (инициализатор) кўрсаткич номи-дан сўнг ёки қавс ичида ёки ‘=’ белгидан кейин берилади. Бошланғич қийматлар қуйидаги усуллар билан берилиши мумкин:

I. Кўрсаткичга мавжуд бўлган объектнинг адресини бериш:

а) адресни олиш амал орқали:

```

int i=5,k=4; // бутун ўзгарувчилар
int *p=&i;    // p кўрсаткичга i ўзгарувчининг
              // адреси ёзилади
int *p1(&k); // p1 кўрсаткичга k ўзгарувчининг
              // адреси ёзилади

```

б) бошқа, инициализацияланган кўрсаткич қийматини бериш:

```

int * r=p; // p олдин эълон қилинган ва қийматга эга
           // бўлган кўрсаткич

```

в) массив ёки функция номини бериш:

```

int b[10]; // массивни эълон қилиш
int *t=b;  // массивнинг бошланғич адресини бериш

```

```
void f(int a){/* ... */} // функцияни аниқлаш
void (*pf)(int); // функцияга кўрсаткични эълон қилиш
pf=f; // функция адресини кўрсаткичга бериш
```

II. Ошкор равишда хотиранинг абсолют адресини бериш:

```
char *vp = (char *)0xB8000000;
```

Бунда 0xB8000000 - ўн олтилик ўзгармас сон ва (char*) - турга келтириш амали бўлиб, у vp ўзгарувчисини хотиранинг абсолют адресидаги байтларни char сифатида қайта ишловчи кўрсаткич турига айлантирилишини англатади.

III. Бўш қиймат бериш:

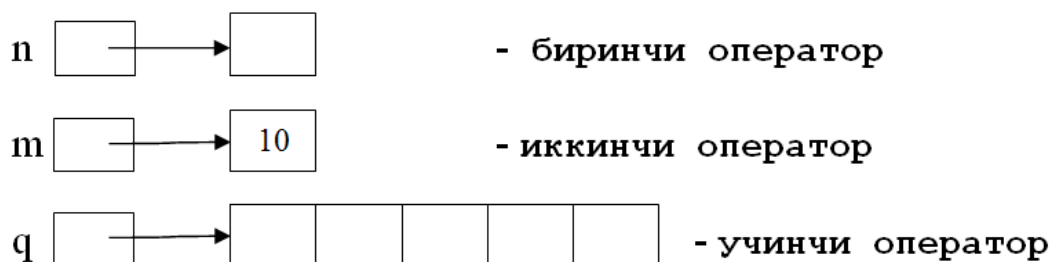
```
int *suxx=NULL;
int *r=0;
```

Биринчи сатрда махсус NULL ўзгармаси ишлатилган, иккинчи сатрда 0 қиймат ишлатилган. Иккала ҳолда ҳам кўрсаткич ҳеч қандай объектга мурожаат қилмайди. Бўш кўрсаткич асосан кўрсаткични аниқ бир объектга кўрсатаётган ёки йўқлигини аниқлаш учун ишла-тилади.

IV. Динамик хотирада new амали билан жой ажратиш ва уни адресини кўрсаткичга бериш:

```
int * n=new int;           // биринчи оператор
int * m=new int(10); // иккинчи оператор
int * q=new int[5];  // учинчи оператор
```

Биринчи операторда new амали ёрдамида динамик хотирада int учун етарли жой ажратиб олиниб, унинг адреси n кўрсаткичга юкланади. Кўрсаткичнинг ўзи учун жой компиляция вақтида ажра-тилади.



6.1-расм. Динамик хотирадан жой ажратиш

Иккинчи операторда жой ажратишдан ташқари m адресига бошланғич қиймат - 10 сонини жойлаштиради.

Учинчи операторда int туридаги 5 элемент учун жой ажра-тилган ва унинг бошланғич адреси q кўрсаткичга берилаяпти.

Хотира new амали билан ажратилган бўлса, у delete амали билан бўшатилиши керак. Юқоридаги динамик ўзгарувчилар билан боғлан-ган хотира қуйидагича бўшатилади:

```
delete n; delete m; delete[]q;
```

Агарда хотира new[] амали билан ажратилган бўлса, уни бўшатиш учун delete [] амалини ўлчови кўрсатилмаган ҳолда қўллаш керак.

Хотира бўшатишга қарамадан кўрсаткични ўзини кейинчалик қайта ишлатиш мумкин.

Кўрсаткич устида амаллар

Кўрсаткич устида қуйидаги амаллар бажарилиши мумкин:

- 1) объектга воситали мурожаат қилиш амали;
- 2) қиймат бериш амали;
- 3) кўрсаткичга ўзгармас қийматни қўшиш амали;
- 4) айириш амали;
- 5) инкремент ва декремент амаллари;
- 6) солиштириш амали;
- 7) турга келтириш амали.

Воситали мурожаат қилиш амали кўрсаткичдаги адрес бўйича жойлашган қийматни олиш ёки қиймат бериш учун ишлатилади:

```
char a; // char туридаги ўзгарувчи эълони.
```

```
char *p=new char; // Кўрсаткични эълон қилиб, унга
```

```
// динамик хотирадан ажратилган
```

```
// хотиранинг адресини бериш
```

```
*p='b'; // p адресига қиймат жойлаштириш
```

```
a=*p; // a ўзгарувчисига p адресидаги қийматни бериш
```

Адресни олиш амали

Адресни олиш қуйидагича эълон қилинади:

<тур> & <ном>;

Бу ерда <тур> - адреси олинган қийматнинг тури, <ном>- адрес олувчи ўзгарувчи номи. Ўртадаги '&' белгисига *адресни олиш амали* дейилади.

Бу кўринишда эълон қилинган ўзгарувчи шу турдаги ўзгарувчининг синоними деб қаралади. Адресни олиш амали орқали битта ўзгарувчига ҳар хил ном билан мурожаат қилиш мумкин бўлади.

Мисол:

```
int kol;
```

```
int & pal=kol; // pal мурожаати, у kol
```

```
// ўзгарувчисининг альтернатив номи
```

```
const char & cr='\n'; // cr - ўзгармасга мурожаат
```

Функцияга кўрсаткич. Функцияга кўрсаткич программа жой-лашган хотирадаги функция кодининг бошланғич адресини кўрса-тади, яъни функция чақирилганда бошқарув айна шу адресга узатила-ди. Кўрсаткич орқали функцияни оддий ёки воситали чақириш амалга ошириш мумкин. Бунда функция унинг номи бўйича эмас, балки функцияга кўрсатувчи ўзгарувчи орқали чақирилади. Функцияни бошқа функцияга аргумент сифатида узатиш ҳам функция кўрсаткичи орқали бажарилади. Функцияга кўрсаткичнинг ёзилиш синтаксиси куйидагича:

`<тур> (* <ном>) (<параметрлар рўйхати>);`

Бунда `<тур>`- функция қайтарувчи қиймат тури; `*<ном>` - кўрсаткич ўзгарувчининг номи; `<параметрлар рўйхати>` - функция параметр-ларининг ёки уларнинг турларининг рўйхати.

Масалан:

`int (*fun)(float,float);`

Бу ерда бутун сон турида қиймат қайтарадиган `fun` номидаги функцияга кўрсаткич эълон қилинган ва у иккита ҳақиқий турдаги параметрларга эга.

Масала. Берилган бутун $n=100$ ва a, b - ҳақиқий сонлар учун $f_1(x) = 5 \sin(3x) + x$, $f_2(x) = \cos(x)$ ва $f_3(x) = x^2 + 1$ функциялар учун $\int_a^b f(x) dx$ интегрални тўғри

тўртбурчаклар формуласи билан тақрибан ҳисоблансин:

$$\int_a^b f(x) dx \approx h[f(x_1) + f(x_2) + \dots + f(x_n)],$$

бу ерда $h = \frac{b-a}{n}$, $x_i = a + ih - h/2, i = 1..n$.

Программа бош функция, интеграл ҳисоблаш ва иккита математик функциялар - $f_1(x)$ ва $f_3(x)$ учун аниқланган функциялардан ташкил топади, $f_2(x) = \cos(x)$ функциянинг адреси «`math.h`» сарлавҳа файлидан олинади. Интеграл ҳисоблаш функциясига кўрсаткич орқали интеграл ҳисобланадиган функция адреси, a ва b - интеграл чегаралари қийматлари узатилади. Оралиқни бўлишлар сони - n глобал ўзгармас қилиб эълон қилинади.

```
#include <iostream.h>
#include <math.h>
const int n=100;
double f1(double x){return 5*sin(3*x)+x;}
double f3(double x){return x*x+1;}
double Integral(double(*f)(double),double a,double b)
{
    double x,s=0;
    double h=(b-a)/n;
    x=a-h/2;
    for(int i=1;i<=n; i++) s+=f(x+=h);
    s*=h;
```

```

return s;
}
int main()
{
double a,b;
int menu;
while(1)
{
cout<<"\nIsh regimini tanlang:\n";
cout<<"1:f1(x)=5*sin(3*x)+x integralini\
hisoblash\n";
cout<<"2:f2(x)=cos(x) integralini hisoblash\n";
cout<<"3:f3(x)=x^2+1 integralini hisoblash\n";
cout<<"0:Programmadan chiqish\n";
do
{
cout<<" Ish regimi-> ";
cin>>menu;
}
while (menu<0 || menu>3);
if(!menu)break;
cout<<"Integral oralig'ining quyi chegarasi a=";
cin>>a;
cout<<"Integral oralig'ining yuqori chegarasi b=";
cin>>b;
cout<<"Funksiya integrali S=";
switch (menu)
{
case 1 : cout<<Integral(f1,a,b)<<endl; break;
case 2 : cout<<Integral(cos,a,b)<<endl; break;
case 3 : cout<<Integral(f3,a,b)<<endl;
}
}
return 0;
}

```

Программанинг иши чексиз такрорлаш оператори танасини бажаришдан иборат.

Такрорлаш танасида фойдаланувчига иш режи-мини танлаш бўйича меню таклиф қилинади:

Ish regimini tanlang:

1: $f_1(x)=5*\sin(3*x)+x$ integralini hisoblash

2: $f_2(x)=\cos(x)$ integralini hisoblash

3: $f_3(x)=x^2+1$ integralini hisoblash

0: Programmadan chiqish

Ish regimi->

Фойдаланувчи 0 ва 3 оралиғидаги бутун сонни киритиши керак. Агар киритилган сон (menu ўзгарувчи қиймати) 0 бўлса, break опера-тори ёрдамида такрорлашдан, кейин программадан чиқилади. Агар menu қиймати 1 ва 3 оралиғида бўлса, интегралнинг қуйи ва юқори чегараларини киритиш сўралади, ҳамда Integral() функцияси мос функция адреси билан чақирилади ва натижа

чоп этилади. Шунга эътибор бериш керакки, интеграл чегараларининг қийматларини тўғри киритилишига фойдаланувчи жавобгар.

Объектга кўрсаткич. Бирор объектга кўрсаткич (шу жумладан ўзгарувчига). Бундай кўрсаткичда маълум турдаги (таянч ёки ҳосила-вий турдаги) берилганларнинг хотирадаги адреси жойлашади. Объектга кўрсаткич қуйидагича эълон қилинади:

`<тур> *<ном>;`

Бу ерда `<тур>` - кўрсаткич аниқлайдиган адресдаги қийматнинг тури, `<ном>` - объект номи (идентификатор). Агар бир турда бир нечта кўрсаткичлар эълон қилинадиган бўлса, ҳар бир кўрсаткич учун `*` белгиси қўйилиши шарт:

`int *i, j,*k;`

`float x,*y,*z;`

Келтирилган мисолда `i` ва `k` - бутун турдаги кўрсаткичлар ва `j` - бутун турдаги ўзгарувчи, иккинчи операторда `x` - ҳақиқий ўзгарувчи ва `y,z` - ҳақиқий турдаги кўрсаткичлар эълон қилинган.

void кўрсаткич. Бу кўрсаткич объект тури олдиндан номаълум бўлганда ишлатилади. `void` кўрсаткичининг муҳим афзалликларидан бири - унга ҳар қандай турдаги кўрсаткич қийматини юклаш мумкин-лигидир. `void` кўрсаткич адресидаги қийматни ишлатишдан олдин, уни аниқ бир турга ошкор равишда келтириш керак бўлади. `void` кўрсаткични эълон қилиш қуйидагича бўлади:

`void *<ном>;`

Кўрсаткичнинг ўзи ўзгармас ёки ўзгарувчан бўлиши ва ўзгармас ёки ўзгарувчилар адресига кўрсатиши мумкин, масалан:

`int i; // бутун ўзгарувчи`

`const int ci=1; // бутун ўзгармас`

`int * pi; // бутун ўзгарувчига кўрсаткич`

`const int *pci; // бутун ўзгармасга кўрсаткич`

`int *const cp=&i; //бутун ўзгарувчига ўзгармас`

`//кўрсаткич`

`const int*const cpc=&ci; // бутун ўзгармасга ўзгармас`

`// кўрсаткич`

Мисоллардан кўриниб турибдики, `*` ва кўрсаткич номи ора-сида турган `const` модификатори фақат кўрсаткичнинг ўзига тегишли ҳисобланади ва уни ўзгартириш мумкин эмаслигини билдиради, `*` белгисидан чапда турган `const` эса кўрсатилган адресдаги қиймат ўзгармас эканлигини билдиради.

Кўрсаткичга қийматни бериш учун `&` - адресни олиш амали ишлатилади.

Кўрсаткич ўзгарувчиларининг амал қилиш соҳаси, яшаш даври ва кўриниш соҳаси умумий қоидаларга бўйсунди.

Назорат саволлари

1. Ko'rsatgich nima?
2. Adres olish amali nimaga kerak?
3. Funksiyaga ko'rsatgichlar qanday ishlatiladi?
4. Funksiya parametrlarida ko'rsatgichni ishlatish nimaga kerak?
5. Havola deb nimaga aytiladi?

Amaliy ish №8

Mavzu: Struktura

Ishdan maqsad: C++ dasturlash tilida strukturada ko'rastgichlarni ishlatish usullarni o'rganish.

Nazariy qism

Struktura nomi ko'cha manzili bu yolg'iz mohiyatga ikki ahamiyatni birlashtirish uchun tariflanishi mumkin. C++da, biz struct odamovi so'zli structurani tariflaymiz:

struct ko'cha manzili

```
{  
    int uy_raqami  
    string ko'cha_ismi;  
}
```

Struktura

Ma'lumki, biror predmet sohasidagi masalani yechishda undagi obyektlar bir nechta, har xil turdagi parametrlar bilan aniqlanishi mumkin. Masalan, tekislikdagi nuqta haqiqiy turdagi x-absissa va y -ordinata juftligi - (x,y) ko'rinishida beriladi. Talaba haqidagi ma'lumotlar – satr turidagi talaba familiyasi, ismi va sharifi, mutaxassislik yo'nalish, talaba yashash adresi, butun turdagi tug'ilgan yili, o'quv bosqichi, haqiqiy turdagi reyting bali, mantiqiy turdagi talaba jinsi haqidagi ma'lumot va boshqalardan shakllanadi.

C++ tilida bir yoki har xil turdagi berilganlarni jamlanmasi struktura deb nomlanadi. Struktura foydalanuvchi tomonidan aniqlangan berilganlarning yangi turi hisoblanadi. Struktura quyidagicha aniqlanadi:

```
struct <struktura nomi> // struktura nomi  
{  
    <1-tur> <1-nom>; // 1-maydonni e'lon qilish  
    <2-tur> <2-nom>; // 2-maydonni e'lon qilish  
    ...  
    <n-tur> <n-nom>; // n-maydonni e'lon qilish  
};
```

Masalan,


```
struct Date
{
int year;
char month, day;
};
```

Masala, Book nomli struktura yarating. Book o‘z ichiga quyidagi maydonlarni olsin:

- Muallif (*satr*);
- Nomi (*satr*);
- Nashr qilingan yili (*butun son*);
- Kitob beti (*butun son*).

Struktura ko‘rinishi quyidagicha bo‘ladi: struct Book {

```
char author[40]; // muallif, satrli
char title[80]; // nomi, satrli
int year;      // nashr qilingan yil, butun son
int pages;     // varaqlar soni, butun son
};
```

Demak biz Book nomli struktura yaratdik. Demak, bu struktura asosida yangi o‘zgaruvchilarni e‘lon qilish mumkin.

Book b; // bu yerda xotira ajratiladi!

Book b1 = {"Yu. Golosinskiy", "Ingliz tili ... ", 2010, 576};

Bunda b va b1 o‘zgaruvchilar e‘lon qilingan. Ularni tiplari esa Book. B ni hali qiymati ma‘lum emas, lekin b1 ning qiymatlari berilgan.

