O‘ZBEKISTON RESPUBLIKASI
OLIY VA O‘RTA MAXSUS TA’LIM VAZIRLIGI

MIRZO ULUG‘BEK NOMIDAGI
O‘ZBEKISTON MILLIY UNIVERSITETI

Sh.Madraximov, A.Ikramov, Q.Maxarov

DASTURLASH ASOSLARI
(*o‘quv qo‘llanma*)

Toshkent – 2017

UDK681.3.01
KBK 32.973.2

Sh.Madraximov, A.Ikramov, Q.Maxarov. Dasturlash asoslari.

Toshkent. “Universitet” nashriyoti-2017. 289bet.

Tuzuvchilar: **Madraximov Shavkat Fayzullayevich,**
Ikramov Axmad Maoripovich,
Maxarov Qodirbek Tolkunovich
Ma’sul muharrir: **O‘zMU f.-m.f.d., professor M.M.Aripov**
Taqrizchilar: **O‘zMU f.-m.f.d., professor v.b. A.M.Polatov**
TATU t.f.n., dotsent K.F.Kerimov

O‘quv qo‘llanmani nashr etishga O‘zbekiston Respublikasi Oliy va o‘rta maxsus ta’lim vazirligining 2017 yil 24 apreldagi 603 sonli buyrug‘iga asosan ruxsat berilgan (ro‘yxatga olish raqami 603-157)

DASTURLASH ASOSLARI
(*O‘quv qo‘llanma*)

O‘quv qo‘llanmada kompyuterda masalalaning qo’yilishi, turli toifadagi masalalarni yechishning algoritmlari va bu algoritmlarni dasturlash tiliga o’girish muammolari qaralgan. Unda C++ algoritmik tilining asosiy tushuncha va atamalari keltirilgan, berilganlar ustida ishlash bilan bog‘liq masalalar ko‘rib chiqilgan, hisoblash jarayonini tashkillashtirish aniq misollarda ko‘rsatilgan. Chiziqli, takrorlanuvchi va tarmoqlanuvchi jarayonlarni, shuningdek massiv elementlari bilan ishlash algoritmlarini amalga oshirish uchun zarur dasturlash saboqlari berilgan.

В учебном пособии рассмотрены вопросы постановки задачи, построения алгоритма различных задач и проблемы преобразования алгоритма в программу. Приведены основные понятия и лексемы языка C++, задачи обработки данных, формирование процессов вычисления показано на конкретных примерах. Представлены знания для реализации процесса линейных, ветвящих и циклических алгоритмов, а также реализация алгоритмов обработки элементов массивов.

In the training manual has describe different types of algorithms and methods for how to solve exercises, C++ algorithmic basic understanding of the language and terminology and discuss issues related to the processing of data, as shown as in the examples in the process of calculation. Linear, repeating and branching processes, as well as elements of the blocks associated with the processing algorithms necessary for the implementation of the program, as well as methods of preparation and use of the features.

Mundarija

Kirish 5
§1. C++ tilining leksikasi va sintaksisi 6
§2. C++ tili dasturining tuzilishi va shakli 12
§3. Berilganlar turlari. C++ tilining tayanch turlari 20
§4. O‘zgaruvchilar va ifodalar 26
§5. Amallar: inkrement, dekrement, sizeof, mantiqiy, razryadli, taqqoslash. Amallarning ustunliklari va bajarilish yo‘nalishlari 37
§6. O‘qish-yozish oqimlari (cin, cout) 48
§7. Operatorlar. Shart operatorlari 59
§8. Takrorlash operatorlari. Boshqaruvni uzatish operatorlari 75
§9. Statik massivlar 96
§10. Funksiyalar e’lon qilish, aniqlash va ularga murojaat qilish 111
§11. Rekursiv funksiyalar 129
§12. Foydalanuvchi tomonidan aniqlangan berilganlar turlari 136
§13. Nomlar fazosi 144
§14. Standart kutubxona funksiyalari 153
§15. Ko‘rsatkichlar va adres oluvchi o‘zgaruvchilar 162
§16. Dinamik massivlar 174
§17. Funksiya va massivlar 183
§18. Satrlar. Satr ustida amallar. Satr funksiyalari 193
§19. Tuzilmalar. Birlashmalar 215
§20. Identifikatorlarning amal qilish doirasi. Makroslarni aniqlash va joylashtirish. 233
§21. Standart oqimlar. Berilganlarni formatlash 243
§22. Fayllar.Matn va binar fayllar 254
§23. Fayldan o‘qish-yozish funksiyalari. Fayl ko‘rsatkichini boshqarish funksiyalari. 262
§24. Berilganlarning dinamik tuzilmalari 276
Nazorat savollari ..... **Ошибка! Закладка не определена.**
Glossariy 294
Foydalanilgan adabiyotlar ro‘yxati 297
Ilovalar 299

Kirish

Dasturlash asoslarini o’zlashtirishdan asosiy maqsad – talabalarga qo’yilgan tadbiqiy masalani anglash, yechish algoritmini ishlab chiqish va dasturiy ta’minotini yaratish asoslarini o’rgatishdir. Mazkur o’quv qo’llanmada masala yechishning matematik modellari, usullari va algoritmlarini yaratish va shu asosida kompyuterda masalanining dasturiy ta’minotini amalga oshirish asoslari keltirilgan. Shuningdek, kompyuterda berilganlar va buyruqlarni tasvirlanishi, C++ tilida dasturlash texnologiyalari, dasturlash tilining tayanch tushunchalari, vizual muhitda ishlash asoslari keltirilgan.

Dasturlash asoslarini o’zlashtirish jarayonida talaba axborot, uni saqlash usullari, qayta ishlash va uzatish, hisoblash tizimlarining matematik va dasturiy ta’minoti, ularni fan sohalarida, ishlab chiqarish va ta’limda qo’llash xususiyatlari, kompyuterni dasturiy ta’minoti, dastur turlari va xususiyatlari, dasturni optimallashtirish va umumlashtirish, dasturlashda modulli tamoyillarni qo’llash, kompyuter texnologiyalari yutuqlarini zamonaviy hisoblash tizimlarining matematik va dasturiy ta’minotida qo’llash, yuqori darajadagi dasturlash tillarini, dasturiy ta’minotni, dasturlash texnologiyalarini, tadbiqiy va hisoblash matematikasi masalalarini yechish algoritmlarini, modulli tahlil va modulli dasturlash asoslarini, samarali dastur va dasturlar kompleksini yaratish usullarini bilish va ulardan foydalana olish, tadbiqiy masalalarni yechish algoritmini tuzish, matematik (kompyuter) modelini qurish va uning dasturiy ta’minotini yaratish ko’nikmalariga ega bo’ladi.

§1. C++ tilining leksikasi va sintaksisi

Dasturlash tillari

Ma’lumki, dastur mashina kodlarining qandaydir ketma-ketligi bo‘lib, aniq bir hisoblash vositasining amal qilishini boshqaradi. Dasturiy ta’minotni yaratish jarayonini osonlashtirish uchun yuzlab dasturlash tillari yaratilgan. Barcha dasturlash tillarini ikki toifaga ajratish mumkin:

– quyi darajadagi dasturlash tillari;

– yuqori darajadagi dasturlash tillari.

Quyi darajadagi dasturlash tillariga Assembler turidagi tillar kiradi. Bu tillar nisbatan ixcham va tezkor bajariluvchi kodlarni yaratish imkoniyatini beradi. Lekin, Assembler tilida dastur tuzish zahmatli, nisbatan uzoq davom etadigan jarayondir. Bunga qarama-qarshi ravishda yuqori bosqich tillari yaratilganki, ularda tabiiy tilning cheklangan ko‘rinishidan foydalangan holda dastur tuziladi. Yuqori bosqich tillaridagi operatorlar, berilganlarning turlari, o‘zgaruvchilar va dastur yozishning turli usullari tilning ifodalash imkoniyatini oshiradi va dasturni «o‘qimishli» bo‘lishini ta’minlaydi. Yuqori bosqich tillariga Fortran, PL/1, Prolog, Lisp, Basic, Pascal, C va boshqa tillarni misol keltirish mumkin. Kompyuter arxitekturasini takomillashuvi, kompyuter tarmog‘ining rivojlanishi mos ravishda yuqori bosqich tillarini yangi variantlarini yuzaga kelishiga, yangi tillarni paydo bo‘lishiga, ayrim tillarni esa yo‘qolib ketishiga olib keldi. Hozirda keng tarqalgan tillarga Object Pascal, C++, C#, Python, Php, Java tillari hisoblanadi. Xususan, C tilining takommillashgan varianti sifatida C++ tilini olishimiz mumkin. 1972 yilda Denis Ritch va Brayan Kernegi tomonidan C tili yaratildi. 1980 yilda Byarn Straustrop C tilining avlodi C++ tilini yaratdi va unda strukturali va obektga yo‘naltirilgan dasturlash texnologiyasiga tayangan holda dastur yaratish imkoniyati tug‘ildi.

C++ tilidagi sodda dastur

Quyida C++ tilidagi sodda dastur matni keltirilgan.

```
#include <iostream>                // sarlavha faylni qo‘shish
```

```
using namespace std;          // std nomlar fazosini ishlatish
int main()                    // bosh funksiya tavsifi
{                             // blok boshlanishi
cout << “Salom Olam! \n”;     //  satrni chop etish
return 0;                     // funksiya qaytaradigan qiymat
}                             // blok tugashi
```

Dastur bajarilishi natijasida ekranga "Salom Olam!" satri chop etiladi.

Dasturning 1-satridagi #include – preprotsessor direktivasi bo‘lib, dastur kodiga oqimli o‘qish-yozish funksiyalari va uning o‘zgaruvchilari e’loni joylashgan «iostream» sarlavha faylini qo‘shadi. Keyingi qatorlarda dasturning yagona, asosiy funksiyasi - main() funksiyasi tavsifi keltirilgan. Shuni qayd etish kerakki, C++ dasturida albatta main() funksiyasi bo‘lishi shart va dastur shu funksiyani bajarish bilan o‘z ishini boshlaydi.

C++ tilida dastur yaratish bir nechta bosqichlardan iborat bo‘ladi. Dastlab, matn tahririda (odatda dasturlash muhitining tahririda) dastur matni teriladi, bu faylning kengaytmasi “*.cpp*” bo‘ladi. Keyingi bosqichda dastur matni yozilgan fayl kompilyatorga uzatiladi, agarda dasturda xatoliklar bo‘lmasa, kompilyator “*.obj*” kengaytmali obekt modul faylini hosil qiladi. Oxirgi qadamda komponovka (yig‘uvchi) yordamida “*.exe*” kengaytmali bajariluvchi fayl dastur hosil bo‘ladi. Bosqichlarda yuzaga keluvchi fayllarning nomlari boshlang‘ich matn faylining nomi bilan bir xil bo‘ladi.

Kompilyatsiya jarayonining o‘zi ham ikkita bosqichdan tashkil topadi. Boshida preprotsessor ishlaydi, u matndagi kompilyatsiya direktivalarini bajaradi, xususan #include direktivasi bo‘yicha ko‘rsatilgan kutubxonalardan C++ tilida yozilgan modullarni dastur tarkibiga kiritadi. Shundan so‘ng kengaytirilgan dastur matni kompilyatorga uzatiladi. Kompilyator o‘zi ham dastur bo‘lib, uning uchun kiruvchi ma’lumot bo‘lib, C++ tilida yozilgan dastur matni hisoblanadi. Kompilyator dastur matnini leksema (atomar) elementlarga ajratadi va uni leksik, keyinchalik sintaktik tahlil qiladi. Leksik tahlil

jarayonidau matnni leksemalarga ajratish uchun “probel ajratuvchisini” ishlatadi. Probel ajratuvchisiga ' ' – probel, '\t' – tabulyasiya, '\n'– keyingi qatorga o‘tish va boshqa ajratuvchilar hamda izohlar (kommentariylar) kiradi.

C++ tilida izohlar

Dastur matni tushunarli bo‘lishi uchun izohlar ishlatiladi. Izohlar kompilyator tomonidan “*o‘tkazib*” yuboriladi va ular dastur amal qilishiga hech qanday ta’sir qilmaydi.

C++ tilida izohlar ikki ko‘rinishda yozilishi mumkin.

Birinchisida "/*" belgilardan boshlanib, "*/" belgalari bilan tugagan barcha belgilar ketma-ketligi izoh hisoblanadi, ikkinchisi “satrli izoh” deb nomlanadi va u "//" belgilardan boshlangan va satr oxirigacha yozilgan belgilar ketma-ketligi bo‘ladi. Izohning birinchi ko‘rinishida yozilgan izohlar bir necha satr bo‘lishi va ulardan keyin C++ operatorlari davom etishi mumkin.

Misol.

```
int main()
{
// bu qator izoh hisoblanadi
int a=0; //int d;
int c;
/* int b=15 */
/*- izoh boshlanishi
a=c;
izoh tugashi */
return 0;
}
```

Dasturda d va b o‘zgaruvchilarning e’lonlari inobatga olinmaydi va a=c amali bajarilmaydi.

Ma’lumotlarni standart oqimga (ekranga) chiqarish uchun

cout <<<ifoda>;

ko’rinishdagi format ishlatiladi. Bu yerda <ifoda> sifatida o‘zgaruvchi yoki sintaksisi to‘g‘ri yozilgan va qandaydir qiymat qabul qiluvchi til ifodasi kelishi mumkin (keyinchalik, burchak qavs ichiga olingan o‘zbekcha satr ostini til tarkibiga kirmaydigan tushuncha deb qabul qilish kerak).

Masalan:

```
int uzg=324;
cout << uzg;        // butun son chop etiladi
```

Berilganlarni standart oqimdan (odatda klaviaturadan) o‘qish quyidagi formatda amalga oshiriladi:

```
cin >> <o‘zgaruvchi>;
```

Bu yerda <o‘zgaruvchi> qiymat qabul qiluvchi o‘zgaruvchining nomi.

Misol:

```
int Yosh;
cout << “Yoshingizni kiriting: ”;
cin >> Yosh;
```

Butun turdagi Yosh o‘zgaruvchisi kiritilgan qiymatni o‘zlashtiradi. Kiritilgan qiymatni o‘zgaruvchi turiga mos kelishini tekshirish mas’uliyati dastur tuzuvchining zimmasiga yuklanadi.

Bir paytning o‘zida probel vositasida bir nechta va har xil turdagi qiymatlarni oqimdan kiritish mumkin. Qiymat kiritish <*Enter*> tugmasini bosish bilan tugaydi. Agar kiritilgan qiymatlar soni o‘zgaruvchilar sonidan ko‘p bo‘lsa, “*ortiqcha*” qiymatlar bufer xotirada saqlanib qoladi.

```
#include <iostream>
using namespace std;
int main()
{
int x,y;
float z;
cin >> x >> y >> z;
```

```
cout <<  “O’qilgan qiymatlar:\n”<< x << ‘\t’ << y << ’\t’ << z;
return 0;
}
```

O‘zgaruvchilarga qiymat kiritish uchun klaviatura orqali

10 20 3.14↵

harakati amalga oshiriladi. Shuni qayd etish kerakki, oqimga qiymat kiritishda probel ajratuvchi hisoblanadi. Bu yerda “↵ “ belgisi “*Enter*” tugmasini bosilishini anglatadi. Haqiqiy sonning butun va kasr qismlari '.' belgisi bilan ajratiladi.

C++ tili alfaviti va leksemalari

C++ tili alfaviti va leksemalariga quyidagilar kiradi:

– katta va kichik lotin alfaviti harflari;

– raqamlar - 0,1,2,3,4,5,6,7,8,9;

– maxsus belgilar:“ { } | [ ] ( ) + - / % \ , ; ‘ : ? < = > _ ! & ~ # ^ . * .

Alfavit belgilaridan tilning leksemalari shakllantiriladi: identifikatorlar; kalit (xizmatchi yoki zahiralangan) so‘zlar; o‘zgarmaslar; amal belgilanishlari; ajratuvchilar.

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‘yilmagan.

Identifikatorlar kalit so‘zlarni, o‘zgaruvchilarni, funksiyalarni, nishonlarni va boshqa obektlarni nomlashda ishlatiladi.

C++ tilining kalit so‘zlariga quyidagilar kiradi:

asm, auto, bool, break, case, catch, char, class, const, const_cast, continue, default, delete, do, double, dynamic_cast, else, enum, explicit, export, extern, false, float, for, friend, goto, if, inline, int, long, mutable, namespace,

new, operator, private, protected, public, register, reinterpret_cast, return, short, signed, sizeof, static, static_cast, struct, switch, template, this, throw, true, try, typedef, typeid, typename, union, unsigned, using, virtual, void, volatile, wchar_t, 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 belgilari orasida probel ishlatish mumkin emas, zarur bo‘lganda uning o‘rniga ‘_’ ishlatish mumkin: Silindr_radiusi, aylana_diametri.

```
#include <iostream>
using namespace std;
int main()
{
int son;
son = 6;
cout << "Mening birinchi C++ dasturim." << endl;
cout << " 2 va 3 ning yig’indisi = " << 5 << endl;
cout << "7 + 8 = " << 7 + 8 << endl;
cout << "Son = " <<son<< endl;
return 0;
}
```

Ushbu dasturni kompilyatsiya qilib ishlatganda ekranga quyidagi to‘rtta qator chiqadi:

Mening birinchi C++ dasturim.

2 va 3 ning yig'indisi = 5

7 + 8 = 15

Son = 6

Nazorat savollari

1. Quyi darajadagi dasturlash tillari nima?
2. Kompilyatsiya jarayonining necha bosqichdan tashkil topadi?
3. Qanday belgilar bilan tugagan barcha belgilar ketma-ketligi izoh hisoblanadi?
4. C++ tilining kalit so‘zlariga qaysilar kiradi?
5. Protsessor registrlarini belgilash uchun qaysi so‘zlar ishlatiladi?
6. Kompilyatsiya jarayoni nima?
7. Kiritilgan qiymatni o‘zgaruvchi turiga mos kelishini qanday tekshirish mumkin?
8. Identifikator nima?
9. “__” (ikkita tagchiziq) belgilaridan boshlangan identifikatorlar nima uchun ishlatiladi?
10. RUN, run, Run – identifikatorlarning farqi nimada?
11. Sintaksis – qanday tilning qoidalari?

§2. C++ tili dasturining tuzilishi va shakli

Til sintaksisi

Avvalgi mavzuda ma’noga ega dastur tuzish uchun kerakli bo‘lgan C++ tushunchalari o‘rganildi. Dastur tuzish uchun avval uning o‘ziga xos strukturasini aniqlash kerak. O‘ziga xos strukturadan foydalanishdan asosiy maqsad – C++tilida tuzilgan dasturni tushunishni osonlashtirish va dasturni o‘zgartirishdan iborat. Sintaktik to‘g‘ri yozilgan, lekin hech qanday strukturaga ega bo‘lmagan dasturni tushunish va o‘zgartirish juda qiyin hamda ko‘p resurs, vaqt talab qiladi. Shuning bilan birga, har bir C++ tilida tuzilgan dastur tildagi aniqlangan qoidalarni qoniqtirishi zarur. C++ dasturida asosiy funksiya – main()

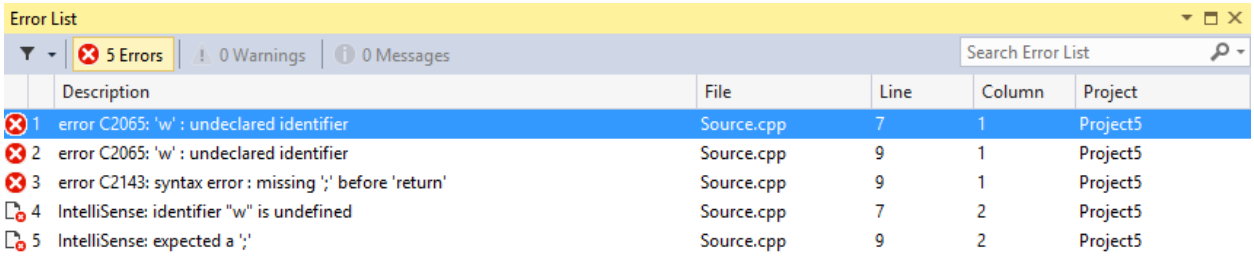
funksiyasi mavjud bo‘lishi kerak. Dastur grammatik qoidalarga o‘xshash bo‘lgan sintaksis qoidalariga, ya’ni tilda nima to‘g‘ri, nima noto‘g‘ri, nima mumkin, nima mumkin emasligini aniqlovchi qoidalarga amal qilish kerak. Shuningdek, maqsadga erishishda dasturlash tilining ma’nosini beruvchi tilning semantik qoidalariga ham amal qilish zarur. Bunda, sintaksis, probellarning ishlatilishi, nuqta, vergul, nuqtali vergul va qavslarning ishlatilishi va ma’nosi, semantika, identifikatorlar va ularning nomlanishi, qatorlarning ishlatilishi, izohlar va ularda ishlatilgan hujjatlash, shuningdek, dasturning tuzilish shakli va yozilish uslubiga (*stiliga*) alohida e’tibor berish kerak.

Sintaksis – til qurilmalarini qanday yozish mumkin va qanday yozish mumkin emasligini aniqlab beruvchi tilning qoidalari. Xatolar kompilyatsiya jarayonida aniqlanadi va dasturchiga ko‘rsatiladi.

Ma’lumki, dastur matni matn muharriri yordamida kompyuterga kiritiladi. Mukammal dastur matnini tuzish juda qiyin, turli ko‘rinishdagi va qiyinchilikdagi xatoliklar bo‘lishi tabiiy. Shuning uchun, kompilyatsiya jarayonidan keyin yuzaga kelgan xatoliklar matn muxarriri tomonidan dasturchiga taqdim etiladi. Shunday xatoliklar bo‘lishi mumkinki, ma’lum bir kod qismidagi xatolik dastur kodining boshqa qismlarida xatolikni yuzaga keltirayotgan bo‘lishi mumkin. Bunday holatlarda asosiy xatolik bo‘lgan qismini to‘g‘irlab, dasturni kompilyatsiya qilgandan so‘ng qolgan xatoliklar to‘liq yoki qisman yo‘qolishi mumkin. Shuning uchun, sintaksis xatolarini kompilyator ko‘rsatgan ketma-ketlikda bartaraf etish maqsadga muvofiq. Shuningdek, kompilyator faqatgina xatolarni aniqlabgina qolmasdan, ularning dastur matnining qaysi qismidaligini va uni bartaraf etish yo‘llarini ham ko‘rsatib beradi. Masalan:

```
#include <iostream>          //1-qator
using namespace std;         //2-qator
int main()                   //3-qator
{                            //4-qator
int x,y;                     //5-qator
```

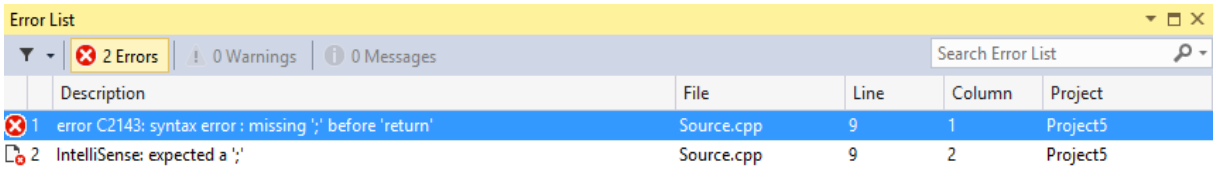
```
cin >> x >> y;                    //6-qator
w=x+y;                            //7-qator
cout << “W = ” << w               //8-qator
return 0;                         //9-qator
}                                 //10-qator
```



Dasturda 5ta xato mavjud, ularning qatorlari ham ko‘rsatilgan. Birinchi xato w identifikatori mavjud emas, ushbu xatoni to‘g‘irlash uchun 5-qator matnini quyidagicha o‘zgartiriladi:

```
int x, y, w;                      //5-qator
```

Keyinchalik kompilsiya jarayonini ishga tushurilganda kompilyator quyidagi xatolarni ko‘rsatib beradi:



Ko‘rinib turibdiki, birinchi xatoning o‘zi uchta kompilyatsiya xatoligini yuzaga keltirib chiqargan.

Probellarni ishlatishda ham alohida e’tiborli bo‘lish kerak. Berilganlar kiritilayotganda bir yoki bir nechta probellar berilganlarni ajratish uchun ishlatiladi. Shuningdek, kalit so‘zlar va identifikatorlarni ham bir-biridan ajratishda ko‘llaniladi. Faqat, kalit so‘z yoki identifikatorlarning o‘zini yozishda probellar umuman ishlatilmaydi.

```
int a, b, c;          //1-qator
dou ble d, e, f;      //2-qator
double birinchi;      //3-qator
float ikki nchi;      //4-qator
```

Ushbu namunalarda, 1-qator to‘g‘ri yozilgan, int kalit so‘zi va a identifikatori bitta probel yordamida ajratilgan. Qolgan identifikatorlardan oldin ham bittadan probel qo‘yilgan. 2-qatorda esa probel noto‘g‘ri ishlatilgan. double kalit so‘zini yozishda noto‘g‘ri probel qo‘yilgan, natijada kalit so‘z ajralib kolgan va bu kompilyatsiya jarayoni xatoligiga olib keladi. 3-qatorda double kalit so‘zi va birinchi identifikatori orasida to‘rtta probel mavjud va bu xatolikka olib kelmaydi. 4-qatorda esa, ikki nchi identifikatorni yozishda probel noto‘g‘ri ishlatilgan, aslida ikkinchi ko‘rinishida ajratilmasdan yozilishi kerak edi. Ushbu qatorda ham kompilyatsiya xatoligi yuzaga keladi.

Barcha C++ko‘rsatmalari (*instruksiyalari*) nuqtali vergul – “;” bilan tugallanishi zarur. Figurali qavslar (“{” va “}”), xattoki ular ko‘pincha bir qatorda va xech qanday dastur matnisiz kelsa ham ko‘rsatma emas. Figurali qavslar dastur qismini bir butunlik deb tushunish uchun ishlatiladi. Vergul (“,”) odatda ro‘yxat elementlarini ajratish uchun ishlatiladi. Masalan, o‘zgaruvchilar e’lon qilinayotganda bir turdagi bir nechta o‘zgaruvchilarni e’lon qilishda verguldan foydalaniladi.

Namuna uchun quyidagi C++ tilida yozilgan dastur qismini ko‘raylik:

```
int x, a;       // 1-qator
int y           // 2-qator
double z;       // 3-qator
y = w + x;      // 4-qator
```

Ushbu qatorlar kompilyatsiya qilinganda 2-qatorda kompilyatsiya xatoligi yuzaga keladi. Chunki, 2-qatorda y o‘zgaruvchisi e’lon qilinishidan keyin nuqtali vergul (“ ; ”) belgisi qo‘yilmagan. Ikkinchi kompilyatsiya xatoligi 4-qatorda yuzaga keladi. Bu qatorda w identifikatori ishlatilmoqda, ammo u e’lon qilinmagan.

Til semantikasi.

Til jumlalarining ma’nosini beruvchi qoidalar to‘plami semantika deyiladi. Masalan, arifmetik operatorlarning bajarilish ketma-ketligi qoidasi semantik qoida.

Agar dasturda sintaktik xatolar bo‘lsa, kompilyator bu haqida xabar beradi. Lekin semantik xato bo‘lganda dastur ishlaydi, lekin kutilgan natijaga erishib bo‘lmaydi. Masalan, quyidagi ikki qator sintaktik to‘g‘ri yozilgan, lekin ma’nolari turlicha (turlicha qiymat hosil bo‘ladi):

```
2 + 3 * 5      // 1-qator      va      (2 + 3) * 5 // 2-qator
```

Birinchi qatorda arifmetik operatorlar bajarilish ketma-ketligi qoidasiga ko‘ra ko‘paytirish amali bajariladi, so‘ngra qo‘shish amali ishga tushadi. Ikkinchi qatorda esa, avval qavs ichi bajariladi, sonlar qo‘shiladi, keyin ko‘paytirish amali bajariladi.

Quyida ikki xil ko‘rinishdagi ko‘rsatmalar keltirilgan:

```
const double a = 0.1;      //almashtirish o’zgarmasi
double x;                  //santimetr uchun o’zgaruvchi
double y;                  //millimetr uchun o’zgaruvchi
x = y * a;
```

va

```
const double SANTIMETR_UCHUN_MILLIMETR = 0.1;
double santimetr;
double millimetr;
santimetr = millimetr * SANTIMETR_UCHUN_MILLIMETR;
```

Hujjatlashgan identifikatorlar

Ikkinchi ko‘rinishdagi dastur ko‘rsatmalaridan foydalanilganda SANTIMETR_UCHUN_MILLIMETR ko‘rinishidagi identifikatorlar odatda hujjatlashgan identifikatorlar deb ataladi. Sababi, dasturchi identifikatordan nima maqsadda foydalanishni doim bilib turadi. Birinchi ko‘rinishdagi dastur ko‘rsatmalarida esa, identifikatorlardan nima maqsadda foydalanilayotganligi identifikatorlarni e’lon qilganda izohlar orqali berib o‘tilgan. Dastur matnining qolgan qismlarida identifikatorning nima maqsadda ishlatilishini bilish uchun doim dastur matni boshiga yoki identifikator e’lon qilingan qismiga o‘tib izoh orqali yozilgan hujjatlash qismidan o‘qib tushunish kerak bo‘ladi. Hujjatlashgan

identifikatorlar izohlardan foydalanishni kamaytirish uchun ishlatiladi. Olmasoni – hujjatlashgan identifikatorini tahlil qilaylik. Ushbu identifikatorni birgalikda qo‘llaniluvchi so‘z (*run-together word*) deyiladi. Hujjatlashgan identifikatorlardan foydalanayotganda birgalikda qo‘llaniluvchi so‘zlarni noto‘g‘ri qo‘llash orqali hujjatlashda aniqlikni kamaytirish mumkin. Birgalikda qo‘llaniluvchi so‘zlarni tushunarliroq qo‘llash uchun bir qancha ko‘rinishlar taklif qilingan. Har bir ma’noga ega so‘zni bosh harf orqali yozish yoki ulardan oldin tag chiziq (“_”) belgisini qo‘yish mumkin. Masalan, tushunarlilik aniqligini oshirish maqsadida olmaSoni, olma_soni ko‘rinishidagi identifikatorlardan foydalanish mumkin. O‘zgarmaslardan foydalanganda, ularni oddiy o‘zgaruvchilardan farqlab turish maqsadida barcha xarflarni bosh harflar yordamida yozish maqsadga muvofiq.

Dasturni formatlash

Dastur ishga tushirilganda foydalanuvchi dasturni tushunsa, bundaydastur yaxshi hujjatlashgan dastur deyiladi. Foydalanuvchi dasturni ishga tushirganda qanday berilganni kiritish kerakligini bilmasa dasturdan foydalanish qiyinlashadi. Shuning uchun foydalanuvchiga nima qilish kerakligini yoki qanday tushunchaga ega ma’lumot kiritilishi yoki chiqarilayotganini ko‘rsatib borish kerak. Masalan, int turidagi son o‘zgaruvchisi uchun quyidagi dastur qismi berilgan bo‘lsin:

```
cout << "1 va 10 oralig’idagi butun sonni kiriting"
<< " va Enter tugmasini bosing" << endl;
cin >> son;
```

Ushbu dastur qismi ishga tushganda foydalanuvchi ekranida quyidagi matn hosil bo‘ladi:

```
1 va 10 oralig’idagi butun sonni kiriting va Enter tugmasini bosing
```

Ushbu qatorni ko‘rgan foydalanuvchi 1 va 10 oralig‘idagi sonni kiritishi, keyin Enter tugmasini bosishi kerakligini tushunadi. Agar shu yozuvlar foydalanuvchiga taqdim etilmasa, foydalanuvchi nimani kiritishini aniq

bilmasligi mumkin. Shuning uchun, odatda, foydalanuvchi kiritishi lozim bo‘lgan qiymatdan oldin foydalanuvchiga ma’lumot chiqarish orqali dastur tuziladi.

Dasturchi yozayotgan dastur faqat uning o‘zi uchun emas, balki boshqalar uchun ham tushunarli bo‘lishi kerak. Shuning uchun dasturchi dasturini hujjatlashtirib tuzishi kerak. Yaxshi hujjatlashgan dasturni birinchi yozilganidan uzoq vaqt o‘tgandan keyin ham tushunish va o‘zgartirish oson bo‘ladi. Bunda hujjatlashgan identifikatorlar yoki izohlardan foydalanish mumkin. Izohlar dastur qismining maqsadi-mazmunini, dasturchi ismini (kim yozganligini) va turli o’ziga xos ma’lumotlarini o’z ichiga olishi kerak. Shuningdek, dastur matnini ham qandaydir qoidalarga asoslanib tuzish maqsadga muvofiq. Masalan, o‘zgaruvchilarni e’lon qilishda:

```
int gramm, tonna;
double x, y;
```

va

```
int gramm,tonna;double x,y;
```

Ikkala ko‘rinish ham xatosiz, to‘g‘ri yozilgan. Ularni tushunishda kompyuter qiyinchilikka duch kelmaydi. Ammo dasturchi uchun birinchi ko‘rinish tushunarliroq. Sababi, o‘zgaruvchilarning turlari alohida ajratilgan, o‘zgaruvchilar ham probel yordamida ajratib yozilgan.

Noto’g’ri formatlangan C++dasturiga misol.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
int son; double vazn;
string ism;
cout << "Son kiriting: "; cin >>son; cout << endl;
```

```
cout<<"son: "<<son<<endl;
cout<<"Ismingizni kiriting: "; cin>> ism;
cout<<endl; cout <<"Vazningizni kiriting: ";cin>>vazn; cout<<endl;
cout<<"Ismingiz: "<<ism<<endl;cout<<"Vazningiz: "
<<vazn; cout <<endl;return 0;
}
```

To’g’ri formatlangan C++dasturi.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
int son;
double vazn;
string ism;
cout << "Son kiriting: ";
cin >>son;
cout << endl;
cout << "son: " <<son<< endl;
cout << "Ismingizni kiriting: ";
cin >>ism;
cout << endl;
cout << "Vazningizni kiriting: ";
cin >>vazn;
cout << endl;
cout << "Ismingiz: " <<ism<< endl;
cout << "Vazningiz: " <<vazn<< endl;
return 0;
```

}

Ushbu ikki dasturdan ko‘rinib turibdiki, noto‘g‘ri formatlangan dasturni tushunish to‘g‘ri formatlangan dasturni tushunishga qaraganda qiyin.

Nazorat savollari

1. C++ ko‘rsatmalari qanday belgi bilan tugallanishi zarur?
2. Tilning ma’nosini beruvchi qoidalar to‘plami qanday nomlanadi?
3. SANTIMETR_UCHUN_MILLIMETR ko‘rinishidagi identifikatorlar qanday ataladi?
4. int turidagi son o‘zgaruvchisi uchun dastur qismi ko‘rsatib bering.
5. Agar dasturda sintaktik xatolar bo‘lsa, kompilyator bu haqida xabar beradimi?
6. Arifmetik operatorlar bajarilish ketma-ketligi qoidasi qanday qoida?
7. Figurali qavslar nima uchun ishlatiladi?
8. Vergul (“,”) odatda nima uchun ishlatiladi?
9. “double” kalit so‘zi nima uchun ishlatiladi?
10. Haqiqiy son turi nima?