Bundan tashqari struktura maydonlarini quyidagicha ham to‘ldirish mumkin:

Dastur orqali	Klaviatura orqali
strcpy (b.author, "Yu.Golosinskiy"); strcpy (b.title, "Ingliz tili ... "); b.year = 2010; b.pages = 576;	printf ("Muallif"); gets (b.author); printf ("Kitob nomi"); gets (b.title); printf ("Nashr qilingan yili, Betlar soni"); scanf ("%d%d", &b.year, &b.pages);

Demak, berilganlar asosida kutubxonadagi kitoblarni ma‘lumotlarini saqlovchi kichkina dasturcha tuzamiz. Uning dasturi quyidagicha:

```
#include<iostream>
```

```

#include<conio.h>
using namespace std;
struct Book {
    char author[40]; // muallif, satrli
    char title[80]; // nomi, satrli
    int year;        // nashr qilingan yil, butun son
    int pages;       // varaqlar soni, butun son
};
void Kutubxona(Book); // Funksiyani e'lon qilish
int main() // asosiy dastur
{
    Book b;
    printf ("Muallifi "); gets(b.author );
    printf ("Kitob nomi "); gets(b.title );
    printf ("Nashr qilingan yili varaqlar soni");
    scanf ("%d%d", &b.year, &b.pages);
    Kutubxona(b);
}
void Kutubxona(Book b) // Funksiya kodi
{
    cout << "Kitob muallifi: " << b.author << endl;
    cout << "Kitob nomi: " << b.title << endl;
    cout << "Nashr qilingan yili: " << b.year << endl;
    cout << "Betlar soni: " << b.pages ;
    getch();
}

```

Struktura bilan ishlash

1. Ma'lumotlar bazasi uchun predmetli sohani tanlash va ma'lumotlar bazasining ayrim qaydlarini yozish uchun tuzilmalar taklif etish. Tanlangan tuzilma ikki yoki undan ortiq turdagi kamida beshta maydon (unsur)ga ega bo'lishi lozim.

Masalan. ***“Davlat” tuzilmasi.***

Tuzilma element (unsur, maydon, komponent)lari:

- mamlakat nomi;
- poytaxt;

- davlat tili;
- aholi soni;
- yer maydoni.

2. Standartga mos kirish oqimi (klaviatura)dan chiqadigan bir o'lchovli tuzilmalar massivini shakllantirish uchun funksiya yozish. Tuzilmalar kiritilayotganda quyidagi mexanizmlardan birini qo'llash mumkin:

- oldindan berilgan tuzilmalar miqdorini kiritish;
- berilgan belgili tuzilmalar paydo bo'lgunga qadar kiritish;
- kiritish amalini davom ettirish zarurati haqida foydalanuvchi bilan dialog o'rnatish;

3. Tuzilmalar massivi fayliga qayd etish uchun funksiya yozish.

4. Tuzilmalar massiviga fayldan o'qish funksiyasini yozish.

5. Mavjud tuzilmalar massivini yangi tuzilmalar bilan to'ldirish funksiyasini yozish.

6. Tanlangan unsur (element)ning berilgan belgi (znachenije)li tuzilmasini izlab topish funksiyasini qayd etish.

7. Tuzilmalar massivi tarkibini displey ekraniga sahifa ko'rinishida chiqarish funksiyasini yozish.

8. Berilgan belgili tuzilmalarni izlab topish funksiyasini yozish (masalan, unsur qiymatlarining berilgan diapazon bo'yicha tuzilmasini tanlash).

9. Berilgan maydon bo'yicha tuzilmalar massivini tartibga solish funksiyasini yozish. Masalan, davlatlarni aholisi soni bo'yicha yoki mamlakatlarni alfavit bo'yicha tartibga keltirish.

10. Faylni to'liq yangilash funksiyasini yozish, masalan, tuzilmalar massivi faylga tartibga keltirilgandan so'ng qayta yoziladi.

Namoyish qilish:

- dastur tugagandan keyin ma'lumotlarni faylda saqlash (xotiraga);
- tuzilmalar majmuini turlicha tartibga keltirish;
- mos keladigan tuzilmalarni izlab topish (element belgisi, element belgisining diapazoni bo'yicha).

3.1-jadvalda talabalar uchun variantlar berilgan. Har bir variant uchun quyidagi amallar bajarilishi lozim:

- har bir strukturada kamida 10 tadan ma'lumot bo'lsin;
- strukturaning ixtiyoriy maydoni bo'yicha saralash amalga oshirilsin (sitrli va sonli maydonlar uchun);
- struktura maydonlaridan ma'lumotlarni qidirsin (masalan, davlat uchun faqat osiyo qit'asidagi davlatlarni yoki ma'lum bir davlatni);
- struktura ma'lumotlarni fayldan o'qib faylga yozsin;
- strukturani sonli maydonlari uchun ma'lum bir oraliqda saralasin (masalan, davlat strukturasiida aholi soni 20 000 000 dan 50 000 000 gacha bo'lgan davlatlarni chiqaring);

- satrli maydonlar uchun biror bir harf bilan boshlanadigan nomlarini chiqaring (masalan, davlat nomi maydonidan faqat “A” harfi bilan boshlanadigan davlatlarni).

Назорат саволлари

1. Ko'rsatgich nima?
2. Adres olish amali nimaga kerak?
3. Funksiyaga ko'rsatgichlar qanday ishlatiladi?
4. Funksiya parametrlarida ko'rsatgichni ishlatish nimaga kerak?
5. Havola deb nimaga aytiladi?
6. Struktura nima?
7. Strukturaning qanday turlari bor?

Topshiriqlar

Ushbu topshiriqlarni bajarishda ko'rsatgich funksiyalardan foydalanilsin

№	Struktura nomi	Struktura maydonlari
1.	Inson	Familiyasi, ismi, jinsi, millati, e'tiqod qiladigan dini, bo'yi vazni, tug'ilgan sanasi (yil, oy, kun), telefon raqami, uy manzili (pochta indeksi, mamlakat, viloyat, tuman, shahar, ko'cha, uy, xonadon).
2.	Maktab o'quvchisi	Familiyasi, ismi, otasining ismi, sinfi, jinsi, tug'ilgan sanasi (yil, oy, kun), uy manzili, sinfi.
3.	Xaridor	Familiyasi, ismi, otasining ismi, manzili, tuman, shahar, ko'cha, uy raqami, xonadon raqami, kredit kartochkasi raqami yoki hisob raqami
4.	Bemor	Familiyasi, ismi, tug'ilgan yili, telefon raqami, tumani, uy manzili, tibbiy kartochka raqami, qon guruhi.
5.	Sport komandasi	Nomi, shahri, nechta o'yin o'ynaganligi, qancha ochko to'plaganligi (yutqazgan, yutgan, durang), o'yinchilar soni.
6.	Stadion	Nomi, sport turi, qurilgan sanasi, manzili, tomoshabinlar sig'imi, arenalar, maydonchalar soni, sektor raqami.
7.	Avtomobil egasi	Familiyasi, ismi, avtomobil raqami, texpasport raqami, tug'ilgan yili, telefon raqami, qayd qilingan YPX bo'limi
8.	Avtomobil	Markasi, rangi, seriyasi nomeri, qayd raqami, eshiklari soni, chiqqan yili, narxi.

9.	Film	Nomi, rejissori (familiyasi, ismi), mamlakati, ekranga chiqqan yili, narxi (ketgan xarajatlar), olingan daromad.
10.	Musiqiy mahsulot	Tashuvchi (gramplastinka, audiokasseta, lazerli disk), katalogdagi tartib raqami, nomi, ijrochisi (familiyasi, ismi), ijro davomiyligi, asarlar soni, katalog bo'yicha narxi
11.	Vokal-cholg'u guruhi albomi	Guruhning nomi, albomning nomi, kasseta, diskdagi qo'shiqlar soni, albom chiqqan yil, ishlab chiqaruvchi firma
12.	Davlat	Mamlakat nomi, poytaxti, davlat tili, aholisi (soni), yer maydoni, pul birligi valyutasining so'm (dollar)ga nisbatan kursi, davlat tuzumi, qit'asi.
13.	Monitor	Davlat, ishlab chiqarilgan yili, markasi, narxi, kafolat vaqti, o'lchami, rangi, netto va boshqalar
14.	Kitob	Muallifi, ham muallif, asar nomi, beti, yili, sohasi, janri, narxi va boshqalar
15.	Antivirus	Nomi, versiyasi, narxi, kalit vaqti, firmasi va boshqalar
16.	Telefon	Nomi, markasi, yili, o'lchami, narxi, kamera o'lchami, operasion tizim tipi, garantiya, davlati va boshqalar
17.	Qaydnoma	1-joriy nazorat (jn), 1-joriy nazorat qayta ishlash (jnqi), 1-oraliq nazorat (on), 1-oraliq nqi, 2-on, 2-onqi, jamon OT + ON, yakuniy nazorat, o'zlashtirish ko'rsatkichi, reyting bali
18.	O'yinchoq	Nomi, narxi, davlati, bolalar yoshiga chegara va boshqalar
19.	Talaba	Familiyasi, ismi, sharifi, tug'ilgan sanasi (kun oy yil), guruhi, kursi, fakulteti, viloyati, tumani, o'qish shakli, stipendiyasi va boshqalar,
20.	Yugurish	Familiya, ismi, sharifi, davlati, yoshi, masofa, shaxsiy rekordi vaqti vaqti, o'rni
21.	Printer	Nomi, markasi, davlati, ishlab chiqarilgan yili, 1 minutda chiqaradigan varaqlar soni, o'g'irligi, kartrij raqami, toneri, rangi, og'irligi va boshqalar
22.	RAM	Nomi, o'lchami, ishlab chiqarilgan yili, narxi, davlati va boshqalar
23.	Futbolchi	Familiyasi, ismi, sharifi, tug'ilgan sanasi (kun oy yil), fuqaroligi, ampulasi, klubi, gollar soni va boshqalar
24.	Qog'oz	Markasi, o'lchami, narxi, og'irligi, rangi, davlati va boshqalar,
25.	Shkaf	Nomi, turi, rangi, material, o'lchami, narxi, davlati va boshqalar

26.	Daftar	Nomi varaqlar soni, qog'ozi, firmasi, abloshkasi (qattiq, silliq), rangi, o'lchami va boshqalar
27.	Universitet	Nomi, tashkil topgan yili, manzili (viloyati, tumani, ko'chasi, raqami), telefoni, faksi, talabalar soni va boshqalar
28.	Bino	Nomi, manzili, qurilgan yili, qurgan tashkilot, ajratilgan mablag', yer maydoni, bino maydoni, etajlar soni, qurish usuli va boshqalar
29.	Mahalla	Nomi, tumani, shirkati, aholi soni, oqsoqoli, telefoni, ko'p qavatli uylar soni, oilalar soni va boshqalar.
30.	Mashina	Nomi, dvigatel nomi, sig'imi, ot kuchi, rangi, narxi, davlari va boshqalar.

Amaliy ish №10

Mavzu: Matnli fayllar bilan ishlash operatorlari

Ishdan maqsad: C++ dasturlash tilida matnli fayllar bilan ishlovchi maxsus funksiyalarni ishlash usullarni o'rganish.

Nazariy qism

Matn fayllarini o'qish va yozish.

C++ da input/output library oqimlar tushunchasiga bog'langan. input stream bu ma'lumotlar manbai, va output stream ma'lumotlar uchun joy. Ma'lumotlar uchun eng keng tarqalgan manbalar va belgilangan joylar sizning qattiq diskizdagi fayllardir. Faylga kirish uchun siz file stream dan foydalanasz. Fayl stream ning asosan 3 turi mavjud: ifstream (input uchun), ofstream (output uchun), va fstream (ikkala input va output uchun). Qachonki siz bu fayllardan birontasidan foydalanganingizda u <fstream> header ni o'z ichiga oladi. Agarda string variable da saqlangan nomni o'tqazishni xohlasangiz, C++ string ni C string ga o'zgartirish uchun c_str funksiyasidan foydalaning:

```
cout << "Please enter the file name:";
```

```
string filename;
```

```
cin >> filename;
```

```
ifstream in_file;
```

```
in_file.open(filename.c_str());
```

Qachonki programma tugallanganda, siz ochgan barcha stream lar avtomatik tarzda yopiladi. Bundan tashqari, qo'lda yaqin komponenti funksiyasi bilan stream ni yopish mumkin:

```
in_file.close();
```

Agar stream variable dan boshqa fayl bilan ma'lumot almashish uchun yana foydalanishni xohlasangiz Manual yopish aynan zarur.

Fayl dan o'qish

File stream dan ma'lumot o'qish bu to'liq to'g'rilik: Siz shunchaki har doim **cin** dan o'qish uchun foydalangan funksiyalarni ishlatasiz:

```
string name;
```

```
double value;
```

```
in_file >> name >> value;
```

Muvaffaqiyatsizlikka uchragan funksiyalardan ma'lum bo'ladiki input ham qulagan. siz allaqachon cin bilan bu funksiyadan foydalangansiz, konsol kiritish xatolar uchun tekshirish. File stream

larga ham shu tarzda yondashiladi. Qachonki fayldan son o'qishga harakat qilsangiz, va keyingi ma'lumot birligi bu prop-erly formatidagi son emas, so'ngra stream omadsizlikka uchraydi. Ma'lumotni o'qigandan so'ng, qayta ishlash muvaffaqiyati uchun oldin sinash kerak:

```
if (!in_file.fail())
{
    Process input.
}
```

shu bilan bir qatorda, siz bir faktdan foydalanishingiz mumkin >> operator “not failed” sharoitiga aylanadi, ya'ni, input bayonoti va test sinovi imkoniyatini bergan holda:

```
if (in_file >> name >> value)
{
    Process input.
}
```

Faylga yozish

Faylga yozish uchun, siz ofstream yoki fstream variable ni aniqlang va uni oching, keyin output file ga ma'lumot jo'nating, bir xil operatsiyalardan foydalangan holda, qaysiki cout lar bilan:

```
ofstream out_file;
out_file.open("output.txt");
out_file << name << " " << value << endl;
```

Quyidagi kod **process_name** funktsiyasi in-file parametr o'zgaruvchan mos yozuvlar parametr ekanligini unutmag. O'qish va yozish o'zgartiruvchilari stream variable dir. Stream variable monitorlari qancha xususiyatlar o'qilgan yoki yozilgan. har bir o'qish va yozish operatsiyalari ma'lumotlarni o'zgartiradi. Shu sababdan, siz har doim stream parametr o'zgaruvchilar mos yozuvlar parametrlarini aniqlash kerak. To'liq programma pastda keltirilgan. Ko'rib turganingizdek, bir fayl o'qish o'qish klaviatura kiritish kabi osondir. Dastur ishlab chiqarishga qarang. Hayratlanarlisi, atigi 69 o'g'il bolalar ismi va qizlar ismining 153 tasi jami tug'ilishlarining yarmini tashkil etgan. Bu shaxsni ko'rsatuvchi biznes bilan shug'ullanuvchilar uchun yaxshi yangilik. 8-betdagi 10- mashqsizdan bu taqsimlash oxirgi yillarda qanday o'zgarganini o'qishni talab qiladi.

```
ch08/babynames.cpp
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4
5 using namespace std;
6
7 /**
8 Nom ma'lumotlarini o'qing, agar nom jami >= 0 bolsa chop qiling, va umumiyini
o'zgartiring.
9 @param in_file Kiruvchi stream
10 @param total hali qayta ishlanishi lozim umumiy ulush
11 */
12 void process_name(ifstream& in_file, double& total)
13 {
14 string name;
15 int count;
16 double percent;
17 in_file >> name >> count >> percent;
```

```

18
19 if (in_file.fail()) { return; } // Har bir kiritishdan keyin muvaffaqiyatsizlikni
tekshiring
20 if (total > 0) { cout << name << " "; }
21 total = total - percent;
22 }
23
24 int main()
25 {
26 ifstream in_file;
27 in_file.open("babynames.txt");
28 if (in_file.fail()) { return 0; } // Ochgandan so'ng muvaffaqiyatsizlikni
tekshirish
29
30 double boy_total = 50;
31 double girl_total = 50;
32
33 while (boy_total > 0 || girl_total > 0)
34 {
35 int rank;
36 in_file >> rank;
37 if (in_file.fail()) { return 0; }
38
39 cout << rank << " ";
40
41 process_name(in_file, boy_total);
42 process_name(in_file, girl_total);
43 44 cout << endl;
45 }
46
47 return 0;
48 }

```

Natija

```

1 Michael Jessica
2 Christopher Ashley
3 Matthew Emily
4 Joshua Sarah
5 Jacob Samantha
6 Nicholas Amanda
7 Andrew Brittany
8 Daniel Elizabeth
9 Tyler Taylor
10 Joseph Megan
...
68 Dustin Gabrielle
69 Noah Katie
70 Caitlin

```


71 Lindsey
 ...
 150 Hayley
 151 Rebekah
 152 Jocelyn
 153 Cassidy

Назорат саволлари

8. Matnli fayllar bilan ish uchun qanday kutubxona ishlatiladi?
9. Fayldan o'qish funksiyasi nima?
10. Faylga yozish funksiyasi nima?
11. Yozish va o'qish uchun nimaga oqimlardan foydalaniladi?