§3. Berilganlar turlari. C++ tilining tayanch turlari

Berilganlarning oddiy turlari

C++ tilining maqsadi berilganlar ustida amallar bajarish orqali ularni boshqarish hisoblanadi. Turli dasturlar turli ko‘rinishdagi berilganlar bilan ishlaydi. Masalan, oylik ish haqini hisoblash dasturida berilganlar qo‘shish, ayirish, ko‘paytirish, bo‘lish amallari orqali ishlanadi, berilganlar ishlagan soat hajmi, ish stavkasi ko‘rinishida aniqlanadi. Shu kabi, talabalarning familiyalarini alfavit bo‘yicha tartiblash dasturi familiyalar ko‘rinishidagi berilganlar bilan ishlaydi. Bu ikki dastur ikki xil ko‘rinishdagi berilganlar bilan ishlaydi, birinchisi sonlar ustida arifmetik amallar bajarsa, ikkinchisi satr ko‘rinishidagi berilganlar ustida amallar bajaradi. Tabiiyki, bu ikkala masalani bir xil amallar orqali yechib bo‘lmaydi. Shuningdek, familiya ko‘rinishidagi berilganlar ustida ko‘paytirish yoki ayirish amalini bajarishning imkoni mavjud emas. Shularni

hisobga olgan holda, C++ tili berilganlarni turlarga ajratadi. Har bir berilganlar turi ustida shu tur uchun oldindan aniqlangan amallarni bajarish mumkin.

C++ tilida berilganlar uchta kategoriyaga ajratiladi:

1. Berilganlarning oddiy turlari (tayanch turlari).
2. Berilganlarning hosilaviy turlari.
3. Ko‘rsatkichlar.

Berilganlarning oddiy turlari C++ tilining tayanch turlari hisoblanadi va ular berilganlarning hosilaviy turlarini qurishda asos bo‘lib xizmat qiladi. Berilganlarning oddiy turlari uchta kategoriyaga bo‘linadi:

1. *Butun son turlari* – butun sonlar bilan ishlovchi turlar yoki haqiqiy qismi bo‘lmagan sonlar turlari.

2. *Suzuvchi nuqtali turlar* – haqiqiy turdagi sonlar ko‘rinishidagi berilganlar bilan ishlovchi turlar.

3. *Sanab o‘tiluvchi tur* – foydalanuvchi aniqlagan berilganlar turi.

Butun son turlari quyidagi to‘qqizta berilganlar turlaridan tashkil topgan: char, short, int, long, bool, unsigned char, unsigned short, unsigned int va unsigned long.

Nima sababdan berilganlarning oddiy turlarining ko‘rinishlari ko‘p degan savol paydo bo‘lishi tabiiy. Har bir tur unga ajratilgan xotira o’lchamidan va qiymatni ifodalash formatidan kelib chiqqan holda o‘ziga xos xususiyatlarga ega bo’ladi. Masalan, char berilganlar turi -128 va 127 oralig‘idagi sonlarni ifodalash uchun ishlatiladi, shuningdek bu tur yordamida belgilarni ekranga chiqarishda foydalanish mumkin. short berilganlar turi -32768 va 32767 oralig‘idagi sonlarni ifodalash uchun ishlatiladi.

Butun son turlari, ularning baytlardagi o‘lchamlari va qiymatlarining chegaralari quyidagi jadvalda keltirilgan.

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
unsignedint (16 razryadli)	2	0..65535
unsignedint (32 razryadli)	4	0..42949667295
unsignedchar	1	0..255
char	1	-128..127

Butun son qiymatlarni qabul qiladigan o‘zgaruvchilar asosan 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. C++ tilida butun sonlar xuddi matematikadagi kabi ko‘rinishda bo‘ladi:

-6728, -13, 0, 47, +485.

Butun sonlar quyidagi o‘nlik, sakkizlik va o‘n oltilik formatlarda bo‘ladi.

O‘nlik o‘zgarmas 0 va 0 raqamidan farqli raqamdan boshlanuvchi raqamlar ketma-ketligi. Masalan: 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‘zgarmas raqamlari oxiriga, probelsiz l yoki L (long), u yoki U (unsigned) yoziladi. Zarur hollarda bitta o‘zgarmas uchun bu belgilarning ikkitasini ham ishlatish mumkin: 45lu, 012Ul, 0xA2L.

Butun sonlar bilan ishlaganda agar son musbat bo‘lsa, “+” belgisini qo‘yish shart emas.

Belgi turi

Belgi turidagi o‘zgaruvchilar char kalit so‘zi bilan beriladi va ular o‘zida belgining ASCII kodini saqlaydi. Ya’ni, kompilyator belgining kodini saqlaydi, chop etayotganda esa belgi ko‘rinishida ekranga chiqariladi. Belgi turidagi qiymatlar nisbatan murakkab bo‘lgan tuzilmalar – satrlar, belgilar massivi va hakozalarni hosil qilishda ishlatiladi. Belgi o‘zgarmaslar “ ’ ” – apostroflar ichiga olingan alohida belgilardan tashkil topadi. Belgi o‘zgarmas uchun xotirada bir bayt joy ajratiladi va unda butun son ko‘rinishidagi belgining ASCII kodi joylashadi. Quyidagilar belgi o‘zgarmaslarga misol bo‘ladi: ’e’, ’@’, ’7’, ’z’, ’W’, ’+’, ’*’, ’a’, ’$’. Apostroflar orasida faqat bitta belgi joylashishi kerak. ’abc’ belgi hisoblanmaydi. C++ dasturda bunday ko‘rinishda ishlatilganda kompilyator xatolik haqida xabar beradi.

Ayrim belgi o‘zgarmaslar ‘\’ belgisidan boshlanadi, bu belgi birinchidan, grafik ko‘rinishga ega bo‘lmagan o‘zgarmaslarni belgilaydi, ikkinchidan, maxsus vazifalar yuklangan belgilar: apostrof ('), so‘roq (?), teskari yon chiziq (\) va qo‘shtirnoq (“) belgilarini 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 ham mumkin. Bunday belgidan boshlangan belgilar escape ketma-ketliklar deyiladi.

C++ tilida escape-belgilar jadvali

escape-belgilar	Ichki kod (16 lik son)	Belgi	Amal
\\	0x5C	\	Teskari yon chiziqni chop etish
\’	0x27	‘	Apostrofnichop etish
\”	0x22	“	Qo‘shtirnoqni chop etish
\?	0x3F	?	So‘roq belgisi
\a	0x07	bel	Tovush signalini berish
\b	0x08	bs	Kursorni 1 belgi orqaga qaytarish
\f	0x0C	ff	Sahifani o‘tkazish

\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
\000	000		Belgi sakkizlik kodi bilan berilganda
\xNN	0xNN		Belgi o‘n oltilik kodi bilan berilganda

Misol:

```
#include <iostream>
using namespace std;
int main()
{
char belgi=99;
cout<< ”Belgi:\t” << belgi << ‘\n’<<”Dastur tugadi!”;
return 0;
}
```

Ushbu dastur ishga tushganda ekranga quyidagi natija chiqadi:

```
Belgi: s
Dastur tugadi!
```

Mantiqiy tur

Bu turdagi o‘zgaruvchi bool kalit so‘zi bilan e’lon qilinadi va xotiradan 1 bayt joy band qiladi va 0 (false, yolg‘on) yoki 0 qiymatidan farqli qiymat (true, rost) qabul qiladi. Mantiqiy turdagi o‘zgaruvchilar qiymatlar o‘rtasidagi munosabatlarni 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*”, “*London-Italiya poytaxti*” va hokazo. Lekin “*0.000001- kichik son*” jumlasi mulohaza hisoblanmaydi, chunki “*kichik son*” tushunchasi juda

ham nisbiy, ya’ni kichik son deganda qanday sonni tushunish kerakligi aniq emas. Shuning uchun yuqoridagi jumlani rost yoki yolg‘onligi haqida fikr bildirish qiyin.

Mulohazalarning rostligi holatlarga bog‘liq ravishda o‘zgarishi mumkin. Masalan “*bugun – chorshanba*” jumlasini rost yoki yolg‘onligi ayni qaralayotgan kunga bog‘liq. Xuddi shunday “$x<0$” jumlasi x o‘zgaruvchisining ayni paytdagi qiymatiga mos ravishda rost yoki yolg‘on bo‘ladi.

C++ tilida mantiqiy tur nomi angliyalik matematik Jorj Bul sharafiga bool so‘zi bilan ifodalangan.

Haqiqiy son turi

C++ tilida haqiqiy sonlar bilan ishlash uchun suzuvchi nuqtali berilganlar turlari mavjud. Haqiqiy o‘zgarmaslar – 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‘zgarmaslarga misollar: 24.56; 13.0; 66.; .87;

– eksponensial shaklda haqiqiy o‘zgarmas 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‘zgarmas);
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‘zgarmas sonlarga misollar: 1e2; 5e+3; .25e4; 31.4e-1.

Haqiqiy sonlar va ularning eksponensial ko‘rinishlari bilan quyidagi jadval orqali tanishish mumkin:

Haqiqiy son	Eksponensial ko‘rinish
75.924	7.592400E1
0.18	1.800000E-1

0.0000453	4.530000E-5
-1.482	-1.482000E0
7800.0	7.800000E3

Haqiqiy sonlar float yoki double kalit so‘zi bilan e’lon qilinadi. Bu turdagi o‘zgaruvchi uchun xotirada 4 bayt joy ajratiladi va

<*ishora*>	<*tartib*>	<*mantissa*>

qolipida sonni saqlaydi. Agar kasrli son juda katta (kichik) qiymatlarni qabul qiladigan bo‘lsa, u xotirada 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.

Tur nomi	Baytlardagi o‘lchami	Qiymat chegarasi
float	4	-3.4E+38..3.4E+38
double	8	-1.7E+308..1.7E+308
long double (32 razryadli)	10	-3.4e-4932..3.4e4932

```
#include <iostream>
using namespace std;
int main()
{
const double pi=3.1415;
const int Radius=3;
double Yuza;
Yuza=pi*Radius*Radius;
cout<<Yuza<<‘\n’;
cout<<”Dastur tugadi!”;
return 0;
}
```

Nazorat savollari

1. Berilganlarning strukturalashgan turlari qanday?
2. Butun son turlari nima?

3. Suzuvchi nuqtali turlar nima?
4. Sanab o‘tiluvchi tur nima?
5. Haqiqiy sonlar qanday kalit so‘zi bilan e’lon qilinadi?
6. O‘nlik fiksirlangan nuqtali format deganda nimani tushuniladi?
7. Eksponensial shaklda haqiqiy o‘zgarmas necha qismdan iborat bo‘ladi va ularga misol ko‘rsating.
8. Harf belgilar qanday registrlarda berilishi mumkin?
9. Kompilyator nimaga qarab unga mos turni belgilaydi?

§4. O‘zgaruvchilar va ifodalar

Tayanch arifmetik amallar

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‘linmaning qoldig‘ini olish (%) amallarini keltirish mumkin. Qo‘shish, ayirish, ko‘paytirish, bo‘lish amallarini butun va haqiqiy turdagi sonli berilganlar bilan ishlatish mumkin. Bo‘linmaning qoldig‘ini olish amali faqat butun turdagi sonli berilganlar bilan ishlatiladi. Shuningdek, bo‘lish amalini butun sonli berilganlar ustida amalga oshirilsa, natija sifatida bo‘lishning butun qismi qaytariladi.

a. -5

b. 8 - 7

c. 3 + 4

d. 2 + y * 5

e. 5.6 + 6.2 * 3

f. x + 2 *5 + 6 / y

Yuqorida keltirilgan namunalarda x va y noma’lum sonlar. Arifmetik ifoda arifmetik operatorlar (amallar) va sonlardan tuziladi. Ifodada qatnashgan

sonlar (noma’lum sonlar ham) operand deb ataladi. a-namunada “–” amali 5 sonining manfiyligini aniqlash uchun ishlatilmoqda. Bu ifodada bitta operand mavjud. Faqatgina bitta operandi bor bo‘lgan arifmetik operator (amal) unar operator deb ataladi. b-namunada “–” amali 8 sonidan 7 sonini ayirish uchun ishlatilmoqda. Bu arifmetik ifodada “–” amalining ikkita operandi bor – 8 va 7. Ikkita operandi mavjud operatorlar binar operatorlar deyiladi.

```
#include <iostream>
using namespace std;
int main()
{
  cout << "2 + 5 = " << 2 + 5 << endl;
  cout << "13 + 89 = " << 13 + 89 << endl;
  cout << "34 - 20 = " << 34 - 20 << endl;
  cout << "45 - 90 = " << 45 - 90 << endl;
  cout << "2 * 7 = " << 2 * 7 << endl;
  cout << "5 / 2 = " << 5 / 2 << endl;
  cout << "14 / 7 = " << 14 / 7 << endl;
  cout << "34 % 5 = " << 34 % 5 << endl;
  cout << "4 % 6 = " << 4 % 6 << endl;
  cout << "5.0 + 3.5 = " << 5.0 + 3.5 << endl;
  cout << "3.0 + 9.4 = " << 3.0 + 9.4 << endl;
  cout << "16.3 - 5.2 = " << 16.3 - 5.2 << endl;
  cout << "4.2 * 2.5 = " << 4.2 * 2.5 << endl;
  cout << "5.0 / 2.0 = " << 5.0 / 2.0 << endl;
  cout << "34.5 / 6.0 = " << 34.5 / 6.0 << endl;
  cout << "34.5 / 6.5 = " << 34.5 / 6.5 << endl;
  return 0;
}
```

Natija:

2 + 5 = 7

13 + 89 = 102

34 - 20 = 14

45 - 90 = -45

2 * 7 = 14

5 / 2 = 2

14 / 7 = 2

34 % 5 = 4

4 % 6 = 4

5.0 + 3.5 = 8.5

3.0 + 9.4 = 12.4

16.3 - 5.2 = 11.1

4.2 * 2.5 = 10.5

5.0 / 2.0 = 2.5

34.5 / 6.0 = 5.75

34.5 / 6.5 = 5.30769

Ikkala operand ham butun son bo‘lganda (5/2) arifmetik ifoda uchun bo‘lish amalining natijasi butun qismni olish uchun ishlaydi. Agar natija sifatida haqiqiy sonni qaytarish kerak bo‘lsa, operandlarning kamida bittasi haqiqiy son turida bo‘lishi kerak, 5.0/2.0, 5./2, 5/2. ko‘rinishlarida ishlatish mumkin. Agar sonning kasr qismi bo‘lmasa, lekin uni haqiqiy son sifatida ishlatish kerak bo‘lsa, 5.0 yoki 5. ko‘rinishida yozish mumkin. Shu sababli ham natija 2 chiqqan. 34%5 ifodasida esa, 34 sonini 5 soniga butun sonli arifmetika qoyidalari bilan bo‘lganda, 6 butun va 4 qoldiq soni chiqadi, % amali qoldiq qismini qaytarishini inobatga olgan holda natija sifatida 4 soni ekranga chiqarilgan. Bo‘linmaning qoldig‘ini olish amalini manfiy sonlar bilan ishlatayotganda juda ehtiyotkor bo‘lish kerak.

Arifmetik ifodada bir nechta amallar qatnashganda kompilyator amallar ketma-ketligini ularning ustunligiga qarab bajaradi. Avval ko‘paytirish yoki bo‘lish (butun yoki bo‘linmaning qoldig‘ini olish) amallari,so‘ngra qo‘shish yoki ayirish amallari bajariladi. Masalan: 3*7-6+2*5/4+6 ifodasi uchun amallarning bajarilish ketma-ketligini quyidagicha tushunish mumkin:

3 * 7 - 6 + 2 * 5 / 4 + 6= (((3 * 7) – 6) + ((2 * 5) / 4)) + 6 (* amali bajariladi)

= ((21 – 6) + (10 / 4)) + 6 (/ amali bajariladi. Operandlar butun son)

= ((21 – 6) + 2) + 6 (– amali bajariladi)

= (15 + 2) + 6 (birinchi +amali bajariladi)

= 17 + 6 (+amali bajariladi)

= 23

Agar ifodadagi barcha operandlar butun sonlardan tashkil topgan bo‘lsa, bu ifoda butun sonli ifoda deyiladi. Agar ifoda operandlari haqiqiy sonlardan tashkil topgan bo‘lsa, haqiqiy sonli yoki suzuvchi nuqtali sonli ifoda deyiladi.

2 + 3 * 5

3 + a - b / 7

a + 2 * (b - c) + 18

12.8 * 17.5 - 34.50

x * 10.5 + y - 16.2

Agar ifodada ham butun turdagi, ham haqiqiy turdagi sonlar yoki o‘zgaruvchilar qatnashsa bunday ifoda aralash ifoda deyiladi.

2 + 3.5

6.3 / 4 + 3.9 – a / b

Aralash ifodada har bir operatorlar (amallar) alohida qism ifoda sifatida qarab hisoblanadi. Agar qism ifoda operandlari bir xil turda bo‘lsa, natija ham shu turda bo‘ladi, agar operandlari turlari farqli bo‘lsa, natija haqiqiy son ko‘rinishida bo‘ladi.

```
#include <iostream>
using namespace std;
```

```
int main()
{
cout << "3 / 2 + 5.5 = " << 3 / 2 + 5.5 << endl;
cout << "15.6 / 2 + 5 = " << 15.6 / 2 + 5 << endl;
cout << "4 + 5 / 2.0 = " << 4 + 5 / 2.0 << endl;
cout << "4 * 3 + 7 / 5 - 25.5 = "<< 4 * 3 + 7 / 5 - 25.5<< endl;
return 0;
}
```

Natija:

```
3 / 2 + 5.5 = 6.5
15.6 / 2 + 5 = 12.8
4 + 5 / 2.0 = 6.5
4 * 3 + 7 / 5 - 25.5 = -12.5
```

Aralash ifodada ifodaning natijasi kompilyator tomonidan hisoblanadi. Butun sonli qiymatlar kasr qismi nolga teng bo‘lgan haqiqiy songa aylantiriladi. Bu jarayon bir turni boshqa turga keltirish deyiladi. C++ tilida bir turni boshqa turga keltirishning oshkor va oshkormas yo‘llari mavjud.

Bir turni boshqa turga keltirish

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>
using namespace std;
```

```
int main()
{
int Integer_1=54, Integer_2;
float Floating=15.854;
Integer_1=(int)Floating;          // turga oshkor keltirish
Integer_2=Floating;               // turga oshkormas keltirish
cout << “Yangi Integer(Oshkor): ”<<Integer_1<<“\n”;
cout<<“YangiInteger(Oshkormas):”<<Integer_2<<”\n”;
return 0;
}
```

Dastur natijasi quyidagi ko‘rinishida bo‘ladi:

```
Yangi Integer(Oshkor): 15
Yangi Integer(Oshkormas): 15
```

Masala. Berilgan belgining ASCII kodi chop etilsin. Masala belgi turidagi qiymatni oshkor ravishda butun son turiga keltirib chop qilish orqali yechiladi.

Dastur matni:

```
#include <iostream.h>
int main()
{
 unsigned char A;
 cout << "Belgini kiriting: "; cin >> A;
 cout << '\'' << A << "'-belgi ASCII kodi=" << (int)A << '\n';
 return 0;
}
```

Dasturning

```
Belgini kiriting so‘roviga:
A ↵
```

amali bajarilsa, ekranga

'A'-belgi ASCII kodi=65

satri chop etiladi.

Shuningdek, cast operatori yordamida ham oshkor ravishda bir turni boshqa turga keltirish mumkin:

static_cast<dataTypeName>(expression),

bu yerda expression – qiymatini boshqa turga o‘tkazish lozim bo‘lgan ifoda, dataTypeName – ifodani o‘tkazish lozim bo’lgan tur nomi.

```
#include <iostream>
using namespace std;
int main()
{
cout << "static_cast<int>(7.9) = "<< static_cast<int>(7.9) << endl;
cout << "static_cast<int>(3.3) = "<< static_cast<int>(3.3) << endl;
cout << "static_cast<double>(25) = "<< static_cast<double>(25) << endl;
cout << "static_cast<double>(5 + 3) = "<< static_cast<double>(5 + 3) << endl;
cout << "static_cast<double>(15) / 2 = "<< static_cast<double>(15) / 2 << endl;
cout << "static_cast<double>(15 / 2) = "<< static_cast<double>(15 / 2) << endl;
cout << "static_cast<int>(7.8 + static_cast<double>(15) / 2) = "
<< static_cast<int>(7.8 + static_cast<double>(15) / 2)<< endl;
cout << "static_cast<int>(7.8 + static_cast<double>(15 / 2)) = "
<< static_cast<int>(7.8 + static_cast<double>(15 / 2))<< endl;
return 0;
}
```

Dastur natijasi quyidagi ko‘rinishida bo‘ladi:

```
static_cast<int>(7.9) = 7
static_cast<int>(3.3) = 3
static_cast<double>(25) = 25
static_cast<double>(5 + 3) = 8
```

static_cast<double>(15) / 2 = 7.5

static_cast<double>(15 / 2) = 7

static_cast<int>(7.8 + static_cast<double>(15) / 2) = 15

static_cast<int>(7.8 + static_cast<double>(15 / 2)) = 14

Shuningdek, cast operatori yordamida belgilarni boshqa turga keltirish mumkin. Bunda, belgining ASCII jadvalidagi kodi orqali boshqa turga keltiriladi, ya'ni:

static_cast<int>('A') => 65

static_cast<int>('8') => 56.

static_cast<char>(65) => 'A'

static_cast<char>(56) => '8'.

O‘zgarmaslar

C++ tilida tuzilgan dasturning asosiy maqsadi hisoblash ishlarini amalga oshirish va berilganlarni boshqarish hisoblanadi. Berilganlar ustida biror ish bajarilishidan oldin ular asosiy xotiradan joy olgan bo‘lishi kerak. Buning uchun o'zgarmaslar ishlatiladi.

O‘zgarmaslar const kalit so‘zi orqali

const dataType identifier = value;

ko'rinishida e'lon qilinadi, bu yerda dataType – tur nomi, identifier – identifikator va value – qiymat.

const double KATTALIK = 2.54;

const int TALABALAR_SONI = 20;

const char PROBEL = ' ';

O‘zgaruvchi – dastur obekti bo‘lib, xotiradagi bir nechta yacheykalarni 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 turiga bog‘liq bo‘ladi. Shuni

qayd etish zarurki, bitta turga turli apparat platformalarda turlicha joy ajratilishi mumkin.

Ifodalarda o‘zgaruvchilardan foydalanish

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.

```
dataType identifier, identifier, . . .;
double Yuza;
int soni;
char ch;
int x, y;
```

O‘zgaruvchilarga ifoda ‘=’ belgisi orqali yuklanadi. Bunda ifodaning natijaviy qiymatining turi o‘zgaruvchi turi bilan mos kelishi kerak. Aks holda, agar kompilyator turlar o‘rtasida bir turdan boshqa turga keltirishni amalga oshira olsa, ifodaning qiymat o‘zgaruvchining turiga o‘tkazib yuklanadi, aks holda kompilyatsiya xatoligi uyzaga keladi.

```
#include <iostream>
using namespace std;
int main()
{
int num1, num2, num3;
double qiymat;
char belgi;
num1 = 4;
cout << "num1 = " << num1 << endl;
num2 = 4 * 5 - 11;
cout << "num2 = " << num2 << endl;
```

```
num3 = 4.5 * 5 - 9;
cout << "num3 = " << num3 << endl;
qiymat = 0.02 * 1000;
cout << "qiymat = " <<qiymat<< endl;
belgi = 'D';
cout << "belgi = " <<belgi<< endl;
return 0;
}
```

Dastur natijasi quyidagi ko‘rinishida bo‘ladi:

```
num1 = 4
num2 = 9
num2 = 13
qiymat = 20
belgi = D
```

C++ tilida “num = num + 2;” ko‘rinishidagi ifoda num o‘zgaruvchisining qiymatini ikkitaga oshirish kerakligini bildiradi. Ifoda bajarilishidan oldin num o‘zgaruvchisiga qiymat berilgan bo‘lishi kerak. Aks holda dastur kutilmagan qiymat chop etadi. Sababi, o‘zgaruvchi e’lon qilinganda unga hech qanday boshlang‘ich qiymat berilmaydi, shu sababli kompilyator uchun num o‘zgaruvchisining boshlang‘ich qiymati mavjud emas.

```
int num1, num2, num3;  //1-qator
num1 = 18;             //2-qator
num1 = num1 + 27;      //3-qator
num2 = num1;           //4-qator
num3 = num2 / 5;       //5-qator
num3 = num3 / 4;       //6-qator
```

Yuqorida keltirib o‘tilgan dastur qismi kodidagi o‘zgaruvchilarning kompilyator uchun qanday ketma-ketlikda qiyamat olishlari quyidagi jadvalda keltirilgan:

Qator	O‘zgaruvchilarning qiymatlari	Izoh
1-qator	num1=? num2=? num3=?	
2-qator	num1=18 num2=? num3=?	
3-qator	num1=45 num2=? num3=?	num1 + 27 = 18 + 27 = 45. Natija num1 o‘zgaruvchisiga, uning oldingi qiymatining o‘rniga yoziladi.
4-qator	num1=45 num2=45 num3=?	num1 o‘zgaruvchisining qiymati num2 o‘zgaruvchisiga yuklanadi
5-qator	num1=45 num2=45 num3=9	num2 / 5 = 45 / 5 = 9. Natija num3 o‘zgaruvchisiga yuklanadi
6-qator	num1=45 num2=45 num3=2	num3 / 4 = 9 / 4 = 2. Natija num3 o‘zgaruvchisiga, uning oldingi qiymatining o‘rniga yoziladi.

Nazorat savollari

1. Aralash ifodada qanday hisoblanadi?
2. Berilgan belgining ASCII kodi chop etilsin. Masala belgi turidagi qiymatni qanday yechiladi?
3. Qaysi operator yordamida oshkor ravishda bir turni boshqa turga keltirish mumkin?
4. cast operatori yordamida belgilar boshqa turga keltirish mumkinmi?
5. C++ tilida tuzilgan dasturning asosiy maqsadi nima?
6. O‘zgaruvchi nima?
7. O‘zgaruvchilarga ifoda qanday belgi orqali yuklanadi?
8. C++ tilida num = num +2; ko‘rinishidagi ifoda nimani bildiradi?

9. Kod qismidagi o‘zgaruvchilarning kompilyator uchun qanday ketma-ketlikda qiymat olishlarini jadvalini yozing.

10. C++ tilida bir turni boshqa turga keltirishning qanday yo‘llari mavjud?

§5. Amallar. Amallarning ustunliklari va bajarilish yo‘nalishlari

Inkrement va dekrement amallari

C++ tilida operand qiymatini birga oshirish va kamaytirishning samarali vositalari mavjud. Bular inkrement (++) va dekrement (--) unar amallardir. Masalan, inkrement amalidan foydalanib

```
soni=soni+1;
```

ifoda kodi o‘rniga

```
soni++;
```

kodni yozish mumkin.

Inkrement amali operand qiymatini bittaga oshirish uchun ishlatiladi. Dekrement amali esa operand qiymatini bittaga kamaytirish uchun ishlatiladi. Operandga nisbatan bu amallarning ikki xil ko‘rinishi: prefiks va postfiks ko‘rinishlari mavjud. 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 inkrement: `++variable`

Postfiks inkrement: `variable++`

Prefiks dekrement: `--variable`

Postfiks dekrement: `variable--`

Prefiks yoki postfiks amal tushunchasi faqat qiymat berish bilan bog‘liq ifodalarda o‘rinli.

```
x = 5;
y = ++x;
```

Ushbu misolda x ning qiymati besh. y ga x ning qiymatini yuklashdan oldin prefiks inkrement amali qo‘llaniladi, x ning qiymati oltiga o‘zgaradi,

so‘ngray o‘zgaruvchisiga olti yuklanadi. Natijada x ning qiymati ham, y ning qiymati ham oltiga teng bo‘ladi.

```
x = 5;
y = x++;
```

Ushbu misolda x ning qiymati besh. y ga x ning qiymatini yuklash jarayonida postfiks inkrement amali qo‘llaniladi, x ning qiymati y o‘zgaruvchisiga yuklanadi, so‘ngra x ning qiymati bittaga oshiriladi. Natijada x qiymati 6, y qiymati 5 bo‘ladi.

Inkrement va dekrement amallarini murakkab ifodaning ichida ham ishlatish mumkin. Faqat bunda, tushunarli bo‘lishi uchun inkrement va dekrement amallarini qavs ichiga olish maqsadga muvofiq.

```
a = 5;
b = 2 + (++a);
```

Birinchi ifodada a o‘zgaruvchisiga besh soni yuklanadi. Keyin esa, 2 + (++a) ifodani bajarish jarayonida avval a o‘zgaruvchisining qiymati bittaga oshiriladi, so‘ngra uning qiymatiga ikki sonini qo‘shib, natija b o‘zgaruvchisiga yuklanadi. Natijada a ning qiymati olti, b ning qiymati sakkizga teng bo‘ladi.

```
a = 5;
b = 2 + (a--);
```

Birinchi ifodada a o‘zgaruvchisiga besh soni yuklanadi. Keyin esa, 2+(a--) ifodani bajarish jarayonida avval a o‘zgaruvchisining qiymati ifodaga qo‘yib hisoblanib, natija b o‘zgaruvchisiga yuklanadi, so‘ngra uning qiymati bittaga kamaytiriladi. Natijada a ning qiymati 4, b ning qiymati esa 7 ga teng bo‘ladi.

sizeof amali

Har xil turdagi o‘zgaruvchilar kompyuter xotirasida turli sondagi baytlarni egallaydi. Bunda, hattoki bir turdagi o‘zgaruvchilar ham qaysi kompyuterda yoki qaysi operatsion sistemada amal qilinishiga qarab turli o‘lchamdagi xotirani band qilishi mumkin.

C++ tilida ixtiyoriy (tayanch va hosilaviy) turdagi o‘zgaruvchilarning o‘lchamini sizeofamali yordamida aniqlanadi. Bu amalni o‘zgarmasga, turga va o‘zgaruvchiga qo‘llash mumkin.

Quyida keltirilgan dasturda kompyuterning platformasiga mos ravishda tayanch turlarining o‘lchamlari chop qilinadi.

```
#include <iostream>
using namespace std;
int main()
{
  short s=-12;
  double d=345.678;
  unsigned long ul=123456789;
  cout << "Size of char = " << sizeof(char) << endl;
  cout << "Size of int = " << sizeof(int) << endl;
  cout << "Size of short = " << sizeof(s) << endl;
  cout << "Size of unsigned int = "<< sizeof(unsigned int)<< endl;
  cout << "Size of long = " << sizeof(long) << endl;
  cout << "Size of bool = " << sizeof(bool) << endl;
  cout << "Size of float = " << sizeof(float) << endl;
  cout << "Size of double = " << sizeof(double) << endl;
  cout << "Size of long double = " << sizeof(d) << endl;
  cout << "Size of unsigned short = "<< sizeof(unsigned short) << endl;
  cout << "Size of unsigned long = "<< sizeof(ul) << endl;
  return 0;
}
```

Dasturning natijasi quyidagicha ko‘rinishga ega:

```
Size of char = 1
Size of int = 4
```

Size of short = 2

Size of unsigned int = 4

Size of long = 4

Size of bool = 1

Size of float = 4

Size of double = 8

Size of long double = 8

Size of unsigned short = 2

Size of unsigned long = 4

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 ishlatsa 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 yetarli 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
×	×	×	1	1	0	0	1

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

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.

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_taxtasi=0;
```

Bu yerda q_taxtasi 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_taxtasi=q_taxtasi|0x04;
```

amalini bajarish kerak, chunki $0x04_{16}=00000100_2$ va mantiqiy YOKI amali natijasida q_taxtasi o‘zgaruvchisi bayti quyidagi ko‘rinishda bo‘ladi:

7	6	5	4	3	2	1	0
0	0	0	0	0	1	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=11111101₂
```

Xuddi shu natijani ‘~’ amalidan foydalangan holda ham olish mumkin. Ikkinchi xona tizimga ulanganligini bildiruvchi bayt qiymati - 00000010_2, 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
A=A>>3;                  //A=00000110₂=0x0616=610
```

Razryadlarni n ta chapga (o‘nga) surish sonni 2^n soniga ko‘paytirish (bo‘lish) amali bilan ekvivalent bo‘lib, biroq nisbatan tez bajariladi. Shuni

e’tiborga olish kerakki, operand ishorali son bo‘lsa, u holda o‘ngga 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; //$B=10001000_2=0x88_{16}$

B=B<<2; //$B=00100000_2=0x20_{16}=32_{10}$

B=-120; // $B=10001000_2=0x88_{16}$

B=B>>3; // $B=11110001_2=0xF1_{16}=-15_{10}$

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

Taqqoslash amallari

C++ tilida qiymatlarni solishtirish uchun taqqoslash amallari aniqlangan. Taqqoslash amali binar amal bo‘lib, quyidagi ko‘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.

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”

Amallarning ustunliklari va bajarilish yo‘nalishlari

An’anaviy arifmetikadagidek C++ tilida ham amallar ma’lum bir tartib va yo‘nalishda bajariladi. Ma’lumki, matematik ifodalarda bir xil ustunlikdagi (prioritetdagi) amallar uchrasa (masalan, qo‘shish va ayirish), ular chapdan

o‘ngga bajariladi. Bu tartib C++ tilida ham o‘rinli, biroq ayrim hollarda amal o‘ngdan chapga bajarilishi mumkin (xususan, qiymat berish amalida).

Ifodalar qiymatini hisoblashda amallar ustunligi hisobga olinadi. Birinchi navbatda eng yuqori ustunlikka ega bo‘lgan amal bajariladi.

Quyidagi jadvalda C++ tilida ishlatiladigan amallar (operatorlar), ularning ustunlik darajalari va bajarilish yo‘nalishlari (‘⇐’ – o‘ngdan chapga, ‘⇒’ – chapdan o‘ngga) keltirilgan.

Amallarning ustunliklari va bajarilish yo‘nalishlari

Operator	Tavsifi	Ustunlik	Yo‘nalish
::	Ko‘rinish sohasiga ruxsat berish	16	⇒
[]	Massiv indeksi	16	⇒
()	Funksiyanga murojaat	16	⇒
.	Struktura yoki sinf elementini tanlash	16	⇒
->			
++	Postfiks inkrement	15	⇐
--	Postfiks dekrement	15	⇐
++	Prefiksinkrement	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 obektni yaratish	14	⇐
delete	Dinamik obektni yo‘q qilish	14	⇐

casting	Turga keltirish	14	
*	Ko‘paytirish	13	⇒
/	Bo‘lish	13	⇒
%	Bo‘linmaning 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‘lishqiymat berish bilan	2	⇐
%=	Bo‘linmaning qoldigini olish qiymat berish bilan	2	⇐
+=	Qo‘shish qiymat berish bilan	2	⇐
- =	Ayirish qiymat berish bilan	2	⇐
<<=	Chapga surishqiymat berish bilan	2	⇐

>>=	O‘ngga surishqiymat 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 guruhlarga jamlash mumkin. Qavs ishlatishga cheklov yo‘q.

Quyidagi dasturda qavs yordamida amallarning bajarilish tartibini o‘zgartirish ko‘rsatilgan.