Topshiriqlar

Matnli fayllarga oid masalalar

№	Masala sharti
1.	<p>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.</p> <p>2. 10 ta raqamdan iborat S satr va shifrlangan matnli fayl berilgan. Text59 masaladagi algoritmi bo'yicha shifrlangan matnni deshifrovchi programma tuzilsin.</p>
2.	<p>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.</p> <p>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.</p>
3.	<p>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".</p> <p>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.</p>
4.	<p>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).</p> <p>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.</p>
5.	<p>1. Satr va matnli fayl berilgan. S satr fayl oxiriga qo'shilsin.</p> <p>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.</p>
6.	<p>1. Ikkita matnli fayl berilgan. Birinchi fayl oxiriga ikkinchi fayl qo'shilsin.</p> <p>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.</p>
7.	<p>1. S satr va matnli fayl berilgan. S satr fayl boshiga qo'shilsin.</p>

	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.
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.
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.
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.
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 matnli fayl va boshlang'ich faylning barcha haqiqiy sonlar qismiga ega bo'lgan haqiqiy sonlar fayli hosil qilinsin.
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.
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.
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 bilan 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.
15.	1. K butun soni va matnli fayl berilgan. Undagi 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.
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.
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.
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, B 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 tarafdin tekislanadi.
19.	<p>1. Matnli fayl berilgan. Fayldagi hamma katta lotin harflari kichik harflarga va aksincha, barcha kichik lotin harflari katta harflarga almashtirilsin.</p> <p>2. N butun soni va A, B 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 tarafdin tekislanadi.</p>
20.	<p>1. Matnli fayl berilgan. Undagi barcha ketma-ket kelgan probellar bitta probelga almashtirilsin.</p> <p>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.</p>
21.	<p>1. Uchtadan ko'p bo'lgan satrga ega bo'lgan matnli fayl berilgan. Shu fayldan oxirgi uchta satr o'chirilsin.</p> <p>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.</p>
22.	<p>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.</p> <p>2. K (> 25) butun soni va chap tarafdin 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 tekislandsin. Satrning oxiridagi probellar o'chirilsin. Formatlangan matn yangi faylda saqlansin.</p>
23.	<p>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.</p> <p>2. K (> 25) butun soni va chap tarafdin 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 tekislandsin.</p>
24.	<p>1. Matnli fayl berilgan. Agar abzas bitta yoki bir nechta bo'sh satrlar bilan ajratilgan bo'lsa, u holda matndagi abzaslar soni aniqlansin.</p> <p>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 tekislandsin (ham chap, ham o'ng tomondan tekislandsin). Matn kengligi 50 ta belgiga teng deb olinsin.</p>
25.	<p>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.</p> <p>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.</p>
26.	<p>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.</p> <p>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.</p>

27.	<p>1. Matnli fayl berilgan. Fayl matnidagi har bir abzasning birinchi satri 5 ta probeldan boshlangan. K butun soni va mantli fayl berilgan. Fayldan K-nomeridagi abzas o'chirilsin. Abzas qizil satr orqali ajratiladi.</p> <p>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).</p>
28.	<p>1. Matnli fayl berilgan. Fayl matnidagi har bir abzasning birinchi satri 5 ta probeldan boshlangan. Abzas <i>qizil satr</i> 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).</p> <p>2. Matnli fayl va lotin harflarining kichik harflaridan C belgi berilgan. Matnli fayl hosil qilinsin va unga boshlang'ich fayldagi barcha shu C 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.</p>
29.	<p>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.</p> <p>2. Matnli fayl va lotin harflarining bosh harflaridan C belgi berilgan. Matnli fayl hosil qilinsin va unga boshlang'ich fayldagi barcha shu C 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.</p>
30.	<p>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.</p> <p>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.</p>

Amaliy ish №11

Mavzu: Binary fayllar bilan ishlash operatorlari

Ishdan maqsad: C++ dasturlash tilida matnli fayllar bilan ishlovchi maxsus funksiyalarni ishlash usullarni o'rganish.

Nazariy qism

Fayllar bilan ishlashning bitli rejimi.

Fayl bilan bitli almashish rejimi gets() va puts() funksiyalari yordamida tashkil etiladi. Bu funksiyalarga esa quyidacha murojaat etiladi:

```
c = gets(fp);
```

```
puts(c,fp);
```

Bu yerda fp-ko'rsatkich

c-int turidagi o'zgaruvchi

Namuna. Klaviaturadan simvol kiritib faylga yozing. Matn oxirini '#' belgisi bilan ko'rsating. Fayl nomi foydalanuvchidan so'raladi. Agar <enter> klavishasi bosilsa faylga CR va LF (qiymatlari 13

va 10) konstantalar yoziladi. Keyinchalik fayldan simvollarini o‘qishda bu konstantalar satrlarni ajratishga imkon beradi.

```
#include <stdio.h>
int main()
{
FILE *fp;
char c;
const char CR = '\015';
const char LF = '\012';
char fname[20];
puts("fayl nomini kiriting:\n");
gets(fname);
if((fp = fopen(fname, "w")) == NULL)
{
perror(fname);
return 1;
}
while ((c = getchar())!= '#')
{
if (c == '\n')
{
putc(CR,fp);
putc(LF,fp);
}
else putc (c,fp);
}
fclose(fp);
}
```

Rasmi Fayldagi Jarayonlar

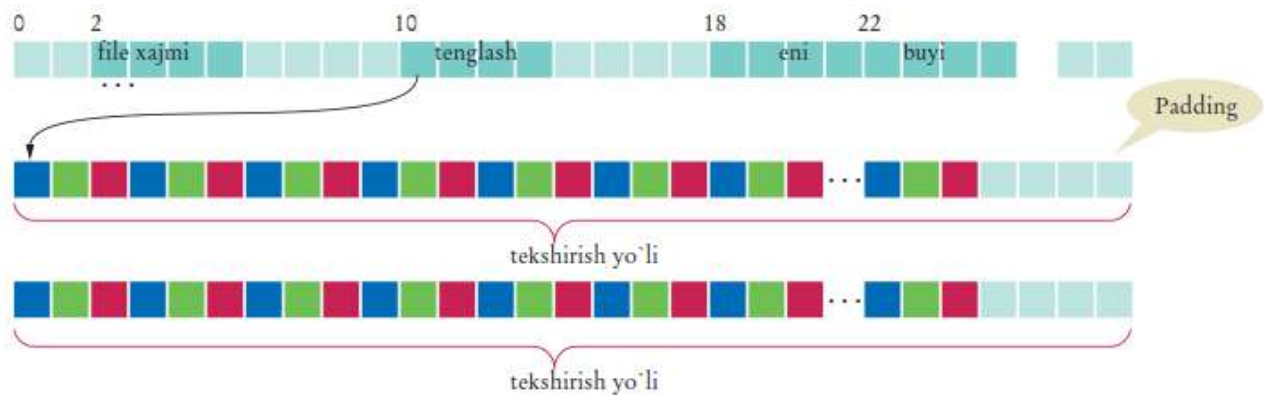
Endi BMP formatli rasmi faylga o'zgartirish kirituvchi programmaga qanday kod yozishni o'rganasiz. BMP odatiy GIF, PNG, va JPEG formatli fayllarga o'xshamagan, chunki u siqilgan ma'lumotlardan foydalanmaydi. Shunga ko'ra BMP fayllar juda katta bo'lib siz uni kamdan kam hollarda ushratasiz.

Biror rasmi faylni BMP formatga qanday o'girsangiz bo'ladi. BMP formatning turli hil versiyalari bor; biz ko'pincha 24-rangli formatli oddiy va sodda turini tanlab olamiz. Bu formatda, har bir piksel uch baytli ketma ketlikni tashkil qiladi , ular ko'k, yashil, va qizil qiymatlar. Masalan, ko'k rang (ko'k va yashil ranglar aralashmasi) bu 255 255 0, qizil esa 0 0 255, va kulrang 128 128 128.

BMP fiayl turli ma'lumot qismlaridan tashkil topgan bosh qismdan boshlanadi.

Bizga faqatgina quyidagilar kerak:

O'rni	Bo'lim
2	Fayl bayt hajmi
10	Rasim boshlanishi
18	Rasim piksellari eniga
22	Rasim piksellari bo'yiga



Rasm quyi qator pikselleridan boshlab pikselni qatorga ketma ketlikda saqlanadi. Har bir piksel qatori ko'k/yashil/qizil uchlikni o'z ichiga oladi. Qator ohiri qo'shimcha baytlar bilan to'ldiriladi shunday qilib qatordagi baytlar soni 4 ga karrali bo'ladi. (5-rasm.) Masalan, agar qator shunday uch pikselni, biri ko'k, biri qizil va kulrangdan iborat bo'lsa qator quyidagicha kodlanadi 255 255 0 0 0 255 128 128 128 x y z bu yerda x y z lar bilan qator to'ldirilib 12 ga tenglanmoqda, bu esa 4 ga karrali bo'ladi. Bu real fayl formatlari bilan ishlashga undovchi qiziqarli tajriba ishiga o'xshaydi. Bo'linma ohiridagi na'muna dasturi BMP fayldagi har bir pikselni o'qiydi hamda ularni qarama qarshi rangi bilan almashtiradi, oqni qoraga, ko'kni qizilga va h.k. Natijada suratga olishda foydalanilgan qadimgi film kameralari olinganidek suratning teskari ranglisi hosil bo'ladi.



```

ch08/imagemod.cpp
1 #include <iostream>
2 #include <fstream>
3 #include <cstdlib>
4
5 using namespace std;
6
7 /**
8
9 @param blue
10 @param green
11 @param red
12 */
13 void process(int& blue, int& green, int& red)
14 {
15 blue = 255 - blue;
16 green = 255 - green;
17 red = 255 - red;
18 }
19
20 /**
21
22 @param stream
23 @param offset
24 @return
25 */
26 int get_int(fstream& stream, int offset)
27 {
28 stream.seekg(offset);
29 int result = 0;
30 int base = 1;
31 for (int i = 0; i < 4; i++)
32 {
33 result = result + stream.get() * base;
34 base = base * 256;
35 }
36 return result;
37 }
38
39 int main()
40 {
41 cout << "Please enter the file name: ";
42 string filename;
43 cin >> filename;
44
45 fstream stream;
46 // Binar faylni ochish
47 stream.open(filename.c_str(), ios::in | ios::out | ios::binary);
48
49 int file_size = get_int(stream, 2); // Rasm hajmini oling
50 int start = get_int(stream, 10);
51 int width = get_int(stream, 18);

```

```

52 int height = get_int(stream, 22);
53
54
55 int scanline_size = width * 3;
56 int padding = 0;
57 if (scanline_size % 4 != 0)
58 {
59     padding = 4 - scanline_size % 4;
60 }
61
62 if (file_size != start + (scanline_size + padding) * height)
63 {
64     cout << "Not a 24-bit true color image file." << endl;
65     return 1;
66 }
67
68 stream.seekg(start); // Piksellar boshiga qaytish
69
70 for (int i = 0; i < height; i++) // har bir linyalar uchun
71 {
72     for (int j = 0; j < width; j++) // har bir piksel uchun
73     {
74         int pos = stream.tellg(); // Piksellar boshiga qaytish
75
76         int blue = stream.get(); // Pikselni o'qish
77         int green = stream.get();
78         int red = stream.get();
79
80         process(blue, green, red); // Pikselni bajarish
81
82         stream.seekp(pos); // Piksellar oxiriga qaytish
83
84         stream.put(blue); // Pikselni yozish
85         stream.put(green);
86         stream.put(red);
87     }
88
89     stream.seekg(padding, ios::cur);
90 }
91
92 return 0;
93 }

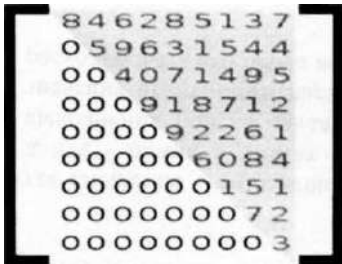
```

Назорат саволлари

1. Binar fayllar bilan ish uchun qanday kutubxona ishlatiladi?
2. Fayldan o'qish funksiyasi nima?
3. Faylga yozish funksiyasi nima?
4. Yozish va o'qish uchun nimaga oqimlardan foydalaniladi?

Topshiriqlar

Binar fayllarga oid masalalar

No	Masala sharti
1.	<p>1. S satr berilgan. Agar S faylning mumkin bolgan nomi bo'lsa, u holda shu nomdagi bo'sh fayl hosil qilinsin va TRUE chop qilinsin. Agar S nomdagi faylni yaratish mumkin bo'lmasa u holda FALSE chop qilinsin.</p> <p>2. Butun sonlar fayli berilgan. Undagi barcha juft nomerdagi elementlari o'chirilsin.</p> <p>3. Tarkibi yuqori uchburchakli Massivdan iborat bo'lgan haqiqiy sonlar fayli berilgan. Tarkibi ushbu berilgan Massivning noldan farqli elementlaridan iborat bo'lgan yangi fayl hosil qilinsin</p> 
2.	<p>1. Fayl nomi N butun soni berilgan ($N > 1$). Berilgan nomdagi fayl hosil qilinsin va unga N ta birinchi musbat juft sonlari chop qilinsin (2,4,...).</p> <p>2. 50 ta elementdan ortiq bo'lgan elementlardan iborat butun sonlar fayli berilgan. Shu fayl elementlarini boshidan boshlab 50ta elementgacha o'chirilsin.</p> <p>3. Tarkibi quyi uchburchakli A va B matritsaning noldan farqli elementlaridan iborat bo'lgan SA va SB nomli haqiqiy sonlar fayli berilgan. Tarkibi A va B matritsaning noldan farqli elementlarini ko'paytmasidan hosil bo'lgan matritsa elementlaridan iborat bo'lgan yangi SC fayl hosil qilinsin. Agar A va B matritsalarini ko'paytirish imkoni bo'lmasa, SC fayl bo'sh holda qoldirilsin.</p>
3.	<p>1. Fayl nomi va A va D haqiqiy sonlar berilgan. Shu nomdagi fayl hosil qilinsin va unga A boshlang'ich hadi va D farqiga ega bo'lgan arifmetik progressiyaning birinchi 10ta hadi yozilsin. A, A+D, A+2*D, A+3*D,...</p> <p>2. Hech bo'lmaganda bitta probel belgisi mavjud bo'lgan belgili fayl berilgan. Shu fayldagi birinchi kelgan probeldan oldin joylashgan barcha elementlar o'chirilsin. (Shu probelni ham hisobga olgan holda)</p> <p>3. Tarkibi yuqori uchburchakli A va B matritsaning noldan farqli elementlaridan iborat bo'lgan SA va SB nomli haqiqiy sonlar fayli berilgan. Tarkibi A va B matritsaning noldan farqli elementlarini ko'paytmasidan hosil bo'lgan matritsa elementlaridan iborat bo'lgan yangi SC fayl hosil qilinsin. Agar A va B matritsalarini ko'paytirish imkoni bo'lmasa, SC fayl bo'sh holda qoldirilsin.</p>
4.	<p>1. 4ta faylning nomi berilgan. Shu fayllarning nechitasi joriy katalogda joylashgani aniqlansin.</p>

	<p>2. Hech bo‘lmaganda bitta probel belgisi mavjud bo‘lgan belgili fayl berilgan. Shu fayldagi oxirgi kelgan probeldan keyin joylashgan barcha elementlar o‘chirilsin. (Shu probelni ham hisobga olgan holda)</p> <p>3. Tarkibi uch dioganalli matritsaning faqat noldan farqli elementlaridan iborat bo‘lgan haqiqiy sonlar fayli berilgan. Yangi fayl hosil qilinsin va u berilgan matritsaning barcha elementlarini saqlasin.</p>
5.	<p>1. Butun sonlar fayli berilgan. Shu fayl tarkibiga kiruvchi elementlar soni aniqlansin. Agar bunday fayl mavjud bo‘lmasa u holda -1 chop qilinsin.</p> <p>2. Hech bo‘lmaganda bitta probel belgisi mavjud bo‘lgan belgili fayl berilgan. Shu fayldagi birinchi kelgan probeldan keyin joylashgan barcha elementlar (probelni ham hisobga olgan holda) o‘chirilsin</p> <p>3. Tarkibi quyi uchburchakli matritsaning faqat noldan farqli elementlaridan iborat bo‘lgan haqiqiy sonlar fayli berilgan. Yangi fayl hosil qilinsin va u berilgan matritsaning barcha elementlarini saqlasin.</p>
6.	<p>1. Manfiy bo‘lmagan butun sonlardan iborat fayl va K soni berilgan (K butun) Faylning K-elementi chop qilinsin (elementlar 1dan boshlab nomerlanadi). Agar bunday element mavjud bo‘lmasa, (-1) chop qilinsin.</p> <p>2. S_1, S_2 satrlar va bir nechta fayllar berilganlarini saqlovchi butun sonlar fayl-arxivi berilgan. Yangi S_1 va S_2 butun sonlar fayli hosil qilinsin va ularning birinchisiga fayl-arxivdagi barcha fayllarning boshlang‘ich elementlari, ikkinchisida esa barcha fayllarning oxirgi elementlari yozilsin (tartibini o‘zgartirmagan holda).</p> <p>3. Tarkibi yuqori uchburchakli matritsaning faqat noldan farqli elementlaridan iborat bo‘lgan haqiqiy sonlar fayli berilgan. Yangi fayl hosil qilinsin va u berilgan matritsaning barcha elementlarini saqlasin.</p>
7.	<p>1. Elementlar soni 4dan ko‘p bo‘lgan butun sonlardan iborat fayl berilgan. Ushbu faylning birinchi, ikkinchi, oxirgi va oxiridan bitta oldingi elementlar chop qilinsin.</p> <p>2. S satr, butun $N (>0)$ soni va S_1, \dots, S_N nomli N ta butun sonlar fayl-arxivi berilgan. Fayl-arxivdan N nomerli fayl tiklansin va S nomi bilan saqlasin. Agar fayl-arxiv N fayldan kam bo‘lgan berilganlarni saqlasa, u holda natijaviy fayl bo‘sh holda qoldirilsin.</p> <p>3. Tarkibi uch dioganalli matritsaning noldan farqli elementlaridan iborat bo‘lgan haqiqiy sonlar fayli hamda I va J butun sonlar berilgan. I-satr va J-ustunda joylashgan matritsa elementi va matritsa tartibi chop qilinsin. (satr va ustunlar 1 dan boshlab nomerlanadi). Agar talab qilingan element matritsaning nol qismida bo‘lsa, u holda 0 qiymati chop qilinsin; agar element mavjud bo‘lmasa, u holda -1 qiymati chop qilinsin.</p>