```
#include <iostream>
using namespace std;
int main()
{
 int x=0, y=0, a=3, b=34, c=82;
 x=a*b+c;
 y=(a*(b+c));
 cout<<”x= ”<<x<<’\n’<<”y= ”<<y<<’\n’;
}
```

Dasturda amallar ustunligiga ko‘ra x qiymatini hisoblashda oldin a o‘zgaruvchi b o‘zgaruvchiga ko‘paytiriladi va unga c o‘zgaruvchi qiymati qo‘shiladi. Navbatdagi ko‘rsatmani bajarishda esa birinchi navbatda ichki qavs ichidagi ifoda – (b+c) qiymati hisoblanadi, keyin bu qiymat a o’zgaruvchisiga ko‘paytirilib, y o‘zgaruvchisiga o‘zlashtiriladi. Dastur bajarilishi natijasida ekranga

```
x=184
y=348
```

satrlari chop etiladi.

Nazorat savollari

1. Inkrement va dekrement amallari nima?
2. Prefiks yoki postfiks amal tushunchasi qanday ifodalarda o‘rinli?
3. C++ tilida ixtiyoriy (tayanch va hosilaviy) turdagi o‘zgaruvchilarning o‘lchamini qanday amali yordamida aniqlanadi?
4. Qo‘yilgan masalani yechishda biror holat ro‘y bergan yoki yo‘qligini ifodalash uchun 0 va 1 qiymat qabul qiluvchi nimalardan foydalaniladi?
5. C++ tilida bayt razryadlari ustida mantiqiy amallar majmuasi jadvalini ko‘rsating.
6. 3-xona indikatorini, uni qanday qiymatda bo‘lishidan qat’iy nazar qarama-qarshi holatga o‘tkazishni qaysi amal yordamida bajarish mumkin?
7. Baytdagi bitlar qiymatini chapga yoki o‘ngga surish uchun, mos ravishda qaysi amallari qo‘llaniladi?
8. Taqqoslash amali qanday amal bo‘lib, u qanday ko‘rinishga ega?
9. Taqqoslash amallarining natijasi - taqqoslash o‘rinli bo‘lsa yoki o‘rinli bo‘lmasa qanday qiymat bo‘ladi?
10. Ifodalar qiymatini hisoblashda nima hisobga olinadi?

§6. O‘qish-yozish oqimlari (cin, cout)

Oqimlar

Dastur uchta asosiy qismdan tashkil topgan, berilganlarni o‘qish, berilganlarni manipulyatsiya qilish, natijalarni chiqarish. Oldingi mavzularda sonli berilganlarni boshqarish va ular ustida arifmetik amallar bajarishni ko‘rdik. Keyingi mavzularimizda sonli bo‘lmagan berilganlarni boshqarish haqida gapiriladi. Modomiki berilganlarni o‘qish va natijalarni chop qilishda bir oz muammoga duch kelinadigan bo‘lsa, C++ tilida berilganlarni o‘qish va chop qilishning keng imkoniyatlari mavjud.

<iostream> sarlavha fayli o’z ichiga kiritish-chiqarish oqimini boshqarish uchun standart obyektlar to‘plamini oladi. Unda klaviaturadan kiritish cin va

ekranga chiqarish uchun cout satndart-obyektlari, hamda “<<” – oqimdan o‘qish, “>>” – oqimiga joylashtirish amallari joylashtirilgan.

Ikki turdagi oqimlar mavjud:

input stream – ilovadan berilganlarning belgilar ketma-ketligini o’qish;

output stream – ilovaga berilganlarning belgilar ketma-ketligini yozish.

Standatr holda berilganlar dastur ilovasi orqali klaviaturadan kiritiladi va natija dastur ilovasiga chiqariladi. Klaviatura orqali kiritilayotgan berilganlar ketma-ketligini qabul qilish va ekranga chiqarish uchun, har bir C++ dastur sarlavhasida iostream faylidan foydalanishi kerak. iostream fayli ikkita oqimdan tashkil topgan, istream – berilganlarni kiritish oqimi va ostream – berilganlarni chiqarish oqimi. Shuningdek sarlavha fayli ikkita operatordan cin – berilganlarni oqimdan o’qish (kiritish) va cout – berilganlarni oqimga yozish (chiqarish).

Bu operatorlar o‘zgaruvchiga o‘xshaydi va quyidagicha tashkil topgan:

```
istream cin;
ostream cout;
```

cin va cout operatorlaridan foydalanish uchun, tuzilayotgan dastur sarlavhasida

```
#include <iostream>
```

preprotsessor direktivasini yozish kerak bo’ladi.

O‘qish oqimi (cin)

istream turi o‘zgaruvchilarni kiritish, ostream obyekti o‘zgaruvchilarni chiqarish oqimi deyiladi. Bundan tashqari, astream obyekti o‘zgaruvchilarni kiritish chiqarishning ixtiyoriy oqimi deyiladi.

cin kalit so‘zidan berilganlarni inputdevice standartidan olish uchun foydalaniladigan operatorlardan va funksiyalardan foydalana olishi mumkin. Ilovadan kiritilgan berilganlarni olish uchun “>>” belgilashidan foydalaniladi.

O‘zgaruvchiga kiritish oqimidan qiymat kiritish quyidagicha amalga oshiriladi:

```
cin >> soni;
```

Kompilyator bu ko‘rsatmani bajarayotganda kiritish oqimidan berilganni olib, xotiradagi soni o‘zgaruvchisida saqlaydi. Shuning uchun foydalanuvchi klaviaturadan 15.50 qiymatini kiritsa soni o‘zgaruvchisining qiymati 15.50 ga teng bo‘ladi;

O‘zgaruvchilarni kiritish belgisi (“>>”) ikkita operanddan tashkil topgan. Operatorning chap tomonida cin kiritish oqimi va operatorning o‘ng tomonida o‘zgaruvchining nomi bo‘ladi.

Berilganlarni oqimdan o‘qish belgisining sintaksisi quyidagicha:

```
cin >> variable >> variable...;
```

Bir nechta o‘zgaruvchiga ham kiritish oqimidan berilganlarni kiritish mumkin, masalan:

```
cin >> soni >> vazni;
```

Shuningdek, yuqoridagi belgilarni quyidagicha yozish ham mumkin:

```
cin >> soni;
cin >> vazni;
```

Bu misolda berilganlar oqimidan oldin soni o‘zgaruvchisiga qiymat kiritiladi va yangi qatorga tushganda vazni o‘zgaruvchisiga qiymat kiritiladi yoki aksincha qora ekranda bitta qatorda kiritilayotgan qiymatlar probel orqali kiritiladi.

```
cin >> soni >> vazni;
```

Quyidagi ko‘rinishda qiymatlar kiritilishi mumkin:

15 48.30↵

yoki:

15↵

48.30↵

Kiritish operatori soni o‘zgaruvchisiga 15 qiymatini va vazni o‘zgaruvchisiga 48.30 qiymatlarini o‘zlashtiradi

O‘zgaruvchiga berilishi mumkin bo‘lgan qiymatlar quyidagi jadvalda keltirilgan:

O‘zgaruvchining turi	Qabul qilish qiymati
char	Probeldan boshqa bitta belgini qabul qiladi
int	Butun turdagi ixtiyoriy qiymatni qabul qiladi
double	Haqiqiy turdagi ixtiyoriy qiymatni qabul qiladi

Faraz qilaylik, o‘zgaruvchilarning turi quyidagicha e’lon qilingan bo‘lsin:

```
int a, b;
double z;
char ch;
```

O‘zgaruvchilarni cin operatori yordamida >> operatoridan foydalanib o‘qish va qiymatlarni konsol (*consol*) ilovadan kiritish quyidagicha amalga oshiriladi:

№	Qiymatlarni kiritish	Qiymatlarni kiritish tartibi ko‘rinishlari	Xotiradagi o‘zgaruvchilar qiymatlari
	cin >> ch;	A ↵	ch=‘A’
	cin >> ch;	AB ↵	ch=‘A’, ch=‘B’
	cin >> a;	48 ↵	a=48
	cin >> a;	46.35 ↵	a=46
	cin >> z;	74.35 ↵	z=74.35
	cin >> z;	39 ↵	z=39.0
	cin >> z >> a;	65.78 38 ↵	z=65.78, a=38
	cin >> a >> b;	4 60 ↵	a=4, b=60
	cin >> a >> z;	46 32.4 68 ↵	a=46, z=35.4,

Agar o‘zgaruvchilarning turi quyidagicha e’lon qilingan bo‘lsa:

```
int a;
double z;
char ch;
```

O‘zgaruvchilarni cin operatoridan foydalanib o‘qish va qiymatlarni konsol ilovadan kiritish quyidagicha amalga oshiriladi:

№	Qiymatlarni kiritish	Qiymatlarni kiritish tartibi ko‘rinishlari	Xotiradagi o‘zgaruvchilar qiymatlari
	cin >> a >> ch >> z;	57 A 26.9 ↵ (Enter)	a=57,ch=’A’, z=26.9
	cin >> a >> ch >> z;	57 A ↵ 26.9 ↵	a=57,ch=’A’, z=26.9
	cin >> a >> ch >> z;	57 ↵ A ↵ 26.9 ↵	a=57,ch=’A’, z=26.9
	cin >> a >> ch >> z;	57A26.9 ↵	a=57,ch=’A’, z=26.9

Ko‘rinib turibdiki yuqoridagi 1-4 hollarda qiymatlarni o‘qib olish bir xil, faqatgina qiymat kiritish har xil. 1-holda qiymatlar bitta qatorda probel yordamida ajratib kiritilmoqda; 2-holda qiymatlar ikkita qatorda kiritilmoqda, ulardan birinchi ikkitasi probel bilan ajratib bitta qatorda uchinchi qiymat yangi qatordan kiritilmoqda; 3- holda barcha qiymatlar yangi qatordan kiritilmoqda; 4-holda barcha qiymatlar probel bilan ajratilmay qo‘shib kiritilmoqda. Bunda “**>>**” operatori avval 57 sonini oqimdan ajratib olib a ga beradi va belgi uchraganda uni belgi turidagi ch o‘zgaruvchisiga beradi, 26.9 ni z o‘zgaruvchisiga beradi.

Agar o‘zgaruvchilarning turi

```
int a, b;
double z;
char ch, ch1, ch2;
```

ko‘rinishda e’lon qilinsa, o‘zgaruvchilarni cin operatori yordamida >> operatoridan foydalanib o‘qish va qiymatlarni konsol ilovadan kiritish quyidagicha amalga oshiriladi:

№	Qiymatlarni kiritish	Qiymatlarni kiritish tartibi ko‘rinishlari	Xotiradagi o‘zgaruvchilar qiymatlari
1.	cin >> z >> ch >> a;	36.78B34 ↵	z=36.78 ch=’B’, a=34
2.	cin >> z >> ch >> a;	36.78 ↵ B34 ↵	z=36.78 ch=’B’, a=34
3.	cin >> z >> ch >> a;	11 34 ↵	a=11 , b=34
4.	cin >> a >> z;	78.49 ↵	a=78 z=0.49
5.	cin >> ch >> a;	256 ↵	ch=’2’ , a=56
6.	cin >> a >> ch;	256 ↵	a=256, kompyuter ch o’zgaruvchisi qiymati kiritilishini kutadi
7.	cin >> ch1 >> ch2 >> a;	A B ↵	ch1=’A’, ch=’B’

Quyida misol sifatida keltirilgan dasturda, dastur kodidan oldin sarlavha fayllari aniqlab olingan. Bir nechta matematik amallardan foydalanish uchun cmath sarlavha fayli va satrlar ustida amallar bajarish uchun string sarlavha fayli qo‘shilgan, length funsiyasi string turidagi satrning uzunligini aniqlab beradi:

```
#include <iostream>
#include <cmath>
#include <string>
using namespace std;
int main()
{
double u, v;
string str;
cout << "Line 1: 2 to the power of 6 = "<< static_cast<int>(pow(2.0, 6.0))<< endl;
u = 12.5;
v = 3.0;
```

```
cout << "Line 4: " << u << " to the power of "<< v << " = " << pow(u, v) << endl;
cout << "Line 5: Square root of 24 = "<< sqrt(24.0) << endl;
u = pow(8.0, 2.5);
cout << "Line 7: u = " << u << endl;
str = "Programming with C++";
cout << "Line 9: Length of str = "<< str.length() << endl;
return 0;
}
```

Dastur natijasi:

```
Line 1: 2 to the power of 6 = 64
Line 4: 12.5 to the power of 3 = 1953.13
Line 5: Square root of 24 = 4.89898
Line 7: u = 181.019
Line 9: Length of str = 20
```

cin operatori va get funksiyasi

Qiymat dastur orqali o‘qib olinganida probellar qiymatlarning ajratuvchisi sifatida qabul qilinadi. Agar probelning o‘zi qiymat sifatida olinishi lozim bo‘lsa, get funksiyasidan foydalaniladi:

```
char ch1, ch2;
int num;
```

va quyidagi qiymatlar kiritilsin

```
A 25
```

Qiymatlarni kiritish uchun til ko’rsatmasi

```
cin>> ch1 >>ch2>> num;
```

ko’rinishda bo’lsin. Bu operator bajarilishida ch1 o‘zgaruvchisi ‘A’ qiymatni qabul qiladi, probel belgisi tashlab yuboriladi va ch2 o‘zgaruvchisi ‘2’ qymatini, num esa 5 qiymatini o’zlashtiradi.

cin operatori orqali kiritish oqimidagi bir nechta funksiyalardan foydalana olishingiz mumkin. Belgilar ketma-ketligini o‘qib olish uchun get funksiyasidan foydalanish mumkin, uning strukturasi quyidagicha:

```
cin.get(varchar);
```

Misol uchun:

```
cin.get(ch1);
cin.get(ch2);
cin >> num;
```

Quyidagi

```
A 25
```

qiymatlar kiritilganda ch1 o‘zgarvchisi‘A’ qiymatni, ch2o‘zgarvchisiprobel belgisini va num o‘zgaruvchisi 25 sonini qabul qiladi.

cin.get() – funksiyasi belgi turidagi o‘zgaruvchiga faqat bitta belgini kiritish uchun mo‘ljallangan.

cin operatori va ignore funksiyasi

Berilganlar oqimidan faqat kerakli qismini kiritish kerak bo‘lsa, u holda kiritish oqimining ignore funksiyasidan foydalanish mumkin. ignore funksiyasining sintaksisi quyidagicha:

```
cin.ignore(inexp, chexp);
```

ignore funksiyasi ikkita parametrga ega bo’lib, birinchi parameter int turida, ikkinchi parametri char turida. Misol tariqasida quyidagi dastur kodini ko‘raylik:

```
int a, b;
cin >> a;
cin.ignore(100, '\n');
cin >> b;
```

Quyidagi qiymatlar kiritilsin:

```
25 67 89 43 72
12 78 34
```

Bu yerda kiritish oqimida 25 qiymati a o‘zgaruvchisiga o‘qiladi. Ikkinchi operator cin.ignore(100, '\n');, 100 ta belgini yoki birinchi uchragan‘\n’ – belgisigacha inkor qiladi va cin >> b, kiritish operatori 12 qiymatini b o‘zgaruvchisiga o‘qib oladi.

Yozish oqimi (cout)

Berilganlarni oqimga chiqarish uchun cout operatoridan foydalaniladi.

```
cout << soni;
```

Bir nechta o‘zgaruvchilar qiymatlarini yozish oqimiga quyidagicha chiqarish mumkin:

```
cout << soni << vazni;
```

Shuningdek, yuqoridagi operatorni quyidagicha yozish ham mumkin:

```
cout << soni;
cout << vazni;
```

Berilganlarni turli formatda va ko‘rinishda chop etish uchun manipulyatorlardan foydalaniladi. Dasturda manipulyatorlarni ishlatish uchun iomanip kutubxonasidan foydalaniladi.

fixed manipulyatori haqiqiy sonni fiksirlangan nuqtali ko‘rinishda chop etadi. Ushbu manipulyatordan foydalanish imkonini

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

funksiyasi orqali o‘chirib qo‘yish mumkin.

scientific manipulyatori esa haqiqiy sonni ilmiy formatda (eksponensial) chop etishda ishlatiladi.

```
#include <iostream>
using namespace std;
int main()
{
double hours = 35.45;
double rate = 15.00;
double tolerance = 0.01000;
```

```
cout << "hours = " << hours << ", rate = " << rate << ", pay = " << hours * rate
<< ", tolerance = " << tolerance << endl << endl;
cout << scientific;
cout << "Scientific notation: " << endl;
cout << "hours = " << hours << ", rate = " << rate<< ", pay = " << hours * rate
<< ", tolerance = " << tolerance << endl << endl;
cout << fixed;
cout << "Fixed decimal notation: " << endl;
cout << "hours = " << hours << ", rate = " << rate<< ", pay = " << hours * rate
<< ", tolerance = " << tolerance << endl << endl;
return 0;
}
```

Dastur natijasi:

```
hours = 35.45, rate = 15, pay = 531.75, tolerance = 0.01
Scientific notation:
hours = 3.545000e+001, rate = 1.500000e+001, pay = 5.317500e+002, tolerance
= 1.000000e-002
Fixed decimal notation:
hours = 35.450000, rate = 15.000000, pay = 531.750000, tolerance = 0.010000
```

setprecision manipulyatori haqiqiy sonlarni chop etishda ishlatiladi. Bu manipulyator orqali son kasr qismining nechta raqamini chop etish keraklagini aniqlash imkoni tug‘iladi.

```
cout << setprecision(2);
double d=123.456;
cout << fixed << setprecision(2);
cout << d;
```

Dastur natijasi:123.45soni ekranga chop etiladi.

setw manipulyatori o‘zgaruvchi yoki ifoda qiymati natijalarini maxsus kataklarda (joy) chiqarish imkonini beradi. setw(n) – ko‘rinishida beriladigan ushbu manipulyatorda n – nechta katakchada chiqarish kerakligini aniqlaydi.

```
#include <iostream>
using namespace std;
int main()
{
int x = 19,  int a = 345;
double y = 76.384;
cout << fixed <<showpoint;
cout << "12345678901234567890" << endl;
cout << setw(5) << x << endl;
cout << setw(5) << a << setw(5) << "Hi"<< setw(5) << x << endl << endl;
cout << setprecision(2);
cout << setw(6) << a << setw(6) << y<< setw(6) << x << endl;
cout << setw(5) << a << x << endl;
cout << setw(2) << a << setw(4) << x << endl;
return 0;
}
```

Dastur natijasi:

```
12345678901234567890
   19
  345   Hi   19
   345 76.38    19
  34519
345  19
```

Nazorat savollari

1. O‘qish oqimi nima?
2. Input stream nima?
3. Output stream nima?
4. C++ dastur sarlavhasida qaysi fayldan foydalanish kerak?
5. iostream fayli nechta oqimdan tashkil topgan?
6. Qiymat dastur orqali o‘qib olinganida nimalar qiymatlarning ajratuvchisi sifatida qabul qilinadi?
7. Qaysi kalit so‘zi orqali kiritish oqimidagi bir nechta funksiyalarda foydalanish mumkin?
8. Berilganlar oqimidan faqat kerakli qismini kiritish kerak bo‘lsa, unda kiritish oqimining qaysi funksiyasidan foydalanish kerak?
9. fixed manipulyatori nimani chop etadi?

§7. Operatorlar. Shart operatorlari

Taqqoslash amallari

C++ tilida qiymatlarni solishtirish uchun taqqoslash amallari aniqlangan. Taqqoslash amali binar amal bo‘lib, quyidagi ko‘rinishga ega:

<operand1><taqqoslash amali><operand2>

Taqqoslash amallarining natijasi – rost 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.

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”

Mantiqiy amallar

Dasturlashda bir emas balki bir nechta shartli ifodalarni tekshirish zaruriyati juda ko‘p uchraydi. Masalan, x o‘zgaruvchisi y o‘zgaruvchisidan, y esa o‘z navbatida z o‘zgaruvchisidan kattami sharti bunga misol bo‘la oladi. Bizning dasturimiz mos amalni bajarishdan oldin bu ikkala shart rost yoki yolg‘onligini tekshirishi lozim.

Yuqori darajada tashkil qilingan signalizatsiya sistemasini tasavvur qiling. Agarda eshikda signalizatsiya o‘rnatilgan bo‘lsa VA kun vaqti kech soat olti VA bugun bayram YOKI dam olish kuni bo‘lmasa politsiya chaqirilsin. Barcha shartlarni tekshirish uchun C++ tilining uchta mantiqiy amali ishlatiladi.

Mantiqiy amallar

Amal	Belgilanishi	Namuna
VA	&&	ifoda1 && ifoda2
YOKI	||	ifoda1 || ifoda2
INKOR	!	! ifoda

Mantiqiy ko‘paytirish amali ikkita ifodani hisoblaydi, agar ikkala ifoda true qiymat qaytarsa VA amali ham true qiymat qaytardi. Agarda sizning qorningiz ochligi rost bo‘lsa VA sizda pul borligi ham rost bo‘lsa siz supermarketga borishingiz va u erdan o‘zingizga tushlik qilish uchun biror bir narsa xarid qilishingiz mumkin. Yoki yana bir misol, masalan,

(x==5) && (y==5)

mantiqiy ifodasi agarda x va y o‘zgaruvchilarini ikkalasining ham qiymatlari 5 ga teng bo‘lsagina true qiymat qaytaradi. Bu ifoda agarda o‘zgaruvchilardan birortasi 5 ga teng bo‘lmagan qiymat qabul qilsa false qiymatini qaytaradi. Mantiqiy ko‘paytirish amali faqatgina o‘zining ikkala ifodasi ham rost bo‘lsagina true qiymat qaytaradi.

Mantiqiy qo‘shish amali ham ikkita ifoda orqali hisoblanadi. Agarda ulardan birortasi rost bo‘lsa mantiqiy qo‘shish amali true qiymat qaytaradi.

Agarda sizda pul YOKI kredit kartochkasi bo‘lsa, to‘lovni amalga oshira olasiz. Bu holda ikkita shartning birdaniga bajarilishi: ham pul, ham kredit kartochkasiga ega bo‘lishingiz shart emas. Sizda ulardan birining bo‘lishi etarli. Bu amalga oid yana bir misolni qaraymiz. Masalan,

(x==5) || (y>13)

ifodasi x o‘zgaruvchi qiymati 5 ga teng bo‘lsa, yoki y o‘zgaruvchi qiymati 13 dan katta bo‘lsa rost qiymat qaytaradi.

Mantiqiy inkor amali tekshirilayotgan ifoda yolg‘on bo‘lsa true qiymat qaytaradi. Agarda tekshirilayotgan ifoda rost bo‘lsa inkor amali false qiymat qaytaradi. Masalan,

!('A' > 'B')yoki !(x<=9)

Mantiqiy amallarining bajarilish jadvali

ifoda1	**ifoda1**	**ifoda1&& ifoda2**	**ifoda1 || ifoda2**	!ifoda1
false (0)	false (0)	false (0)	false (0)	true (1)
false (0)	true (0 emas)	false (0)	true (1)	true (1)
true (0 emas)	false (0)	false (0)	true (1)	false (0)
true (0 emas)	true (0 emas)	true (1)	true (1)	false (0)

Shart operatorlari

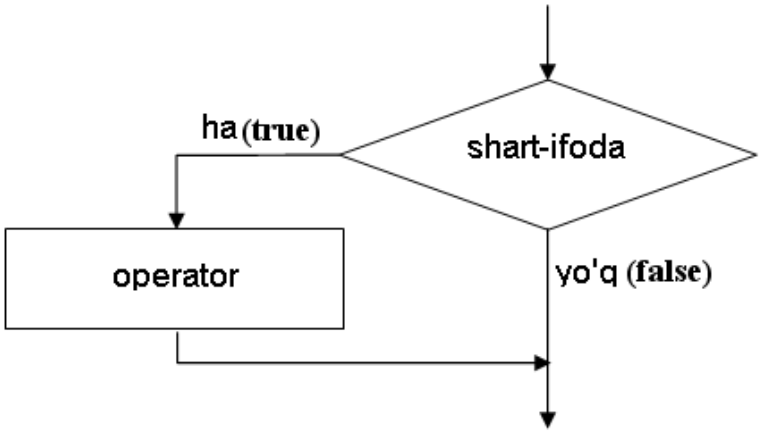
Oldingi mavzularda misol tariqasida keltirilgan dasturlarda amallar yozilish tartibida ketma-ket va faqat bir marta bajariladigan holatlar, ya’ni chiziqli algoritmlar keltirilgan. Amalda esa kamdan-kam masalalar shu tariqa yechilishi mumkin. Aksariyat masalalar yuzaga keladigan turli holatlarga bog‘liq ravishda mos qaror qabul qilishni (yechimni) talab etadi. C++ tili dasturning alohida bo‘laklarini bajarilish tartibini boshqarishga imkon beruvchi qurilmalarning yetarlicha katta majmuasiga ega. Masalan, dastur bajarilishining biror qadamida qandaydir shartni tekshirish natijasiga ko‘ra boshqaruvni dasturning u yoki bu bo‘lagiga uzatish mumkin (*tarmoqlanuvchi algoritm*). Tarmoqlanishni amalga oshirish uchun shartli operatordan foydalaniladi.

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

if (<shart ifoda>) <operator>;

Bu yerda <shart ifoda> har qanday ifoda bo‘lishi mumkin.

Agar <shart ifoda> qiymati 0 qiymatidan farqli yoki rost (true) bo‘lsa, <operator> bajariladi, aks holda, ya’ni 0 yoki yolg‘on (false) bo‘lsa, hech qanday amal bajarilmaydi va boshqaruv if operatoridan keyingi operatorga o‘tadi. Bunday qurilma bir tomonlama tanlov deb ham ataladi. Ushbu holat quyidagi rasmda ko‘rsatilgan.



Quyidagi dasturda if operatoridan foydalanish ko‘rsatilgan.

```
#include <iostream>
using namespace std;
int main()
{
 int b;
 cin >> b;
 if (b > 0) cout << "b - musbat son";
 b –=4;
if (b<0)cout<< "b - manfiy son";
return 0;
}
```

Dastur bajarilishi jarayonida butun turdagi b o‘zgaruvchi e’lon qilinadi va uning qiymati klaviaturadan o‘qiladi. Keyin b o‘zgaruvchining qiymatini 0 sonidan kattaligi tekshiriladi, agar shart bajarilsa (true), u holda ekranga "b - musbat son" xabari chiqadi. Agar shart bajarilmasa, bu operatorlar cheklab o‘tiladi. b o‘zgaruvchisining qiymatidan to‘rt ayiriladi. Navbatdagi shart operatori b o‘zgaruvchi qiymati manfiylikka tekshiradi, agar shart bajarilsa, ekranga "b - manfiy son" xabari chiqadi.

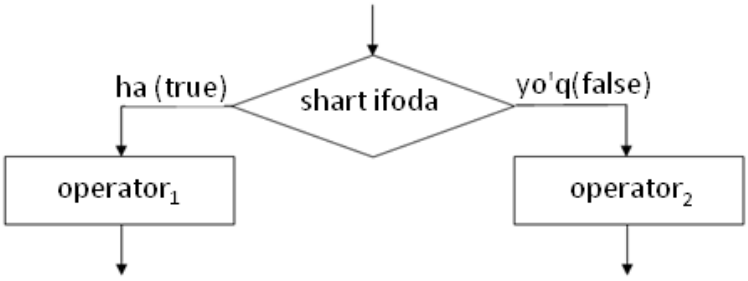
If...else **operatori**

Agar dastur bajarilishi jarayonida shartning natijasiga qarab u yoki bu amalni bajarish kerak bo‘lsa, shart operatorining ikki tomonlama tanlovli ko‘rinishidan foydalaniladi. Bunday ko‘rinish quyidagicha sintaksisga ega:

if (<shart ifoda>) <operator1>;

else <operator2>;

Bu yerda <shart ifoda>0 qiymatidan farqli yoki true bo‘lsa, <operator1>, aks holda <operator2> bajariladi. if...else shart operatori mazmuniga ko‘ra algoritmning tarmoqlanuvchi blokini ifodalaydi: <shart ifoda> - shart bloki (romb) va <operator1> blokning “*ha*” shoxiga, <operator2> esa blokning “*yo‘q*” shoxiga mos keluvchi amallar bloklari deb qarash mumkin.



Quyidagi dasturda if...else operatoridan foydalanish keltirilgan.

```
#include <iostream>
using namespace std;
int main()
{
int a, b, c;
```

```
cin >> a >> b;
if (a + b>30) c=a + b;
 else c=a * b;
cout<< "c = " << c;
return 0;
}
```

Faraz qilaylik, dastur bajarilishi jarayonida a va b butun turdagi o‘zgaruvchilarga mos ravishda 9 va 13 sonlari kiritildi. Shart tekshirilishi jarayonida a va b o‘zgaruvchilari qiymatlari qo‘shiladi va o‘ttiz sonidan kattaligi tekshiriladi. Agar natija o‘ttizdan katta bo‘lsa c o‘zgaruvchisiga a va b o‘zgaruvchilari qiymatlari yig‘indisi yuklanadi, aks holda ularning ko‘paytmasi yuklanadi. Shart bajarilmaganligi uchun if...else operatorining else qismi bajariladi va ekranga c = 117 xabari chiqadi.

C++ tilining qurilmalari operatorlarni blok ko‘rinishida tashkil qilishga imkon beradi. *Blok* – ‘{‘ va ‘}’ belgilari oralig‘iga olingan operatorlar ketma-ketligi bo‘lib, u kompilyator tomonidan yaxlit bir operator deb qabul qilinadi. Blok ichida e’lon operatorlari ham bo‘lishi mumkin va ularda e’lon qilingan o‘zgaruvchilar faqat shu blok ichida ko‘rinadi (amal qiladi), blokdan tashqarida ko‘rinmaydi. Blokdan keyin ‘;’ belgisi qo‘yilmasligi mumkin, lekin blok ichidagi har bir ifoda ‘;’ belgisi bilan yakunlanishi shart.

Masala. Berilgan to‘rt xonali ishorasiz sonning boshidagi ikkita raqamining yig‘indisi qolgan raqamlar yig‘indisiga teng yoki yo‘qligi aniqlansin (raqamlar yig‘indisi deganda ularga mos son qiymatlarining yig‘indisi tushuniladi). Sonning raqamlarini ajratib olish uchun butun sonlar arifmetikasi amallaridan foydalaniladi:

```
#include <iostream>
using namespace std;
int main()
{
```

```
 unsigned int n, a3, a2, a1, a0;
 cout << "\nn - qiymatini kiriting: ";cin >> n;
 if ((n < 1000) || (n > 9999))cout << "Kiritilgan son 4 xonali emas!";
  else{
    a3 = n / 1000;
    a2 = n % 1000 / 100;
    a1 = n % 100 / 10;
    a0 = n % 10;
    if(a3 + a2 == a1 + a0) cout << "a3+a2 = a1+a0";
      else cout << "a3+a2 != a1+a0";
  }
return 0;
}
```

Dastur ishorasiz butun son kiritishni taklif qiladi. Agar kiritilgan son 4 xonali bo‘lmasa (n < 1000) yoki (n > 9999), bu haqda xabar beriladi va dastur o‘z ishini tugatadi. Aks holda n sonining raqamlari ajratib olinadi, hamda boshidagi ikkita raqamning yig‘indisi – (a3+a2) qolgan ikkita raqamlar yig‘indisi – (a1+a0) bilan solishtiriladi va ularning teng yoki yo‘qligiga qarab mos javob chop qilinadi.

Shart operatorida e’lon qilish operatorlarini ishlatish man etiladi, lekin undagi bloklarda o‘zgaruvchilarni e’lon qilish mumkin va bu o‘zgaruvchilar faqat blok ichida amal qiladi. Quyidagi misolda bu holat bilan bog‘liq xatolik ko‘rsatilgan:

```
if (j > 0){
int i;
i=2*j;
}
else i = -j; // xato, chunki i blokdan tashqarida ko‘rinmaydi.
```

Misol tariqasida diskriminantni hisoblash usuli yordamida $ax^2+bx+c=0$ ko‘rinishidagi kvadrat tenglama ildizlarini topish masalasini ko‘raylik:

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
 float a,b,c;
 float D,x1,x2;
 cout <<"ax^2+bx+c=0 tenglama ildizini topish. ";
 cout <<"\n a - koeffisiyentini kiriting: ";  cin >> a;
 cout <<"\n b - koeffisiyentini kiriting: "; cin >> b;
 cout <<"\n c - koeffisiyentini kiriting: "; cin >> c;
 D = b * b – 4 * a * c;
 if (D < 0){
cout <<"Tenglama haqiqiy ildizga ega emas!";
return 0;
}
if (D == 0){
cout <<"Tenglama yagona ildizga ega: ";
x1 = -b / (2 * a);
cout <<"\nx= "<< x1;
}
else{
cout <<"Tenglama ikkita ildizga ega: ";
x1 = (-b + sqrt(D)) / (2 * a);
x2 = (-b - sqrt(D)) / (2 * a);
 cout <<"\nx1= "<< x1 <<"\nx2= "<< x2;
```

```
}
return 0;
}
```

Dastur bajarilganda, birinchi navbatda tenglama koeffitsientlari - a,b,c o‘zgaruvchilar qiymatlari kiritiladi, keyin diskriminant –D o‘zgaruvchi qiymati hisoblanadi. Keyin D qiymatining manfiy ekanligi tekshiriladi. Agar shart o‘rinli bo‘lsa, yaxlit operator sifatida keluvchi ‘{‘ va ‘}’ belgilari orasidagi operatorlar bajariladi va ekranga “*Tenglama haqiqiy ildizga ega emas!*” xabari chiqadi va dastur o‘z ishini tugatadi (“return 0;” operatorini bajarish orqali). Diskriminant noldan kichik bo‘lmasa, navbatdagi shart operatori uni nolga tengligini tekshiradi. Agar shart o‘rinli bo‘lsa, keyingi qatorlardagi operatorlar bloki bajariladi – ekranga “*Tenglama yagona ildizga ega:*” xabari, hamda x1 o‘zgaruvchi qiymati chop etiladi, aks holda, ya’ni D qiymati noldan katta holati uchun else kalit so‘zidan 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 funksiyaning return ko‘rsatmasini bajarish orqali dastur o‘z ishini tugatadi.

O‘z navbatida <operator1> va <operator2> ham shartli operator bo‘lishi mumkin. Ifodadagi har bir else kalit so‘zi, o‘zidan oldingi eng yaqin if kalit so‘ziga tegishli hisoblanadi (xuddi ochiluvchi va yopiluvchi qavslardek). Buni inobatga olmaslik mazmunan xatoliklarga olib kelishi mumkin.

Masalan:

```
if (x==1)
if (y==1) cout << "x=1 va y=1";
else cout << "x <> 1";
```

Bu misolda “x<>1”xabari x qiymati 1 va y qiymati 1 bo‘lmagan holda ham chop etiladi. Quyidagi variantda ushbu mazmunan xatolik bartaraf etilgan:

```
if (x==1) {
if (y==1) cout << "x=1 va y=1";
```

```
}
else cout << "x<>1";
```

Ikkinchi misol tariqasida uchta butun sonning maksimal qiymatini topadigan dastur bo‘lagini keltirishimiz mumkin:

```
int x, y, z, max;
cin >> x >> y >> z;
if (x > y)
if (x < z) max = z;
else max = x;
else
  if (y < z) max = z;
   else max = y;
```

?: shart amali

Agar tekshirilayotgan shart nisbatan sodda bo‘lsa, shart amalining “?:” ko‘rinishini ishlatish mumkin:

<shart ifoda>?<ifoda1>:<ifoda2>;

Shart amali if shart operatoriga o‘xshash holda ishlaydi: agar <shart ifoda> 0 qiymatidan farqli yoki true bo‘lsa, <ifoda1>, aks holda <ifoda2> bajariladi. Odatda ifodalar qiymatlari birorta o‘zgaruvchiga o‘zlashtiriladi. Misol tariqasida ikkita butun son maksimumini topish masalasini ko‘raylik.

```
if (a >= b) max = a;
else max = b;
```

Dasturni “?:” operatori yordamida quyidagicha yozish mumkin:

```
#include <iostream>
using namespace std;
int main()
{
 int a, b, c;
```

```
 cout << "a va b sonlar maksimumini topish.";
 cout << "\na - qiymatini kiriting: ";  cin >> a;
 cout << "\nb - qiymatini kiriting: "; cin >> b;
 c = a > b ? a : b;
 cout<< "\nSonlar maksimumi: " << c;
 return 0;
}
```

Dasturdagi shart operatori qiymat berish operatorining tarkibiga kirgan bo‘lib, a o‘zgaruvchining qiymati b o‘zgaruvchining qiymatidan kattaligi tekshiriladi. Agar shart rost bo‘lsa, c o‘zgaruvchisiga a o‘zgaruvchining qiymatini, aks holda b o‘zgaruvchining qiymatini o‘zlashtiradi va c o‘zgaruvchisining qiymati chop etiladi.

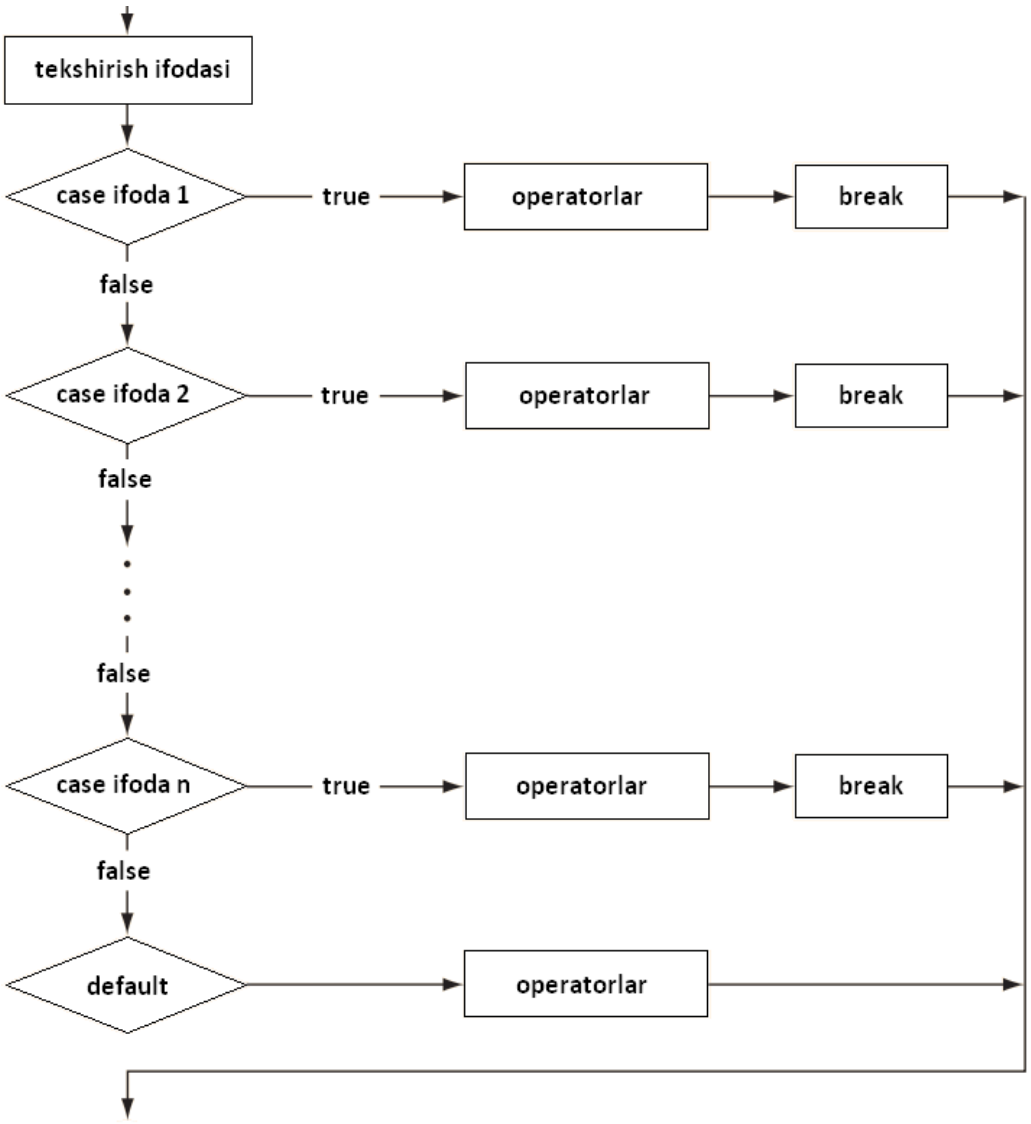
?: operatorining qiymat qaytarish xossasidan foydalangan holda, uni bevosita cout operatoriga yozish orqali ham qo‘yilgan masalani yechish mumkin:

```
#include <iostream>
using namespace std;
int main()
{
 int a,b;
 cout << "a va b sonlar maksimumini topish.";
 cout << "\na - qiymatini kiriting: ";  cin >> a;
 cout << "\nb - qiymatini kiriting: "; cin >> b;
 cout<<"\nSonlar maksimumi: "<<(a>b)?a:b;
 return 0;
}
```

switch **operatori**

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

```
switch (<ifoda>)
{
 case <o‘zgarmas ifoda_1> : <operatorlar guruhi_1>;  break;
 case <o‘zgarmas ifoda_2> : <operatorlar guruhi_2>; break;
 . . .
 case <o‘zgarmas ifoda_n> : <operatorlar guruhi_n>; break;
default :  <operatorlar guruhi_n+1>;
}
```



Bu operator quyidagicha amal bajaradi: birinchi navbatda <ifoda> qiymati hisoblanadi, keyin bu qiymat case kalit so‘zi bilan ajratilgan <o‘zgarmas ifoda_i> bilan solishtiriladi. Agar ular ustma-ust tushsa, shu qatordagi ‘:’ belgisidan boshlab, toki break kalit so‘zigacha bo‘lgan <operatorlar guruhi_i> bajariladi va boshqaruv tarmoqlanuvchi operatordan keyin joylashgan operatorga o‘tadi. Agar <ifoda> birorta ham <o‘zgarmas ifoda_i> bilan mos kelmasa, qurilmaning default qismidagi <operatorlar guruhi_n+1> bajariladi. Shuni qayd etish kerakki, qurilmada default kalit so‘zi faqat bir marta uchrashi mumkin.

Namuna uchun, char turidagi belgi o‘zgaruvchisi orqali tekshirish jarayoni bajarilayotgan quyidagi namuna ko‘rilsin:

```
switch (belgi)
{
 case 'A': cout << "Oradagi masofa 10m"; break;
 case 'B': cout << "Oradagi masofa 8m";  break;
 case 'C': cout << "Oradagi masofa 6m"; break;
 case 'D': cout << "Oradagi masofa 3m"; break;
 case 'F': cout << "Oradagi masofa 0m"; break;
 default: cout << "Masofa belgisi noto’g’ri kiritilgan.";
}
```

Ushbu misolda tekshirish uchun o‘zgaruvchi kelmoqda. O‘zgaruvchining turi belgi bo‘lgani uchun har bir qiymat case kalit so‘zidan keyin kelgan belgi bilan solishtirilgan.

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 tanlash operatordan keyingi operatorlarni bajarish bilan davom ettiradi, aks holda “*Jarayon tugadi!*” javobi beriladi va dastur o‘z ishini tugatadi. Bunda,

foydalanuvchining 'y' yoki 'Y' 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’): ";  cin >> Javob;
 switch(Javob)
{
 case ‘Y’:
 case ‘y’:  cout << ”Jarayon davom etadi!\n”;  break;
 default: cout << ”Jarayon tugadi!\n”;
 return 0;
}
 return 0;
}
```

Umuman olganda, tanlash operatorida break va default kalit so‘zlarini ishlatish majburiy emas. Lekin bu holatda operator mazmuni buzilishi mumkin. Masalan, default qismi bo‘lmagan holda, agar <ifoda> birorta <o‘zgarmas ifoda_i> bilan ustma-ust tushmasa, operator hech qanday amal bajarmasdan boshqaruv tanlash operatordan keyingi operatorga o‘tadi. Agar break bo‘lmasa, <ifoda> birorta <o‘zgarmas ifoda_i> bilan ustma-ust tushgan holda, unga mos keluvchi operatorlar guruhini bajaradi va “*to‘xtamasdan”* keyingi qatordagi operatorlar guruhini ham bajarishda davom etadi. Masalan, yuqoridagi misolda break operatori bo‘lmasa va jarayonni davom ettirishni tasdiqlovchi ('Y') javob bo‘lgan taqdirda ekranga

```
Jarayon davom etadi!
Jarayon tugadi!
```

xabarlari chiqadi va dastur o‘z ishini tugatadi (return operatorining bajarilishi natijasida).

Tanlash operatori sanab o‘tiluvchi turdagi o‘zgarmaslar bilan birgalikda ishlatilganda samara beradi. Quyidagi dasturda ranglar gammasini toifalash masalasi yechilgan.

```
#include <iostream>
using namespace std;
int main()
{
 enum Ranglar {Qizil,Tuq_sariq,Sariq,Yashil, Kuk,Zangori,Binafsha};
 Ranglar Rang = 4;
 switch (Rang)
 {
  case Qizil:case Tuq_sariq:case Sariq:
    cout << "Issiq gamma tanlandi.\n"; break;
  case Yashil:case Kuk:case Zangori:case Binafsha:
    cout << "Sovuq gamma tanlandi.\n"; break;
  default: cout << "Kamalak bunday rangga ega emas.\n";
  }
 return 0;
}
```

Dastur bajarilishida boshqaruv tanlash operatoriga kelganda, Rang qiymati Qizil yoki Tuq_sariq yoki Sariq bo‘lsa, “*Issiq gamma tanlandi*” xabari, agar Rang qiymati Yashil yoki Kuk 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.

switch operatorida e’lon operatorlari ham uchrashi mumkin. Lekin switch operatori bajarilishida “*sakrab o‘tish*” holatlari bo‘lishi hisobiga blok ichidagi ayrim e’lonlar bajarilmasligi va buning oqibatida dastur ishida xatolik ro‘y berishi mumkin:

```
//...
int k=0,n=0;
cin >>n;
switch (n)
{
int i=10;              // xato, bu operator bajarilmaydi
case 1:  int j=20;     // agar n=2 bo‘lsa, bu e’lon bajarilmaydi
case 2: k+=i+j;        // xato, chunki i,j o‘zgaruvchilar noma’lum
}
cout<<k;
//...
```

Masala. r birlikda berilgan x o‘zgaruvchisining qiymati metrlarda chop qilish dasturi tuzilsin.

```
#include <iostream>
using namespace std;
int main()
{
  enum Birlik {desimetr, kilometr, metr, millimetr, santimetr};
  float x,y;
  Birlik r; int p;
  cout << "Uzunlikni kiriting: x=";  cin>>x;
  cout<<" Uzunlik birliklari\n";
  cout<<" 0- desimetr\n";
  cout<<" 1- kilometr\n";
```

```
 cout<<" 2- metr\n";
 cout<<" 3- millimetr\n";
 cout<<" 4- santimetr\n";
 cout<<" Uzunlikni birligini tanlang: r=";  cin>>p;
 r=p;
 switch(r)
 {
   case desimetr: y=x/10; break;
   case kilometr: y=x*1000; break;
   case metr: y=x; break;
   case millimetr: y=x/1000; break;
   case santimetr: y=x/100; break;
  default: cout<<"Uzunlik birligi noto'g'ri kiritildi!";  return 0;
 }
 cout<<y<<" metr";
 return 0;
}
```

Nazorat savollari

1. Mantiqiy qo‘shish operatori nechta ifoda orqali hisoblanadi?
2. Mantiqiy inkor operatori tekshirilayotgan ifoda yolg‘on bo‘lsa qanday qiymat qaytaradi?
3. if operatori nima?
4. C++ tilining qurilmalari operatorlarni blok ko‘rinishida tashkil qilishga imkon beradimi? Buni tushuntirib bering.
5. Blok – nima?
6. Shart operatorida e’lon qilish operatorlarini ishlatish mumkinmi?
7. <operator1> va <operator2> shartli operator bo‘lishi mumkinmi?
8. Agar tekshirilayotgan shart nisbatan sodda bo‘lsa nima ishlatish mumkin?
9. switch tarmoqlanish operatori nima?

10. break va default kalit so‘zlari nimauchun ishlatiladi?
11. switch operatorida e’lon operatorlari ham uchrashi mumkinmi?
12. switch operatori bajarilishida “sakrab o‘tish” holatlari bo‘lishi hisobiga blok ichidagi ayrim e’lonlar bajarilmasligi va buning oqibatida dastur ishida xatolik ro‘y berishi mumkinmi?
13. switch operatori nima uchun ishlatiladi?
14. Sanab o‘tiluvchi turlar va shu turdagi o‘zgaruvchilarga misol keltiring.
15. Mantiqiy operatorlarga nimalar kiradi?

§8. Takrorlash operatorlari. Boshqaruvni uzatish operatorlari

Takrorlanuvchi jarayonlar

Beshta sonning o‘rta arifmetigini topish masalasi ko‘rilsin. Buning uchun quyidagi dastur kodi qismidan foydalanish mumkin:

```
cin >> num1 >> num2 >> num3 >> num4 >> num5;
sum = num1 + num2 + num3 + num4 + num5;
average = sum / 5;
```

Quyidagi savol tug‘ilishi tabiiy: sonlar miqdori ko‘p bo‘lsa nima qilish kerak? O‘zgaruvchilar soni ko‘payib ketadi. Ammo bitta o‘zgaruvchi bilan ham ushbu masalani yechish mumkin. Buning uchun quyidagi dastur kodi qismidan foydalanish mumkin:

```
1. sum = 0;
2. cin >> num;
3. sum = sum + num;
```

Birinchi ifodada sum o‘zgaruvchisiga boshlang‘ich qiymat yuklanadi. Ikkinchi ifodada num o‘zgaruvchisiga ekran orqali qiymat kiritiladi. Uchinchi ifodada esa sum o‘zgaruvchisiga num o‘zgaruvchisining qiymati qo‘shiladi.

```
num = 5
sum = sum + num = 0 + 5 = 5
num = 3
```

sum = sum + num = 5 + 3 = 8

va xokazo...

Agar o‘rta arifmetigi topilishi kerak bo‘lgan sonlar miqdori ko‘p bo‘lsa dastur kodi ko‘p bo‘lib ketadi. Ikkinchi va uchinchi ifodani barcha sonlar uchun takroran yozish kerak bo‘ladi. Shunday vaziyatlarda takrorlash operatorlaridan foydalanish maqsadga muvofiq.

Takrorlash operatori “*takrorlash sharti*” deb nomlanuvchi ifodaning rost qiymatida dasturning ma’lum bir qismidagi operatorlarni (takrorlash tanasini) ko‘p marta takror ravishda bajaradi (iterativ jarayon).

Takrorlash o‘zining kirish va chiqish nuqtalariga ega, lekin chiqish nuqtasining bo‘lmasligi mumkin. Bu holda takrorlashga *cheksiz takrorlash* deyiladi. Cheksiz takrorlash uchun takrorlashni davom ettirish sharti doimo rost bo‘ladi.

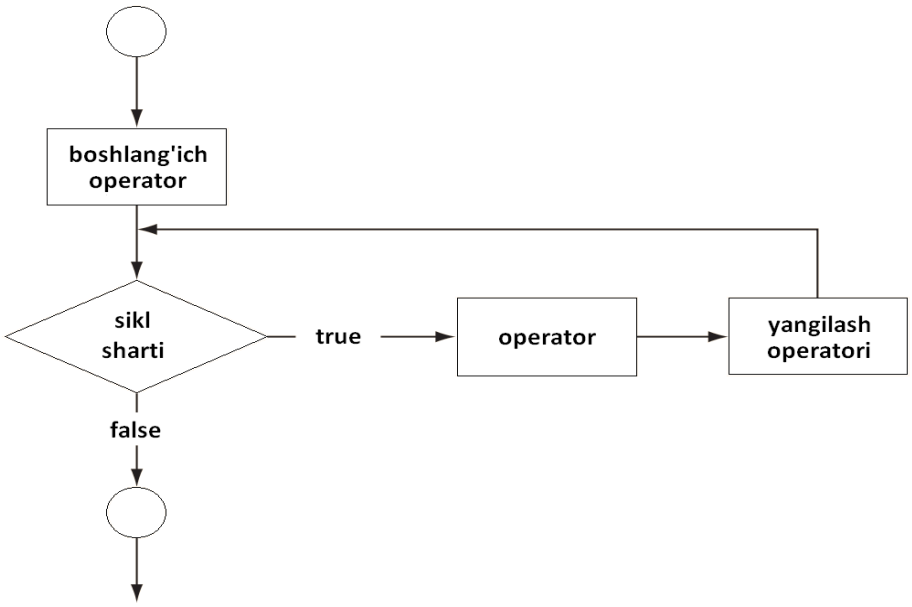
Takrorlash shartini tekshirish takrorlash tanasidagi operatorlarni bajarishdan oldin tekshirilishi mumkin (for, while) yoki takrorlash tanasidagi operatorlari bir marta bajarilgandan keyin tekshirilishi mumkin (do-while).

for takrorlash operatori

for takrorlash operatorining sintaksisi quyidagi ko‘rinishga ega:

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

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 yoki operatorlar bloki bo‘lishi 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++)Summa+=i;
cout<< "Yig'indi="<<Summa;
return 0;
}
```

Dasturdagi takrorlash operatori o‘z ishini, i takrorlash parametriga (takrorlash sanagichiga) boshlang‘ich qiymat 10 sonini berishdan boshlaydi va har bir takrorlash qadamidan 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 ningqiymati 21 bo‘lganda “i <= 20” takrorlash sharti false 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 ifodalariga 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 uchrashi 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 qiluvchi boshqa amallar bo‘lishi mumkin.

Takrorlash operatorida ham bloklardan foydalanish mumkin. Bir nechta operatorlar takrorlanishi kerak bo‘lganda bloklardan foydalanish mumkin. Buni quyidagi misolda yaqqol ko‘rish mumkin:

```
for (i = 1; i <= 3; i++)
{
 cout << "Hello!" << endl;
 cout << "***" << endl;
}
```

Ushbu misol natijasi quyidagicha ko‘rinishga ega bo‘ladi:

```
Hello!
***
Hello!
***
Hello!
***
```

Agar aynan shu misolda blok ishlatilmasa dastur quyidagi ko‘rinishda ishlaydi:

```
for (i = 1; i <= 3; i++) cout << "Hello!" << endl;
cout << "***" << endl;
```

Dastur ishlashi natijasi:

```
Hello!
Hello!
Hello!
***
```

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= j +10)
{
 s = s + i;
 p = p + j;
}
```

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

for operatorida takrorlash tanasi bo‘lmasligi ham mumkin. Masalan, dastur bajarilishini ma’lum bir muddatga “*to‘xtatib*” turish zarur bo‘lsa, bunga takrorlashni hech qanday qo‘shimcha ishlarni bajarmasdan amal qilishi orqali erishish mumkin:

```
#include <iostream>
using namespace std;
int main()
{
```

```
 int delay;
 ...
 for(delay=5000; delay>0; delay--);// bo‘sh operator
 ...
 return 0;
}
```

10 dan 20 gacha bo‘lgan sonlar yig‘indisini bo‘sh tanali takrorlash operatori orqali hisoblash mumkin:

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

Takrorlash operatorini blok tanasi sifatida ishlatishni faktorialni hisoblash misolida ko‘rsatish mumkin:

```
#include <iostream>
using namespace std;
int main()
{
 int a;
 unsigned long fact=1;
 cout << "Butun sonni kiriting: ";cin >>a;
 if ((a>=0)&&(a<=33)) {
 for (int i=1; i<=a; i++) fact*=i;
```

```
 cout<<a<< "!= "<<fact<<'\n';
}
return 0;
}
```

Dastur foydalanuvchi tomonidan 0 dan 33 gacha oraliqdagi son kiritilganda amal qiladi, chunki 34! qiymati unsigned long uchun ajratilgan razryadlarga sig‘maydi.

Takrorlash operatori ichma-ich joylashgan bo‘lishi ham mumkin. Bunda har bir tashqarida joylashgan takrorlash qadami uchun ichki takrorlash to‘la aylanadi.

Misol sifatida qatorlar uchun qator soni miqdoriga teng yulduzcha belgisini chop etish masalasini ko‘raylik:

```
#include <iostream>
using namespace std;
int main()
{
 for (i = 1; i <= 5; i++){
  for (j = 1; j <= i; j++)cout << "* ";
 cout << endl;
}
return 0;
}
```

Dastur ishlashi natijasi:

```
*
**
***
****
*****
```

Takrorlash operatorining ichma-ich joylashuviga misol sifatida raqamlari bir-biriga o‘zaro teng bo‘lmagan uch xonali natural sonlarni o‘sish tartibida chop qilish masalasini ko‘rish mumkin:

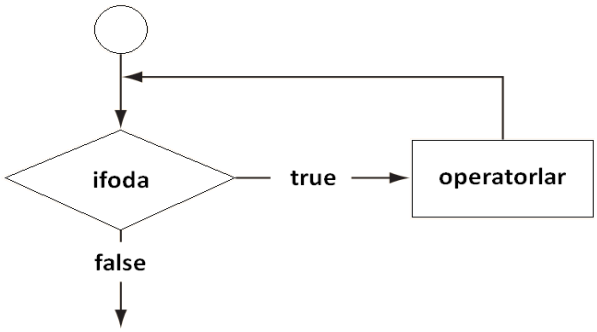
```
#include <iostream>
using namespace std;
int main()
{
 unsigned char a3,a2,a1;          //uch xonali son raqamlari
 for (a1='1'; a1<='9'; a1++)      //sonning 1-raqami
  for (a2='0'; a2<='9'; a2++)     //sonning 2-raqami
   for (a3='0'; a3<='9'; a3++)    //sonning 3-raqami
// raqamlarni o‘zaro teng emasligini tekshirish
    if (a1!=a2 && a2!=a3 && a1!=a3)    //o‘zaro teng emas
     cout<<a1<<a2<<a3<<'\n';
 return 0;
}
```

Dasturda uch xonali sonning har bir raqami takrorlash operatorlarining parametrlari sifatida hosil qilinadi. Birinchi, tashqi takrorlash operatori bilan 1-xonadagi raqam (a1 takrorlash parametri) hosil qilinadi. Ikkinchi, ichki takrorlash operatorida (a2 takrorlash parametri) son ko‘rinishining 2-xonasidagi raqam va nihoyat, unga nisbatan ichki bo‘lgan a3 parametrli takrorlash operatorida 3-xonadagi raqamlar hosil qilinadi. Har bir tashqi takrorlashning bir qadamiga ichki takrorlash operatorining to‘liq bajarilishi to‘g‘ri kelishi hisobiga barcha uch xonali sonlar ko‘rinishi hosil qilinadi.

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>;



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.

```
i = 0;
while (i <= 20){
cout << i << " ";
i = i + 5;
}
cout << endl;
```

Dastur qismi ishlashi natijasi:

```
0 5 10 15 20
```

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 ifodaning qiymati rost(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 bajarilishi mumkin. Masalan, son va uning kvadratlarini chop qiladigan 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=" << n << " n^2 = " << n2 << endl;
 return 0;
}
```

Dasturdagi takrorlash operatori bajarilishida 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 qaraladi va n qiymati 0 bo'lganda takrorlash tugaydi.

Keyingi dasturda berilgan o'nlik sonning ikkilik ko'rinishini chop qilish masalasini yechishda while operatorini qo'llash ko'rsatilgan.

```
#include <iostream>
using namespace std;
int main()
{
 int sanagich = 4;
 short son10, jarayon = 1;
 while (jarayon) {                // cheksiz takrorlash
  cout << "O'nlik sonni kiriting (0..15) ";cin >> son10;
  cout <<'\n'<< son10 << "Sonining ikkilik ko'rinishi: ";
 while (sanagich){
  if (son10 & 8) cout <<'1';  //son10 & 00001000
```

```
  else cout <<'0';
 son10 <<= 1;        // razryadlarni chapga surish
 sanagich--;
}
cout <<'\n' ;
cout << "Jarayonni to'xtasin(0), davom etsin(1): ";cin >> jarayon;
sanagich = 4;
}
return 0;
}
```

Dasturda ichma-ich joylashgan takrorlash operatorlari ishlatilgan. Birinchisi, sonning ikkilik ko‘rinishini chop qilish jarayonini davom ettirish sharti bo‘yicha amal qiladi. Ichki joylashgan ikkinchi takrorlash operatoridagi amallar – har qanday, 0 dan 15 gacha bo‘lgan sonlar to‘rtta razryadli ikkilik son ko‘rinishida bo‘lishiga asoslangan. Unda kiritilgan sonning ichki, ikkilik ko‘rinishida uchinchi razryadida 0 yoki 1 turganligi aniqlanadi (“son10 & 8”). Shart natijasi 1 (rost) bo‘lsa, ekranga ‘1’, aks holda ‘0’ belgisi chop etiladi. Keyingi qadamda son razryadlari chapga bittaga suriladi va yana uchinchi razryaddagi raqam chop etiladi. Takrorlash sanagich qiymati 0 bo‘lguncha ya’ni to‘rt marta bajariladi va boshqaruv ichki takrorlash operatoridan chiqadi.

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

```
#include <iostream>
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;
}
```

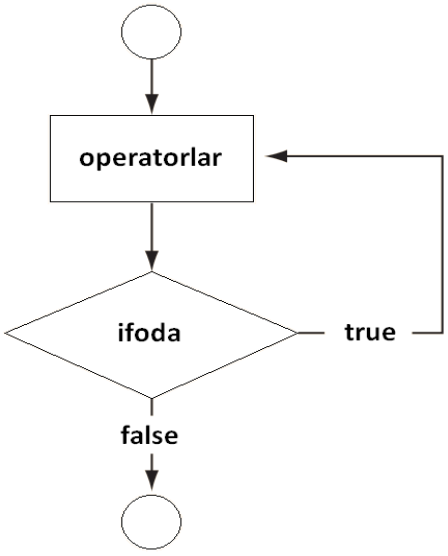
Butun turdagi a va b qiymatlari oqimdan o‘qilgandan keyin toki ularning qiymatlari o‘zaro teng bo‘lmaguncha takrorlash jarayoni ro‘y beradi. Takrorlashning har bir qadamida a va b sonlarning kattasidan kichigi ayriladi. Takrorlashdan keyingi ko‘rsatmada a o‘zgaruvchining qiymati natija sifatida chop etiladi.

do-while takrorlash operatori

do-while takrorlash operatori while operatoridan farqli ravishda oldin operator yoki blokni bajaradi, keyin takrorlash shartini tekshiradi. Bu qurilma takrorlash 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 –takrorlash boshlamasdan turib, takrorlash shartini tekshirishning iloji bo‘lmagan holatlar hisoblanadi. 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
{
 cout<< "dastur tanasi\n";
 cout<< "Jarayonni to’xtatish (N): ";cin>>javob;
}
 while(javob != 'N');
 return 0;
}
```

Dastur toki “Jarayonni to’xtatish (N):” so‘roviga 'N' belgisi (javobi) kiritilmaguncha davom etadi.

Bu operator ham cheksiz takrorlanishi mumkin:

```
do
{
 cout << "cheksiz takrorlash tanasi ";
}
while(1);
```

do-while takrorlash operatori ham boshqa takrorlash operatorlari kabi ichma ich joylashib kelishi mumkin.

Masala: Har qanday 7 dan katta butun sondagi pul miqdorini 3 va 5 so‘mliklarda berish mumkinligi isbotlansin. Qo‘yilgan masala p=3n+5m

tenglamani qanoatlantiruvchi m va 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.

```
#include <iostream>
using namespace std;
int main()
{
 unsigned int Pul;
 unsigned n3, m5;
 bool xato=false;
 do {
  if (xato) cout<<"Pul qiymati 7 dan kichik!";
  xato=true;
  cout<<"\nPul qiymatini kiriting (>7): ";cin>>Pul;
 }
 while(Pul<=7);
n3=0;
do {
 m5=0;
 do {
  if(3*n3+5*m5==Pul)cout<<n3<<" ta 3 so’mlik + "<<m5<<" ta 5 so’mlik\n";
   m5++;
 }
  while(3*n3+5*m5<=Pul);
 n3++;
 }
  while(3 * n3 <= Pul);
 return 0;
}
```

Dastur pul qiymatini kiritishni so‘raydi (Pul o‘zgaruvchisiga). Agar pul qiymati 7 sonidan kichik bo‘lsa, bu haqda xabar beriladi va takror ravishda qiymat kiritish talab qilinadi. Pul qiymati 7 dan katta bo‘lganda, 3 va 5 so‘mliklarning mumkin bo‘lgan to‘la kombinatsiyasini amalga oshirish uchun ichma-ich takrorlashlar amalga oshiriladi. Tashqi takrorlash n3 (3 so‘mliklar miqdori) bo‘yicha, ichki takrorlash esa m5 (5 so‘mliklar miqdori) bo‘yicha, toki bu miqdordagi pullar qiymati Pul qiymatidan oshib ketmaguncha davom etadi. Ichki takrorlashda m5 o‘zgaruvchisining har bir qiymatida “3*n3+5*m5==Pul” sharti tekshiriladi, agar u o‘rinli bo‘lsa, yechim varianti sifatida n3 va m5 o‘zgaruvchilar qiymatlari chop etiladi. Pul qiymati 30 so‘m kiritilganda (Pul=30), ekranga

0 ta 3 so’mlik + 6 ta 5 so’mlik

5 ta 3 so’mlik + 3 ta 5 so’mlik

10 ta 3 so’mlik + 0 ta 5 so’mlik

yechim variantlari chop etiladi.

break **operatori**

Takrorlash operatorlarining bajarilishida shunday holatlar yuzaga kelishi mumkinki, unda qaysidir qadamda, takrorlashni yakuniga yetkazmasdan takrorlashdan chiqish zarurati bo‘lishi mumkin. Boshqacha aytganda, takrorlashni “*uzish*” kerak bo‘lishi mumkin. Bunda break operatoridan foydalaniladi. break operatorini takrorlash operatori tanasining ixtiyoriy (zarur) joylariga qo‘yish orqali shu joylardan takrorlashdan chiqishni amalga oshirish mumkin. E’tibor beradigan bo‘lsak, switch-case operatorining tub mohiyatiga ham break operatorini qo‘llash orqali erishilgan.

Ichma - ich joylashgan takrorlash va switch operatorlarida break operatori faqat o‘zi joylashgan blokdan chiqish imkoniyatini beradi.

Quyidagi dasturda ikkita ichma-ich joylashgan takrorlash operatoridan foydalangan holda foydalanuvchi tomonidan kiritilgan qandaydir sonni 3 va 7 sonlariga nisbatan qanday oraliqqa tushishi aniqlanadi. Tashqi takrorlashda “*Son*

*kiriting (0-to‘xtash):*_” so‘rovi beriladi va kiritilgan qiymat javob_son o‘zgaruvchisiga o‘qiladi. Agar son noldan farqli bo‘lsa, ichki takrorlash operatorida bu sonning qandaydir oraliqqa tushishi aniqlanib, shu haqida xabar beriladi va ichki takrorlash operatoridan chiqiladi. Tashqi takrorlashdagi so‘rovga javob tariqasida 0 kiritilsa, dastur o‘z ishini tugatadi.

```
#include <iostream>
using namespace std;
int main()
{
  int javob_son=0;
  do {
   while(javob_son) {
    if(javob_son<3) {
     cout<<”3 kichik!”;
     break;
     }
    if(3<=javob_son&&javob_son<=7){
     cout<<”3 va 7 oraligida !”;
     break;
    }
   if(javob_son>7) {
    cout<<”7 dan katta !”;
    break;
  }
 }
 cout<<”\nSon kiriting (0-to’xtash):_”;cin>>javob_son;
 }
  while(javob_son !=0);
return 0;
}
```

Amaliyotda break operatoridan cheksiz takrorlashdan chiqishda foydalaniladi.

```
for (;;) {
// 1- shart
 if (...) {
 ...
 break;
 }
// 2- shart
if (...) {
 ...
 break;
}
...
}
```

Bu misolda for cheksiz takrorlashidan 1- yoki 2- shart bajarilganda chiqiladi.

Masala. Ishorasiz butun sonlar ketma-ketligi 0 qiymati bilan tugaydi, 0 ketma-ketlik hadi hisoblanmaydi. Ketma-ketlikni kamaymaydigan holda tartiblangan yoki yo‘qligi aniqlansin.