8.	<p>1. Ikkita haqiqiy sonlar fayli berilgan. Shu fayllarning birinchisi bo'sh bo'lmagani ma'lum va ikkinchisi joriy katalogda mavjud emas. Mavjud bo'lmagan bo'sh fayl yaratilsin va mavjud bo'lgan faylning birinchi va oxirgi elementi shu faylga yozilsin. (tartibi buzilmagan holda).</p> <p>2. S satr, butun $N (>0)$ soni va binar formatdagi bir nechta fayllar (6 tadan ko'p bo'lmagan) berilganlarini saqllovchi butun sonlar fayl-arxivi berilgan. Arxivda saqlanuvchi har bir fayl uchun uning barcha elementlarini o'rta arifmetigi (haqiqiy son) topilsin va topilgan sonlarni (tartibini buzmagani holda) S nomli haqiqiy sonlar fayliga yozilsin.</p> <p>3. Tarkibi quyi uchburchakli matritsaning noldan farqli elementlaridan iborat bo'lgan haqiqiy sonlar fayli hamda I va J butun sonlar berilgan. I-satr va J-ustunda joylashgan matritsa elementi va matritsa tartibi chop qilinsin. (satr va ustunlar 1 dan boshlab nomerlanadi). Agar talab qilingan element matritsaning nol qismida bo'lsa, u holda 0 qiymati chop qilinsin; agar element mavjud bo'lmasa, u holda -1 qiymati chop qilinsin.</p>
9.	<p>1. Ikkita haqiqiy sonlar fayli berilgan. Shu fayllardan biri (birinchisi bo'lishi shart emas) bo'sh bo'lmagani ma'lum va boshqasi esa joriy katalogda mavjud emas. Mavjud bo'lmagan fayl yaratilsin va mavjud bo'lgan faylning birinchi va oxirgi elementi shu faylga yozilsin. (tartibi buzilmagan holda).</p> <p>2. S satr, butun $N(>0)$ soni va S_1, \dots, S_N nomli N ta butun sonlar fayl-arxivi berilgan. Fayl-arxivdan N nomerli fayl tiklansin va S nomi bilan saqlansin. Agar fayl-arxiv N fayldan kam bo'lsa, u holda natijaviy fayl bo'sh holda qoldirilsin.</p> <p>3. Tarkibi uch dioganalli matritsadan iborat bo'lgan haqiqiy sonlar fayli berilgan. Tarkibi ushbu berilgan matritsaning noldan farqli elementlaridan iborat bo'lgan yangi fayl hosil qilinsin.</p>
10.	<p>1. Haqiqiy sonlar fayli berilgan. Berilgan fayl elementlarini teskari tartibda saqllovchi yangi fayl hosil qilinsin.</p> <p>2. SO satri, butun $N(\leq 4)$ soni va S_1, \dots, S_N nomli N ta butun sonlar fayli berilgan. Ularning tarkibini quyidagi formatdan foydalangan holda SO nomli yangi fayl-arxivda birlashtirilsin: fayl -arxivning birinchi elementi sifatida N soni, keyingi N ta elementi esa har bir boshlang'ich fayllarning o'lchami (elementlar soni) va ulardan so'ng ketma - ket har bir boshlang'ich fayllarning berilganlari joylashtiriladi.</p> <p>3. Tarkibi yuqori uchburchakli matritsaning noldan farqli elementlaridan iborat bo'lgan haqiqiy sonlar fayli hamda I va J butun sonlar berilgan. I-satr va J-ustunda joylashgan matritsa elementi va matritsa tartibi chop qilinsin. (satr va ustunlar 1 dan boshlab nomerlanadi). Agar talab qilingan element matritsaning nol qismida bo'lsa, u holda 0 qiymati chop qilinsin; agar element mavjud bo'lmasa, u holda -1 qiymati chop qilinsin.</p>

11.	<p>1. Haqiqiy sonlar fayli berilgan. Ikkita yangi fayl hosil qilinsin. Ularning biriga boshlang'ich faylning toq nomerdagi elementlarini (1, 3,...), ikkinchisi esa - juft nomerdagi elementlarini (2, 4,...) saqlovchi programma tuzilsin.</p> <p>2. Elementlari kamayish tartibida joylashgan S_1, S_2, va S_3 haqiqiy sonlar fayli berilgan. Shu fayllarni yangi S_4 faylga shunday birlashtirilsinki, natijada elementlar yana kamayish tartibida joylashsin.</p> <p>3. Tarkibi quyi uchburchakli matritsadan iborat bo'lgan haqiqiy sonlar fayli berilgan. Tarkibi ushbu berilgan matritsaning noldan farqli elementlaridan iborat bo'lgan yangi fayl hosil qilinsin.</p>
12.	<p>1. Butun sonlar fayli berilgan. Ikkita yangi fayl hosil qilinsin. Ulardan birinchisi boshlang'ich faylning juft sonlarini, ikkinchisi esa toq sonlarini o'zida saqlasin. Agar boshlang'ich faylda juft yoki toq sonlar mavjud bo'lmasa u holda natijaviy fayl bo'sh holda qoldirilsin.</p> <p>2. Haqiqiy sonlar o'sish tartibida S_1 va S_2 fayllariga joylashirilgan. Shu fayllarni yangi S_3 fayliga shunday birlashtirilsinki, natijada elementlar yana o'sish tartibida joylashsin. Ya'ni S_1 faylda (1 2 3) S_2 faylda (4 5 6) joylashgan bo'lishi mumkin. Yoki aksincha S_1 faylda (4 5 6) S_2 faylda (1 2 3) joylashgan. Har ikkala holda ham S_3 fayl elementlari (1 2 3 4 5 6) tartibida bo'lishi kerak.</p> <p>3. Tarkibi A va B to'rtburchak matritsaning elementlarini saqlovchi ikkita SA va SB nomli haqiqiy sonlar fayli berilgan. Bunda har bir faylning birinchi elementi mos matritsalarining ustunlar o'lchamini saqlaydi. Tarkibi A va B matritsalarining ko'paytmasidan hosil bo'lgan matritsa elementlaridan iborat bo'lgan, xudda o'sha strukturadagi SC fayl hosil qilinsin. Agar A va B matritsalarini ko'paytirish imkoni bo'lmasa, u holda SC fayl bo'sh qoldirilsin.</p>
13.	<p>1. Butun sonlar fayli berilgan. Ikkita yangi fayl hosil qilinsin. Ularning birinchisi boshlang'ich faylning musbat sonlaridan (teskari tartibda), ikkinchisi esa manfiy sonlaridan (teskari tartibda) iborat bo'lsin. Agar boshlang'ich faylning manfiy yoki musbat sonlari mavjud bo'lmasa, u holda natijaviy fayl bo'sh holda qoldirilsin.</p> <p>2. Bir xil turdagi va bir xil o'lchamdagi SA, SB, SC, SD butun sonlar fayli va SE satr berilgan. Yangi SE nomli fayl hosil qilinsin va uning elementlari boshlang'ich fayl elementlarining bir xil nomerlilari bilan joylashsin.</p> <p>3. Tarkibi to'rtburchak matritsaning elementlari bo'lgan haqiqiy sonlar fayli berilgan. Bunda faylning birinchi elementi matritsaning ustunlari sonini saqlaydi. Tarkibi boshlang'ich matritsani transponerlanganidan hosil bo'lgan matritsa elementlarini saqlovchi huddi o'sha strukturadagi yangi fayl hosil qilinsin.</p>
14.	<p>1. Haqiqiy sonlar fayli berilgan. Shu fayl elementlarining o'rta arifmetigi topilsin.</p> <p>2. Bir xil turdagi va bir xil o'lchamdagi S_A, S_B, S_C butun sonlar fayli va S_D satr berilgan. Yangi S_D nomli fayl hosil qilinsin va unda elementlar boshlang'ich fayl elementlari bir xil nomerlilari bilan joylashsin:</p>

	<p>A1, B1, C1, A2, B2, C2</p> <p>3. Ikkita I va J butun sonlari hamda tarkibi to'rtburchak matritsaning elementlari bo'lgan haqiqiy sonlar fayli berilgan. Bunda faylning birinchi elementi matritsaning ustunlari sonini saqlaydi. I-satr va J-ustunda joylashgan matritsa elementi chop qilinsin (satr va ustunlar 1 dan boshlab nomerlanadi). Agar talab qilingan element mavjud bo'lmasa, nol chop qilinsin.</p>
15.	<p>1. Haqiqiy sonlar fayli berilgan. Shu faylning juft nomerdagi elementlari yig'indisi hisoblansin.</p> <p>2. Bir xil turdagi ikkita fayl berilgan. Birinchi fayl tarkibiga ikkinchi faylniki, ikkinchi fayl tarkibiga birinchi fayl tarkibi qo'shilsin.</p> <p>3. A va B kvadrat matritsaning elementlarini saqlovchi ikkita SA va SB nomli haqiqiy sonlar fayli berilgan. Tarkibi A va B matritsaning ko'paytmasidan hosil bo'lgan matritsa elementlaridan iborat bo'lgan SC nomli yangi fayl hosil qilinsin. Agar A va B matritsani ko'paytirish mumkin bo'lmasa, u holda SC fayl bo'sh holda qoldirilsin.</p>
16.	<p>1. Butun sonlar fayli berilgan. Shu fayl tarkibiga kiruvchi seriyalar soni topilsin. Seriya deb, ketma-ket kelgan bir xil elementlar guruhiga aytiladi. Masalan, 1, 5, 5, 5, 4, 4, 5 elementga ega bo'lgan fayl uchun natija 4.</p> <p>2. Butun N soni va SO satri berilgan ($N \leq 4$) va bir xil turdagi N ta fayl berilgan. S1 SN fayllar nomlari. Shu fayllar tarkibi yangi SO nomli faylda birlashtirilsin (tartibini buzmaganda).</p> <p>3. Kvadrat matritsa elementlaridan iborat bo'lgan haqiqiy sonlar fayli berilgan. Boshlang'ich matritsaning transponerlanganidan hosil bo'lgan matritsaning elementlaridan iborat bo'lgan yangi fayl hosil qilinsin.</p> $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$
17.	<p>1. Butun sonlar fayli berilgan. Boshlang'ich faylning barcha seriyalari uzunligiga ega bo'lgan yangi butun sonlar fayli hosil qilinsin. Seriya deb, ketma-ket kelgan bir xil elementlar guruhida aytiladi. Seriya uzunligi esa, bu elementlar soni. (seriya uzunligi 1 bo'lishi mumkin). Masalan, 1,5,5,5,4,4,5 elementlariga ega bo'lgan boshlang'ich faylni elementlarini orqali yaratilgan yangi fayl tarkibi 1, 3, 2, 1 bo'ladi.</p> <p>2. Bir xil turdagi lekin o'lchamlari turli bo'lgan uchta fayl berilgan. Shu fayllar orasidan tarkibi eng kalta fayl bilan eng uzun tarkibidagisiga almashtirilsin.</p> <p>3. Ikkita I va J butun sonlari, hamda kvadrat matritsa elementlaridan iborat bo'lgan haqiqiy sonlar fayli berilgan. I - satr va J - ustunda joylashgan matritsa elementi chop qilinsin. (satr va ustunlar 1 dan boshlab nomerlanadi). Agar talab qilingan element mavjud bo'lmasa, u holda nol qiymati chop qilinsin.</p>

18.	<p>1. Haqiqiy sonlar fayli berilgan. Shu faylning birinchi lokal minimumi topilsin. (lokal minimum deb o'z qo'shnilaridan kichik bo'lgan elementga aytiladi).</p> <p>2. Bir xil turdagi lekin o'lchamlari turli bo'lgan uchta fayl berilgan. Shu fayllar orasidan tarkibi eng uzun fayl bilan eng kalta tarkibdasisiga almashtirilsin.</p> <p>3. Tarkibida "kun/oy/yil" formatdagi sanalar bo'lgan satrli fayl beilgan. Yangi satr fayli hosil qilinsin va boshlang'ich fayldagi sanalar kamayish tartibida joylashtirilsin.</p>
19.	<p>1. Haqiqiy sonlar fayli berilgan. Shu faylning birinchi lokal maksimumi topilsin. (lokal maksimum deb o'z qo'shnilaridan katta bo'lgan elementga aytiladi).</p> <p>2. Ixtiyoriy fayl berilgan. Shu fayl nusxasi yangi nom bilan yaratilsin.</p> <p>3. Tarkibida "kun/oy/yil" formatdagi sanalar bo'lgan satrli fayl berilgan. Shu fayldan tarkibida kuzning eng kech sanasi bo'lgan satr topilsin. Agar talab qilingan sana faylda mavjud bo'lmasa, u holda bo'sh satr chop qilinsin.</p>
20.	<p>1. Haqiqiy sonlar fayli berilgan. Shu fayldagi umumiy lokal ekstremumlari soni topilsin. (Ya'ni lokal minimumlar va lokal maksimumlarning umumiy soni. Lokal minimum deb o'z qo'shnilaridan kichik bo'lgan elementga, lokal maximum esa o'z qo'shnilaridan katta bo'lgan elementga aytiladi).</p> <p>2. Ixtiyoriy ikkita fayl berilgan. Fayl ichidagi ma'lumotlar almashtirilsin.</p> <p>3. Tarkibida "kun/oy/yil" formatdagi sanalar bo'lgan satrli fayl berilgan. Shu fayldan tarkibida bahorning eng erta sanasi bo'lgan satr topilsin. Agar talab qilingan sana faylda mavjud bo'lmasa, u holda bo'sh satr chop qilinsin.</p>
21.	<p>1. Haqiqiy sonlar fayli berilgan. Boshlang'ich faylning barcha lokal maksimumlarining nomerlarini o'sish tartibida joylashgan holda saqlovchi yangi butun sonlar fayli hosil qilinsin. (lokal maximum deb o'z qo'shnilaridan katta bo'lgan elementga aytiladi).</p> <p>2. Butun sonlar fayli berilgan. Undagi barcha musbat sonlarni uchta nolga almashtirilsin.</p> <p>3. Tarkibida "kun/oy/yil" formatdagi sanalar bo'lgan satrli fayl berilgan. Tarkibida boshlang'ich fayldagi barcha qishki sanalarni saqlovchi yangi satrli fayl hosil qilinsin (teskari tartibda). Agar qo'yilgan shartdagi sanalar boshlang'ich faylda mavjud bo'lmasa, u holda natijaviy fayl bo'sh holda qoldirilsin.</p>
22.	<p>1. Haqiqiy sonlar fayli berilgan. Boshlang'ich faylning barcha lokal ekstremumlarining nomerlarini kamayish tartibida joylashgan holda saqlovchi yangi butun sonlar fayli hosil qilinsin. (lokal ekstremumi deb lokal minimum va lokal maximumlar soniga aytiladi).</p> <p>2. Butun sonlar fayli berilgan. Undagi juft nomerdagi elemetlar ikkita nolga almashtirilsin.</p> <p>3. Tarkibida "kun/oy/yil" formatdagi sanalar bo'lgan satrli fayl berilgan. Tarkibida boshlang'ich fayldagi barcha yozgi sanalarni saqlovchi yangi satrli fayl hosil qilinsin (tartibini saqlagan holda).</p>

	Agar qo'yilgan shartdagi sanalar boshlang'ich faylda mavjud bo'lmasa, u holda natijaviy fayl bo'sh holda qoldirilsin.
23.	<p>1. Haqiqiy sonlar fayli berilgan. Boshlang'ich faylning kamayib boruvchi elementlar ketma-ketliklari uzunligiga ega bo'lgan yangi butun sonlar fayli hosil qilinsin. Masalan, 1.7, 4.5, 3.4, 2.2 elementlariga ega bo'lgan boshlang'ich fayl uchun natijaviy yaratilgan fayl tarkibi quyidagicha bo'ladi: 3, 2.</p> <p>2. Butun sonlar fayli berilgan. Undagi barcha toq nomerdagilari ikkilantirilsin.</p> <p>3. Tarkibida "kun/oy/yil" formatdagi sanalar bo'lgan satrli fayl berilgan. Ikkita butun sonlar fayli hosil qilinsin va ularning birinchisida boshlang'ich satrli fayldagi oylar, ikkinchisida esa shu oylar uchun yillar qiymatlari saqlansin (tartibni saqlagan holda).</p>
24.	<p>1. Haqiqiy sonlar fayli berilgan. Boshlang'ich fayl elementlarining barcha monoton ketma-ketliklariga ega bo'lgan yangi butun sonlar fayli hosil qilinsin. Masalan, 1.7, 4.5, 3.4, 2.2, 8.5, 1.2 elementlariga ega bo'lgan boshlang'ich fayl uchun yaratilgan natijaviy fayl tarkibi quyidagicha bo'ladi: 2, 3, 2, 2.</p> <p>2. Butun sonlar fayli berilgan. Shu fayl oxiriga boshlang'ich elementlarni yozish orqali fayl o'lchami 2 marta orttirilsin.(teskari tartibda)</p> <p>3. Tarkibida "kun/oy/yil" formatdagi sanalar bo'lgan satrli fayl berilgan. Bunda kun va oy ikkita o'rinni, yil esa 4 ta o'rinni egallaydi (masalan, "05/07/2012"). Ikkita butun sonlar fayli hosil qilinsin va ularning birinchisi boshlang'ich satrli fayldagi kunlar qiymatini, ikkinchisi esa shu kunlar uchun oylar qiymatini saqlasin. (tartibini saqlagan holda).</p>
25.	<p>1. Haqiqiy sonlar fayli berilgan. Undagi barcha elementlarni kvadratlariga almashtirilsin.</p> <p>2. Butun sonlar fayli berilgan. Shu fayl oxiriga boshlang'ich elementlarni yozish orqali fayl o'lchami 2 marta orttirilsin.(tartibini buzmaganda)</p> <p>3. Satrli fayl berilgan. Boshlang'ich fayldagi satrlarni liksografik tartibda, ya'ni satrdagi birinchi belgidan boshlab belgilarning kodlarini o'sish tartibida joylashishidan hosil bo'lgan yangi satrli fayl hosil qilinsin.</p>
26.	<p>1. Haqiqiy sonlar fayli berilgan. Undadi eng kata va eng kichik elementlar o'rinlari almashtirilsin.</p> <p>2. 50ta elementdan kam bo'lgan butun sonlar fayli berilgan. Shu faylning boshiga kerakli nollar yozish orqali elementlar miqdorini 50tagacha ochirilsin.</p> <p>3. Satrli fayl berilgan. Boshlang'ich fayl tarkibidagi barcha eng katta uzunglikka ega bo'lgan satrlarni saqllovchi yangi satrli fayl hosil qilis. (Tartibini saqlagan holda).</p>
27.	<p>1. A_1, A_2, \dots, A_n (n fayldagi elementlar soni) elementlardan iborat butun sonlar fayli berilgan. Shu faylning boshlang'ich joylashishini elementlarning quyidagi joylashishiga almashtirilsin. $A_1, A_n, A_2, A_{n-1}, A_3, \dots$</p> <p>2. Butun sonlar fayli berilgan. Undagi barcha manfiy sonlar o'chirilsin.</p>

	3. Satrli fayl berilgan. Boshlang'ich fayl tarkibidagi barcha eng kichik uzunglikka ega bo'lgan satrlarni saqlovchi yangi satrli fayl hosil qilis. (Tartibini saqlagan holda).
28.	<p>1. Haqiqiy sonlar fayli berilgan. Fayldagi oxirgi va birinchi turgan elementdan boshqa barcha elementlarini o'zidan oldingi va keying turgan elementlarning o'rta arifmetigiga almashtirilsin.</p> <p>2. Butun sonlar fayli berilgan. Undagi barcha juft nomerdagi elementlari o'chirilsin.</p> <p>3. Butun $K (>0)$ soni va satrli fayl berilgan. 2 ta yangi fayl hosil qiling. 1 - fayl satrli, 2 - fayl belgili. Satrli fayl boshlang'ich faylning har bir satridan dastlabki K ta belgini o'zida saqlasin. Belgili fayl esa, faqat K - belgini o'zida saqlasin. Qaralayotgan satr belgilari soni K dan kichik bo'lsa, satrli faylga to'liq yozilsin. Belgili faylga probel yozilsin.</p>
29.	<p>1. 50 ta elementdan ortiq bo'lgan elementlardan iborat butun sonlar fayli berilgan. Shu fayl elementlarini oxiridan boshlab 50 ta elementgacha o'chirilsin.</p> <p>2. Juft miqdordagi elementlarga ega bo'lgan butun sonlar fayli berilgan. Shu faylning birinchi yarmi o'chirib tashlansin.</p> <p>3. Belgili fayl berilgan. Undagi elementlar kodlari o'sish tartibida joylashgan holda tartiblansin.</p>
30.	<p>1. Juft miqdordagi elementlarga ega bo'lgan butun sonlar fayli berilgan. Shu faylning ikkinchi yarmi o'chirib tashlansin.</p> <p>2. 50 ta elementdan ortiq bo'lgan elementlardan iborat butun sonlar fayli berilgan. Shu fayl elementlarini boshidan boshlab 50ta elementgacha o'chirilsin.</p> <p>3. Hech bo'lmaganda bitta probel belgisi mavjud bo'lgan belgili fayl berilgan. Shu fayldagi oxirgi kelgan probeldan oldin joylashgan barcha elementlar o'chirilsin. (Shu probelni ham hisobga olgan holda)</p>

Amaliy ish № 9,12,13

Mavzu: Obektga yo`naltirilgan dasturlash

Ishdan maqsad: C++ dasturlash tilida sinflarni yaratish usullarini o'rganish.

Nazariy qism

Sinflarni eng soda holda qo'yidagicha tasvirlash mumkin: Sinf-kaliti Sinf-soni {komponentalar ruyhati}. Sinf komponentalari sodda holda tiplangan ma'lumotlar va funktsiyalardan iborat bo'ladi. Figurali qavslarga olingan komponentalar ro'yhati sinf tanasi deb ataladi. Sinfga tegishli funktsiyalar komponenta-funktsiyalar yoki sinf funktsiyalari deb ataladi. Sinf kaliti sifatida Struct hizmatchi so'zi ishlatilishi mumkin. Masalan qo'yidagi konstruktsiya kompleks son sinfini kiritadi.

Struct complex 1

{ double real;


```

double imag;
void define (double re=0.0, double im=0.0)
{ real=re; imag=im;}
void display (void)
{cout<<"real="<<real;
cout<<"imag="<<imag;
}
};

```

Strukturadan bu sinfning farqi shuki komponenta ma'lumotlardan (real, imag) tashqari ikkita komponenta funktsiya (define() va display ()) kiritilgan. Bu kiritilgan sinf o'zgaruvchilar tipi deb qaralishi mumkin. Bu tiplar yordamida konkret ob'ektlarni qo'yidagicha tasvirlash mumkin:

Misol uchun:

Complex x,y;

Complex dim[8];

Complex *p=1x;

Sinfga tegishli ob'ektlar qo'yidagicha tasvirlanadi;

Sinf-nomi . ob'ekt-nomi Dasturda ob'ekt komponentasiga quyidagicha murojaat qilish mumkin:

Sinf-nomi.ob'ekt-nomi :: komponenta-nomi yoki soddaroq holda ob'ekt-nomi. Element-nomi

Misol uchun:

x!=real=1.24;

x!=imag=0.0;

dim[3]. Real=0.25;

dim[3]. Imag=0.0;

Sinfga tegishli funktsiyalarga qo'yidagicha murojaat qilinadi: funktsiya-nomi.ob'ekt-nomi;

Misol uchun:

X. define.(Bu holda real=0.9 va imag=0.0)

X. define.(Bu holda kompleks son 4.3+i*20.0)

Display funktsiyasi ekranda kompleks son qiymatlarini tasvirlaydi. Sinfga tegishli ob'ektga ko'rsatkich orqali komponentalarga quyidagicha murojat qilinadi: Ob'ektga-ko'rsatkich>element-nomi Yuqorida ko'rsatilgan P ko'rsatkich orqali H ob'ekt elementlariga qo'yidagicha qiymat berish mumkin:

P>real=2.3

P>imag=6.1

Huddi shu shaklda sinfga tegishli funktsiyalarga murojat qilinadi:

P>display;

P>define(2.3, 5.4);

Kompanenta o'zgaruvchilar va kompanenta funktsiyalar. Sinf kompanenta o'zgaruvchilari sifatida o'zgaruvchilar, massivlar, ko'rsatkichlar ishlatilishi mumkin. Elementlar ta'riflanganda initsializatsiya qilish mumkin emas. Buning sababi shuki sinf uchun hotiradan joy ajratilmaydi. Kompanenta elementlariga kompanenta funktsiyalar orqali murojat qilinganda faqat nomlari ishlatiladi. Sinfdan tashqarida sinf elementlariga emas ob'ekt elementlariga murojaat qilish mumkin. Bu murojaat ikki hil bo'lishi mumkindir.

Ob'ekt- nomi . Element - nomi.

Ob'ktga – korsatgich – element nomi.

Sinf elementlari sinfga tegishli funktsiyalarida ishlatilishidan oldin ta'riflangan bo'lishi shart emas. Huddi shunday bir funktsiyadan hali ta'rifi berilmagan ikkinchi funktsiyaga murojaat qilish mumkin. Komponentalariga murojaat huquqlari. Komponentalariga murojaat huquqi murojaat spetsifikatorlari yordamida boshqariladi.

Bu spetsifikatorlar :

Protected – himoyalangan;

Private – hususiy;

Public – umumiy;

Himoyalangan komponentalardan sinflar ierarhiyasi qurilganda foydalaniladi. Oddiy holda Protected spetsifikatori Private spetsifikatoriga ekvivalentdir. Umumiy ya'ni Public tipidagi komponentalariga dasturning ixtiyoriy joyida murojaat qilinishi mumkin. Hususiy ya'ni Private tipidagi komponentalariga sinf tashqarisidan murojaat qilish mumkin emas. Agar sinflar Struct hizmatchi so'zi bilan kiritilgan bo'lsa, uning hamma komponentalari umumiy Public bo'ladi, lekin bu huquqni murojaat spetsifikatorlari yordamida o'zgartirish mumkin. Agar sinf Class hizmatchi so'zi orqali ta'riflangan bo'lsa, uning hamma komponentalari hususiy bo'ladi. Lekin bu huquqni murojaat spetsifikatorlari yordamida uzgartirish mumkindir. Bu spetsifikator yordamida Sinflar umumiy holda quyidagicha ta'riflanadi:

class class_name

```
{  
  int data_member; // Ma'lumot-element  
  void show_member(int); // Funktsiya-element  
};
```

Sinf ta'riflangandan so'ng, shu sinf tipidagi o'zgaruvchilarni(ob'ektlarni) qo'yidagicha ta'riflash mumkin:

class_name object_one, object_two, object_three;

Qo'yidagi misolda employee, sinfi kiritilgandir:

```
class employee  
{  
  public:
```

```

char name[64] ;
long employee_id;
float salary;
void show_employee(void)
{
cout << "Imya: " << name << endl;
cout << "Nomer slujathego: " << employee_id << endl;
cout << "Oklad: " << salary << endl;
};
};

```

Bu sinf uch o'zgaruvchi va bitta funktsiya-elementga ega. Qo'yidagi EMPCLASS.CPP dastur ikki employee ob'ektini yaratadi. Nuqta operatoridan foydalanib ma'lumot elementlarga qiymat beriladi so'ngra show_employee elementidapn foydalanib hizmatchi haqidagi ma'lumot ekranga chiqariladi:

```

#include <iostream.h>
#include <string.h>
class employee
{
public:
char name [64];
long employee_id;
float salary;
void show_employee(void)
{
cout << "Imya: " << name << endl;
cout << "Nomer slujathego: " << employee_id << endl;
cout << "Oklad: " << salary << endl;
};
};
void main(void)
{
employee worker, boss;
strcpy(worker.name, "John Doe");
worker.employee_id = 12345;
worker.salary = 25000;
strcpy(boss.name, "Happy Jamsa");
boss.employee_id = 101;
boss.salary = 101101.00;
worker.show_employee();
boss.show_employee();
}

```

Konstruktorlar

Konstruktorlar bu sinf komponenta funktsiyalari bulib ,ob'ektlarni avtomatik initsializatsiya qilish uchun ishlatiladi. Konstruktorlar ko'rinishi qo'yidagicha bo'lishi mumkin :

Sinf nomi (formal parametrlar ruyhati) {konstruktor tanasi}. Bu komponenta funktsiya nomi sinf nomi bilan bir hil bulishi lozim. Misol uchun complex sinfi uchun konstruktorni qo'yidagicha kiritish mumkin :

Mplex(double re = 0.0; double im = 0.0)

{real=re; imag=im;}

Tovarlar sinfi uchun konstruktorni qo'yidagicha kiritish mumkin.

Goods(char* new _ name, float new _ price)

{name= new _ name; price= new _ price; }

Konstruktorlarda percent kabi statik elementlarning ham qiymatlarini o'zgartirish mumkindir. Konstruktorlar uchun qaytariluvchi tiplar, hatto void tipi ham ko'rsatilmaydi. Dasturchi tomonidan ko'rsatilmagan holda ham ob'ekt yaratilganda konstruktor avtomatik ravishda chaqiriladi. Masalan ss ob'ekt Copmlex cc; shaklida aniqlangan bo'lsa, konstruktor avtomatik chaqirilib real va imag parametrlari avtomatik ravishda 0.0 qiymatlariga ega bo'ladi. Ko'rsatilmagan holda parametrsiz konstruktor va qo'yidagi tipdagi nusxa olish konstruktorlari yaratiladi: **T :: T (const T&)**

Misol uchun

Class F

{.....

public : F(const T&)

.....

}

Sinfda bir nechta konstruktorlar b'olishi mumkin, lekin ularning faqat bittasida parametrlar qiymatlari oldindan ko'rsatilgan bo'lishi kerak. Konstruktor adresini hisoblash mumkin emas. Konstruktor parametri sifatida uz sinfining nomini ishlatish mumkin emas, lekin bu nomga ko'rsatkichdan foydalanish mumkin. Konstruktorni oddiy komponenta funktsiya sifatida chakirib bo'lmaydi. Konstruktorni ikki hil shaklda chaqirish mumkin : Sinf_nomi ,Ob'ekt_nomi (konstruktor_hakikiy_parametlari) Sinf_nomi (konstruktor_hakikiy_parametlari). Birinchi shakl ishlatilganda haqiqiy parametrlar ro'yhati bo'sh bo'lmasligi lozim. Bu shakldan yangi ob'ekt ta'riflanganda foydalaniladi:

Complex SS(10.3; 0.22)

// real=10.3; SS.imag= 0.22;

Complex EE (2.3)

```
// EE . real= 2.3;
```

```
EE.imag= 0.0;
```

Complex D() // hato

Konstruktorni ikkinchi shaklda chaqirish nomsiz ob'ekt yaratilishiga olib keladi. Bu nomsiz ob'ektdan ifodalarda foydalanish mumkin.

Misol uchun :

Complex ZZ= complex (4.0;5.0);

Bu ta'rif orqali ZZ ob'ekt yaratilib, unga nomsiz ob'ekt qiymatlari(real= 4.0; imag= 5.0) beriladi; Konstruktorlar yordamida ob'ektlar qiymatlarini initsializatsiya qilish uchun initsializatsiya ro'yhatidan foydalanish mumkin: Sinf_nomi (parametrlar ro'yhati);

Komponenta_uzgaruvchilar_initsializatsiya ruyhati {konstruktor tanasi}

Initsializatsiya ruyhatining har bir elementi konkret komponentaga tegishli bo'lib, qo'yidagi ko'rinishga ega:

Komponenta_uzgaruvchi_nomi (ifoda)

Misol:

Class AZ

```
{ int ii ; float ee ; char cc ;
```

```
public:
```

```
AZ (int in ; float en ; char cn) : ii(5),
```

```
EE (ii+en+in) , CC(en) { }
```

```
.....
```

```
};
```

```
AZ A(2,3.0,'d');
```

```
AZ X=AZ (0,2.0,'z');
```

Konstruktor nomi sinf nomi bilan bir hil bo'lishi lozimdir. Misol uchun siz employee sinfdan foydalansangiz, konstruktor ham employee nomga ega bo'ladi. Agar dasturda konstruktor ta'rifi berilgan bo'lsa ob'ekt yaratilganda avtomatik chaqiriladi. Qo'yidagi CONSTRUC.CPP nomli dasturda employee nomli sinf kiritilgandir:

```
class employee
```

```
{
```

```
public:
```

```
employee(char *, long, float); //Konstruktor
```

```
void show_employee(void);
```

```
int change_salary(float);
```

```
long get_id(void);
```

```
private:
```

```
char name [64];
```

```
long employee_id;
```

```
float salary;
```

```
};
```

Konstruktor ta'rifi:

```
employee::employee(char *name, long employee_id, float salary)
{
    strcpy(employee::name, name) ;
    employee::employee_id = employee_id;
    if (salary < 50000.0)
        employee::salary = salary;
    else // Nedopustimihy oklad
        employee::salary = 0.0;
}
```

CONSTRUC.CPP dasturi:

```
#include <iostream.h>
```

```
#include <string.h>
```

```
class employee
```

```
{
```

```
public:
```

```
    employee(char *, long, float);
```

```
    void show_employee(void);
```

```
    int change_salary(float) ;
```

```
    long get_id(void);
```

```
private:
```

```
    char name [64] ;
```

```
    long employee_id;
```

```
    float salary;
```

```
};
```

```
employee::employee(char *name, long employee_id, float salary)
```

```
{
```

```
    strcpy(employee::name, name) ;
```

```
    employee::employee_id = employee_id;
```

```
    if (salary < 50000.0)
```

```
        employee::salary = salary;
```

```
    else // Nedopustimihy oklad
```

```
        employee::salary = 0.0;
```

```
}
```

```
void employee::show_employee(void)
```

```
{
```

```
    cout << "Slujathiy: " << name << endl;
```

```
    cout << "Nomer slujathego: " << employee_id << endl;
```

```
    cout << "Oklad: " << salary << endl;
```

```
}
```

```
void main(void)
```

```
{
```

```
    employee worker("Happy Jamsa", 101, 10101.0);
```

```
worker.show_employee();  
}
```

Konstruktrdan foydalanilganda ob'ekt ta'riflanganda parametr uzatish mumkin:

```
employee worker("Happy Jamsa", 101, 10101.0);
```

Agar dasturda employee tipidagi ob'ektlar mavjud bo'lsa har birini qo'yidagicha initsializatsiya qilish mumkin

```
employee worker("Happy Jamsa", 101, 10101.0);
```

```
employee secretary("John Doe", 57, 20000.0);
```

```
employee manager("Jane Doe", 1022, 30000.0);
```

Konstruktorlarda kuzda tutilgan qiymatlardan ham foydalanish mumkindir. Misol uchun qo'yidagi konstruktor employee oklad qiymatini dasturda ko'rsatilmagan bo'lsa 10000.0 teng qilib oladi.:

```
employee::employee(char *name, long employee_id, float salary = 10000.00)
```

```
{
```

```
strcpy(employee::name, name);
```

```
employee::employee_id = employee_id;
```

```
if (salary < 50000.0)
```

```
employee::salary = salary;
```

```
else // Nedopustimihy oklad
```

```
employee::salary = 0.0;
```

Destruktorlar

Sinfning biror ob'ekti uchun ajratilgan hotira ob'ekt yo'qotilgandan so'ng bo'shatilishi lozimdir. Sinflarning mahsus komponentalari destruktorlar, bu vazifani avtomatik bajarish imkonini yaratadi.