```
#include <iostream>
using namespace std;
int main()
{
 unsigned int Ai_1=0,Ai;
 cout<<"Sonlar ketma-ketligini kiriting"
 cout<<"0-tugash alomati):\n ";
 cin>>Ai;
```

```
 while(Ai) {
   Ai_1=Ai;
   cin>>Ai;
    if (Ai_1>Ai) break;
   }
   if(Ai_1) {
    cout<<"Ketma-ketlik tartiblangan";
    if(!Ai)cout<<" emas!";
     else cout<<”!”;
  }
 else cout<<"Ketma-ketlik bo’sh!";
 return 0;
}
```

Dastur ishga tushganda, avval ketma-ketlikning birinchi hadi alohida o‘qib olinadi (Ai o‘zgaruvchisiga). Keyin Ai qiymati nolga teng bo‘lmaguncha takrorlash operatori amal qiladi. Takrorlash tanasida Ai qiymati oldingi qiymat sifatida Ai_1 o‘zgaruvchisida eslab qolinadi va navbatdagi had Ai o‘zgaruvchisiga o‘qiladi. Agar oldingi had navbatdagi haddan katta bo‘lsa, break operatori yordamida takrorlash jarayoni uziladi va boshqaruv takrorlashdan keyingi shart operatoriga o‘tadi. Bu erdagi shart operatorlarining mazmuni quyidagicha: agar Ai_1 noldan farqli bo‘lsa, ketma-ketlikning kamida bitta hadi kiritilgan bo‘ladi (ketma-ketlik mavjud) va oxirgi kiritilgan had tekshiriladi. O‘z navbatida agar Ai noldan farqli bo‘lsa, bu holat hadlar o‘rtasida kamaymaslik sharti bajarilmaganligi sababli hadlarni kiritish jarayoni uzilganini bildiradi va bu haqda xabar chop etiladi. Aks holda ketma-ketlik kamaymaydigan holda tartiblangan bo‘ladi.

continue **operatori**

continue operatori xuddi break operatoridek takrorlash operatori tanasini bajarishni to‘xtatadi, lekin takrorlashdan chiqib ketmasdan keyingi qadamiga “*sakrab*” o‘tishini tayinlaydi.

continue operatorini qo‘llanishiga misol tariqasida 2 va 50 sonlar oralig‘idagi tub sonlarni topadigan dastur matnini keltiramiz.

```
#include <iostream>
using namespace std;
int main()
{
  bool bulinadi=false;
  for(int i=2; i<50; i++) {
    for (int j=2; j<i/2;j++) {
     if (i%j)continue;
  bulinadi=true;
  break;
 }
 // break bajarilganda boshqaruv o‘tadigan joy
 if(!bulinadi)cout<<i<<" ";
 bulinadi=false;
}
 return 0;
}
```

Keltirilgan dasturda qo‘yilgan masala ichma-ich joylashgan ikkita takrorlash operatorlari yordamida yechilgan. Birinchi takrorlash operatori 2 dan 50 gacha sonlarni hosil qilishga xizmat qiladi. Ichki takrorlash esa har bir hosil qilinayotgan sonni 2 sonidan toki shu sonning yarmigacha bo‘lgan sonlarga bo‘lib, qoldig‘ini tekshiradi, agar qoldiq 0 sonidan farqli bo‘lsa, navbatdagi songa bo‘lish davom etadi, aks holda bulinadi o‘zgaruvchisiga true qiymat berib, ichki takrorlash uziladi (son o‘zining yarmigacha bo‘lgan qandaydir songa bo‘linar ekan, demak u tub emas va keyingi sonlarga bo‘lib tekshirishga hojat yo‘q). Ichki j bo‘yicha takrorlashdan chiqqandan keyin bulinadi qiymati false bo‘lsa (!bulinadi), i soni tub bo‘ladi va u chop qilinadi.

goto operatori va nishonlar

Nishon – bu davomida ikkita nuqta (‘:’) qo‘yilgan identifikator. Nishon bilan qandaydir operator belgilanadi va keyinchalik, dasturning boshqa bir qismidan unga shartsiz o‘tish amalga oshiriladi. Nishon bilan har qanday operator belgilanishi mumkin, shu jumladan e’lon operatori va bo‘sh operatori ham. Nishon faqat funksiyalar ichida amal qiladi.

Nishonga shartsiz o‘tish goto operatori yordamida o‘tiladi. goto operatori orqali faqat uning o‘zi joylashgan funksiya ichidagi operatorlarga o‘tish mumkin. goto operatorining sintaksisi quyidagicha:

```
goto <nishon>;
```

Ayrim hollarda, goto operatorining “*sakrab o‘tishi*” hisobiga xatoliklar yuzaga kelishi mumkin. Masalan,

```
int i=0, j=0;
i++;
if (i==1) goto m;
j+= 5;
m: j+=i;
```

Shartsiz o‘tish operatori dasturni tuzishdagi kuchli va shu bilan birgalikda xavfli vositalardan biri hisoblanadi. Kuchliligi shundaki, uning yordamida algoritmning “*boshi berk*” joylaridan chiqib ketish mumkin. Ikkinchi tomondan, bloklarning ichiga o‘tish, masalan, takrorlash operatorlarini ichiga “*sakrab*” kirish kutilmagan holatlarni yuzaga keltirishi mumkin.

Garchi, nishon yordamida dasturning ixtiyoriy joyiga o‘tish mumkin bo‘lsa ham, boshlang‘ich qiymat berish e’lonlaridan sakrab o‘tish man etiladi, lekin bloklardan sakrab o‘tish mumkin.

Quyidagi dasturda ikkita natural sonning eng katta umumiy bo‘luvchisini (EKUB) topish masalasidagi takrorlash jarayonini nishon va goto operatori vositasida amalga oshirish ko‘rsatilgan:

```
#include <iostream>
```

```
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;
 nishon:
 if(a==b){
  cout << "Bu sonlar EKUBi: " << a;
  return 0;
 }
 a>b?a-=b:b-=a;
 goto nishon;
}
```

Dasturdagi nishon bilan belgilangan operatorda a va b sonlarni tengligi tekshiriladi. Agar ular teng bo‘lsa, ixtiyoriy bittasi, masalan a soni EKUB bo‘ladi va funksiyadan chiqiladi. Aks holda, bu sonlarning kattasidan kichigi ayriladi va goto orqali ularning tengligi tekshiriladi. Takrorlash jarayoni a va b sonlar o‘zaro teng bo‘lguncha davom etadi.

Nazorat savollari

1. For operatori qanday vazifani bajaradi?
2. While takrorlash qanday takrorlash operatori hisoblanadimi?
3. Takrorlash operatorida ham bloklardan foydalanish mumkinmi?
4. C++ tilining qurilmalari operatorlarni blok ko‘rinishida tashkil kilishga imkon beradimi?
5. Agar <ifoda> rost qiymatli o‘zgarmas ifoda bo‘lsa, takrorlash qanday buladi?
6. Takrorlash operatorlarining bajarilishida qanday holatlar yuzaga kelishi mumkin?

7. Takrorlash operatori ichma-ich joylashgan bo‘lishi mumkunmi?
8. do-while takrorlash operatori qanday vazifani bajaradi?
9. continue operatori qanday vazifani bajaradi?
10. break operatori qanday vazifani bajaradi?

§9. Statik massivlar

Massivlar haqida tushuncha

Xotirada ketma-ket (regulyar) joylashgan bir xil turdagi qiymatlarga massiv deyiladi. Odatda massivlarga zarurat, katta hajmdagi, lekin cheklangan miqdordagi va bir xil turdagi qiymatlarni qayta ishlash bilan bog‘liq masalalarni yechishda 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 (konkret) talabaning reytingi haqida ma’lumot berish va boshqa masala ostilarini yechish zarur bo‘lsin. Qayd etilgan masalalarni yechish uchun berilganlarning (reytinglarning) tartiblangan 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: Reyting1,…, ReytingN. 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, uning qiymatlariga Reyting_1,…, 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 qandaydir qiymatlarning (massiv elementlarining) 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 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.

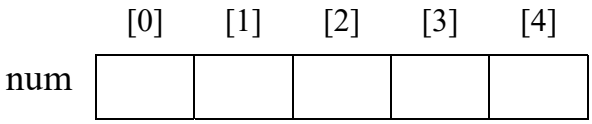
Murojaat sintaksisi:<massiv nomi >[<indeks>]

Bu ko‘rinishga xususiy o‘zgaruvchi deyiladi, chunki uning qiymati massivning alohida elementidir. Bizning misolda Reyting massivining alohida elementlariga Reyting[1],…,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 bir indeksli o‘zgaruvchi orqali massivning barcha elementlarini belgilash (aniqlash) mumkin bo‘ladi.

Massiv elementiga murojaat qilish

Massivning elementlariga murojaat indekslari orqali bo‘ladi. Masalan, Reyting[i] o‘zgaruvchisi orqali i o‘zgaruvchining qiymatiga bog‘liq ravishda Reyting massivining ixtiyoriy elementiga murojaat qilish mumkin. Indeks sifatida butun turdagi o‘zgaruvchilardan foydalanish mumkin. Haqiqiy turdagi (float, double) qiymatlar to‘plami cheksiz bo‘lganligi sababli ular indeks sifatida ishlatilmaydi. Massiv elementlarining indekslarini quyidagi rasmda ko’rish mumkin:



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

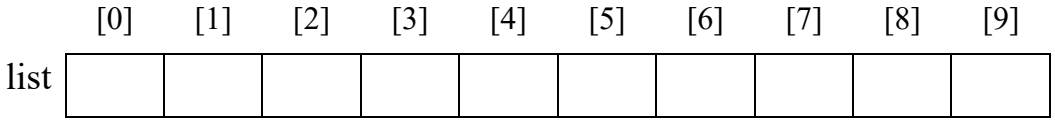
Massiv e'loni quyidagicha bo'ladi:

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

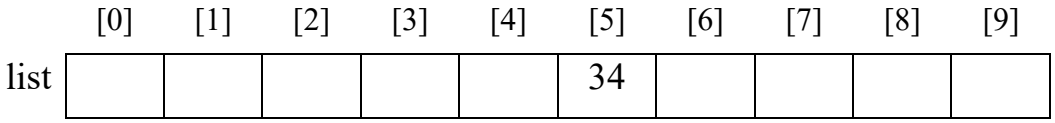
Bu yerda **<uzunlik>** – o'zgarmas ifoda (konstanta).

Misol: int list[10];

Buerda list nomli massiv elementlari 10 ta bo'lsa, uning elementlari list[0], list[1],…,list[9] bo'ladi:



Dastur matnida "list[5]=34;" ko'rsatmasi bilan 34 sonini massivning 5-joyiga joylashtirish mumkin.



Butun turdagi i o'zgaruvchi bilan "list[i]=63;", "list[i]=5*list[i]" yoki "list[2*i-3]=58;" ko'rsatmalari massiv elementlari ustida amal bajarilishiga misol bo'la oladi.

Quyidagi misollarni ko'raylik:

```
list[3]=10;
list[6]=35;
list[5]= list[3]+list[6];
```

Yuqoridagi misolda birinchi list massivining uchinchi elementiga 10 qiymatini o'zlashtirmoqda, massivning oltinchi elementiga 35 qiymatini o'zlashtirmoqda va massivning uchinchi va oltinchi elementlari yig'indisi massivning beshinchi elementiga yuklanmoqda:

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
list				10		45	35			

Xuddi shuningdek massivni quyidagicha e’lon qilish mumkin:

```
const int ARRAY_SIZE = 10;
int list[ARRAY_SIZE];
```

bu yerda birinchi o‘rinda butun turdagi o‘zgarmas e’lon qilinmoqda va massiv e’lon qilinib o‘lchamlari o‘rnatilmoqda.

Ko‘p o‘lchovli massivlar

C++tilida massiv elementining turiga cheklovlar qo‘yilmaydi, lekin bu turlar chekli o‘lchamdagi obektlarning turi bo‘lishi kerak. Chunki kompilyator massivning xotiradan qancha joy (bayt) egallashini hisoblay olishi kerak. Xususan, massiv elementi massiv bo‘lishi mumkin (“*vektorlar-vektori*”), natijada matritsa deb nomlanuvchi ikki o‘lchovli massiv hosil bo‘ladi.

	[0]	[1]	[2]	[3]	[4]
[0]					
[1]					
[2]					
[3]					
[4]					
[5]					
[6]					
[7]					
[8]					
[9]					

Agar matritsaning elementi ham vektor bo‘lsa, uch o‘lchovli massivlar - kub hosil bo‘ladi. Shu yo‘l bilan yechilayotgan masalaga bog‘liq ravishda ixtiyoriy o‘lchovdagi massivlarni yaratish mumkin.

Ikki o‘lchovli massivning sintaksisi quyidagi ko‘rinishda bo‘ladi:

<tur><nom> [<uzunlik >] [<uzunlik>]

Masalan, 10x20 o‘lchovli haqiqiy sonlar massivining e’loni quyidagicha bo‘ladi:

```
float a[10][20];
```

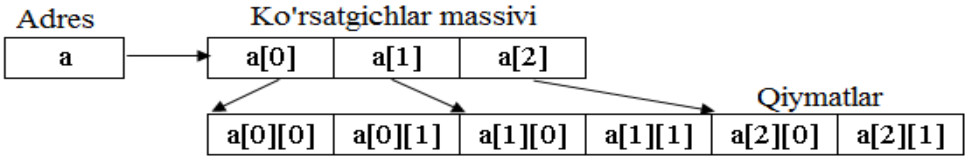
E’lon qilingan a matritsani ko‘rinishi quyidagi rasmda keltirilgan.

$$
\begin{array}{cl}
 & \qquad\qquad\qquad\qquad\qquad\quad j \\
 & a_0: \quad (a_{0\,0}, \quad a_{0\,2} \quad \ldots \quad \ldots \quad a_{0\,18}, \quad a_{0\,19}), \\
 & a_1: \quad (a_{1\,0}, \quad a_{1\,1}, \quad \ldots \quad \ldots \quad a_{1\,18}, \quad a_{1\,19}), \\
 & \ldots \\
i & a_i: \quad (\ldots, \quad \ldots, \quad \ldots \quad a_{ij} \quad \ldots \quad \ldots, \quad \ldots), \\
 & \ldots \\
 & a_9: \quad (a_{9\,0}, \quad a_{9\,1}, \quad \ldots \quad \ldots \quad a_{9\,18}, \quad a_{9\,19}).
\end{array}
$$

Endi adres nuqtai - nazaridan ko‘p o‘lchovli massiv elementlariga murojaat qilishni ko‘raylik. Quyidagi e’lonlar berilgan bo‘lsin:

```
int a[3][2];
float b[2][2][2];
```

Birinchi e’londa ikki o‘lchovli massiv, ya’ni 2 satr va 3 ustundan iborat matritsa e’lon qilingan, ikkinchisida uch o‘lchovli - 3 ta 2x2 matritsadan iborat bo‘lgan massiv e’lon qilingan. Uning elementlariga murojaat sxemasi:



Bir o‘lchovli massivlar bilan ishlash

Massivlar ustida bajariladigan asosiy amallar berilganlarni massiv elementlariga yuklash, massiv elementlari ustida amallar bajarish va massiv elementlarini chop qilishdan iborat. Agar massiv elementlari butun sonlardan iborat bo‘lsa unda massiv elementlari yig‘indisini, o‘rta arifmetigini va boshqa amallarni bajarish mumkin. Bunda massivnig har bir elementlariga murojaat qilishga to‘g‘ri keladi, buni boshqarish oson. Misol sifatida massiv e’loni quyidagicha bo‘lsin.

```
int list[100];
int i;
```

Quyidagi takrorlash operatori orqali massivning har bir elementiga murojaat qilishimiz mumkin bo‘ladi va murojaat massivning birinchi elementidan boshlanadi:

```
for (i=0; i<100; i++) ...
```

Massiv elementlari ustida amallar bajarish uchun berilganlarni massivning har bir elementiga o‘qib olish kerak, bu cin operatori orqali amalga oshiriladi. Misol sifatida quyidagi ifoda massivning 100 ta elementlarini o‘qib oladi:

```
for (i=0; i<100; i++) cin>>list[i];
```

Bir o‘lchovli massivning e’loni quyidagicha bo‘lsin:

```
double sales[10];
int index;
double largestSale,sum,average;
```

Yuqoridagi misolda haqiqiy turdagi 10 ta elementdan tashkil topgan sales massivi e’lon qilingan. Bu massiv ustida quyidagi amallarni bajaramiz:

a. *Massiv elementlariga qiymat berish:* Quyida sales massivning har bir elementiga 0.0 qiymati berilgan:

```
for(index=0; index<10; index++)  sales[index]=0.0;
```

b. *Massiv elementlarini o‘qib olish:* Klaviaturadan kiritilayotgan berilganlar massiv elementlariga o’zlashtiriladi:

```
for (index=0; index<10; index++) cin>>sales[index];
```

c. *Massiv elementlarini chop qilish*: Massivning har bir elementi probel bilan ajratilib chop qilinadi:

```
for (index=0; index<10; index++) cout<<sales[index]<<" ";
```

d. *Massivning elementlari yig‘indisini va massiv elementlarining o‘rta arifmetigini topish*:

```
int sum=0;
for (index=0; index<10; index++)  sum+= sales[index];
average=sum/10;
```

e. *Massiv elementlaridan eng kattasini topish*: Ushbu masalani yechish uchun butun turdagi maxI (masivning eng katta elementining indexi) o‘zgaruvchisi e’lon qilinadi va unga 0 beriladi. Takrorlash operatori yordamida sales massivining elementlari maxI indeksli elementi bilan solishtiriladi, agar massivning birorta elementi ushbu elementidan katta bo’lsa, maxI o‘zgaruvchisi

katta element indeksi oladi va tekshirish davom etadi. Takrorlash tugagandan keyin sales massivining maxI indeksli elementi largestSale o'zgaruvchisiga o'zlashtiriladi.

```
maxI=0;
for(int i= 1; i<10; i++) if(sales[maxI]<sales[i]) maxI=i;
largestSale=sales[maxI];
```

Bu algoritmni massivning quyidagi qiymatlarida tekshirib ko‘rish mumkin:

	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
sales	12.50	8.35	19.60	25.00	14.00	39.43	35.90	98.23	66.65	35.64

Amallar bajarilishining har bir qadamidagi holatlar quyidagi jadvalda keltirilgan.

i	maxI	sales[maxI]	sales[i]	sales[maxI]<sales[i]
1	0	12.50	8.35	12.50 < 8.35 = false
2	0	12.50	19.60	12.50 < 19.60 = true, maxI= 2
3	2	19.60	25.00	19.60 < 25.00 =true, maxI = 3
4	3	25.00	14.00	25.00 < 14.00 = false
5	3	25.00	39.43	25.00 < 39.43 = true, maxI = 5
6	5	39.43	35.90	39.43 < 35.90 = false
7	5	39.43	98.23	39.43 < 98.23 = true, maxI = 7
8	7	98.23	66.45	98.23 < 66.65 = false
9	7	98.23	35.64	98.23 < 35.64 = false

Ko‘rib o‘tilgan ketma-ketlik (algoritm) asosida bu masalaning dasturini C++ da tuzamiz:

```
#include <iostream>
using namespace std;
int main()
{
 int item[5];          // massiv elementlari beshta bo‘lsin
```

```
 int sum, counter;
 cout<<"Enter five numbers: ";
 sum=0;
 for(counter=0; counter<5; counter++){
cin>>item[counter];
sum=sum + item[counter];
}
cout<<endl;
cout<<" Massiv elementlari yig‘indisi: "<<sum<<endl;
 cout<<" Teskari tartibdagi massiv elemetlari: ";
for(counter=4; counter>=0; counter--) // Qiymatlarni teskari tartibda chop qilish
 cout<<item[counter]<< " ";
 cout<<endl;
 return 0;
}
```

Dasturni ishga tushirilib

12 76 34 52 89

qiymatlari kiritilsa, natija

Massiv elementlari yig‘indisi: 263

Teskari tartibdagi massiv elemetlari: 89 52 34 76 12

ko’rinishida bo’ladi.

Bir o‘lchovli massivlarni initsializatsiyalash

Massiv e’lonida uning elementlariga boshlang‘ich qiymatlar berish mumkin. Misol uchun elementlari 5 ta bo‘lgan haqiqiy turdagi sales massivi berilgan bo‘lsin. Unga boshlang’ich qiymatlarni berish quyidagicha amalga oshirilishi mumkin:

1) initsalizatsiya ro’yxati (*qiymatlar ketma-ketligi*) orqali :

```
double sales[5]= {12.25, 32.50, 16.90, 23, 45.68};
```

Bunda massivning har bir elementi ro’yxatdagi o’z indeksiga mos o’rindagi qiymatlarni qabul qiladi.

2) bevosita har elementga qiymat berish orqali:

sales[0]=12.25, sales[1]=32.50, sales[2]=16.90,

sales[3]=23.00, sales[4]=45.68 .

Massivning o‘lchamini bermagan holda ham uni initsializatsiya qilish mumkin, bunda massivning o‘lchami kompilyator tomonidan berilgan qiymatlar sonidan kelib chiqqan holda aniqlab olinadi.

```
double sales[]= {12.25, 32.50, 16.90, 23, 45.68};
```

Massiv elementlarini to‘liqmas initsializatsiya qilish

Massiv elementlarini e’lon qilish va bir vaqtda massivning barcha elementlarini initsializatsiya qilish mumkin, ammo massivni initsializatsiya qilishda uning qisman elementlarini initsializatsiya qilish mumkin bunga massivni to‘liqmas initsializatsiya qilish deyiladi.

Misollar:

1) `int list[10]={0};`

10 ta elementdan iborat bo‘lgan massiv e’lon qilingan bo’lib, uning barcha elementlari 0 qiymat qabul qiladi.

2) `int list[10]={8,5,12};`

bunda list[0]=8, list[1]=5 va list[2]=12 qiymatlar qabul qiladi, qolgan elementlari esa 0 qiymat qabul qiladi.

3) `int list[25]={4,7};`

bunday holda massivning birinchi ikkita elementlari 4 va 7 qiymatlarni qolgan elementlar 0 qiymat qabul qiladi.

Shuni qayd etish kerakki, massivning boshidagi yoki o‘rtasidagi elementlariga qiymatlar bermasdan, uning oxiridagi elementlariga 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.

Masala. Massivda musbat elemenlar soni va yig‘indisini hisoblash.

```
#include <iostream>
#include <conio.h>
using namespace std;
void main()
{
 int  x[]={-1,2,5,-4,8,9};
 clrscr();
 int s ,k, I;
 for (s=0,k=0, I=0; I<6; I++) {
  if (x[I]<=0) continue;
  k++;
  s+=x[I];
 };
cout << k << ' ' << s;
}
```

Massivning eng katta, eng kichik elementi va o‘rta qiymatini aniqlash:

```
#include <iostream>
using namespace std;
void main()
{
  int i,j,n;
  float a,b,d,x[100],s=0,max,min;
  while(1) {
   cout<< "\n n=";  cin>>n;
   if(n>0 && n<=100) break;
   cout<<"\n Xato  0<n<101 bulishi kerak";
   }
```

```
 cout << "\n elementlar kiymatlarini kiriting:\n";
 for(i=0; i<n; i++) {
  cout<<"x["<<i<<"]=";  cin>>x[i];
}
max=x[0];
min=x[0];
for(s=0,i=0;i<n;i++) {
 s+= x[i];
 if(max<x[i]) max=x[i];
 if(min>x[i]) min=x[i];
};
s/=n;
cout<<"\n max="<<max <<"\n min="<<min<<"\n o’rta qiymat="<<s;
}
```

Ko‘p o‘lchovli massivlarni initsializatsiyalash

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}};
intc[3][3][3]={{{0}},{{100,101},{110}},
{{200,201,202},{210,211,212},{220,221,222}};
```

Birinchi operatorda boshlang‘ich qiymatlar ketma-ket yozilgan, ikkinchi operatorda qiymatlar guruhlashgan, uchinchi operatorda ham guruhlashgan, lekin ba’zi guruhlarda oxirgi qiymatlar berilmagan.

Ikki o’lchovli massivlar matematikada matritsa yoki jadval tushunchasiga mos keladi. Jadvallarni initsializatsiya qilish qoidasi, ikki o‘lchovli massivning elementlari massivlardan iborat bo‘lgan bir o‘lchovli massiv ta’rifiga asoslangandir. Misol uchun 2 qator va 3 ustundan iborat bo‘lgan haqiqiy turga tegishli d massiv boshlang‘ich qiymatlari quyidagicha ko‘rsatilishi mumkin:

```
float d[2][3]={{1,-2.5,10}, {-5.3,2,14}};
```

Bu yozuv quyidagi qiymat berish operatorlariga mosdir:

d[0][0]=1; d[0][1]=-2.5; d[0][2]=10; d[1][0]=-5.3; d[1][1]=2; d[1][2]=14;

Bu qiymatlarni bitta ro‘yhat bilan hosil qilish mumkin:

float d[2][3]={1,-2.5,10,-5.3,2,14};

Initsializatsiya yordamida boshlang‘ich qiymatlar aniqlanganda massivning hamma elementlariga qiymat berish shart emas.

Misol uchun: int x[3][3]={{1,-2,3},{1,2},{-4}}.

Bu yozuv quyidagi qiymat berish operatorlariga mosdir:

x[0][0]=1; x[0][1]=-2; x[0][2]=3; x[1][0]=-1; x[1][1]=2; x[2][0]=-4;

Initsializatsiya yordamida boshlang‘ich qiymatlar aniqlanganda massivning birinchi indeksi chegarasi ko‘rsatilishi shart emas, lekin qolgan indekslar chegaralari ko‘rsatilishi shart.

Misol uchun:

double x[][2]={{1.1,1.5},{-1.6,2.5},{3,-4}};

Bu misolda avtomatik ravishda qatorlar soni 3 teng deb olinadi.

Quyida jadvalning har bir qatoridagi maksimal elementi aniqlash va bu elementlar orasida eng kichigi topish dasturining matni keltirilgan:

```
#include <iostream>
using namespace std;
const int n=4, m=3;
void main()
{
 double a[n][m],s,max[n], min;
 int i,j;
 for(i=0;i<n;i++)
  for (j=0;j<m;j++) {
   cout<<“a[”<<i<<“][”<<j<<“]=”; cin>>s;
   a[i][j]=s;
   }
```

```
}
for(i=0;i<n;i++)
{
 max[i]=a[i][0];
  for (j=1;j<m;j++)
   if (max[i]<a[i][j]) max[i]=a[i][j];
  };
min=max[0];
for(i=1; i<n; i++)
 if(min>max[i]) min=max[i];
cout <<"\n min="<< min;
}
```

Misol uchun, matritsani vektorga ko‘paytmasi C=A*b ni hisoblash masalasini ko‘raylik. Bu yerda $A=\{a_{ij}\}$, $b=\{b_j\}$, $C=\{c_{ij}\}$, $0 \le i<n$, $0 \le j<m$.

Hisoblash natijasida hosil bo‘ladigan C vektor elementlari

$$c_i = \sum_{j=0}^{n-1} a_{ij} b_j$$

formula bilan hisoblanadi.

Mos dastur matni quyidagi ko‘rinishda bo‘ladi:

```
void main()
{
 const int n=4,m=5;
 float a[n][m], b[m], c[n];
 int i,j;
 float s;
 for(i=0; i<n; i++)
  for(j=0; j<m; i++) cin>>a[i][j];
 for(i=0; i<m; i++) cin>>b[i];
```

```
 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];
 return;
}
```

Belgili massivlar

C ++ tilida satrlar belgili massivlar sifatida ta'riflanadi. Belgili massivlar quyidagicha ifodlanishi mumkin:

```
char pas[10];
```

Belgili massivlar quyidagicha initsializatsiya qilinadi:

```
char capital[]="TASHKENT";
```

Bunday holda massiv elementlari soni avtomatik ravishda aniqlanadi va massiv oxiriga satr ko'chirish '\0' belgisi (satr tugash belgisi) qo'shiladi.

Yuqoridagi initsializatsiyani quyidagicha amalga oshirish mumkin:

```
char capital[]={'T','A','S','H','K','E','N','T','\0'};
```

Bu holda so'z oxirida '\0' belgisi aniq ko'rsatilishi shart.

Misol uchun palindrom masalasini ko'raylik. Palindrom deb oldidan ham ohiridan ham bir hil o'qiladigan so'zlarga aytiladi. Misol uchun "*non*" satri. Dasturda kiritilgan satr palindrom ekanligi aniqlanadi:

```
#include <iostream>
using namespace std;
void main()
{
 char a[100];
 gets(a);       // a satrni kiritish
 int i=0, j=0;
 for(j=0; a[j]!='\0'; j++);
```

```
 while(i<j) if(a[i++]!=a[--j]) break;
 if(j-i>1) cout<<"Palindrom emas";
  else cout<<"Palindrom";
}
```

C++ tilida satr massivlari ikki o‘lchovli belgili massivlar sifatida ta’riflanadi. Misol uchun: char Name[4][5].

Bu e’londa har biri 5 ta harfdan iborat bo‘lgan 4 ta satrlar massivi e’lon qilingan. So‘zlar massivlari quyidagicha initsializatsiya qilinishi mumkin:

char Name[3][8]={“Anvar”, “Mirkomil”, “Yusuf”}.

Bu ta’rifda har bir so‘z uchun hotiradan 8 bayt joy ajratiladi va har bir so‘z oxiriga ‘\0’ belgisi qo‘yiladi.

Satrlar massivlari initsializatsiya qilinganda satrlar soni ko‘rsatilmasligi mumkin. Bu holda satrlar soni avtomatik aniqlanadi:

char comp[][9]={“kompyuter”, “printer”, “kartridj”}.

Quyidagi dasturda berilgan harf bilan boshlanuvchi satrlar ro’yxati bosib chiqariladi:

```
#include <iostream>
using namespace std;
void main()
{
 char a[10][10];
 char c;
 for (int i=0; i<10; i++) gets(a[i]);
 c=getchar();
 for(int i=0; i<10; i++)
  if(a[i][0]==c) puts(a[i]);
}
```

Nazorat savollari

1. Massiv deb nimaga aytiladi?
2. Massiv indeksi sifatida qanday son ishlatiladi?

3. Dasturda ishlatiladigan har bir konkret massiv qanday nomga ega?
4. Massiv elementiga murojaat qilish qanday amalga oshiriladi?
5. C++tilida massivlar elementining turiga cheklovlar kuyiladimi?
6. Ikki o‘lchamli massivning sintaksisi qanday ko’rinishda bo‘ladi?
7. So‘zlar massivlari initsializatsiya qilinganda so‘zlar soni ko‘rsatilmasligi mumkin. Bu holda so‘zlar soni qanday aniqlanadi?
8. Misolda massiv elementlar soni keltirilmagan bulsa massiv elementlar soni qanday aniklanadi?

§10. Funksiyalar e’lon qilish, aniqlash va ularga murojaat qilish

Funksiyalardan foydalanish

Dastur ta’minotini yaratish amalda murakkab jarayon hisoblanadi. Dastur tuzuvchi dastur kompleksini bir butunlikdagi va uning har bir bo‘lagining ichki mazmunini va ularning sezilmas farqlarini hisobga olishi kerak bo‘ladi.

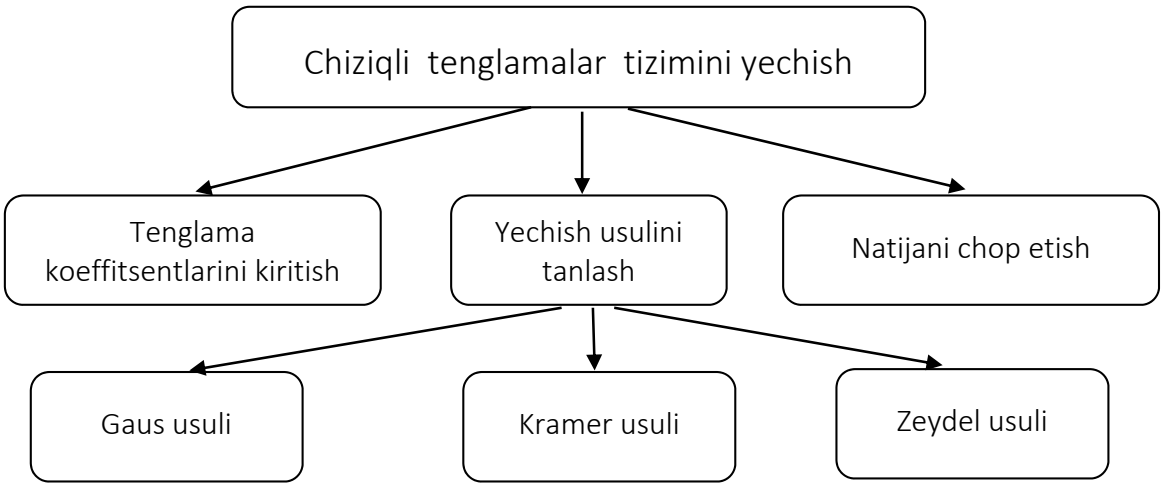
Dasturlashga strukturali yondoshuv shundan iboratki, dastur tuzuvchi oldiga qo‘yilgan masalani odatda bir nechta masala ostilarga bo‘ladi. O‘z navbatida bu masalaostilari ham yana kichik masalaostilariga bo‘linishi mumkin. Bu jarayon toki mayda masalalarni oddiy standart amallar yordamida yechish mumkin bo‘lguncha davom etadi. Shu yo‘l bilan masalani dekompozitsiysi amalga oshiriladi.

Ikkinchi tomondan, dasturlashda shunday holatlar kuzatiladiki, unda dasturning turli joylarida mazmunan bir xil algoritmlarni bajarishga to‘g‘ri keladi. Algoritmning bu bo‘laklari asosiy yechilayotgan masaladan ajratib olingan qandaydir masala ostini yechishga mo‘ljallangan bo‘lib, yetarlicha mustaqil qiymatga (natijaga) egadir. Misol uchun quyidagi masalani ko‘raylik:
Berilgan a0,a1,...,a30, b0,b1,...,b30, c0,c1,...,c30 va x,y,z haqiqiy sonlar uchun

$$\frac{\left(a_0x^{30}+a_1x^{29}+...+a_{30}\right)^2-\left(b_0y^{30}+b_1y^{29}+...+b_{30}\right)}{c_0(x+z)^{30}+c_1(x+z)^{29}+...+c_{30}}$$

ifodaning qiymati hisoblansin.

Bu misolni yechishda kasrning surat va maxrajidagi ifodalar bir xil algoritm bilan hisoblanadi va dasturda har bir ifodani (masala ostisini) hisoblash uchun bu algoritmni 3 marta yozishga to‘g‘ri keladi. Masaladagi 30-darajali ko‘phadni hisoblash algoritmini, masalan, Gorner algoritmini alohida, bitta nusxada yozib, unga turli parametrlar – bir safar a vektor va x qiymatini, ikkinchi safar b vektor va y qiymatini, hamda c vektor va (x+z) qiymatlari bilan murojaat qilish orqali asosiy masalani yechish mumkin bo‘ladi. Funksiyalar qo‘llanishining yana bir sababini quyidagi masalada ko‘rishimiz mumkin - berilgan chiziqli tenglamalar tizimini Gauss, Kramer, Zeydel usullarining birortasi bilan yechish talab qilinsin. U holda asosiy dasturni quyidagi bo‘laklarga bo‘lish maqsadga muvofiq bo‘lar edi:



Qo’yilgan masala tenglama koeffitsentlarini kiritish, yechish usulini tanlash, Gauss, Kramer va Zeydel usullarini amalga oshirish, hamda natijani chop qilish ko‘rinishida bo‘laklarga bo’linishi maqsadga muvofiqdir. Har bir bo‘lak uchun o‘z funksiyasi yaratib, zarur bo‘lganda ularga bosh funksiya tanasidan murojaatni amalga oshirish orqali masalani yechish samarali hisoblanadi.