Destruktorni standart shakli qo'yidagicha :

```
~sinf_nomi ( ) {destruktor tanasi}
```

Destruktor parametri yoki qaytariluvchi qiymatga ega bo'lishi mumkin emas. (hatto void tipidagi)

Nazorat savollari

1. Quyidagi sigment kodi nimani chop etadi?

```
CashRegister reg;
```

```
reg.clear();
```

```
reg.add_item(0.95);
reg.add_item(0.95);
cout << reg.get_count() << " " << reg.get_total() << endl;
```

2. Quyidagi kod sigmentining xatosi nimada?

```
CashRegister reg;
reg.clear();
reg.add_item(0.95);
cout << reg.get_amount_due() << endl;
```

3. CashRegister cinfining get_dollars komponentlik funksiyasini e'lon qiling, u savdoning umumiy miqdoridan dollarni ajratib ko'rsatsin.
4. **string** sinfi ikki aksessor komponentlik funksiyasini nomlang.
5. **ifstream** cinfining get komponentlik funksiyasi aksessormi yoki mutatormi?

Topshiriqlar

Nº	Masala sharti
1.	Talaba sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
2.	Avtomashina sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
3.	Mijoz sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
4.	Tovar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
5.	Avia reys sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
6.	Dars sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
7.	Kitob sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
8.	Kompyuter sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
9.	Odam sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
10.	O`quv xonasi nomli sinfni yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
11.	Olimlar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
12.	O`qituvchi sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
13.	Telefon sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
14.	Shahar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
15.	Metro sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.

16.	Nuqta sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
17.	Uchburchak sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
18.	To'rtburchak sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
19.	Doira sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
20.	Aylana sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
21.	Kub sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
22.	Ko'pburchak sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
23.	Matematika sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
24.	Hayvonlar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
25.	Marketlar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
26.	Bekat sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
27.	Geometriya sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
28.	Printerlar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
29.	Modemlar sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.
30.	Kompleks sinfini yarating. Unda kamida 5 ta maydon va ularni ekranga chiqaruvchi, qayta ishlovchi usullarni yarating.

Amaliy ish № 14

Mavzu: Operatorlarni qayta yuklash

Ishdan maqsad: Operatorlarni qayta yuklash ko'nikmalarini egallash.

Nazariy qism

C++ tilida o'rnatilgan operatorlarni qayta yuklash imkoniyati mavjud. Operatorlar global ravishda yoki sinf chegarasida qayta yukla–nishi mumkin. Qayta yuklangan operatorlar operator kalit so'zi yordamida funksiya ko'rinishida amalga oshiriladi. Qayta yuklanuvchi funksiya *operator funksiya* nomlanadi va nomi operatorX ko'rinishida bo'lishi kerak, bu erda X – qayta yuklanuvchi operator. S++ tilida qayta yukla–nishi mumkin bo'lgan operatorlar ro'yxati 13.1-jadvalida keltirilgan. Masalan, qo'shish operatorini qayta yuklash uchun operator+ nomli funksiyani aniqlash kerak bo'ladi. Agar qo'shish qiymat berish amali bilan kelgan holini qayta yuklash uchun operator+= ko'rinishida funksiya aniqlash zarur bo'ladi. Odatda kompilyator programma kodida qayta yuklangan operatorlar uchraganda ularni oshkormas ravishda qo'llaydi. Zarur bo'lganda ularni oshkor chaqirish mumkin:

Nuqta nuqta1, nuqta2, nuqta3;

// Qayta yuklangan qo'shish operatorini oshkor chaqirish

nuqta3=nuqta1.operator+(nuqta2);

13.1–jadval. Qayta yuklanuvchi operatorlar

Operator	Tavsifi	Toifasi
,	Vergul	Binar
!	Mantiqiy inkor	Unar
!=	Teng emas	Binar
%	Bo'lish qoldig'i	Binar
%=	Modulli bo'lish qiymat berish bilan	Binar
&	Razryadli VA	Binar
&	Adresni olish	Unar
&&	Mantiqiy VA	Binar
&=	Razryadli VA qiymat berish bilan	Binar
()	Funksiyani chaqirish	–
*	Ko'paytirish	Binar
*	Vositali murojaat	Binar
*=	Ko'paytirish qiymat berish bilan	Binar
+	Qo'shish	Binar
+	Unar plyus	Unar
++	Inkrement	Unar
+=	Qo'shish qiymat berish bilan	Binar
–	Ayirish	Binar
–	Unar minus	Unar
--	Dekrement	Unar
– =	Ayirish qiymat berish bilan	Binar
->	Elementini tanlash	Binar
->*	Elementini ko'rsatkich orqali tanlash	Binar
/	Bo'lish	Binar
/=	Bo'lish qiymat berish bilan	Binar
<	Kichik	Binar
<=	Kichik yoki teng	Binar
<<	Razryad bo'yicha chapga surish	Binar
<<=	CHapga surish qiymat berish bilan	Binar

=	Qiymat berish	Binar
==	Teng	Binar
>	Katta	Binar
>=	Katta yoki teng	Binar
>>	Razryad bo'yicha o'ngga surish	Binar
>>=	O'ngga surish qiymat berish bilan	Binar
[]	Massiv indeksi	–
^	Razryadli istisno qiluvchi YOKI	Binar
^=	Razryadli istisno qiluvchi YOKI qiymat berish bilan	Binar
	Razryadli YOKI	Binar
=	Razryadli YOKI qiymat berish bilan	Binar
	Mantiqiy YOKI	Binar
~	Bitli mantiqiy INKOR	Binar
delete	Dinamik ob'ektni yo'qotish	–
new	Dinamik ob'ektni yaratish	–

13.2–jadvalda keltirilgan operatorlar qayta yuklanmaydigan operatorlar hisoblanadi.

13.2–jadval. Qayta yuklanmaydigan operatorlar

Operator	Tavsifi
.	A'zoni tanlash
::	Ko'rinish sohasiga ruxsat berish operatori
.*	Ko'rsatkich bo'yicha a'zoni tanlash
?:	SHart amali
#	Preprotssessor belgilari
##	Preprotssessor belgilari

Qayta yuklanadigan operatorlarning operator funksiyalari, new va delete operatorlaridan tashqari, quyidagi qoidalarga bo'ysunishi kerak:

1) operator funksiya sinfning nostatik funksiya–a'zosi bo'lishi kerak yoki operator funksiya sinf yoki sanab o'tiladigan turdagi argument qabul qilishi kerak yoki operator funksiya sinf yoki sanab o'tiladigan turga ko'rsatkich yoki murojaat bo'lgan argumentlarni qabul qilishi kerak.

Masalan,

```
class Nuqta
{
```

public:

//«kichik» operatori uchun operator funksiya-a'zoni

// e'lon qilish

Nuqta operator<(Point&);

...

// Qo'shish operatorlarini e'lon qilish

friend Nuqta operator+(Point&, int);

friend Nuqta operator+(int, Point&);

};

Bu misolda «kichik» operatori sinfnining funksiya–a'zosi sifatida e'lon qilingan, qo'shish operatori esa sinfnining do'sti sifatida e'lon qilingan va u bitta operatorni qayta yuklashning bir nechta varianti bo'lishi mumkinligini ko'rsatadi;

2) operator funksiya operatorning argumentlar (operandlar) sonini, ularning ustunligi va bajarilish tartibini o'zgartira olmaydi;

3) sinf funksiya a'zosi sifatida e'lon qilingan unar operatorning operator funksiyasi parametrغا ega bo'lmasligi kerak. Agar operator funksiya global funksiya bo'lsa, u faqat bitta parametrغا ega bo'ladi;

4) sinf funksiya a'zosi sifatida e'lon qilingan binar operatorning operator funksiyasi bitta parametrغا ega bo'lishi kerak. Agar operator funksiya global funksiya bo'lsa, u faqat ikkita parametrغا ega bo'ladi;

5) operator funksiya kelishuv bo'yicha parametrlarga ega bo'lmasligi kerak;

6) sinf funksiya a'zosi sifatida e'lon qilingan operator funksiyaning birinchi parametri (agar u bo'lsa) sinf turida bo'lishi kerak. Chunki aynan shu sinf ob'ekti uchun mazkur operator chaqiriladi. Birinchi argument ustida hech qanday turga keltirish amali bajaril–masligi kerak;

7) qiymat berish operatorining operator funksiyasidan tashqari barcha operator funksiyalar vorislik bilan o'tadi;

8) =, (), [] va -> operatorlarning operator funksiyalari sinfnining nostatik funksiya a'zolari bo'lishi kerak (va ular global funksiya bo'la olmaydi).

Operatorlarni qayta yuklash orqali, sinf chegarasida operatorning mohiyatini tubdan o'zgartirib yuborish mumkin. Lekin bu ishni zarurat bo'lgandagina amalga oshirgan ma'qul. Aks holda bajariladigan amal–larda mazmuniy xatolar yuzaga kelishi mumkin.

Binar operatorlarni qayta yuklash

Binar operatorning operator funksiyasi sinfnining nostatik funksiya–a'zosi sifatida e'lon qilinganda u quyidagi sintaksisga ega bo'lishi kerak:

<qaytariladigan qiymat turi>operatorX(<parametr turi><parametr>);

Bu erda <qaytariladigan qiymat turi> – funksiya qaytaradigan qiymat turi, X– qayta yuklanadigan operator, <parametr turi> –parametr turi va <parametr> – funksiya parametri.

Funksiya parametriga operatorning o'ng tomonidagi ob'ekt uzatiladi, operatorning chap tomonidagi ob'ekt esa nooshkor ravishda this ko'rsatkichi bilan uzatiladi.

Agar operator funksiya global deb e'lon qilinsa, u quyidagi ko'rinishga ega bo'ladi:

<qaytariladigan qiymat turi>operatorX(<parametr turi₁><parametr₁>,
<parametr turi₂><parametr₂>);

Bu erda funksiya parametrlarining kamida bittasi operator qayta yuklanayotgan sinf turida bo'lishi kerak.

Garchi operator funksiya qaytaradigan qiymat turiga hech qanday cheklov bo'lmasa ham, u sinf turida yoki sinfga ko'rsatkich bo'ladi.

Operator funksiyalarni yozishning bir nechta misollarini keltiramiz. Bu misollar operatorlarni qayta yuklashning to'liq imkoniyatlarini ochib bermasa ham, uning muhim qirralarini ko'rsatadi.

Birinchi navbatda operator funksiyaning sinfnig funksiya–a'zosi ko'rinishida aniqlashni ko'ramiz.

Quyidagi programmada Nuqta sinfi uchun qo'shish va ayirish operatorlarini qayta yuklash amalga oshirilgan.

```
#include <iostream.h>
class Nuqta
{
    int x,y;
public:
    Nuqta(){x=0; y=0;}
    Nuqta(int _x,int _y){x=_x; y=_y;}
    void Nuqta_Qiymati(int & _x,int & _y){_x=x; _y=y;}
    Nuqta operator+(Nuqta& ob);
    Nuqta operator-(Nuqta& ob);
};
Nuqta Nuqta::operator+(Nuqta& ob)
{
    Nuqta OraliqOb;
    OraliqOb.x=x+ob.x;
    OraliqOb.y=y+ob.y;
    return OraliqOb;
}
Nuqta Nuqta::operator-(Nuqta& ob)
{
    Nuqta OraliqOb;
```

```

OraliqOb.x=x-ob.x;
OraliqOb.y=y-ob.y;
return OraliqOb;
}

int main()
{
    int x,y;
    Nuqta A(100,200), B(50,100),C;
    C=A+B; // qayta yuklangan qo'shish operatori amal qiladi
    C.Nuqta_Qiymati(x,y);
    cout<<" C=A+B: "<<"C.x="<<x<<" C.y="<<y<<endl;
    A=A-B; // qayta yuklangan ayirish operatori amal qiladi
    A.Nuqta_Qiymati(x,y);
    cout<<" A=A-B: "<<"A.x="<<x<<" A.y="<<y<<endl;
    return 0;
}

```

Programma ishlashi natijasida ekranga quyidagi ko'rinishidagi natijalar chop etiladi:

C=A+B amali natijasi: C.x=150 C.y=300

A=A-B amali natijasi: A.x=50 A.y=100

Programmada shu narsaga e'tibor berish kerakki, operator funksiya parametri sinf ob'ektga murojaat ko'rinishida aniqlangan. Umuman olganda argument sifatida ob'ektni o'zini ham chaqirish mumkin, lekin funksiyadan chiqishda bu ob'ekt destruktord yordamida yo'qotiladi. Funksiya parametri sinf ob'ektga murojaat ko'rinishida bo'lishining afzalligi shundaki, funksiya chaqirilganda unga ob'ekt emas, balki ob'ektga ko'rsatkich uzatiladi va sinf nusxasi uchun chaqiriladigan destruktorni ishlatilmaydi. Operator funksiyalarning qaytaruvchi qiymati ayni shu sinf turida va hol ob'ektlarni nisbatan murakkab ifodalarda qo'llash imkonini beradi. Masalan, quyidagi amallar programma uchun ruxsat etilgan til ko'rsatmasi hisoblanadi:

C=A+B-C;

Ikkinchi tomondan, quyidagi ifoda ham o'rinli:

(A+B).Nuqta_Qiymati(x,y);

Bu ifodada qo'shish operatoring operator funksiyasidagi vaqtincha (OraliqOb) ob'ektning Nuqta_Qiymati() funksiyasi ishlatiladi.

Keyingi misol operator funksiya parametri sifatida sanab o'tiladigan turdagi berilgan kelgan holatini namoyon qiladi. Bu berilgan operatorning o'ng tomonida kelishiga e'tibor berish kerak.

```

#include <iostream.h>
class Nuqta
{
    int x,y;

```

```

public:
    Nuqta(){x=0; y=0;}
    Nuqta(int _x,int _y){x=_x; y=_y;}
    void Nuqta_Qiymati(int & _x,int & _y){_x=x; _y=y;}
    Nuqta operator+(Nuqta& ob);
    Nuqta operator+(int n);
};
Nuqta Nuqta::operator+(Nuqta& ob)
{
    Nuqta OraliqOb;
    OraliqOb.x=x+ob.x;
    OraliqOb.y=y+ob.y;
    return OraliqOb;
}
Nuqta Nuqta::operator+(int n)
{
    Nuqta OraliqOb;
    OraliqOb.x=x+n;
    OraliqOb.y=y+n;
    return OraliqOb;
}
int main()
{
    int x,y;
    Nuqta A(100,200), B(50,100),C;
    C=A+B; // parametri sinf turidagi ob'ekt bo'lgan
           // qayta yuklangan qo'shish operatori amal qiladi
    C.Nuqta_Qiymati(x,y);
    cout<<" C=A+B: "<<"C.x="<<x<<" C.y="<<y<<endl;
    C=A+30; // parametri sanab o'tiladigan turidagi ob'ekt
    //bo'lgan qayta yuklangan qo'shish operatori amal qiladi
    C.Nuqta_Qiymati(x,y);
    cout<<" C=A+30: "<<"C.x="<<x<<" C.y="<<y<<endl;
    return 0;
}

```

Programma ishlashi natijasida ekranga quyidagi ko'rinishidagi natijalar chop etiladi:

C=A+B amali natijasi: C.x=150 C.y=300

C=A+30 amali natijasi: C.x=130 C.y=230

Operator funksiya parametri operatorning o'ng tomonidagi operand ekanligi sababli kompilyator quyidagi ko'rsatmalarni to'g'ri «tushunadi»:

C=A+30;

Lekin kompilyator

C=30+A;

ko'rsatmasini qabul qilmaydi.

Bu muammoni operator funksiyaning «ichki» imkoniyatlari bilan hal qilib bo'lmaydi. Muammoni do'st operator funksiyalardan foydalanish orqali echish mumkin. Ma'lumki, do'st funksiyalarga yashiringan this ko'rsatkichi uzatilmaydi. SHuning uchun binar operator funksiyasi ikkita argumentga ega bo'lishi kerak – birinchisi chap operand uchun, ikkinchisi o'ng operand uchun.

```
#include <iostream.h>
class Nuqta
{
    int x,y;
public:
    Nuqta(){x=0; y=0;}
    Nuqta(int _x,int _y){x=_x; y=_y;}
    void Nuqta_Qiymati(int & _x,int & _y){_x=x; _y=y;}
    friend class Nuqta operator+(Nuqta& ob1, Nuqta& ob2);
    friend class Nuqta operator+(Nuqta& ob,int n);
    friend class Nuqta operator+(int n, Nuqta& ob);
};
Nuqta operator+(Nuqta& ob1,Nuqta& ob2)
{
    Nuqta OraliqOb;
    OraliqOb.x=ob1.x+ob2.x;
    OraliqOb.y=ob1.y+ob2.y;
    return OraliqOb;
}
Nuqta operator+(Nuqta& ob,int n)
{
    Nuqta OraliqOb;
    OraliqOb.x=ob.x+n;
    OraliqOb.y=ob.y+n;
    return OraliqOb;
}
Nuqta operator+(int n, Nuqta& ob)
{
    Nuqta OraliqOb;
    OraliqOb.x=ob.x+n;
    OraliqOb.y=ob.y+n;
    return OraliqOb;
}
int main()
{
    int x,y;
    Nuqta A(100,200), B(50,100),C;
    C=A+B;
```



```

C.Nuqta_Qiymati(x,y);
cout<<" C=A+B: "<<"C.x="<<x<<" C.y="<<y<<endl;
C=A+30;
C.Nuqta_Qiymati(x,y);
cout<<" C=A+30: "<<"C.x="<<x<<" C.y="<<y<<endl;
C=30+A;
C.Nuqta_Qiymati(x,y);
cout<<" C=30+A: "<<"C.x="<<x<<" C.y="<<y<<endl;
return 0;
}

```

Do'st funksiyalarni qayta yuklash hisobiga

```
C=A+30;
```

```
C=30+A;
```

til ko'rsatmalarini bajarish imkoniyati yuzaga keldi.

Taqqoslash va mantiqiy operatorlarni qayta yuklash

Taqqoslash va mantiqiy operatorlari, garchi binar operatorlar bo'lsa ham ular alohida qaraladi. Chunki ularga mos keluvchi operator-funksiyalar o'zlari aniqlangan sinf turini emas, balki mantiqiy qiymatlarni qaytarishi kerak (yoki true va false sifatida qabul qilinuvchi butun son qiymatini). Misol tariqasida, == va && operatorlarini qayta yuklashni ko'raylik.

```

#include <iostream.h>
class Nuqta
{
    int x,y;
public:
    Nuqta(int _x,int _y){x=_x; y=_y;}
    Nuqta(){x=0; y=0;}
    Qiymat_xy(int & _x, int & _y){_x=x; _y=y;}
    bool operator==(Nuqta ob);
    bool operator&&(Nuqta ob);
};
bool Nuqta::operator==(Nuqta ob)
{
    return (x==ob.x && y==ob.y);
}
bool Nuqta::operator&&(Nuqta ob)
{
    return (x && ob.x) && (y && ob.y);
}
int main()
{
    Nuqta Nuqta1(10,20), Nuqta2(10,25),

```

```

        Nuqta3(10,20),Nuqta4;
    if(Nuqta1==Nuqta2)
        cout<<"Nuqta1 va Nuqta2 o'zaro teng.\n";
    else cout<<"Nuqta1 va Nuqta2 o'zaro teng emas.\n";
    if(Nuqta1==Nuqta3)
        cout<<"Nuqta1 va Nuqta3 o'zaro teng.\n";
    else cout<<"Nuqta1 va Nuqta3 o'zaro teng emas.\n";
    if(Nuqta1 && Nuqta2) cout<<"Nuqta1 && Nuqta2 rost.\n";
    else cout<<"Nuqta1 && Nuqta2 yolg'on.\n";
    if(Nuqta1 && Nuqta4) cout<<"Nuqta1 && Nuqta4 rost.\n";
    else cout<<"Nuqta1 && Nuqta4 yolg'on.\n";
    return 0;
}

```

Programma ishlashi natijasida ekranga quyidagilar chop etiladi:

Nuqta1 va Nuqta2 o'zaro teng emas.

Nuqta1 va Nuqta3 o'zaro teng.

Nuqta1 && Nuqta2 rost.

Nuqta1 && Nuqta4 yolg'on.

Operatorlarni qayta yuklash orqali koordinata nuqtalari orasi–dagi yangi mazmundagi munosabatlar aniqlandi.

Qiymat berish operatorini qayta yuklash

Qiymat berish operatori ham binar operator hisoblanadi, lekin uni qayta yuklash bir qator o'ziga xosliklarga ega:

- qiymat berish operatorining operator funksiyasi global ravishda e'lon qilinishi mumkin emas, ya'ni u faqat sinfnining funksiya–a'zosi bo'lishi kerak;
- qiymat berish operatorining operator funksiyasi hosilaviy sinfga vorislik bilan o'tmaydi;
- kompilyator qiymat berish operatorining operator funksiyasi hosil qilishi mumkin, agar u sinfdan aniqlanmagan bo'lsa.

Kompilyator tomonidan kelishuv bo'yicha hosil qilingan qiymat berish operatori sinfnining har bir statik bo'lmagan a'zolariga qiymat berish amalini bajaradi. Lekin, agar sinfdan ko'rsatkichlar bo'ladigan bo'lsa, bunday qiymat berish operatori ishlamaydi.

Qayd etish kerakki, qiymat berish operatori bajarilgandaen keyin chap tomondagi operand o'zgaradi, chunki unga yangi qiymat beriladi. SHuning uchun qiymat berish operatorining operator funksiyasi uni chaqirilgan ob'ektga ko'rsatkichni qaytarishi shart. Buning uchun funksiya nooshkor ravishda sinf funksiyalariga birinchi parametr sifatida uzatiladigan this ko'rsatkichini qaytarishi etarli. O'z navbatida funksiyaning this ko'rsatkichini qaytarishi quyidagi ko'rinishdagi qiymat berish amallarini «tushunish» imkonini beradi:

Nuqta1=Nuqta2=Nuqta3;

Quyidagi misol qiymat berish operatorini qayta yuklashni namoyon qiladi:

```
#include <iostream.h>
```

```
class Nuqta
```

```
{
```

```
    int x,y;
```

```
    public:
```

```
    Nuqta(int _x,int _y){x=_x; y=_y;}
```

```
    Nuqta(){x=0; y=0;}
```

```
    Qiymat_xy(int & _x,int & _y){_x=x; _y=y;}
```

```
    bool operator==(Nuqta ob);
```

```
    Nuqta & operator=(Nuqta & ob);
```

```
};
```

```
bool Nuqta::operator==(Nuqta ob)
```

```
{
```

```
    return (x==ob.x && y==ob.y);
```

```
}
```

```
Nuqta & Nuqta::operator=(Nuqta & ob)
```

```
{
```

```
    if (this==&ob) return *this;
```

```
    x=ob.x;
```

```
    y=ob.y;
```

```
    return *this;
```

```
}
```

```
int main()
```

```
{
```

```
    int a,b;
```

```
    Nuqta Nuqta1(10,20), Nuqta2(20,25), Nuqta3;
```

```
    Nuqta3=Nuqta2;
```

```
    if(Nuqta2==Nuqta3)
```

```
        cout<<"Nuqta2 va Nuqta3 o'zaro teng.\n";
```

```
    else cout<<"Nuqta2 va Nuqta3 o'zaro teng emas.\n";
```

```
    Nuqta3=Nuqta2=Nuqta1;
```

```
    if(Nuqta1==Nuqta3)
```

```
        cout<<"Nuqta1 va Nuqta3 o'zaro teng.\n";
```

```

else cout<<"Nuqta1 va Nuqta3 o'zaro teng emas.\n";
Nuqta3.Qiymat_xy(a,b);
cout<<"Nuqta3.x="<a<<" Nuqta3.y="<b<<endl;
return 0;
}

```

Programma ishlashi natijasida ekranga

Nuqta2 va Nuqta3 o'zaro teng.

Nuqta1 va Nuqta3 o'zaro teng.

Nuqta3.x=10 Nuqta3.y=20

xabarlari chop etiladi.

Qiymat berish amalini qayta yuklashda mazmunan xatoga olib keladigan

Nuqta1=Nuqta1;

ko‘rinishdagi o‘zini o‘ziga yuklash holati alohida nazorat qilinishi kerak bo‘ladi. Chunki qiymat berish operatori bajarilishida oldin chap tarafdagi operand xotirasi tozalanadi va keyinchalik shu joyga o‘ng tomondagi operandning haqiqatga to‘g‘ri kelmaydigan qiymatini joylashtiradi. SHu sababli, yuqoridagi misolda operator=() funksiyasi

if (this==&ob) return *this;

nazorat ko‘rsatmasiga ega va u xatolik ro‘y berishiga yo‘l qo‘ymaydi.

Unar operatorlarni qayta yuklash

Unar operatorlar uchun faqat bitta operand kerak bo‘ladi. Unar operatorni sinfning funksiya–a‘zosi ko‘rinishida qayta yuklashda bu yagona operand bu amalni chaqirgan ob‘ektning o‘zi hisoblanadi. SHu sababli, unar operatorning operator funksiyasi nostatik funksiya–a‘zo sifatida e‘lon qilinadi va u quyidagi ko‘rinishga ega bo‘ladi:

<qaytaruvchi qiymat turi > operatorX();

bu erda <qaytaruvchi qiymat turi > funksiya qaytaradigan qiymat turi, X– qayta yuklanayotgan unar operator.

Agar operator funksiya global ravishda e‘lon qilinganda, u quyidagi sintaksisga javob berishi kerak:

<qaytaruvchi qiymat turi > operatorX(<parametr turi> <parametr>);

bu erda <parametr turi> – parametr turi va <parametr> – funksiya parametri.

Plyus va minus unar operatorlarni qayta yuklashga misol ko‘raylik.

#include <iostream.h>

class Nuqta

{

int x,y;

```

public:
    Nuqta(int _x,int _y){x=_x; y=_y;}
    Nuqta(){x=0; y=0;}
    Qiymat_xy(int & _x,int & _y){_x=x; _y=y;}
    Nuqta operator+();
    Nuqta operator-();
};

Nuqta & Nuqta::operator+()
{
    x+=x;
    y+=y;
    return *this;
}

Nuqta & Nuqta::operator-()
{
    x=-x;
    y=-y;
    return *this;
}

int main()
{
    int a,b;
    Nuqta N1(-10,20);
    N1=+N1;           //qayta yuklangan plyus operatorini chaqirish
    N1.Qiymat_xy(a,b);
    cout<<"N1.x="<<a<<" N1.y="<<b<<endl;
    N1=-N1;           //qayta yuklangan plyus operatorini chaqirish
    N1.Qiymat_xy(a,b);
    cout<<"N1.x="<<a<<" N.y="<<b<<endl;
    return 0;
}

```

Programma ishlashi natijasida ekranda

N1.x=-10 N1.y=20

N1.x=10 N1.y=-20

satrlari paydo bo‘ladi.

Xuddi shu natijalarga global operator funksiyalarni sinfning do‘st funksiyalari ko‘rinishida e‘lon qilish orqali erishish mumkin:

```

friend Nuqta operator+(Nuqta & ob);
friend Nuqta operator-(Nuqta & ob);

Bu funksiyalar aniqlanishi

friend Nuqta operator+(Nuqta & ob)
{

```

```

ob.x+=ob.x;
ob.y+=ob.y;
return ob;
}
friend Nuqta operator-(Nuqta & ob)
{
ob.x=-ob.x;
ob.y=-ob.y;
return ob;
}

```

Ushbu funksiyalarni chaqirish natijalari yuqoridagi funksiyalar bilan bir xil bo‘ladi.

Inkrement va dekrement operatorlarini qayta yuklash

Inkrement va dekrement operatorlari, ularning prefiks va postfiks ko‘rinishlari bo‘lishi hisobiga qayta yuklash nuqtai-nazaridan alohida kategoriyaga tushadi. Prefiks va postfiks ko‘rinishlarni farqlash uchun quyidagi qoidalarga amal qilinadi: prefiks ko‘rinishni qayta yuklash, odatdagi unar operatorni qayta yuklash bilan bir xil; postfiks ko‘rinish uchun operator funksiya qo‘shimcha int turidagi parametrga ega bo‘ladi. Amalda bu parametr ishlatilmaydi va funktsiyani chaqirishda uning qiymati 0 bo‘ladi (zarurat bo‘lganda ishlatilishi mumkin). Bu parametrning vazifasi – kompilyatorga operatorning postfiks ko‘rinishi ishlatilayotganligini bildirishdir. Quyidagi misolda Nuqta sinfi uchun inkrement va dekrement operatorlarini qayta yuklash ko‘rsatilgan:

```

#include <iostream.h>
class Nuqta
{
int x,y;
public:
Nuqta(int _x,int _y){x=_x; y=_y;}
Nuqta(){x=0; y=0;}
Qiymat_xy(int & _x,int & _y){_x=x; _y=y;}
Nuqta & operator++(); // prefiks inkrement uchun
Nuqta operator++(int); // postfiks inkrement uchun
Nuqta & operator--(); // prefiks dekrement uchun
Nuqta operator--(int); // postfiks dekrement uchun
};
Nuqta & Nuqta::operator++()
{
x++;
y++;
return *this;
}
Nuqta Nuqta::operator++(int)
{

```

```

    Nuqta Oraliq=*this;
    ++*this;
    return Oraliq;
}
Nuqta & Nuqta::operator--()
{
    x--;
    y--;
    return *this;
}
Nuqta Nuqta::operator--(int)
{
    Nuqta Oraliq=*this;
    --*this;
    return Oraliq;
}

```

```

int main()
{
    int a,b;
    Nuqta N1(-10,20);N2(15,25),N3;
    ++N1;          //prefiks inkrement operatorini chaqirish
    N1.Qiymat_xy(a,b);
    cout<<"(++N1).x="<<a<<" (++N1).y="<<b<<endl;
    N1++;          //postfiks inkrement operatorini chaqirish
    N1.Qiymat_xy(a,b);
    cout<<"(N1++).x="<<a<<" (N1++).y="<<b<<endl;
    N3=--N2; //prefiks dekrement operatorini chaqirish
    N3.Qiymat_xy(a,b);
    cout<<"N3=--N2; => N3.x="<<a<<" N3.y="<<b<<endl;
    //postfiks dekrement operatorini chaqirish
    (N3=N1--).Qiymat_xy(a,b);
    cout<<"(N3=N1--).x="<<a<<" (N3=N1--).y="<<b<<endl;
    N1.Qiymat_xy(a,b);
    cout<<"N1.x="<<a<<" N1.y="<<b<<endl;
    N2.Qiymat_xy(a,b);
    cout<<"N2.x="<<a<<" N2.y="<<b<<endl;
    return 0;
}

```

Programma ishlashi natijasida ekranga

(++N1).x=11 (++N1).y=21

(N1++).x=12 (N1++).y=22

N3=--N2; => N3.x=14 N3.y=24

(N3=N1--).x=12 (N3=N1--).y=22

N1.x=11 N1.y=22

N2.x=14 N2.y=24

xabarlari chiqadi.

Programmada inkrement va dekrement operatorlarining prefiks va postfiks ko‘rinishlarini qayta yuklashni amalga oshirishda o‘ziga xos yo‘l tanlangan. Masalan, inkrement operatorining prefiks ko‘rinishi uchun aniqlangan operator funksiyaning qaytaradigan qiymati sinf ob’ektiga murojaat, chunki inkrement operatorining postfiks ko‘rinish uchun aniqlangan operator funksiya shu funksiyani chaqiradi va o‘zgargan ob’ektni qaytarib olishi kerak. Umuman olganda, bu funksiyalarni bir–biriga bog‘liqmas ravishda aniqlash mumkin:

Nuqta Nuqta::operator++()

```
{  
    x++;  
    y++;  
    return *this;  
}
```

Nuqta Nuqta::operator++(int)

```
{  
    x++;  
    y++;  
    return *this;  
}
```

Lekin bu variantda bir xil amallar ketma-ketligini takror yoziladi va inkrement operatorini turlicha talqin qilish bilan bog‘liq xatolarni yuzaga kelishiga zamin bo‘ladi. Ma’qul variant– bu operatorning prefiks ko‘rinishining operator funksiyasida operator mazmuni aniqlanadi va postfiks ko‘rinishni qayta yuklash unga tayanadi.

Endi inkrement va dekrement operatorlarini do‘st funksiyalar orqali qayta yuklashni ko‘ramiz. SHunga e’tibor berish kerakki, do‘st funksiyaga murojaat ko‘rinishidagi argument uzatilishi va u o‘zgartirilib funksiya tomonidan qaytarilishi kerak bo‘ladi.