Bunday hollarda dasturni ixcham va samarali qilish uchun C++tilida dastur bo‘lagini alohida ajratib olib, uni funksiya ko‘rinishida aniqlash imkoni mavjud.

Funksiya bu – C++tilida masala yechishdagi kalit elementlaridan biridir. Funksiyalar modullar deb ham ataladi. Funksiyalar oldindan aniqlangan va foydalanuvchi tomonidan aniqlanadigan funksiyalarga bo‘linadi.

Oldindan aniqlangan funksiyalar

Oldindan aniqlangan funksiyalar tilning turli kutubxona fayllari orqali aniqlangan. Ularga matematik funksiyalar, turlarni tekshirish funksiyalari, belgi va satrlar bilan ishlash funksiyalari misol bo‘ladi. Masalan:

Funksiya	**Kutubxona fayli**	**Bajaradigan amali**
abs(x)	<cmath>	x butun sonining absolyut qiymatini qaytaradi
fabs(x)	<cmath>	x haqiqiy sonining absolyut qiymatini qaytaradi
log(x)	<cmath>	x sonining natural logarifmini qaytaradi
pow(x,y)	<cmath>	x^yhisoblaydi
sqrt(x)	<cmath>	x sonining kvadrat ildizini qaytaradi
islower(x)	<cctype>	x qiymatini kichik harfligini tekshiradi
isupper(x)	<cctype>	x qiymatini katta harfligini tekshiradi
tolower(x)	<cctype>	x qiymatini kichik harf ko‘rinishiga aylantiradi
toupper(x)	<cctype>	x qiymatini katta harf ko‘rinishiga aylantiradi

Foydalanuvchi tomonidan aniqlanadigan funksiyalar

Dasturda ishlatiladigan har qanday foydalanuvchi tomonidan aniqlanadigan funksiyalar e’lon qilinishi kerak. Funksiyalar qiymat qaytaruvchi va qiymat qaytarmaydigan ko‘rinishida bo‘ladi.

Odatda funksiyalar e’loni sarlavha fayllarda e’lon qilinadi va #include direktivasi yordamida dastur matniga qo‘shiladi.

Funksiya e’lonini *funksiya prototipi* tavsiflaydi (ayrim hollarda *signatura* deyiladi). Funksiya prototipi quyidagi ko‘rinishda bo‘ladi:

<qaytaruvchi qiymat turi><funksiya nomi>(<parametrlar ro‘yxati >);

Bu yerda <qaytaruvchi qiymat turi> – funksiya ishlashi natijasida u tomonidan qaytaradigan qiymatning turi. Agar qaytariladigan qiymat turi

ko‘rsatilmagan bo‘lsa, kelishuv bo‘yicha funksiya qaytaradigan qiymat turi int deb hisoblanadi, <parametrlar ro‘yxati> – vergul bilan ajratilgan funksiya parametrlarining turi va nomlari ro‘yxati. Parametr nomini yozmasa ham bo‘ladi. Ro‘yxat bo‘sh bo‘lishi ham mumkin. Funksiya prototiplariga misollar:

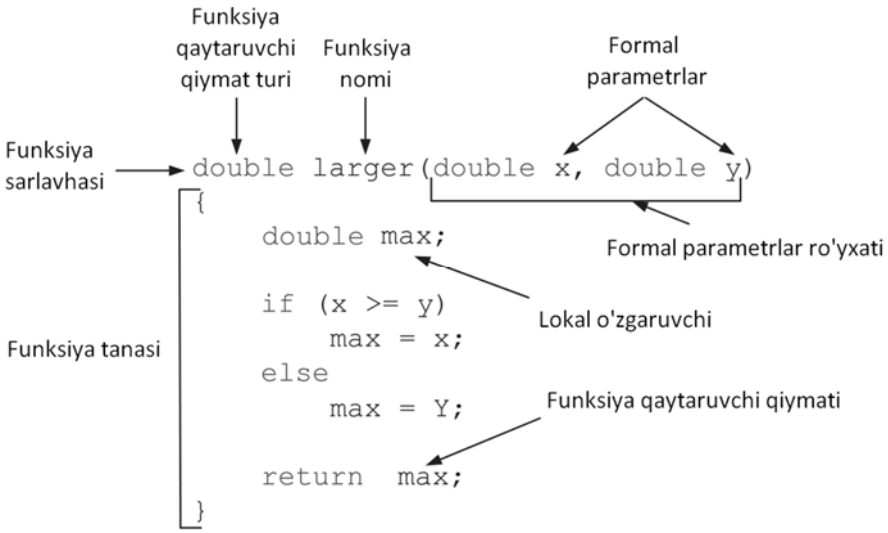
```
int almashsin(int, int);
double max(double x, double y);
void func();
void chop_etish(void);
```

Funksiya prototipi tushirib qoldirilishi mumkin, agar dastur matnida funksiya aniqlanishi uni chaqiradigan funksiyalar matnidan oldin yozilgan bo‘lsa. Lekin bu holat yaxshi uslub hisoblanmaydi, ayniqsa o‘zaro bir-biriga murojaat qiluvchi funksiyalarni e’lon qilishda muammolar yuzaga kelishi mumkin.

Funksiya aniqlanishi – funksiya sarlavhasi va figurali qavsga (‘{‘, ’}’) olingan qandaydir amaliy mazmunga ega tanadan iborat bo‘ladi. Agar funksiya qaytaruvchi turi void turidan farqli bo‘lsa, uning tanasida albatta mos turdagi parametrga ega return operatori bo‘lishi shart. Funksiya tanasida bittadan ortiq return operatori bo‘lishi mumkin. Ularning ixtiyoriy birortasini bajarish orqali funksiyadan chiqib ketiladi. Agar funksiyaning qiymati dasturda ishlatilmaydigan bo‘lsa, funksiyadan chiqish uchun parametrsiz return operatori ishlatilishi mumkin yoki umuman return ishlatilmaydi. Oxirgi holda funksiyadan chiqish - oxirgi yopiluvchi qavsga yetib kelganda ro‘y beradi.

Funksiya dasturning birorta modulida yagona ravishda aniqlanishi kerak, uning e’loni esa funksiyani ishlatadigan modullarda bir necha marta yozilishi mumkin. Funksiya aniqlanishida sarlavhadagi barcha parametrlar nomlari yozilishi shart.

Odatda dasturda funksiya ma’lum bir ishni amalga oshirish uchun chaqiriladi. Funksiyaga murojaat qilganda, u qo‘yilgan masalani yechadi va o‘z ishini tugatishida qandaydir qiymatni natija sifatida qaytaradi.



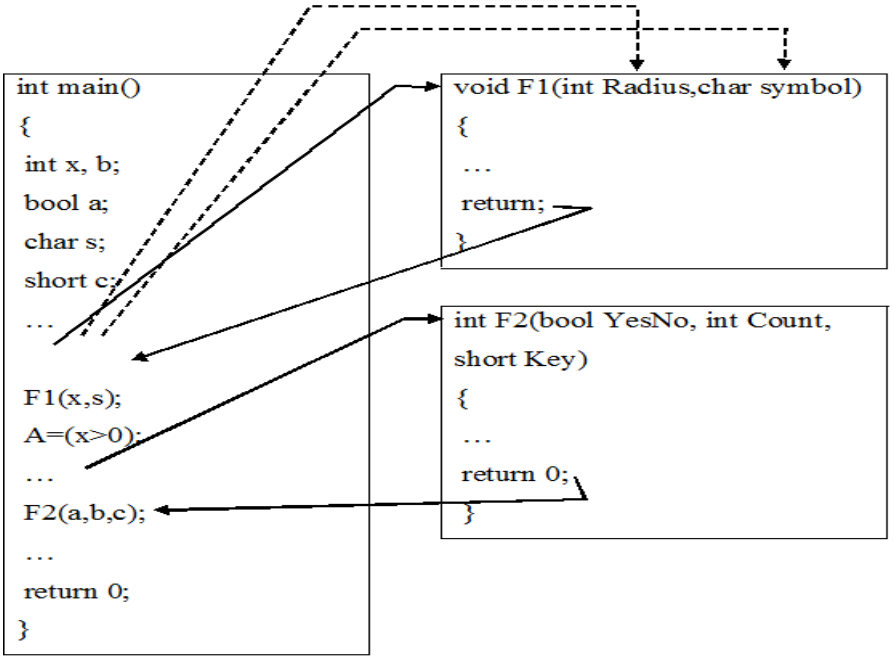
Funksiyaga murojaat qilish uchun uning nomi va undan keyin qavs ichida argumentlar ro‘yxati beriladi:

<funksiya nomi>(<argument1>, <argument2>,…,<argumentn >);

Bu yerda har bir <argument>– funksiya tanasiga uzatiladigan va keyinchalik hisoblash jarayonida ishlatiladigan o‘zgaruvchi, ifoda yoki o‘zgarmasdir. Argumentlar ro‘yxati bo‘sh bo‘lishi mumkin.

Oldingi boblarda ta’kidlab o‘tilganidek, C++ tilidagi har qanday dasturda albatta main() bosh funksiyasi bo‘lishi kerak. Ayni shu funksiyani yuklagich tomonidan chaqirilishi bilan dastur bajarilishi boshlanadi.

Quyidagi rasmda bosh funksiyadan boshqa funksiyalarga murojaat qilish va ulardan qaytish sxemasi ko‘rsatilgan.



Dastur main() funksiyasini bajarishdan boshlanadi va F1(x,s)– funksiyaga murojaatgacha davom etadi va keyinchalik boshqaruv F1() funksiya tanasidagi amallarni bajarishga o‘tadi. Bunda Radius parametrining qiymati sifatida funksiya x o‘zgaruvchi qiymatini, symbol parametri sifatida s o‘zgaruvchisining qiymati ishlatiladi. Funksiya tanasi return operatorigacha bajariladi. return operatori boshqaruvni main() funksiyasi tanasidagi F1() funksiyasiga murojaat qilingan operatordan keyingi operatorga o‘tishni ta’minlaydi, ya’ni funksiyadan qaytish ro‘y beradi. Shundan keyin main() funksiyasi operatorlari bajarilishda davom etadi va F2(a,b,c)–funksiyaga murojaati orqali boshqaruv F2() funksiya tanasiga o‘tadi va hisoblash jarayonida mos ravishda YesNo sifatida a o‘zgaruvchisining, Count sifatida b o‘zgaruvchicining va Key sifatida c o‘zgaruvchisining qiymatlari ishlatiladi. Funksiya tanasidagi return operatori yoki oxirgi operator bajargandan keyin bosh funksiyaga qaytish amalga oshiriladi.

Aksariyat hollarda main() funksiyasining parametrlar ro‘yxati bo‘sh bo‘ladi. Agar yuklanuvchi dasturni ishga tushirishda, buyruq satri orqali yuklanuvchi dastur ishga tushirilganda, unga parametrlarni uzatish (berish) zarur bo‘lsa, main() funksiyasining sintaksisi o‘zgaradi:

```
int main(int argc, char* argv[]);
```

bu yerda argc– uzatiladigan parametrlar soni, argv[] – bir-biridan punktuatsiya belgilari (va probel) bilan ajratilgan parametrlar ro‘yxatini o‘z ichiga olgan massivga ko‘rsatkich.

Quyida funksiyalarni e’lon qilish, funksiyalarga murojaat qilish va aniqlashga misollar keltirilgan:

```
// funksiyalar e’loni
int Mening_funksiyam(int Number, float Point);
char Belgini_uqish();
void bitni_urnatish(short Num);
void Amal_yoq(int,char);
```

```
// funksiyalarga murojaat qilish
result = Mening_funksiyam(Varb1, 3.14);
symb = Belgini_uqish();
bitni_urnatish(3);
Amal_yoq(2, Smbl);
// funksiyalarni aniqlash
int Mening_funksiyam(int Number, float Point)
{
int x;
...
return x;
}
char Belgini_uqish()
{
char Symbol;
cin >> Symbol;
return Symbol;
};
void bitni_urnatish(short number)
{
global_bit = global_bit | number;
};
void Amal_yoq(int x, char ch){};
```

Funksiyaning dasturdagi o‘rnini yanada tushunarli bo‘lishi uchun son kvadratini hisoblash masalasida funksiyadan foydalanishni ko‘raylik. Bunda funksiya prototipini sarlavha.h sarlavha faylida joylashtiramiz:

```
long Son_Kvadrati(int);
```

Asosiy dasturga ushbu sarlavha faylini qo‘shish orqali Son_Kvadrati() funksiya e’loni dastur matniga kiritiladi:

```
#include <iostream>
using namespace std;
#include "sarlavha.h"
int main()
{
int Uzgaruvchi=5;
cout<<Son_Kvadrati(Uzgaruvchi);
return 0;
}
long Son_Kvadrati(int x) {return x*x;}
```

Xuddi shu masalani sarlavha faylidan foydalanmagan holda, funksiya e’lonini dastur matniga yozish orqali ham hal qilish mumkin:

```
#include <iostream>
using namespace std;
long Son_Kvadrati(int);
int main()
{
int Uzgaruvchi=5;
cout<<Son_Kvadrati(Uzgaruvchi);
return 0;
}
long Son_Kvadrati(int x){ return x*x;}
```

Dastur ishlashida o‘zgarish bo‘lmaydi va natija sifatida ekranga 25 sonini chop etadi.

Masala. Ikkita tub son “*egizak*” deyiladi, agar ular bir-biridan 2 soniga farq qilsa (masalan, 41 va 43 sonlari). Berilgan natural n uchun [n..2n]

oraliqdagi barcha “*egizak*” sonlar juftliklari chop etilsin. Masalani yechish uchun berilgan k sonini tub son yoki yo‘qligini aniqlaydigan mantiqiy funksiyani tuzish zarur bo‘ladi. Funksiyada k soni 2..k/2 gacha sonlarga bo‘linadi, agar k bu sonlarning birortasiga ham bo‘linmasa, u tub son hisoblanadi va funksiya true qiymatini qaytaradi. Bosh funksiyada, berilgan n uchun [n..2n] oraliqdagi (n, n+2), (n+1, n+3), ..., (2n-2, 2n) son juftliklarini tub sonlar ekanligi tekshiriladi va shartni qanoatlantirgan juftliklar chop etiladi.

Dastur matni:

```
bool TubSon(unsigned long k);
int main()
{
  unsigned long n,i;
  unsigned char egizak=0;
  cout<<"n -> ";   cin>>n;
  cout<<'['<<n<<".."<<2*n<<']';
  for(i=n; i<=2*n-2; i++)
   if(TubSon(i) && TubSon(i+2)){
    if (!egizak) cout<<" oralig'idagi egizak tub sonlar:\n";
else cout<<"; ";
egizak=1;
cout<<'{'<<i<<','<<i+2<<'}';
};
  if(!egizak)  cout<<" oralig'ida egizak tub sonlar mavjud emas.";
else cout<<'.';
return 0;
}
bool TubSon(unsigned long k)
{
```

```
 unsigned long m;
 for (m=2; m<=k/2; m++)
 if (k%m==0) return false;
 return true;
}
```

Natural n soni uchun 100 kiritilsa, dastur quyidagi sonlar juftliklarini chop qiladi:

[100..200] oralig'idagi egizak tub sonlar:

{101,103}; {107,109}; {137,139}; {149,151}; {179,181}; {191,193}; {197,199}.

Kelishuv bo‘yicha argumentlar

C++ tilida funksiyaga murojaat qilinganda ayrim argumentlarni tushirib qoldirish mumkin. Bunga funksiya prototipida ushbu parametrlarni kelishuv bo‘yicha qiymatini ko‘rsatish orqali erishish mumkin. Masalan, quyida prototipi keltirilgan funksiya turli chaqirishga ega bo‘lishi mumkin:

```
void Butun_Son(int I, bool Bayroq=true, char Blg='\n');  // funksiya prototipi
//funksiyaga murojaat variantlari
Butun_Son(1, false, 'a');
Butun_Son(2, false);
Butun_Son(3);
```

Birinchi murojaatda barcha parametrlar mos argumentlar orqali qiymatlarini qabul qiladi, ikkinchi holda I parametri 2 qiymatini, bayroq parametri false qiymatini va Blg o‘zgaruvchisi kelishuv bo‘yicha ‘\n’ qiymatini qabul qiladi.

Kelishuv bo‘yicha qiymat berishning bitta sharti bor – parametrlar ro‘yxatida kelishuv bo‘yicha qiymat berilgan parametrlardan keyingi parametrlar ham kelishuv bo‘yicha qiymatga ega bo‘lishlari shart. Yuqoridagi misolda I parametri kelishuv bo‘yicha qiymat qabul qilingan holda, Bayroq yoki Blg parametrlari qiymatsiz bo‘lishi mumkin emas. Misol tariqasida berilgan sonni ko‘rsatilgan aniqlikda chop etuvchi dasturni ko‘raylik. Qo‘yilgan masalani

yechishda sonni darajaga oshirish funksiyasi – pow() va suzuvchi nuqtali uzun sondan modul olish fabsl() funksiyasidan foydalaniladi. Bu funksiyalar prototipi “cmath” sarlavha faylida joylashgan:

```
#include <iostream>
using namespace std;
#include <cmath>
void Chop_qilish(double Numb, double Aniqlik=1, bool Bayroq=true);
int main()
{
 double Mpi=-3.141592654;
 Chop_qilish(Mpi, 4, false);
 Chop_qilish(Mpi, 2);
 Chop_qilish(Mpi);
 return 0;
}
void Chop_qilish(double Numb,double Aniqlik=1, bool Bayroq = true)
{
 if(!Bayroq)Numb=fabsl(Numb);
 Numb=(int)(Numb*pow(10,Aniqlik));
 Numb=Numb/pow(10,Aniqlik);
 cout<<Numb<<‘\n’;
}
```

Dasturda sonni turli aniqlikda (Aniqlik parametri qiymati orqali) chop etish uchun har xil variantlarda Chop_qilish() funksiyasga murojaat qilingan. Dastur ishlashi natijasida ekranga quyidagi sonlar chop etiladi:

```
3.1415
-3.14
-3.1
```

Parametrning kelishuv bo‘yicha beriladigan qiymati o‘zgarmas, global o‘zgaruvchi yoki qandaydir funksiya tomonidan qaytaradigan qiymat bo‘lishi mumkin.

Ko‘rinish sohasi. Lokal va global o‘zgaruvchilar

O‘zgaruvchilar funksiya tanasida yoki undan tashqarida e’lon qilinishi mumkin. Funksiya ichida e’lon qilingan o‘zgaruvchilarga *lokal o‘zgaruvchilar* deyiladi. Bunday o‘zgaruvchilar xotiradagi dastur stekida joylashadi va faqat o‘zi e’lon qilingan funksiya tanasida amal qiladi. Boshqaruv asosiy funksiyaga qaytishi bilan lokal o‘zgaruvchilar uchun ajratilgan xotira bo‘shatiladi (o‘chiriladi).

Har bir o‘zgaruvchi o‘zining amal qilish sohasi va yashash vaqti xususiyatlari bilan xarakterlanadi.

O‘zgaruvchi *amal qilish sohasi* deganda o‘zgaruvchini ishlatish mumkin bo‘lgan dastur sohasi (qismi) tushuniladi. Bu tushuncha bilan o‘zgaruvchining *ko‘rinish sohasi* uzviy bog‘langan. O‘zgaruvchi amal qilish sohasidan chiqqanda ko‘rinmay qoladi. Ikkinchi tomondan, o‘zgaruvchi amal qilish sohasida bo‘lishi, lekin ko‘rinmasligi mumkin. Bunda ko‘rinish sohasiga ruxsat berish amali “::” yordamida ko‘rinmas o‘zgaruvchiga murojat qilish mumkin bo‘ladi.

O‘zgaruvchining *yashash vaqti* deb, u mavjud bo‘lgan dastur bo‘lagining bajarilishiga ketgan vaqt intervaliga aytiladi.

Lokal o‘zgaruvchilar o‘zlari e’lon qilingan funksiya yoki blok chegarasida ko‘rinish sohasiga ega. Blokdagi ichki bloklarda xuddi shu nomdagi o‘zgaruvchi e’lon qilingan bo‘lsa, ichki bloklarda bu lokal o‘zgaruvchi ham amal qilmay qoladi. Lokal o‘zgaruvchi yashash vaqti - blok yoki funksiyani bajarish vaqti bilan aniqlanadi. Bu hol shuni anglatadiki, turli funksiyalarda bir-biriga umuman bog‘liq bo‘lmagan bir xil nomdagi lokal o‘zgaruvchilarni ishlatish mumkin.

Quyidagi dasturda main() va sum() funksiyalarida bir xil nomdagi o‘zgaruvchilarni ishlatish ko‘rsatilgan. Dasturda ikkita sonning yig‘indisi hisoblanadi va chop etiladi:

```
#include <iostream>
```

```
using namespace std;
int sum(int a, int b);        // funksiya prototipi
int main()
{
 int x=1;  int y=4;           // lokal o‘zgaruvchilar
 cout << sum(x, y);
 return 0;
}
int sum(int a,int b)
{
 int x=a+b;                   // lokal o‘zgaruvchi
 return x;
}
```

Global o‘zgaruvchilar dastur matnida funksiya aniqlanishidan tashqarida e’lon qilinadi va e’lon qilingan joyidan boshlab dastur oxirigacha amal qiladi.

```
#include <iostream>
using namespace std;
int f1();
int f2();
int main()
{
 cout<<f1()<<" "<<f2()<<endl;
 return 0;
}
int f1()
{
 return x;     // kompilyatsiya xatosi ro‘y beradi
}
```

```
int x=10;      // global o‘zgaruvchi e’loni
int f2()
{
 return x*x;
}
```

Yuqorida keltirilgan dasturda kompilyatsiya xatosi ro‘y beradi, chunki f1() funksiya uchun x o‘zgaruvchisi noma’lum hisoblanadi.

Dastur matnida global o‘zgaruvchilarni ular e’lonidan keyin yozilgan ixtiyoriy funksiyada ishlatish mumkin. Shu sababli, global o‘zgaruvchilar dastur matnining boshida yoziladi. Funksiya ichidan global o‘zgaruvchiga murojat qilish uchun funksiyada uning nomi bilan mos tushadigan lokal o‘zgaruvchilar bo‘lmasligi kerak. Agar global o‘zgaruvchi e’lonida unga boshlang‘ich qiymat berilmagan bo‘lsa, ularning qiymati 0 hisoblanadi. Global o‘zgaruvchining amal qilish sohasi uning ko‘rinish sohasi bilan ustma-ust tushadi.

Shuni qayd etish kerakki, tajribali dastur tuzuvchilar imkon qadar global o‘zgaruvchilarni ishlatmaslikka harakat qilishadi, chunki bunday o‘zgaruvchilar qiymatini dasturning ixtiyoriy joyidan o‘zgartirish xavfi mavjudligi sababli dastur ishlashida mazmunan xatolar yuzaga kelishi mumkin. Bu fikrimizni tasdiqlovchi dasturni ko‘raylik.

```
#include <iostream>
using namespace std;
// global o‘zgaruvchi e’loni
int test = 100;
void Chop_qilish(void);
int main()
{
 int test=10;          // lokal o‘zgaruvchi e’loni
 Chop_qilish();        // global o‘zgaruvchi chop qilish funksiyasini chaqirish
cout << “Lokal o’zgaruvchi: ” << test << ‘\n’;
```

```
return 0;
}
void Chop_qilish(void)
{
 cout<< “Global o’zgaruvchi: ” << test << ‘\n’;
}
```

Dastur boshida test global o‘zgaruvchisi 100 qiymati bilan e’lon qilinadi. Keyinchalik, main() funksiyasida test nomi bilan lokal o‘zgaruvchisi 10 qiymati bilan e’lon qilinadi. Dasturda, Chop_qilish() funksiyasiga murojaat qilinganida, asosiy funksiya tanasidan vaqtincha chiqiladi va natijada main() funksiyasida e’lon qilingan barcha lokal o‘zgaruvchilarga murojaat qilish mumkin bo‘lmay qoladi. Shu sababli Chop_qilish() funksiyasida global test o‘zgaruvchisining qiymatini chop etiladi. Asosiy dasturga qaytilgandan keyin, main() funksiyasidagi lokal test o‘zgaruvchisi global test o‘zgaruvchisini “*berkitadi*” va lokal test o‘zgaruvchini qiymati chop etiladi. Dastur ishlashi natijasida ekranga quyidagi natijalar chop etiladi:

```
Global o‘zgaruvchi: 100
Lokal o‘zgaruvchi: 10
```

:: amali

Yuqorida qayd qilinganidek, lokal o‘zgaruvchi e’loni xuddi shu nomdagi global o‘zgaruvchini “*berkitadi*” va bu joydan global o‘zgaruvchiga murojat qilish imkoni bo‘lmay qoladi. C++ tilida bunday holatlarda ham global o‘zgaruvchiga murojat qilish imkoniyati saqlanib qolingan. Buning uchun “*ko‘rinish sohasiga ruxsat berish*” amalidan foydalanish mumkin va o‘zgaruvchi oldiga ikkita nuqta – “::” qo‘yish zarur bo‘ladi. Misol tariqasida quyidagi programani keltiramiz:

```
#include <iostream>
using namespace std;
//global o‘zgaruvchi e’loni
```

```
int uzg=5;
int main()
{
int uzg=70; // lokal o‘zgaruvchi e’loni
cout << uzg << ‘/n’;   // lokal o‘zgaruvchini chop etish
cout << ::uzg << ‘/n’;  //global o‘zgaruvchini chop etish
return 0;
}
```

Dastur ishlashi natijasida ekranga oldin 70 va keyin 5 sonlari chop etiladi.

Joylashtiriladigan (inline) funksiyalar

Kompilyator ishlashi natijasida har bir funksiya mashina kodi ko‘rinishida bo‘ladi. Agar dasturda funksiyaga murojaat ko‘rsatmasi bo‘lsa, shu joyda funksiyani adresi bo‘yicha chaqirishning mashina kodi shakllanadi. Odatda funksiyani chaqirish protsessor tomonidan qo‘shimcha vaqt va xotira resurslarini talab qiladi. Shu sababli, agar murojaat qilinadigan funksiya hajmi unchalik katta bo‘lmagan hollarda, kompilyatorga funksiyani chaqirish kodi o‘rniga funksiya tanasini joylashtirishga ko‘rsatma berish mumkin. Bu ish funksiya prototipini inline kalit so‘zi bilan e’lon qilish orqali amalga oshiriladi. Natijada hajmi oshgan, lekin nisbatan tez bajariladigan dastur kodi yuzaga keladi.

Funksiya kodi joylashtiriladigan dasturga misol.

```
#include <iostream>
using namespace std;
inline int Summa(int, int);
int main()
{
 int a=2,b=6,c=3;
 char yangi_qator = ‘\n’;
 cout << Summa(a,b) << yangi_qator;
 cout << Summa(a,c) << yangi_qator;
```

```
 cout << Summa(b,c) << yangi_qator;
 return 0;
}
int Summa(int x, int y){ return x+y;}
```

Keltirilgan dastur kodini hosil qilishda Summa() funksiyasiga murojaat qilingan joylarga uning tanasidagi buyruqlar joylashtiriladi.

Qayta yuklanuvchi funksiyalar

Ayrim algoritmlar berilganlarning har xil turdagi qiymatlari uchun qo‘llanishi mumkin. Masalan, ikkita sonning maksimumini topish algoritmida bu sonlar butun yoki haqiqiy turda bo‘lishi mumkin. Bunday hollarda bu algoritmlarni amalga oshiradigan funksiyalarni nomlari bir xil bo‘lgani ma’qul. Bir nechta funksiyani bir xil nomlash, lekin har xil turdagi parametrlar bilan ishlatish *funksiyani qayta yuklash* deyiladi.

Kompilyator parametrlar turiga va soniga qarab mos funksiyaga murojaatni amalga oshiradi. Bunday amalni “*hal qilish amali*” deyiladi va uning maqsadi parametrlarga ko‘ra aynan (nisbatan) to‘g‘ri keladigan funksiyaga murojaat qilishdir. Agar bunday funksiya topilmasa kompilyator xatolik haqida xabar beradi. Funksiyani aniqlashda funksiya qaytaruvchi qiymat turining ahamiyati yo‘q. Misol:

```
#include <iostream>
using namespace std;
int max(int,int);
char max(char,char);
float max(float,float)
int max(int,int,int);
void main()
{
int a, b, k; char c, d; float x,y;
 cin>>a>>b>>k>>c>>d>>x>>y;
```

```
 cout<<max(a,b)<<max(c,d)<<max(a,b,k)<<max(x,y);
}
int max(int i,int j){return (i>j)?i:j;}
char max(char s1,char s2){return (s1>s2)?s1:s2;}
float max(float x,float y){return (x>y)?x:y;}
int max(int i,int j,int k){return (i>j)?(i>k? i:k;):((j>k)?j:k;);}
```

Agar funksiyaga murojaat qilinganida argument turi uning prototipidagi xuddi shu o‘rindagi parametr turiga mos kelmasa, kompilyator uni parametr turiga keltirilishga harakat qiladi – bool va char turlarini int turiga, float turini double turiga va int turini double turiga.

Qayta yuklanuvchi funksiyalardan foydalanishda quyidagi qoidalarga rioya qilish kerak:

– qayta yuklanuvchi funksiyalar bitta ko‘rinish sohasida bo‘lishi kerak;

– qayta yuklanuvchi funksiyalarda kelishuv bo‘yicha parametrlar ishlatilsa, bunday parametrlar barcha qayta yuklanuvchi funksiyalarda ham ishlatilishi va ular bir xil qiymatga ega bo‘lish kerak;

– agar funksiyalar parametrlarining turi faqat const va ‘&’ belgilari bilan farq qiladigan bo‘lsa, bu funksiyalar qayta yuklanmaydi.

Nazorat savollari

1. Funksiya bu…?
2. Funksiyalar modullar deb ham atalishi mumkunmi?
3. C++ tilida funksiya chaqirilganda ayrim argumentlarni tushirib qoldirish mumkunmi va bunga qanday erishish mumkun?
4. Lokal o‘zgaruvchilar o‘zlari e’lon qilingan funksiya yoki blok chegarasida ko‘rinish sohasiga ega buladimi?
5. Funksiyalar qanday turlarga bo‘linadi?
6. Kelishuv bo‘yicha qiymat berishning nechta sharti bor?

7. Qayta yuklanuvchi funksiyalardan foydalanishda qanday qoidalarga rioya qilish kerak?
8. Funksiyalar qanday ko‘rinishda bo‘ladi?
9. Funksiya qanday aniklanadi?
10. Kompilyator ishlashi natijasida har bir funksiya qanday ko‘rinishida bo‘ladi?

§11. Rekursiv funksiyalar

Odatda rekursiya matematikada keng qo‘llaniladi. Chunki aksariyat matematik formulalar rekursiv aniqlanadi. Misol tariqasida faktorialni hisoblash formulasini

$$n! = \begin{cases} 1, & \text{arap } n = 0; \\ n * (n-1)!, & \text{arap } n > 0, \end{cases}$$

va sonning butun darajasini hisoblashni ko‘rishimiz mumkin:

$$x^n = \begin{cases} 1, & \text{arap } n = 0; \\ x * x^{n-1}, & \text{arap } n > 0. \end{cases}$$

Ko‘rinib turibdiki, navbatdagi qiymatni hisoblash uchun funksiyaning “*oldingi qiymati*” ma’lum bo‘lishi kerak. C++ tilida rekursiya matematikadagi rekursiyaga o‘xshash.

Quyidagi masala qaralsin: matematikada manfiy bo‘lmagan butun sonlarning faktorialini aniqlash quyidagi formula yordamida amalga oshiriladi:

0! = 1 (1)

n! = n * (n-1)! (2)

Agar n=3 bo‘lsa masala quyidagi formulalar yordamida ishlanadi:

3! = 3*2!

2! = 2*1!

1! = 1*0!

0! = 1

(1) formulada ifodaning o‘ng qismida faktorialni hisoblash mavjud bo‘lmaganligi uchun u 1 ga teng deb olinadi. (2) formulada esa ifodaning o‘ng

qismida yana faktorialni hisoblash kerak. (1) va (2) formulalar rekursiv formulalar deyiladi. (1) ifoda asos ifoda, (2) ifoda umumiy ifoda deyiladi.

Rekursiya deb funksiya tanasida shu funksiyaning o‘zini chaqirishiga aytiladi. Rekursiya uchun quyidagi aniqlanishlar o‘rinli:

1. Har bir rekursiv formula kamida bitta asos ifodaga ega bo‘lishi kerak.
2. Umumiy ifoda doim asos ifodaga yo‘naltirilgan bo‘lishi kerak.
3. Asos ifoda rekursiyani to‘xtatishi kerak.

Buni yuqoridagi misollar uchun tuzilgan funksiyalarda ko‘rish mumkin. Faktorial uchun:

```
long F(int n)
{
 if (n == 0) return 1;
 else return n * F(n-1);
}
```

Berilgan haqiqiy x soning n - darajasini hisoblash funksiyasi:

```
double Butun_Daraja(double x, int n)
{
if (n == 0) return 1;
 else return x * Butun_Daraja(x, n-1);
}
```

Agar factorial funksiyasiga n>0 qiymat berilsa, quyidagi holat ro‘y beradi: shart operatorining elses hoxidagi qiymati (n qiymati) stekda eslab qolinadi. Hozircha qiymati noma’lum n-1 faktorialni hisoblash uchun shu funksiyaning o‘zi n-1 qiymati bilan chaqiriladi. O‘z navbatida, bu qiymat ham eslab qolinadi (stekka joylanadi) va yana funksiya chaqiriladi va hakoza. Funksiya n=0 qiymat bilan chaqirilganda if operatorining sharti (n == 0) rost bo‘ladi va “return 1;” amali bajarilib, ayni shu chaqirish bo‘yicha 1 qiymati qaytariladi. Shundan keyin “*teskari*” jarayon boshlanadi – stekda saqlangan qiymatlar ketma-ket olinadi va ko‘paytiriladi: oxirgi qiymat – aniqlangandan keyin (1), u undan oldingi

saqlangan qiymatga 1 qiymatiga ko‘paytirib F(1) qiymati hisoblanadi, bu qiymat 2 qiymatiga ko‘paytirish bilan F(2) topiladi va hakoza. Jarayon F(n) qiymatini hisoblashgacha “*ko‘tarilib*” boradi. Bu jarayonni, n=4 uchun faktorial hisoblash sxemasini quyida ko‘rish mumkin:

↓	F(4)=4*F(3)
↓	F(3)=3*F(2)
↓	F(2)=2*F(1)
↓	F(1)=1*F(0)
↑	F(0)=1

↓	F(4)=4*F(3)
↓	F(3)=3*F(2)
↓	F(2)=2*F(1)
↑	F(1)=1*1

↓	F(4)=4*F(3)
↓	F(3)=3*F(2)
↑	F(2)=2*1

↓	F(4)=4*F(3)
↑	F(3)=3*2

↑	F(4)=4*6

Rekursiv funksiyalarni to‘g‘ri amal qilishi uchun rekursiv chaqirishlarning to‘xtash sharti bo‘lishi kerak. Aks holda rekursiya to‘xtamasligi va o‘z navbatida funksiya ishi tugamasligi mumkin. Faktorial hisoblashida rekursiv tushishlarning to‘xtash sharti funksiya parametri n=0 bo‘lishidir (shart operatorining rost shoxi).

Rekursiya ikki xil bo‘ladi:

1. *oddiy*– agar funksiya o‘z tanasida o‘ziga murojaat qilsa;

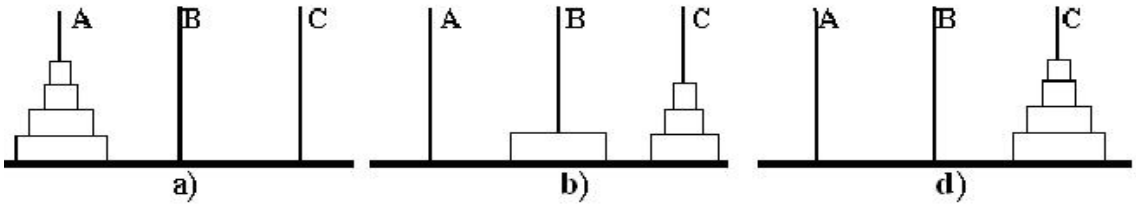
2. *vositali* – agar birinchi funksiya ikkinchi funksiyaga murojaat qilsa, ikkinchisi esa o‘z navbatida birinchi funksiyaga murojaat qilsa.

Har bir rekursiv murojaat qo‘shimcha xotira talab qiladi – funksiyalarning lokal obektlari (o‘zgaruvchilari) uchun har bir murojaatda stekdan yangidan joy ajratiladi. Masalan, rekursiv funksiyaga 100 marta murojaat bo‘lsa, jami 100 lokal obektlar uchun joy ajratiladi. Ayrim hollarda, ya’ni rekursiyalar soni etarlicha katta bo‘lganda, stek o‘lchami cheklanganligi sababli (real rejimda 64Kb o‘lchamgacha) u to‘lib ketishi mumkin. Bu holatda dastur o‘z ishini “*Stek to‘lib ketdi*” xabari bilan to‘xtadi.

“*Xanoy minorasi*”*masalasi*

Uchta A, B, C qoziq va n-ta har xil o‘lchamli xalqalar mavjud. Xalqalarni o‘lchamlari o‘sish tartibida 1 dan n gacha tartiblangan. Qolgan barcha xalqalar

A qoziqqa rasmdagidek joylashtirilgan. A qoziqdagi barcha xalqalarni C qoziqqa, yordamchi B qoziqdan foydalangan holda, quyidagi qoidalarga amal qilgan holda o‘tkazish talab etiladi: xalqalarni bittadan ko‘chirish kerak va katta o‘lchamli xalqani kichik o‘lchamli xalqa ustiga qo‘yish mumkin emas.



Amallar ketma-ketligini chop etadigan (“*halqa q dan r ga o‘tkazilsin”* ko‘rinishida, bunda q va r – ’A’, ’B’ yoki ’C’) masalani n ta halqa uchun yechadigan ilova tuzilsin.

Ko‘rsatma: halqalarni A dan C ga to‘g‘ri o‘tkazishda quydagi rasmda keltirilgan 5-qadamdagidek holat uchraydi, y’ani n ta xalqani ‘A’ dan ‘C’ ga o’tkazish masalasi n-1 xalqani ‘B’ dan ‘C’ ga o’tkazish masalasiga keladi.

```
#include <iostream>
using namespace std;
void Hanoy(int n, char a = 'A', char b = 'C', char c = 'B')
{
 if(n) {
   Hanoy(n-1, a, c, b);
   cout << "Xalqa " << a << " dan " << b<< " ga o'tkazilsin\n";
  Hanoy(n-1, c, b, a);
 }
}
int main()
{
 unsigned int Xalqalar_Soni;
 cout << "Hanoy minorasi masalasi" << endl;
 cout << "Xalqalar sonini kiriting: "; cin >> Xalqalar_Soni;
```

```
 Hanoy(Xalqalar_Soni);
 return 0;
}
```

Xalqalar soni 3 bo‘lganda (Xalqalar_Soni=3) dastur ekranga xalqalarni ko‘chirish bo‘yicha amallar ketma-ketligini chop etadi:

Xalqa A dan C ga o’tkazilsin

Xalqa A dan B ga o’tkazilsin

Xalqa C dan B ga o’tkazilsin

Xalqa A dan C ga o’tkazilsin

Xalqa B dan A ga o’tkazilsin

Xalqa B dan C ga o’tkazilsin

Xalqa A dan C ga o’tkazilsin

Tahlil qilib ko‘rilsa, A qoziqdagi barcha xalqalarni C qoziqqa o‘tkazish uchun 2^3-1=7 ta jarayon bajarildi. Xalqalar soni 64ta bo‘lganda bu jarayonlar soni 2^{64}-1 ga teng bo‘ladi.

$2^{10}= 1024\approx 1000 = 10^3$

Bundan kelib chiqadiki:

$2^{64}= 2^4\times2^{60}\approx2^4\times10^{18}=1.6\times10^{19}$

Bir yildagi sekundlar soni 3.2×10^7 ga teng. Bir dona diskni bir qoziqdan boshqasiga olib o‘tish uchun bir sekund sarflanadi deb hisoblansa quyidagiga kelish mumkin:

$1.6 \times 10^{19} = 5 \times 3.2 \times 10^{18}= (3.2 \times 10^7) \times (5 \times 10^{11})$

Barcha 64ta diskni A qoziqdan C qoziqqa olib o‘tish uchun (5×10^{11}) yil kerak bo‘ladi. Kompyuter bir sekundda bir milliard (10^9) operatsiya bajara oladi deb hisoblansa, bir yilda quyidagicha operatsiya bajara oladi:

$(3.2 \times 10^7) \times 10^9= 3.2 \times 10^{16}$

64 ta diskni A qoziqdan C qoziqqa olib o‘tish uchun kompyuterga quyidagicha miqdorda vaqt kerak bo‘ladi:

$2^{64} \approx 1.6 \times 10^{19} = 1.6 \times 10^{16} \times 10^{3} = (3.2 \times 10^{16}) \times 500$

Ya’ni, 500 yil vaqt kerak bo‘ladi.

Fibonachchi sonlarini topish masalasi

Fibonachchi sonlari qyidagicha aniqlanadi:

$$f_0 = f_1 = 1,\ f_n = f_{n-1} + f_{n-2},\ n = 2,3,...$$

Fibonachchi sonlaridan hosil bo‘lgan ketma-ketlikning n – hadi topilsin. Rekursiyani qo‘llagan holda fibonachchi sonlarini topish formulasini quyidagicha yozish mumkin:

$$rFibNum(a,b,n) = \begin{cases} a & agar\ n = 1 \\ b & agar\ n = 2 \\ rFibNum(a,b,n-1) + rFibNum(a,b,n-2) & agar\ n > 2 \end{cases}$$

```
#include <iostream>
using namespace std;
int rFibNum(int a, int b, int n);
int main()
{
 int firstFibNum, secondFibNum, n;
 cout << "Birinchi fibonachchi sonini kiriting: "; cin >> firstFibNum;
 cout << endl;
 cout << "Ikkinchi fibonachchi sonini kiriting: "; cin >> secondFibNum;
 cout << endl;
 cout << "Qidirilayotgan fibonachchi soni o’rnini kiriting: "; cin >> n;
 cout << endl;
 cout << n<< " – o’rindagi Fibonachchi soni: "
      << rFibNum(firstFibNum, secondFibNum, n)<<endl;
 return 0;
}
```

```
int rFibNum(int a, int b, int n)
{
  if (n == 1) return a;
   else if (n == 2) return b;
  return rFibNum(a, b, n - 1)+ rFibNum(a, b, n - 2);
}
```

Rekursiya chiroyli, ixcham ko‘ringani bilan xotirani tejash va hisoblash vaqtini qisqartirish nuqtai-nazaridan uni imkon qadar iterativ hisoblash bilan almashtirilgani ma’qul. Masalan, x haqiqiy sonining n-darajasini hisoblashning quyidagi yechim varianti nisbatan kam resurs talab qiladi (n – butun ishorasiz son):

```
double Butun_Daraja(double x, int n)
{
 double p=1;
 for(int i=1; i<=n; i++) p*=x;
 return p;
}
```

Ikkinchi tomondan, shunday masalalar borki, ularni yechishda rekursiya juda samarali, hattoki yagona usuldir. Xususan, grammatik tahlil masalalarida rekursiya juda oson hisoblanadi.

Nazorat savollari

1. Rekursiya deb nimaga aytiladi?
2. Rekursiya uchun qanday aniqlanishlar o‘rinli?
3. Agar faktorial funksiyasiga n>0 qiymat berilsa, qanday holat ro‘y beradi?
4. Rekursiv funksiyalarni to‘g‘ri amal qilishi uchun qanday chaqirishlarning to‘xtash sharti bo‘lishi kerak?
5. Har bir rekursiv murojaat qo‘shimcha xotira talab qiladimi?

6. Rekursiya chiroyli, ixcham ko‘ringani bilan xotirani tejash va hisoblash vaqtini qisqartirish nuqtai-nazaridan uni imkon qadar iteraktiv hisoblash bilan almashtirilgani ma’qulmi?
7. Rekursiya qanday to‘xtatiladi?
8. Har bir rekursiv formula nechta ifodaga ega bo‘lishi kerak?

§12. Foydalanuvchi tomonidan aniqlangan berilganlar turlari

Sanab o‘tiluvchi tur

C++ da berilganlarning oddiy turlari uchta kategoriyaga bo‘linadi: butun, suzuvchi nuqtali va sanab o‘tiluvchi tur. Berilganlarning int turi – 2147483648 dan 2147483647 gacha bo‘lgan butun sonlardan tashkil topgan va ular ustida arifmetik amallar bajariladi (+, –, *, /, va %). Modomiki dasturning asosiy maqsadi berilganlarni boshqarish ekan, ixtiyoriy dasturlash tilida berilganlarning turlari asosiy tushunchlaridan biri hisoblanadi.

Biz hozirgacha foydalanib kelayotgan berilganlarning int, bool, char va double turlari berilganlar turlarining asosiy turlari hisoblanadi. Ammo bu turlar o‘ziga xos masalalarni yechishda yetarli emas.

C++ tili dasturlash imkoniyatlarini oshirish uchun berilganlarning sanab o‘tiluvchi turini yaratishga imkon beradi.

Berilganlarning sanab o‘tiluvchi turini aniqlash quyidagi qismlardan iborat:

1. Berilganlar turining nomi;
2. Berilganlar turining qiymatlari;

Ko‘p miqdordagi, mantiqan bog‘langan o‘zgarmaslardan foydalanilganda sanab o‘tiluvchi turdan foydalangan ma’qul. Sanab o‘tiluvchi o‘zgarmaslar enum kalit so‘zi bilan aniqlanadi. Mazmuni bo‘yicha bu o‘zgarmaslar oddiy butun sonlardir. Sanab o‘tiluvchi o‘zgarmaslar C++ standarti bo‘yicha butun turdagi o‘zgarmaslar hisoblanadi. Har bir o‘zgarmasga (songa) mazmunli nom beriladi va bu identifikatorni dasturning boshqa joylarida nomlash uchun ishlatilishi mumkin emas.

Sanab o‘tiluvchi tur quyidagi ko‘rinishga ega:

enum <tur nomi>{<nom_1>=<qiymat_1>, …,<nom_n>=<qiymat_n>};

Bu yerda, enum – kalit so‘z (inglizcha *enumarate – sanamoq*);

<tur nomi> – o‘zgarmaslar ro‘yxatining nomi; <nom_i> – butun qiymatli o‘zgarmasning nomlari; <qiymat_i> – majburiy 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‘zgarmaslarini 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‘zgarmaslar quyidagi xossaga ega: agar o‘zgarmas qiymati ko‘rsatilmagan bo‘lsa, u oldingi o‘zgarmas qiymatidan bittaga ortiq bo‘ladi. Kelishuv bo‘yicha birinchi o‘zgarmas qiymati 0 bo‘ladi.

Initsializatsiya yordamida o‘zgarmas 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‘zgarmaslarning nomlari har xil bo‘lishi kerak, lekin ularning qiymatlari bir xil bo‘lishi mumkin:

enum {nol=0, toza=0, bir, ikki, juft=2, ush};

O‘zgarmasning qiymati ifoda ko‘rinishda berilishi mumkin, faqat ifodadagi nomlarning qiymatlari shu qadamgacha aniqlangan bo‘lishi kerak:

enum {ikki=2,turt=ikki*2};

O‘zgarmasni qiymatlari manfiy son bo‘lishi mumkin:

enum {ikki=-2,turt=ikki*2};

Misol:

enum ranglar { JIGARRANG, KOK, QIZIL, YASHIL, SARIQ} ;

Yuqorida foydalanuvchi tomonidan yangi ranglar nomli berilganlarning turi aniqlandi, uning qiymatlari JIGARRANG, KOK, QIZIL, YASHIL va SARIQ.

Quyidagi misolga e’tibor bering:

```
enum harflar {'A', 'B', 'C', 'D', 'F'};  // berilganlarning taqiqlangan qiymatlari
enum sonlar {1_bir,2_ikki,3_uch};// berilganlarning taqiqlangan qiymatlari
```

Yuqoridagi misolda o‘zgaruvchilarning yangi aniqlangan turlari bu qiymatlarni qabul qila olmaydi, sababi ular identifikator bo‘lishi kerak.

```
enum harflar {A, B, C, D, F};
enum sonlar {bir, ikki, uch};
```

Agar berilganlarni sanab o‘tiluvchi turi qabul qiladigan qiymatlari ichidagi birortasi oldin boshqa aniqlangan tur qiymatlarida foydalanilgan bo‘lsa, undan foydalanib bo‘lmaydi.

Masalan:

```
enum mat_Talaba {ALI, JALOL, OYSHA, FARRUX};
enum inf_Talaba {SEVARA, VALI, JALOL, JAMOL };    //xatolik
```

Bu misolda inf_Talaba nomi bilan aniqlangan turda o’zgaruvchi qabul qiladigan qiymatlari ichidagi JALOL qiymati oldin aniqlangan mat_Talaba turida mavjudligi sababli xatolik yuzaga keladi.

Dasturda foydalaniladigan standart turdagi o‘zgaruvchilar qanday e’lon qilinsa enum turidagi o‘zgaruvchilar ham xuddi shunday e’lon qilinadi. O‘zgaruvchilarni enum turida e’lon qilish sintaksisi quyidagicha:

```
yangiTur identifikator, identifikator,...;
```

Misol tariqasida quyidagi e’lonni ko’raylik:

```
enum sport {BASKETBALL, FOOTBALL, HOCKEY, SOCCER, VOLLEYBALL};
sport mashhurSport, meningSportim;
```

Yuqorida sport turida bo‘lgan mashhurSport va meningSportim o‘zgaruvchilari e’lon qilindi. E’lon qilingan o‘zgaruvchilarga qiymatlar yuklash mumkin.

```
mashhurSport = FOOTBALL;
meningSportim = mashhurSport;
```

Bu amaldan keyin mashhurSport o‘zgaruvchisi sport turi qabul qilishi mumkin bo‘lgan qiymatlardan FOOTBALL qiymatini o‘zlashtirdi. O’z navbatida meningSportim o‘zgaruvchi mashhurSport o‘zgaruvchisining qiymatini o‘zlashtirib oladi.

Sanab o‘tiluvchi tur ustida amallar bajarish

Shuni aytib o‘tish lozimki sanab o‘tiluvchi turlar ustida quyidagi arifmetik amallarni bajarib bo‘lmaydi. Quyidagi misolda ko‘raylik:

```
meningSportim = mashhurSport + 2;  // xatolik
mashhurSport = FOOTBALL + SOCCER;  //xatolik
mashhurSport = mashhurSport * 2;  // xatolik
```

Shuningdek inkrement va dekrement amallari ham bajarib bo‘lmaydi.

```
mashhurSport++;  // xatolik
mashhurSport– –;  // xatolik
```

Agar mashhurSport o‘zgaruvchisining qiymatini 1 ga oshirmoqchi bo‘lsak, static_cast operatoridan foydalanishimiz mumkin. Kompliyator bu o‘zgaruvchiga sanab o‘tiluvchi tur qabul qilishi mumkin bo‘lgan qiymatlar ichidan o‘zi qabul qilgan qiymatdan keying qiymatni yuklaydi:

```
mashhurSport = FOOTBALL;
mashhurSport = static_cast<sport>(mashhurSport + 1);
```

Yuqoridagi misolda mashhurSport o‘zgaruvchisiga FOOTBALL qiymati yuklandi va mashhurSport o‘zgaruvchisining qiymati birga oshirilganda u HOCKEY qiymatini oladi.

```
mashhurSport = FOOTBALL;
mashhurSport = static_cast<sport>(mashhurSport - 1);
```

mashhurSport o‘zgaruvchisi BASKETBALL qiymatini oldi.

Taqqoslash operatorlaridan foydalanish

Sanab o‘tiluvchi turdagi o‘zgaruvchilar qiymat qabul qilsa, demak bu o‘zgaruvchilar ustida taqqoslash operatorlaridan foydalanish mumkin. Quyidagi

misolda sports sanab o‘tiluvchi turida e’lon qilingan popularSport va mySport nomli o‘zgaruvchilar ustida taqqoslash amallariga doir misol keltirilgan:

```
FOOTBALL <= SOCCER          // rost
HOCKEY > BASKETBALL         // rost
VOLLEYBALL < FOOTBALL       // yolg‘on
mashhurSport = SOCCER;
meningSportim = VOLLEYBALL;
mashhurSport < meningSportim        // rost
```

O‘zgaruvchilarni kiritish va chiqarish

Berilganlarni kiritish va chiqarish faqat satandart int, char, double turdagi o‘zgaruvchilar uchun o‘rinli, sanab o‘tiluvchi turdagi o‘zgaruvchilarga berilganlarni to‘g‘ridan to‘g‘ri o‘qib, chiqarib bo‘lmaydi.

```
enum darslar {ALGEBRA, DASTURLASH, GEOMETRIYA, ASTRONOMIYA, KIMYO, GEOGRAFIYA};
darslar bugungi;
```

Yuqoridagi bugungi o‘zgaruvchisining qiymatini to’g’ridan-to’gri cin orqali o’qib bo’lmaydi. Bu amalni faqat skalayar turdagi o’zgaruvchi vositasida o’qish mumkin. Quyidagi misol buni namoyon etadi.

```
char ch1, ch2;
cin >> ch1 >> ch2; // char turidagi ikkita o‘zgaruvchini o‘qib olish
switch (ch1)
{
 case 'a':  case 'A':
   if (ch2 == 'l' || ch2 == 'L') bugungi = ALGEBRA;
    else bugungi = ASTRONOMIYA;
  break;
case 'd': case 'D':
 bugungi = DASTURLASH;
```

```
  break;
case 'k': case 'K':
  bugungi = KIMYO;
  break;
case 'g': case 'G':
  if (ch2 == 'm' || ch2 == 'M') bugungi = GEOMETRIYA;
  else bugungi = GEOGRAFIYA;
  break;
default:
  cout << "Noto’g’ri kiritildi." << endl;
}
```

Funksiyalar va enum turlar

enum turidagi o‘zgaruvchilarni oddiy turdagi o‘zgaruvchilar singari funksiya parametri sifatida, hamda funksiya qaytaradigan qiymat turi sifatida ishlatish mumkin. Quyidagi keltirilgan misolda funksiya yordamida enum turidagi qiymatlarni ekranga chop etish ko’rsatilgan.

```
void printEnum(darslar bugungi)
{
 switch (bugungi) {
   case ALGEBRA:   cout << "Algebra"; break;
   case ASTRONOMIYA: cout << "Astronomiya"; break;
   case DASTURLASH: cout << "Dasturlash"; break;
   case KIMYO: cout << "Kimyo"; break;
   case GEOMETRIYA: cout << "Geometriya"; break;
   case GEOGRAFIYA: cout << "Geografiya"; break;
    }              // switch tugashi
}
```

typedef bilan ishlash

Foydalanuvchi tomonidan aniqlanadigan tur typedef kalit so‘zi bilan boshlanadi, undan keyin mavjud tur ko‘rsatiladi va identifikator yoziladi. Oxirida yozilgan identifikator – yangi yaratilgan turning nomi hisoblanadi. Masalan,

```
typedef unsigned char byte;
```

ifodasi byte deb nomlanuvchi yangi turni yaratadi va o‘z mazmuniga ko‘ra unsigned char turi bilan ekvivalent bo‘ladi. Keyinchalik, dasturda xotiradan bir bayt joy egallaydigan va [0..255] oraliqdagi qiymatlarni qabul qiladigan byte turidagi o‘zgaruvchi (o‘zgarmaslarni) e’lon qilish mumkin:

```
byte c=65;
byte Byte=0xFF;
```

Massiv ko‘rinishidagi foydalanuvchi tomonidan aniqlanuvchi tur e’loni quyidagicha bo‘ladi:

```
typedef char Ism[30];
Ism ism;
```

Ism turidagi ism o‘zgaruvchisi e’loni – bu 30 belgidan iborat massiv (satr) e’lonidir.

Odatda yechilayotgan masalaning predmet sohasi terminlarida ishlash uchun strukturalar qayta nomlanadi. Natijada murakkab tuzilishga ega bo‘lgan va zarur xususiyatlarni o‘ziga jamlagan yangi turlarni yaratishga muvofiq bo‘linadi.

Masalan, kompleks son haqidagi ma’lumotlarni o‘z ichiga oluvchi Complex turi quyidagicha aniqlanadi:

```
typedef struct {
  double re;
  double im;
 } Complex;
```

Endi kompleks son e’lonini

```
Complex KSon;
```

ko‘rinishida yozish mumkin va uning maydonlariga murojaat qilish mumkin:

```
KSon.re=5.64;
KSon.im=2.3;
```

Nazorat savollari

1. Sanab o‘tiluvchi turlar nima maqsadda ishlatiladi?
2. Sanab o‘tiluvchi turni aniqlash qanday qismlardan iborat?
3. Sanab o‘tiluvchi tur qanday xossalarga ega?
4. Sanab o‘tiluvchi turlar ustida qanday amallar bajarib bo‘lmaydi?
5. Sanab o‘tiluvchi turlar ustida amal bajarishga misol keltiring.
6. Sanab o‘tiluvchi turga berilganlarni kiritish va chiqarish qanday o‘zgaruvchilar uchun o‘rinli?
7. enum turidagi o‘zgaruvchilardan qanday maqsadlarda foydalanish mumkin?
8. enum turidagi o‘zgaruvchi e’loniga misol keltiring.
9. Sanab o‘tiluvchi turlar ustida taqqoslash amaliga misol keltiring.
10. typedef kalit so‘zi yordamida yangi tur xosil qilishga misol keltiring.

§13. Nomlar fazosi

Ma’lumki, dasturga qo‘shilgan sarlavha fayllarida e’lon qilingan identifikator va o‘zgarmaslar kompilyator tomonidan yagona global nomlar fazosiga kiritiladi. Agar dastur ko‘p miqdordagi sarlavha fayllarni ishlatsa va undagi identifikatorlar (funksiya nomlari va o‘zgaruvchilar nomlari, sinflar nomlari va h.k.) va o‘zgarmaslar nomlari turli dastur tuzuvchilar tomonidan mustaqil ravishda aniqlangan bo‘lsa, bir xil nomlarni ishlatish bilan bog‘liq muammolar yuzaga kelish ehtimoli katta bo‘ladi. Nomlar fazosi tushunchasini kiritilishi mazkur muammoni ma’lum bir ma’noda hal qilishga yordam beradi. Agar dasturda yangi identifikatorni aniqlash kerak bo‘lsa va xuddi shu nomni boshqa modullarda yoki kutubxonalarda ishlatishi xavfi bo‘ladigan bo‘lsa, bu identifikatorlar uchun o‘zining shaxsiy nomlar fazosini aniqlash mumkin. Bunga namespace kalit so‘zidan foydalanilgan holda erishiladi:

```
namespace <nomlar fazosining nomi>
{
 // e'lonlar
}
```

Nomlar fazosi ichida e'lon qilingan identifikatorlar faqat <nomlar fazosining nomi> ko'rinish sohasida bo'ladi va yuzaga kelishi mumkin bo'lgan kelishmovchiliklarning oldi olinadi.

Misol tariqasida quyidagi nomlar fazosini yarataylik:

```
namespace Shaxsiy_nomlar
{
 int x, y, z;
 const int soni = 100;
 double d=7.25;
 void Mening_funksiyam(char belgi);
}
```

Kompilyatorga aniq nomlar fazosidagi nomlarni ishlatish kerakligini ko'rsatish uchun ko'rinish sohasiga ruxsat berish amalidan foydalanish mumkin:

```
Shaxsiy_nomlar::x=5;
```

Agar funksiyani ishlatish kerak bo'lsa, mos holda funksiyaga murojaat qilish ko'rinishi yoziladi:

```
Shaxsiy_nomlar::Mening_funksiyam('A');
```

Agar dastur matnida aniq nomlar fazosiga nisbatan ko'p murojaat qilinadigan bo'lsa using namespace qurilmasini ishlatish orqali yozuvni soddalashtirish mumkin:

```
using namespace <nomlar fazosinomi>;
```

Masalan,

```
using namespace Shaxsiy_nomlar;
```

ko'rsatmasi kompilyatorga, bundan keyin toki navbatdagi using uchramaguncha Shaxsiy_nomlar fazosidagi nomlar ishlatilishi kerakligini bildiradi:

```
x = 0; y = z = 10;
Mening_funksiyam('A');
```

Dastur va unga qo‘shilgan sarlavha fayllari tomonidan aniqlanadigan nomlar fazosi std deb nomlanadi. Standart fazoga o‘tish kerak bo‘lsa

```
using namespace std;
```

ko‘rsatmasi beriladi.

Agar birorta nomlar fazosidagi alohida bir nomga murojaat qilish zarur bo‘lsa, using qurilmasini boshqa shaklida foydalaniladi. Misol uchun

```
using Shaxsiy_nomlar::soni;
```

ko‘rsatmasi soni identifikatorini Shaxsiy_nomlar fazosidan ishlatish kerakligini bildiradi.

Shuni qayd etish kerakki, using namespace qurilmasi standart nomlar fazosi ko‘rinish sohasini berkitadi va undagi nomga murojaat qilish uchun ko‘rinish sohasiga ruxsat berish amalidan (std::) foydalanish zarur bo‘ladi.

Nomlar fazosi funksiya ichida e’lon qilinishi mumkin emas, lekin ular boshqa nomlar fazosi ichida e’lon qilinishi mumkin. Ichma-ich joylashgan nomlar fazosidagi identifikatorga murojaat qilish uchun uni qamrab olgan barcha nomlar fazosi nomlar ketma-ket ravishda ko‘rsatilishi kerak. Misol uchun, quyidagi ko‘rinishda nomlar fazosi e’lon qilingan bo‘lsin:

```
namespace Yuqori
{...
        namespace Urta
        {...
                namespace Ichki {int Ichki_n;}
        }
}
```

Ichki_n o‘zgaruvchisiga murojaat quyidagi ko‘rinishda bo‘ladi:

```
Yuqori::Urta::Ichki::Ichki_n=0;
```

Nomlar fazosida funksiyani e’lon qilishda nomlar fazosida faqat funksiya

prototipini e’lon qilish va funksiya tanasini boshqa joyda e’lon qilish ma’qul variant hisoblanadi. Bu holatning ko‘rinishiga misol:

```
namespace Nomlar_fazosi
{
 char c;
 int I;
 void Funcsiya(char Bayroq);
}
...
void Nomlar_fazosi::Funcsiya(char Bayroq)
{
 // funksiya tanasi
}
```

Umuman olganda, o‘z nomiga ega bo‘lmagan nomlar fazosini e’lon qilish mumkin. Bu holda namespace kalit so‘zidan keyin hech nima yozilmaydi. Misol uchun

```
namespace {
char c_nomsiz;
int i_nomsiz;
}
```

ko‘rinishidagi nomlar fazosi elementlariga murojaat hech bir prefiks ishlatmasdan amalga oshiriladi. Nomsiz nomlar fazosi faqat o‘zi e’lon qilingan fayl chegarasida amal qiladi.

C++ tili nomlar fazosining psevdonimlarini aniqlash imkonini beradi. Bu yo‘l orqali nomlar fazosini boshqa nom bilan ishlatish mumkin bo‘ladi. Masalan, nomlar fazosi nomi uzun bo‘lganda unga qisqa nom bilan murojaat qilish:

```
namespace Juda_uzun_nomli_fazo {
float y;
```

```
}
Juda_uzun_nomli_fazo::y = 0;
namespace Qisqa_nom = Juda_uzun_nomli_fazo;
Qisqa_nom::y = 13.2;
```

Xotira sinflari

O‘zgaruvchilarning ko‘rinish sohasi va amal qilish vaqtini aniqlovchi o‘zgaruvchi modifikatorlari mavjud.

O‘zgaruvchi modifikatorlari

Modifikator	Qo‘llanishi	Amal qilish sohasi	Yashash davri
auto	lokal	blok	vaqtincha
register	lokal	blok	vaqtincha
extern	global	blok	vaqtincha
static	lokal	blok	doimiy
	global	fayl	doimiy
volatile	global	fayl	doimiy

Avtomat o‘zgaruvchilar

auto modifikatori lokal o‘zgaruvchilar e’lonida ishlatiladi. Odatda lokal o‘zgaruvchilar e’lonida bu modifikator kelishuv bo‘yicha qo‘llaniladi va shu sababli amalda uni yozishmaydi:

```
#include <iostream>
using namespace std;
int main()
{
 auto int X=2;        // int X=2; bilan ekvivalent
 cout<<X;
 returu 0;
}
```

auto modifikatori blok ichida e’lon qilingan lokal o‘zgaruvchilarga qo‘llaniladi. Bu o‘zgaruvchilar blokdan chiqishi bilan avtomatik ravishda o‘chiriladi.

Registr o‘zgaruvchilar

register modifikatori kompilyatorga, ko‘rsatilgan o‘zgaruvchini protsessor registrlariga joylashtirishga harakat qilishni tayinlaydi. Agar bu harakat natija bermasa o‘zgaruvchi auto turidagi lokal o‘zgaruvchi sifatida amal qiladi.

O‘zgaruvchilarni registrlarda joylashtirish dastur kodini bajarish tezligi bo‘yicha optimallashtiradi, chunki protsessor xotiradagi berilganlarga nisbatan registrdagi qiymatlar bilan ancha tez ishlaydi. Lekin registrlar soni cheklanganligi uchun har doim ham o‘zgaruvchilarni registrlarda joylashtirishning iloji bo‘lmaydi.

```
#include <iostream>
using namespace std;
int main()
{
 register int Reg;
 ...
 return 0;
}
```

register modifikatori faqat lokal o‘zgaruvchilariga nisbatan qo‘llaniladi, global o‘zgaruvchilarga qo‘llash kompilyatsiya xatosiga olib keladi.

volatile modifikatori o‘zgaruvchilari dasturda birorta tashqi qurilma yoki boshqa dastur bilan bog‘lash uchun ishlatish zarur bo‘ladi. Kompilyator bunday modifikatorli o‘zgaruvchini registrga joylashtirishga harakat qilmaydi. Bunday o‘zgaruvchilar e’loniga misol quyida keltirilgan:

```
volatile short port_1;
volatile const int Adress=0x00A2;
```

Tashqi o‘zgaruvchilar

Agar dastur bir nechta moduldan iborat bo‘lsa, ular qandaydir o‘zgaruvchi orqali o‘zaro qiymat almashishlari mumkin (fayllar orasida). Buning uchun o‘zgaruvchi birorta modulda global tarzda e’lon qilinadi va u boshqa faylda

(modulda) ko‘rinishi uchun u yerda extern modifikatori bilan e’lon qilinishi kerak bo‘ladi. extern modifikatori o‘zgaruvchini boshqa faylda e’lon qilinganligini bildiradi. Tashqi o‘zgaruvchilar ishlatilgan dastur.

```
//Sarlavha.h faylidagi e’lon
void Bayroq_Almashsin(void);
// modul_1.cpp faylidagi e’lon
bool Bayroq;
void Bayroq_Almashsin(void){Bayroq=!Bayroq;}
// masala.cpp fayl tanasi
#include <iostream>
using namespace std;
#include <Sarlavha.h>
#include <modul_1.cpp>
extern bool Bayroq;
int main()
{
 Bayroq_Almashsin();
 if(Bayroq) cout <<“Bayroq TRUE” << endl;
  else cout <<“Bayroq FALSE” << endl;
 return 0;
}
```

Oldin Sarlavha.h faylida Bayroq_Almashsin() funksiya sarlavhasi e’lon qilinadi, keyin modul_1.cpp faylida tashqi o‘zgaruvchi e’lon qilinadi va Bayroq_Almashsin() funksiyasining tanasi aniqlanadi va nihoyat, masala.cpp faylida Bayroq o‘zgaruvchisi tashqi deb e’lon qilinadi.

Statik o‘zgaruvchilar

Statik o‘zgaruvchilar static modifikatori bilan e’lon qilinadi va o‘z xususiyatiga ko‘ra global o‘zgaruvchilarga o‘xshaydi. Agar bu turdagi

o‘zgaruvchi global bo‘lsa, uning amal qilish sohasi – e’lon qilingan joydan dastur matnining oxirigacha bo‘ladi. Agar statik o‘zgaruvchi funksiya yoki blok ichida e’lon qilinadigan bo‘lsa, u funksiya yoki blokka birinchi kirishda initsializatsiya qilinadi. O‘zgaruvchining bu qiymati funksiya keyingi chaqirilganida yoki blokka qayta kirishda saqlanib qoladi va bu qiymatni o‘zgartirish mumkin. Statik o‘zgaruvchilarni tashqi deb e’lon qilib bo‘lmaydi.

Agar statik o‘zgaruvchi initsializatsiya qilinmagan bo‘lsa, uning birinchi murojatdagi qiymati 0 hisoblanadi.

Misol tariqasida birorta funksiyani necha marotaba chaqirilganligini aniqlash masalasini ko‘raylik:

```
#include <iostream>
using namespace std;
int Sanagich(void);
int main()
{
  int natija;
  for (int i = 0; i < 30; i++) natija = Sanagich();
   cout << natija;
  return 0;
}
int Sanagich(void)
{
  static short sanagich = 0;
 ...
 sanagich++;
 return sanagich;
}
```

Bu yerda asosiy funksiyadan statik o‘zgaruvchiga ega Sanagich() funksiyasiga 30 marta murojaat qilinadi. Funksiyaga birinchi marta murojaat

qilinganda sanagich o‘zgaruvchiga 0 qiymatini qabul qiladi va uning qiymati birga ortgan holda funksiya qiymati sifatida qaytariladi. Statik o‘zgaruvchilar qiymatlarini funksiyani bir chaqirilishidan ikkinchisiga saqlanib qolinishi sababli, keyingi har bir chaqirishlarda sanagich qiymati bittaga ortib boradi.