#include <iostream.h>

class Nuqta

{

int x,y;

public:

Nuqta(int _x,int _y){x=_x; y=_y;}

Nuqta(){x=0; y=0;}

Qiymat_xy(int & _x,int & _y){_x=x; _y=y;}

friend Nuqta & operator++(Nuqta &);// prefiks inkrement


```

//postfiks inkrement
friend Nuqta operator++(Nuqta&,int);
friend Nuqta & operator--(Nuqta&); // prefiks dekrement
// postfiks dekrement
friend Nuqta operator--(Nuqta&int);
};
Nuqta & operator++(Nuqta & ob)
{
    ob.x++;
    ob.y++;
    return ob;
}
Nuqta operator++(Nuqta & ob,int)
{
    Nuqta Oraliq=ob;
    ++ob;
    return Oraliq;
}
Nuqta & operator--(Nuqta &)
{
    ob.x--;
    ob.y--;
    return ob;
}
Nuqta operator--(Nuqta &,int)
{
    Nuqta Oraliq=ob;
    --ob;
    return Oraliq;
}

```

```

int main()
{
    int a,b;
    Nuqta N1(-10,20);N2(15,25),N3;
    ++N1;          //prefiks inkrement operatorini chaqirish
    N1.Qiymat_xy(a,b);
    cout<<"(++N1).x="<<a<<" (++N1).y="<<b<<endl;
    N1++;          //postfiks inkrement operatorini chaqirish
    N1.Qiymat_xy(a,b);
    cout<<"(N1++).x="<<a<<" (N1++).y="<<b<<endl;
    N3=--N2; //prefiks dekrement operatorini chaqirish
    N3.Qiymat_xy(a,b);
    cout<<"N3=--N2; => N3.x="<<a<<" N3.y="<<b<<endl;
    //postfiks dekrement operatorini chaqirish
    (N3=N1--).Qiymat_xy(a,b);
}

```

```

cout<<"(N3=N1--).x="<<a<<" (N3=N1--).y="<<b<<endl;
N1.Qiymat_xy(a,b);
cout<<"N1.x="<<a<<" N1.y="<<b<<endl;
N2.Qiymat_xy(a,b);
cout<<"N2.x="<<a<<" N2.y="<<b<<endl;
return 0;
}

```

Programma ishlashi natijasida ekranga xuddi oldingi misoldagidek xabarlar chop etiladi.

YUqorida qayd qilingandek, int turidagi argument odatda ishlatishmaydi, lekin zarur bo'lganda ishlatishi mumkin. Bu argumentni ishlatishga misol:

```

#include <iostream.h>
class Nuqta
{
    int x,y;
public:
    Nuqta(int _x,int _y){x=_x; y=_y;}
    Nuqta(){x=0; y=0;}
    Qiymat_xy(int & _x,int & _y){_x=x; _y=y;}
    Nuqta & operator++(int);
};
Nuqta & Nuqta::operator++(int n)
{
    if (n!=0)
    {
        x+=n;
        y+=n;
    }
    else
    {
        x++;
        y++;
    }
    return *this;
}
int main()
{
    Nuqta N1(10,20);
    N1.operator++(100);          //100 soniga inkrement
    return 0;
}

```

Bu holatning o'ziga xosligi shundaki, postfix inkrement operatorining operator funksiyasini oshkor ravishda chaqirishga to'g'ri keladi. Chunki kompilyator

```

N1++(100);

```

ifodasini operatorgacha bo'lgan qismini alohida ajratib

N1++;

ko'rsatma deb tushunadi.

Indekslash operatorini qayta yuklash

Kvadrat qavslar ('[' , ']') bilan yoziladigan indekslash operatori binar operator hisoblanadi va u qayta yuklanishida operator funksiya sinfning bitta argumentli nostatik funksiya–a'zosi sifatida aniqlanishi kerak. Funksiya argumenti ixtiyoriy turda bo'lishi va u sinf ob'ektlari massivining indeksi deb qabul qilinadi. Quyidagi misol buni namoyon qiladi:

```
#include <iostream.h>
class BS_Massiv
{
    int MaxIndex;
    int * kButun;
public:
    BS_Massiv(int Elem_Soni)
    ~BS_Massiv(){delete kButun;}
    int & operator[](int index);
};
BS_Massiv::BS_Massiv(int Elem_Soni)
{
    kInt=new int(Elem_Soni);
    MaxIndex=Elem_Soni;
}
int & BS_Massiv::operator[](int index)
{
    static int iXato=-1;
    if(index>=0 && index<MaxIndex) return kInt[index];
    else
    {
        cout<<"Xato: Massiv chegarasidan chiqish ro'y berdi!";
        cout<<endl;
        return iXato;
    }
}
main()
{
    BS_Massiv vector(5);
    for(int i=0; i<5; i++)
        vector[i]=i;
    for(int i=0; i<=5; i++)
        cout<<" vector["<<i<<"]="<<vector[i]<<endl;
    return 0; }
```

Programmada 5 ta, butun son turidagi elementlardan tashkil topgan vector massivi BS_Massiv sinfining ob'ekti sifatida e'lon qilingan va unga qiymatlar berilib, keyin chop qilingan. Indeks argumenti i=5 bo'lganda xatolik haqida xabar beriladi. Programma ishlashi natijasida ekranga quyidagilar chiqadi:

vector[0]=0

vector[1]=1

vector[2]=2

vector[3]=3

vector[4]=4

Xato: Massiv chegarasidan chiqish ro'y berdi!

vector[5]=-1

SHunga e'tibor berish kerakki, operator[] funksiyasi murojaat qaytaradi (son qiymatini emas) va shu sababli bu funksiyani qiymat berish operatorining ikki tomonida ham qo'llash imkoni yuzaga keladi.

Nazorat savollari

1. Operatorlarni qayta yuklash nima?
2. Operatorlarni qayta yuklash afzalliklari nimada?
3. Operatorlarni qayta yuklash qanday amalga oshiradi va nimaga?
4. Qayta yuklash nima?
5. Operatorlarni qayta yuklash qaysi operatorlar uchun mumkin emas?
6. Operatorlarni qayta yuklash kamchilliklari va yutuqlari nimada?

Amaliy ish № 26, 27

Mavzu: Qoliplar yaratish va foydalanish

Ishdan maqsad: Qoliplar yaratish ko'nikmalarini egallash.

Nazariy qism

Funksiyalar qoliplari

Qoliplar – berilganlar turlaridan, funksiyalar va sinflar aniqlanishlaridan foydalanish ma'nosida umumlashtirishga imkon beruvchi tushunchalardir. SHu sababli ularni *parametrlashtirilgan funksiyalar* yoki *parametrlashtirilgan sinflar* deyiladi. Aksariyat hollarda «qolipli funksiyalar» va «qolipli sinflar» terminlari ishlatiladi.

Funksiya qolipi funksiyaning umumlashgan aniqlanishi bo'lib, uning asosida kompilyator foydalanuvchi tomonidan berilgan turdagi funksiya vakilini yaratadi.

Funksiya qolipini e'lonining sintaksisi quyidagi ko'rinishga ega:

```

template <class T1|T1 <identifiktor1>,
class T2|T2 <identifiktor2>,
..., class Tn| Tn <identifiktorn> >
<Qaytaruvchi qiymat turi><funksiya nomi>
(<parametrlar ro'yxati>)
{
// funksiya tanasi
}

```

template kalit so'zidan keyin burchakli qavs ichida bir-biridan vergul bilan ajratilgan parametrlar ro'yxati keladi. Har bir parametr - slass kalit so'zi yoki tur nomi va undan keyin identifikatordan iborat bo'ladi. Ayrim hollarda, berilganlarning parametrlashgan turini berish uchun class so'zi o'rniga typename kalit so'zi ishlatilishi mumkin.

E'londagi class yoki typename kalit so'zidan keyingi qolip parametrlariga *parametrlashgan turlar* deyiladi. Ular kompilyatorga qolipda parametr sifatida qanday berilganlar turi ishlatilayotganini bildiradi. Tur nomi va identifikatordan iborat qolip parametri kompilyatorga qolip parametri ko'rsatilgan turdagi konstanta ekanligini bildiradi.

Funksiya qolipini aniqlash va chaqirishga misol ko'raylik.

```

#include <iostream.h>
template <class T>
T Kvadrat(T x)
{ return x*x; }
template <class T>
T* Almashtirish(T * t, int ind1, int ind2)
{
    T Vaqtincha=t[ind1];
    t[ind1]=t[ind2];
    t[ind2]=Vaqtincha;
    return t;
}
template <typename T1, typename T2>
void Ekranga(T1 x, T2 y)
{ cout<<x<<"\t"<<y<<endl; }
template <class T, int siljish>
void Obect_Adresi(T * obj, unsigned int * pAdres)
{

```

```

    *pAdres=(unsigned int) & obj[0]+siljish*sizeof(T);
}

int main()
{
    int n=10, kv_n, i=1, j=3;
    double d=10.21, kv_d;
    char * satr="Qolip";
    kv_n=Kvadrat(n);
    kv_d=Kvadrat(d);
    int Massiv[10];
    unsigned int adres=0;
    cout<<"n="<<n<<" Qvadrat n="<<kv_n<<endl;
    cout<<"d="<<n<<" Qvadrat d="<<kv_d<<endl;
    cout<<"satr="<<satr<<" O'zgargan satr= "
    << Almashtirish(satr,i,j)<<endl;
    cout<<"Son va uning qvadrati:\n";
    Ekranga(n, d);
    Ekranga(kv_n, kv_d);
    Obect_Adresi<int,5>(Massiv, &adres);
    cout<<"Arr[5] element adresi = "
    <<hex<<showbase<<adres;

return 0;
}

```

Programma ishlashi natijasida ekranga quyidagilar chop etiladi:

```

n=10 Qvadrat n=100
d=10 Qvadrat d=104.244
satr=Qilop O'zgargan satr=Qilop
Son va uning qvadrati:
10      10.21
100     104.244
Arr[5] element adresi = 0x12ff50

```

Xuddi oddiy funksiyalardek funksiyalar qolipining prototipini e'lon qilish mumkin.

Masalan:

```

template <class T>

```

T Kvadrat(T x);

Funksiya qolipi va uning prototipidagi parametrlar nomi ustma – ust tushmasligi
mumkin:

// Funksiya qolipi prototipi

template <class T, class S>

void Fun(T,S);

...

// Funksiya qolipi aniqlanishi

template <class U, class V>

void Fun(U,V)

{

// Funksiya tanasi

}

Funksiya qolipining prototipidagi har bir parametr qolip aniqlanishida ishlatilishi kerak.

Masalan, quyidagi holat xatolikka olib keladi:

template <class T, class S>

T Fun(S);

Funksiya qolipining parametrlarining nomlari bir xil bo'lmashligi kerak. Quyidagi e'lon xato hisoblanadi:

template <class T, class T>

T Fun(T,T);

Qolipli funksiya tashqi, statik va joylashuvchi deb e'lon qilinishi mumkin. Buning uchun mos kalit so'zlar template qatorining oxirida yozilishi kerak bo'ladi:

// Tashqi qolipli funksiya prototipi

template <class T, class S> extern

void Fun2(T,S);

// Statik qolipli funksiya prototipi

template <class T> static

T Fun2(T);

// Joylashuvchi qolipli funksiya

prototipi

template <class T, class S> static

T* Fun3(T,S);

Qolipli funksiyaning har bir tur bilan birinchi chaqirilishida uning vakili yaratiladi va bu jarayonga *qolipli funksiyaning konkretlash* deyiladi. YUqorida keltirilgan programmadagi

```
kv_n=Kvadrat(n);
```

```
kv_d=Kvadrat(d);
```

chaqirishlar butun tur uchun Kvadrat() funksiyasini, ikkinchi ko'rsatmani bajarish uchun esa bu funksiyasining haqiqiy tur uchun vakilini yaratadi.

Funksiya qolipini konkretlashtirishda uning parametrlari uchun boshqa turdan oshkor ravishda kutilgan turga keltirish amalini bajarish mumkin.

Masalan:

```
template <class T>
```

```
void Fun1(T){...};
```

```
...
```

```
void Fun2(char ch)
```

```
{
```

```
Fun1<int>(ch);
```

```
}
```

Ushbu misolda kompilyator char turidagi ch o'zgaruvchisini int turiga o'tkazadi.

Nazorat savollari

1. Qoliplar nima?
2. Qoliplarni yaratishni afzalliklari nimada?
3. Qoliplarni yaratish qanday amalga oshiradi?
4. Qoliplarni qayta yuklash mumkinmi?
5. Nima uchun qoliplarni qo'shimcha direktivalarda ishlatish mumkin emas?
6. Qoliplarni yaratishning kamchilliklari va yutuqlari nimada?

MUSTAQIL TA'LIM MAVZULARI

1. Matematik amallardan foydalanishni ko'rsatuvchi dastur tuzing.
2. Mantiqiy amallardan foydalanishni ko'rsatuvchi dastur tuzing.
3. Nisbat amallardan foydalanishni ko'rsatuvchi dastur tuzing.
4. Munosabat amallardan foydalanishni ko'rsatuvchi dastur tuzing.
5. Son absolyut qiymatini shartli amal yordamida hisoblovchi dastur tuzing.
6. Berilgan eps aniqlikda umumiy xadi $1/n!$ bo'lgan ketma-ketlik yig'indisini hisoblovchi dastur tuzing.
7. Umumiy xadi $x/n!$ bo'lgan ketma-ketlik n ta xadi yig'indisini hisoblovchi dastur tuzing.
8. Kiritilgan n ta son qat'iy o'suvchi ekanligini tekshiruvchi dastur yarating.
9. Kiritilgan n simvoldan nechitasi o'nli harf ekanligini switch operatori yordamida hisoblovchi dastur tuzing.
10. Rekursiya yordamida Paskal uchburchagini hisoblovchi funksiya tuzing. Bu funksiya yordamida uchburchakni ekranga chiqaruvchi funksiya tuzib dasturda foydalaning.
11. Berilgan massiv qat'iy kamayuvchi ekanligini tekshiruvchi funksiya tuzing va dasturda foydalaning.
12. Matrisani vektorga ko'paytirish funksiyasini tuzing va dasturda foydalaning.
13. Berilgan satr telefon raqami ekanligini aniqlovchi funksiya tuzing va dasturda foydalaning.
14. Berilgan satr o'zgaruvchi ekanligi tekshiruvchi funksiya tuzing va dasturda foydalaning.
15. Berilgan jumladan eng qisqa so'z ajratib oluvchi funksiya tuzing va dasturda foydalaning.
16. Abituriyent (ismi, tug'ilgan yili, yiqqan bali, attestat o'rta bali) strukturasini yarating. Struktura tipidagi massiv yarating.
17. Massiv elementidan bir elementni olib tashlab, yangi massiv elementini tashkil eting.
18. Xodim (ismi, lavozimi, tug'ilgan yili, oyligi) strukturasini yarating. Struktura tipidagi massiv yarating.
19. Ko'rsatilgan familiyali elementni olib tashlang va ko'rsatilgan raqamli elementdan so'ng element qo'shing.
20. Mamlakat (nomi, poytaxti, axoli soni, egallagan maydoni) strukturasini yarating. Struktura tipidagi massiv yarating. Ko'rsatilgan aholi sonidan kichik bo'lgan elementni olib tashlang va ko'rsatilgan nomga ega bo'lgan elementdan so'ng element qo'shing.

21. Davlat (nomi, davlat tili, pul birligi, valyuta kursi) strukturasini yarating. Struktura tipidagi massiv yarating. Ko'rsatilgan nomga ega bo'lgan elementni o'chiring va fayl oxiriga ikkita element qo'shing.

22. Inson (ismi, yashash manzili, telefon raqami, yoshi) strukturasini yarating. Struktura tipidagi massiv yarating. Ko'rsatilgan yoshga ega bo'lgan elementni o'chiring va berilgan telefon raqamidagi elementdan oldin element qo'shing.

23. Berilgan satr o'zgaruvchi ekanligini aniqlovchi funksiya tuzing va dasturda foydalaning. Funksiya tanasida faqat ko'rsatkichlar ustida amallardan foydalaning.

24. Matrisani vektorga ko'paytirish funksiyasini yaratib dasturda foydalaning. Matrisa ko'rsatkichlar massivi sifatida kiritilsin.

25. Dinamik ravishda o'quvchilar familiyalari va baholari massivlarini hosil qiluvchi funksiyalar yarating. Hamma a'lochilar familiyalarini chiqaruvchi funksiya tuzib dasturda foydalaning.

26. Uchburchak dinamik massiv yordamida Paskal uchburchagini hisoblovchi funksiya tuzing. Bu funksiya tashqari uchburchakni ekranga chiqaruvchi funksiya tuzib dasturda foydalaning.

27. Dixotomiya usuli yordamida $f(x) = 0$ tenglamani yechish uchun funksiya tuzing. Funksiyaga ko'rsatkich parametr sifatida uzatilsin.

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 ixtisoslashtirilgan 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 гурӯҳи	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	Маҳаллий ҳисоблаш тармоғи	Metropolitan Area Network
WAN	Худудий ҳисоблаш тармоғи	Wide Area Network
ISO	Халқаро 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 attestatsiyadan 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 attestatsiyadan 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. “Vorish-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