Masala. Berilgan ishorasiz butun sonning barcha tub bo‘luvchilari aniqlansin. Masalani yechish algoritmi quyidagi takrorlanuvchi jarayondan iborat bo‘ladi: berilgan son tub songa (1-qadamda 2 ga) bo‘linadi. Agar qoldiq 0 bo‘lsa, tub son chop qilinadi va bo‘linuvchi sifatida bo‘linma olinadi, aks holda navbatdagi tub son olinadi. Takrorlash navbatdagi tub son bo‘linuvchiga teng bo‘lguncha davom etadi.

Dastur matni:

```
#include<iostream>
using namespace std;
#include<cmath>
int Navb_tub();
int main()
{
 unsigned int n,p;
 cout << “\nn qiymatini kiritng: ”;  cin >> n;
 cout << "\n1";
 p = Navb_tub();
 while(n >= p) {
  if(n % p == 0) {
   cout << "*" << p;
   n = n / p;
  }
 else p = Navb_tub();
}
 return 0;
```

```
}
int Navb_tub()
{
  static unsigned int tub = 1;
  for( ; ; ){
   tub++;
   short int ha_tub = 1;
   for(int i = 2; i <= tub / 2 ; i++)
     if (tub % i == 0) ha_tub = 0;
   if (ha_tub) return tub;
  }
 return 0;
}
```

Dasturda navbatdagi tub sonni hosil qilish funksiya ko‘rinishida amalga oshirilgan. Navb_tub() funksiyasining har chaqirilishida oxirgi tub sondan keyingi tub son topiladi. Oxirgi tub sonni “*eslab*” qolish uchun tub o‘zgaruvchisi static qilib aniqlangan.

Dastur ishga tushganda klaviaturadan n o‘zgaruvchisining qiymati sifatida 60 soni kiritilsa, ekranga quyidagi ko‘paytma chop etiladi:

```
1*2*2*3*5
```

Nazorat savollari

1. Nomlar fazosi nima uchun xizmat qiladi?
2. Nomlar fazosini dasturga ulash qanday amalga oshiriladi?
3. Nomlar fazosi o‘zgaruvchilariga qanday murojaat qilinadi?
4. Statik o‘zgaruvchilar nima uchun xizmat qiladi?
5. Registr o‘zgaruvchilar nima uchun xizmat qiladi?
6. Avtomat o‘zgaruvchilar nima uchun xizmat qiladi?
7. Tashqi o‘zgaruvchilar nima uchun xizmat qiladi?
8. volatile o‘zgaruvchilar nima uchun xizmat qiladi?

9. O‘zgaruvchining amal qilish soxasi nima uchun kerak?
10. Lokal va global o‘zgaruvchilarning bir-biridan farqi nimada?

§14. Standart kutubxona funksiyalari

C++ ning standart kutubxonasi ko‘pgina aniqlangan funksiyalar, o‘zgarmaslar va berilganlarning maxsus turlaridan tashkil topgan.

cctype (ctype.h) **kutubxona fayli**

Quyidagi jadvalda cctype(ctype.h) kutubxona faylining standart funksiyalari keltirilgan.

Funksiya nomlari va parametrlari	**Parametrlar turlari**	**Funksiya qaytaradigan qiymat**
isalnum(ch)	ch – char turidagi o’zgaruvchi	Funksiya quyidagi hollarda butun turdagi qiymatlarni qaytaradi: agar ch o‘zgaruvchi qiymati 'A'-'Z', 'a'-'z', '0'-'9' oralig‘ida bo‘lsa 1 (true); aks holda 0 (false) qiymat qaytaradi;
iscntrl(ch)	ch – char turidagi o’zgaruvchi	Funksiya quyidagi hollarda butun turdagi qiymatlarni qaytaradi: agar ch o‘zgaruvchi qiymati ASCII jadvalidagi (0-31 va 127) bo‘lsa 1(true), aks holda 0 (false);
isdigit(ch)	ch – char turidagi o’zgaruvchi	Funksiya quyidagi hollarda butun turdagi qiymatlarni qaytaradi: agar ch o‘zgaruvchi qiymati '0' - '9' belgilar belgilarini qabul qilsa 1(true), aks holda, 0(false);
islower(ch)	ch – char turidagi o’zgaruvchi	Funksiya quyidagi hollarda butun turdagi qiymatlarni qaytaradi: agar ch o‘zgaruvchi qiymati 'a'- 'z' oralig‘idagi belgilardan biri bo‘lsa, nol

		bo‘lmagan qiymat (true) qaytaradi, aks holda0(false);
isprint(ch)	ch – char turidagi o’zgaruvchi	Funksiya quyidagi hollarda butun turdagi qiymatlarni qaytaradi: agar ch o‘zgaruvchi qiymati ASCII jadvalidagi, probel yoki “~” qiymatni qabul qilsa, 1 (true), aks holda 0 (false);
ispunct(ch)	ch – char turidagi o’zgaruvchi	Funksiya quyidagi hollarda butun turdagi qiymatlarni qaytaradi: agar ch o‘zgaruvchi qiymati punktuatsiya belgilarini qabul qilsa 1 (true), aks holda, 0 (false);
isspace(ch)	ch – char turidagi o’zgaruvchi	Funksiya quyidagi hollarda butun turdagi qiymatlarni qaytaradi: agar ch o‘zgaruvchi qiymati (probel, yangi qator, tab) belgilaridan biri bo‘lsa nol bo‘lmagan qiymat (true) qaytaradi, aks holda, 0 (false);
isupper(ch)	ch – char turidagi o’zgaruvchi	funksiya quyidagi hollarda butun turdagi qiymatlarni qaytaradi:agar ch o‘zgaruvchi qiymati katta harflarni ('A'-'Z') qabul qilsa 1 (true), aks holda0(false).

Matematik funksiyalar kutubxonasi(math.h)

Funksiyaprototipi	Funksiya qaytaradigan qiymat
int abs(int i)	argumentning absolyut qiymatini qaytaradi
double acos(double x)	radianda berilgan x argumentni arkkosinus qiymatini qaytaradi
double asin(double x)	radianda berilgan x argumentni arksinus

	qiymatini qaytaradi
double atan(double x)	radianda berilgan x argumentni arktangens qiymatini qaytaradi
double atan2(double x, double y)	radianda berilgan x/y nisbatning arktangens qiymatini qaytaradi
double ceil(double x)	haqiqiy x qiymatini unga eng yaqin katta butun songacha aylantiradi va uni haqiqiy ko‘rinishda qaytaradi
double cos(double x)	x radianga teng bo‘lgan burchakni kosinusini qaytaradi
double cosh(double x)	x radianga teng bo‘lgan burchakni giperbolik kosinusini qaytaradi
double exp(double x)	e^x qiymatni qaytaradi
double fabs(double x)	haqiqiy sonni absolyut qiymatini qaytaradi
double floor(double x)	haqiqiy x qiymatni eng yaqin kichik songa aylantiradi va uni haqiqiy son ko‘rinishida qaytaradi
double fmod(double x, double y)	x sonini y soniga bo‘lish natijasidagi qoldiqni qaytaradi. % amaliga o‘xshash, faqat natija haqiqiy sonda qaytariladi
double frexpr(double x, int *expptr)	x sonni mantissasini va darajasini ajratib, mantissa qiymatini qaytaradi va darajasini ko‘rsatilgan expptr adresiga joylashtiradi
double hypot(double x, double y)	to‘g‘ri uchburchakni katetlari bo‘yicha gipotenuzasini hisoblaydi
long int labs(long int num)	num uzun butun sonning absolyut qiymatini qaytaradi
double ldexp(double x, int exp)	$x*2^{exp}$ qiymatni qaytaradi
double log(double x)	x sonining natural logarifmini qaytaradi

math standart kutubxonasidagi funksiyalardan foydalanish uchun cmath sarlavha faylini qo’shib qo’yish kerak.

Misol:

```
#include <iostream>
#include <cmath>
#include <cctype>
using namespace std;
int main()
{
 int x;
 double u, v;
 cout<< "Line 1: Uppercase a is" <<static_cast<char>(toupper('a'))<<endl;
 u = 4.2;
v = 3.0;
cout <<"Line 4: " <<u<< " to the power of "<<v <<" = "<<pow(u, v)<<endl;
cout << "Line 5: 5.0 to the power of 4 = "<<pow(5.0, 4) << endl;
u = u + pow(3.0, 3);
cout << "Line 7: u = " << u << endl;
x = -15;
cout << "Line 9: Absolute value of " << x << " = " << abs(x) << endl;
return 0;
}
```

Dastur ishlashining natijasi:

```
Line 1: Uppercase a is A
Line 4: 4.2 to the power of 3 = 74.088
Line 5: 5.0 to the power of 4 = 625
Line 7: u = 31.2
Line 9: Absolute value of -15 = 15
```

Belgilar bilan ishlash funksiyalari

Dasturlash amaliyotida belgilarni qaysidir oraliqqa tegishli ekanligini bilish zarur bo‘ladi. Buni “ctype.h” sarlavha faylida e’lon qilingan funksiyalar yordamida aniqlash mumkin.

Quyida ularning bir qismining tavsifi keltirilgan:

isalnum() – belgi raqam (true) yoki raqam emasligini (false) aniqlaydi;

isalpha() – belgini harf (true) yoki yo‘qligini (false) aniqlaydi;

isascii() – belgini kodi 0..127 oralig‘ida (true) yoki yo‘qligini (false) aniqlaydi;

isdigit() – belgini raqamlar diapazoniga tegishli (true) yoki yo‘qligini (false) aniqlaydi.

Bu funksiyalardan foydalanishga misol keltiramiz.

```
#include <iostream>
#include <ctype.h>
#include <string.h>
using namespace std;
int main()
{
 char satr[5];
 int xato;
 do {
  xato=0;
  cout<<"\nTug\'ilgan yilingizni kiriting: ";
  cin.getline(satr,5);
  for (int i=0; i<strlen(satr) && !xato; i++)   {
   if(isalpha(satr[i])) {
    cout<<"Harf kiritildi!";
    xato=1;
```

```
   }
    else
   if(iscntrl(satr[i])) {
    cout<<"Boshqaruv belgisi kiritildi!";
    xato=1;
   }
    else
     if(ispunct(satr[i])) {
      cout<<"Punktuatsiya belgisi kiritildi!";
      xato=1;
     }
     else
      if (!isdigit(satr[i])) {
       cout<<"Raqamdan farqli belgi kiritdildi!";
       xato=1;
      }
  }
  if (!xato) {
   cout << "Sizni tug\'ilgan yilingiz: "<<satr;
   return 0;
  }
 }
  while (1);
}
```

Dasturda foydalanuvchiga tug‘ilgan yilini kiritish taklif etiladi. Kiritilgan sana satr o‘zgaruvchisiga o‘qiladi va agar satrning har bir belgisi (satr[i]) harf yoki boshqaruv belgisi yoki tinish belgilari (punktuatsiya) belgisi bo‘lsa, shu haqda xabar beriladi va tug‘ilgan yilni qayta kiritish taklif etiladi. Dastur

tug‘ilgan yil (to‘rtta raqam) to‘g‘ri kiritilganda “*Sizni tug'ilgan yilingiz: XXXX*” satrini chop qilish bilan o‘z ishini tugatadi.

Turlarni o‘zgartirish funksiyalari

Satrlar bilan ishlashda satr ko‘rinishida berilgan sonlarni, son turlaridagi qiymatlarga aylantirish yoki teskari amalni bajarishga to‘g‘ri keladi. C++ tilining “*stdlib.h*” kutubxonasida bu amallarni bajaruvchi funksiyalar to‘plami mavjud. Quyida nisbatan ko‘p ishlatiladigan funksiyalar tavsifi keltirilgan.

atoi() funksiyasining sintaksisi

```
int atoi(const char* ptr);
```

ko‘rinishga ega bo‘lib, ptr ko‘rsatuvchi ASCIIZ-satrni int turidagi songa o‘tkazishni amalga oshiradi. Funksiya satr boshidan belgilarni songa aylantira boshlaydi va satr oxirigacha yoki birinchi raqam bo‘lmagan belgigacha ishlaydi. Agar satr boshida songa aylantirish mumkin bo‘lmagan belgi bo‘lsa, funksiya 0 qiymatini qaytaradi. Lekin, shunga e’tibor berish kerakki, “0” satri uchun ham funksiya 0 qaytaradi. Agar satrni songa aylantirishdagi hosil bo‘lgan son int chegarasidan chiqib ketsa, sonning kichik ikki bayti natija sifatida qaytariladi.

Misol uchun

```
#include <stdlib.h>
#include <iostream.h>
int main()
{
  char str[]="32secund";
  int i=atoi(str);
  cout<<i<<endl;
  return 0;
}
```

dastursining natijasi sifatida ekranga 32 sonini chop etadi. Agar str qiymati ”100000” bo‘lsa, ekranga –31072 qiymati chop etiladi, chunki 100000

soning ichki ko‘rinishi 0x186A0 va uning oxirgi ikki baytidagi 0x86A0 qiymati 31072 sonining qo‘shimcha koddagi ko‘rinishidir.

atol() funksiyasi xuddi atoi() funksiyasidek amal qiladi, faqat funksiya natijasi long turida bo‘ladi. Agar hosil bo‘lgan son qiymati long chegarasiga sig‘masa, funksiya kutilmagan qiymatni qaytaradi.

atof() funksiyasi e’loni

```
double atof (const char* ptr);
```

ko‘rinishida bo‘lib, ptr ko‘rsatuvchi ASCIIZ-satrni double turidagi suzuvchi nuqtali songa o‘tkazishni amalga oshiradi. Satr suzuvchi nuqtali son formatida bo‘lishi kerak. Songa aylantirish birinchi formatga mos kelmaydigan belgi uchraguncha yoki satr oxirigacha davom etadi.

strtod() funksiyasi atof() funksiyasidan farqli ravishda satrni double turidagi songa o‘tkazishda konvertatsiya jarayoni uzilgan paytda aylantirish mumkin bo‘lmagan birinchi belgi adresini ham qaytaradi. Bu o‘z navbatida satrni xato qismini qayta ishlash imkonini beradi.

strtod() funksiyasining sintaksisi

```
double strtod(const char *s, char **endptr);
```

ko‘rinishga ega va endptr ko‘rsatkichi konvertatsiya qilinishi mumkin bo‘lmagan birinchi belgi adresi.

itoa() va ltoa() funksiyalari mos ravishda int va long turidagi sonlarni satrga ko‘rinishga o‘tkazadi. Bu funksiyalar mos ravishda quyidagi sintaksisga ega:

```
char* itoa(int num, char *str, int radix);
```

va

```
char* ltoa(long num, char *str, int radix);
```

Bu funksiyalar num sonini radix argumentda ko‘rsatilgan sanoq sistemasidagi ko‘rinishini str satrda hosil qiladi.

Nazorat savollari

1. <operand1><taqqoslash amali>< operand2> quyidagi amal nimani anglatadi?
2. “&&” “||” “!” operatorlari nimani anglatadi?
3. (x==3) && (y==5) agar x va y qiymatlari xar hil bo‘lsa qanday qiymat qaytaradi?
4. Mantiqiy ko‘paytirish operatori qanday belgi orqali belgilanadi?
5. Mantiqiy qo‘shish operatori qanday belgi orqali belgilanadi?
6. Beshta sonning o‘rta arifmetigi qanday topiladi?

§15. Ko‘rsatkichlar va adres oluvchi o‘zgaruvchilar

Dastur matnida o‘zgaruvchi e’lon qilinganda, kompilyator o‘zgaruvchiga xotiradan joy ajratadi. Boshqacha aytganda, dastur kodi xotiraga yuklanganda berilganlar uchun, ular joylashadigan segmentning boshiga nisbatan siljishini, ya’ni *nisbiy adresini* aniqlaydi va obekt kod hosil qilishda o‘zgaruvchi uchragan joyga uning adresini joylashtiradi.

Umuman olganda, dasturdagi o‘zgarmaslar, o‘zgaruvchilar, funksiyalar va sinf obektlari adreslarini xotiraning alohida joyida saqlash va ular ustida amallar bajarish mumkin. Qiymatlari adres bo‘lgan o‘zgaruvchilarga *ko‘rsatkich o‘zgaruvchilar* deyiladi.

Ko‘rsatkich uch xil turda bo‘lishi mumkin:

- birorta obektga, xususan o‘zgaruvchiga ko‘rsatkich;
- funksiyaga ko‘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.

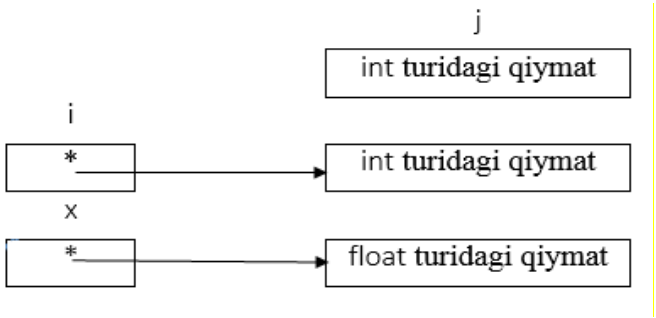
Obektga ko‘rsatkich. Biror obektga ko‘rsatkich (shu jumladan o‘zgaruvchiga). Bunday ko‘rsatkichda ma’lum turdagi (tayanch yoki hosilaviy turdagi) berilganlarning xotiradagi adresi joylashadi. Obektga ko‘rsatkich quyidagicha e’lon qilinadi:

<tur>*<nom>;

Bu yerda <tur> – ko‘rsatkich aniqlaydigan adresdagi qiymatning turi, <nom>– obekt nomi (identifikator). Agar bir turda bir nechta ko‘rsatkichlar e’lon qilinadigan bo‘lsa, har bir ko‘rsatkich uchun ‘*’ belgisi qo‘yilishi shart:

```
int j,*i;
float *x;
```

Keltirilgan misolda i – butun turdagi ko‘rsatkich, x – haqiqiy turdagi ko‘rsatkichlar e’lon qilingan. Ushbu holatni quyidagi rasmda ko’rsatish mumkin.



Funksiyaga ko‘rsatkich. 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 funksiyani oddiy yoki vositali chaqirishni amalga oshirish mumkin. Bunda funksiya uning nomi bo‘yicha emas, balki funksiyaga ko‘rsatuvchi o‘zgaruvchi orqali chaqiriladi. Funksiyani boshqa funksiyaga 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);
```

Bu yerda butun son turida qiymat qaytaradigan fun nomidagi funksiyaga ko‘rsatkich e’lon qilingan va u ikkita haqiqiy turdagi parametrlarga ega.

Masala. Berilgan butun n=100 va a, b - haqiqiy sonlar uchun $f_1(x) = 5\sin(3x) + x$, $f_2(x) = \cos(x)$ va $f_3(x) = x^2 + 1$ funksiyalar uchun $\int_a^b f(x)dx$ integralini to‘g‘ri to‘rtburchaklar formulasi bilan taqriban hisoblansin:

$$\int_a^b f(x)dx \approx h[f(x_1) + f(x_2) + ... + f(x_n)],$$

bu yerda $h = \frac{b-a}{n}$, $x_i = a + ih - h/2, i = 1..n$.

Dastur bosh funksiya, integral hisoblash va ikkita matematik funksiyalar – $f_1(x)$ va $f_3(x)$ uchun aniqlangan funksiyalardan tashkil topadi, $f_2(x) = \cos(x)$ funksiyaning adresi “*cmath.h*” sarlavha faylidan olinadi. Integral hisoblash funksiyasiga ko‘rsatkich orqali integrali hisoblanadigan funksiya adresi, a va b - integral chegaralari qiymatlari uzatiladi. Oraliqni bo‘lishlar soni –n global o‘zgarmas qilib e’lon qilinadi.

```
#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 rejimini 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:Dasturdan chiqish\n";
  do {
   cout<<" Ish rejimi-> "; 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;
}
```

Dasturning ishi cheksiz takrorlash operatori vositasida foydalanuvchiga ish rejimini tanlash bo‘yicha menyuni taklif qilishdan iborat:

```
Ish rejimini tanlang:
1: f1(x)=5*sin(3*x)+x integralini hisoblash
2: f2(x)=cos(x) integralini hisoblash
3: f3(x)=x^2+1 integralini hisoblash
0: Dasturdan chiqish
Ish rejimi->
```

Foydalanuvchi 0 va 3 oralig‘idagi butun sonni kiritishi kerak. Agar kiritilgan son (menu o‘zgaruvchi qiymati) 0 bo‘lsa, break operatori yordamida takrorlashdan, keyin dasturdan chiqiladi. Agar menu qiymati 1 va 3 oralig‘ida bo‘lsa, integralning quyi va yuqori chegaralarini kiritish so‘raladi, hamda Integral() funksiyasi mos funksiya adresi bilan chaqiriladi va natija chop etiladi. Shunga e’tibor berish kerakki, integral chegaralarining qiymatlarini to‘g‘ri kiritilishiga foydalanuvchi javobgar.

void ko‘rsatkich

Bu ko‘rsatkich obekt 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 *<nom>;
```

Ko‘rsatkichning o‘zi o‘zgarmas yoki o‘zgaruvchi bo‘lishi va o‘zgarmas yoki o‘zgaruvchilar adresini ko‘rsatishi mumkin, masalan:

```
int i;
const int ci=1;
```

```
int * pi;              // butun o‘zgaruvchiga ko‘rsatkich
const int *pci;        // butun o‘zgarmasga ko‘rsatkich
int *const cp=&i;   //butun o‘zgaruvchiga o‘zgarmas ko‘rsatkich
const int*const cpc=&ci; // butun o‘zgarmasga o‘zgarmas ko‘rsatkich
```

Misollardan ko‘rinib turibdiki, ‘*’ va ko‘rsatkich nomi orasida turgan const modifikatori faqat ko‘rsatkichning o‘ziga tegishli hisoblanadi va uni o‘zgartirish mumkin emasligini bildiradi, ‘*’ belgisidan chapda turgan const esa ko‘rsatilgan adresdagi qiymat o‘zgarmas ekanligini bildiradi.

Ko‘rsatkichga qiymat berish uchun ‘&’ - adresni olish amali ishlatiladi.

Ko‘rsatkich o‘zgaruvchilarining amal qilish sohasi, yashash davri va ko‘rinish sohasi umumiy qoidalarga bo‘ysunadi.

Ko‘rsatkichga boshlang‘ich qiymat berish

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). Keyinchalik, bu xotira bo‘lagi dastur tomonidan boshqa maqsadda yana ishlatilishi mumkin. 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‘rsatkichlarni e’lon qilishda unga boshlang‘ich qiymatlar berish mumkin. Boshlang‘ich qiymat ko‘rsatkich nomidan so‘ng yoki qavs ichida, yoki ‘=’ belgisidan keyin beriladi. Boshlang‘ich qiymatlar quyidagi usullar bilan berilishi mumkin:

Ko‘rsatkichga mavjud bo‘lgan obektning adresini berish:

a) adresni olish amali orqali:

```
int i=5,k=4;
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 egabo‘lgan ko‘rsatkich

d) massiv yoki funksiya nomini berish:

```
int b[10];
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;
```

Bu yerda 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 obektga murojaat qilmaydi. Bo‘sh ko‘rsatkich asosan ko‘rsatkichni aniq bir obektga ko‘rsatayotgan yoki yo‘qligini aniqlash uchun ishlatiladi.

Ko‘rsatkich ustida amallar

Ko‘rsatkich ustida quyidagi amallar bajarilishi mumkin:

1) obektga vositali murojaat qilish amali;
2) qiymat berish amali;
3) ko‘rsatkichga o‘zgarmas qiymatni qo‘shish amali;
4) ayirish amali;
5) inkrement va dekrement amallari;

6) solishtirish amali;

7) turga keltirish amali.

Vositali murojaat qilish amali ko‘rsatkichdagi adres bo‘yicha joylashgan qiymatni olish yoki qiymat berish uchun ishlatiladi:

```
char a;
char *p=new char; // ko‘rsatkichni e’lon qilish va unga xotira adresini berish
*p=’b’; // p adresiga qiymat joylashtirish
a=*p; // a o‘zgaruvchisiga p adresni 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 turli mazmundagi qiymatlar chop etiladi:

```
cc77ffaa ffaa aa
```

O‘zgaruvchilar bir 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. Masalan:

```
int n=5;
float x=1.0;
int *pi=&n;
float *px=&x;
void *p;
```

```
int *r,*r1;
px=(float *)&n; // int qiymatli joyga float ko'rsatkich
p=px; // void ixtiyoriy ko'rsatkichga moslashadi
r=(int *)p; // float ko'rsatkichli joyga int ko'rsatkich
```

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 ustida bajariladigan arifmetik amallarda turlarning o‘lchami avtomatik ravishda 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 elementiningadresiga ko‘chiradi. Bunda ko‘rsatkichning qiymati sizeof(<massiv elementining 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 ga oshadi
```

Ko‘rsatkichlarning ayirmasi deb, ular ayirmasining tur o‘lchamiga bo‘linishiga aytiladi. Ko‘rsatkichlarni o‘zaro qo‘shish mumkin emas.

Adresni olish amali

Adresni olish quyidagicha e’lon qilinadi:

<tur>&<nom>;

Bu yerda <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 – adres oluvchi o’zgaruvchi,
                 // kol - o‘zgaruvchisining alternativ nomi
const char & cr='\n'; // cr– o‘zgarmasga murojaat
```

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 qilingan 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.

Ko‘rsatkichlar va adres oluvchi o‘zgaruvchilar funksiya parametri sifatida

Funksiya prototipida yoki aniqlanish sarlavhasida ko‘rsatilgan parametrlar “*formal parametrlar”* deyiladi, funksiyaga murojaat qilinganda ko‘rsatilgan argumentlarga “*amaldagi parametrlar”*deyiladi.

Funksiyaga murojaat qilinganda amaldagi parametrning turi mos o‘rindagi formal parametr turiga to‘g‘ri kelmasa yoki shu turga keltirishning iloji bo‘lmasa kompilyatsiya xatosi ro‘y beradi.

Amaldagi parametrlarni funksiyaga ikki xil usul bilan uzatish mumkin: qiymati yoki adresi bilan.

Funksiyaga murojaat qilinganda argument qiymat bilan uzatilganda, argument yoki uning o‘rnida 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 argumentning 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. Funksiyaga murojaat qilinganda mos argument sifatida o‘zgaruvchining nomi yoki ifoda bo‘lishi mumkin.

Amaldagi parametr adres bilan uzatilganda unga mos keluvchi formal parametrni ikki xil usul bilan yozish mumkin: *ko‘rsatkich orqali* yoki *adresni oluvchi parametrlar orqali*. Ko‘rsatkich orqali yozilganda formal parametr turidan keyin ‘*’ belgisi yoziladi, mos argumentda esa o‘zgaruvchining adresi (‘&’ amal orqali) yoki massiv nomi, yoki funksiya nomi bo‘lishi mumkin. Adresni olish amali orqali parametr uzatishda formal parametrda turidan keyin ‘&’ belgisi yoziladi va funksiyaga murojaat qilinganda mos argument sifatida o‘zgaruvchi nomi keladi.

Misol:

```
#include <iostream>
using namespace std;
void f(int,int*,int &)
void main()
{
 int i=1,j=2,k=3;
 cout<<i<<” “<<j<<” “<<k;
 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;
}
```

Dastur ishlashi natijasida ekranga quyidagi qiymatlar chop qilinadi:

```
1 2 3
1 6 8
```

Bu misolda birinchi parametr i qiymat bilan uzatiladi (“int i"). Uning qiymati funksiya ichida o‘zgaradi, lekin tashqaridagi i o‘zgaruvchisining qiymati o‘zgarmaydi. Ikkinchi parametrni ko‘rsatkich orqali adresi bilan uzatilishi talab qilinadi (“int *j”), adresni uzatish uchun ‘&’ – adresni olish amali ishlatilgan (“&j”). Funksiya tanasida argument adresidan qiymat olish uchun ‘*’- qiymat olish amali qo‘llanilgan. Uchinchi parametrda murojaat orqali (“&k”) argumentning adresi uzatish ko‘zda tutilgan. Bu holda funksiya chaqirilishida mos argument o‘rnida o‘zgaruvchi nomi turadi, funksiya ichida esa qiymat olish amalini ishlatishning hojati yo‘q. Funksiya ishlash natijasidagi qiymatlarni argumentlar ro‘yxati orqali olish qulay va tushunarli usul hisoblanadi.

Agar funksiya ichida adres bilan uzatiladigan parametr qiymati o‘zgarmasdan qolishi zarur bo‘lsa, bu parametr const modifikator bilan yozilishi kerak:

```
fun(int n, const char*str);
```

Agarda funksiyaga murojaatda argumentlar faqat nomlari bilan berilgan bo‘lsa, kelishuv bo‘yicha massivlar va funksiyalar adresi bilan, qolgan turdagi parametrlar qiymatlari bilan uzatilgan deb hisoblanadi.

Nazorat savollari

1. Qiymatlari adres bo‘lgan o‘zgaruvchilarga nima deyiladi?
2. Ko‘rsatkich necha turda bo‘ladi?
3. Funksiyaga ko‘rsatkichning yozilish sintaksisi qanday bo‘ladi?
4. Obektga ko‘rsatkich e’loni qanday bo‘ladi?
5. void ko‘rsatkichining muxim afzalliklari nimalardan iborat?
6. Dinamik o‘zgaruvchilar deb nimaga aytiladi?
7. Ko‘rsatkichga boshlang‘ich qiymat berish qay tarzda amalga oshiriladi?
8. Ko‘rsatkich ustida qanday amallar bajarilishi mumkin?
9. Adres oluvchi o‘zgaruvchining ko‘rsatkichdan farqi nimadan iborat?

§16. Dinamik massivlar

new operatori

Xotiradan obektlar uchun dinamik taqsimlanuvchi sohadan joy ajratish uchun new operatori ishlatiladi. new operatoridan keyin xotiraga joylashtiriladigan obekt turini ko‘rsatish lozim. Bu obektni saqlash uchun talab etiladigan xotira sohasi o‘lchovini aniqlash uchun kerak bo‘ladi. Masalan, “new unsigned short int” ifodasi dinamik taqsimlanuvchi xotiradan ikki bayt joy ajratish kerakligini bildiradi. Xuddi shuningdek, new long satri orqali obektlar dinamik taqsimlanuvchi sohadan to‘rt bayt joy ajratiladi.

new operatori natija sifatida belgilangan xotira katagining adresini qaytaradi. Bu adres ko‘rsatkichga o‘zlashtirilishi lozim. Masalan, unsigned short turidagi 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 turidagi

qiymatni saqlovchi dinamik soha xotirasining katagini ko‘rsatib turadi. Endi pPointer ko‘rsatkichini shu turdagi ixtiyoriy o‘zgaruvchiga ko‘rsatkich sifatida qo‘llash mumkin. Ajratilgan xotira sohasiga biror bir qiymat joylashtirish uchun quyidagicha yozuv yoziladi:

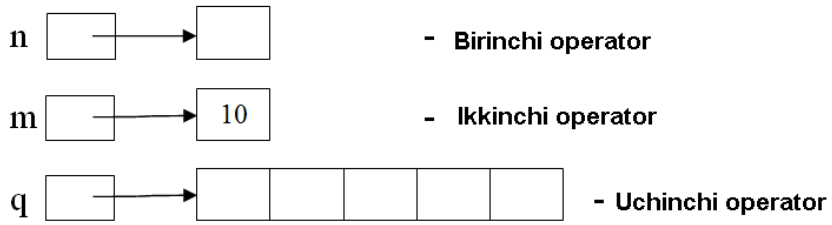
```
*pPointer=72;
```

Bu satr quyidagi ma’noni anglatadi –“*pPointer ko‘rsatkichida adresi saqlanayotgan xotiraga 72 sonini yozilsin*”. Dinamik xotira sohasi albatta chegaralangan bo‘ladi. U to‘lib qolganda new operatori orqali xotiradan joy ajratishga urinsak xatolik yuz beradi.

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 operatorda new amali yordamida dinamik xotirada int uchun yetarli joy ajratib olinib, uning adresi n ko‘rsatkichga yuklanadi. Ko‘rsatkichning o‘zi uchun joy kompilyatsiya vaqtida ajratiladi. Dinamik xotiradan joy ajratish quyidagi rasmda keltirilgan:



Ikkinchi operatorda joy ajratishdan tashqari m adresiga boshlang‘ich qiymat – 10 sonini joylashtiradi.

Uchinchi operatorda int turidagi 5 element uchun joy ajratilgan va uning boshlang‘ich adresi q ko‘rsatkichga berilayapti.

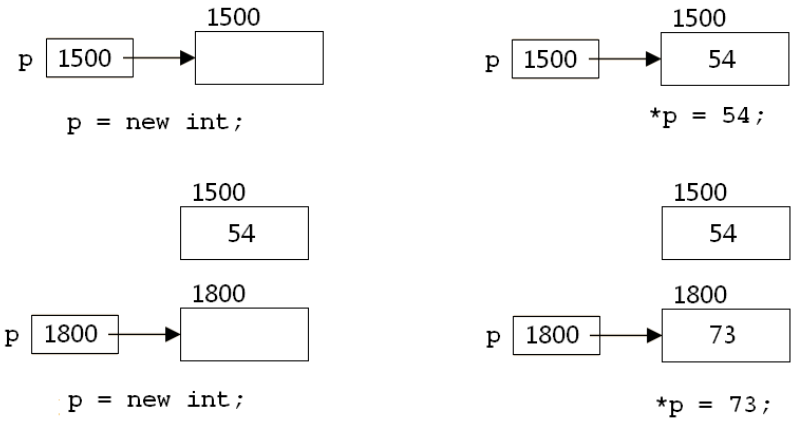
delete operatori

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 yo‘qolishi*” deb atashadi. Haqiqatan ham dastur ishini yakunlaguncha xotiraning bu qismidan foydalanib bo‘lmaydi.

```
int *p;
p = new int;
*p = 54;
p = new int;
*p = 73;
```

Ushbu ro’rsatmalarning bajarilish ketma-ketligi quyidagi rasmda ko’rsatilgan.



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.

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. Quyidagi matnda dinamik o‘zgaruvchi uchun qanday xotira ajratilishi, uni ishlatish va ajratilgan xotirani bo‘shatishga oid misol keltirilgan.

```
#include <iostream>
using namespace std;
int main()
{
 int   local variable = 5;
 int * pLocal = & local variable;
 pHeap = 7;
 cout << “local variable:” << local variable <<”\n”;
 cout << “ *pLocal: ”<< *pLocal  << “\n”;
 cout << “ *pHeap: ” << *pHeap  << “\n”;
 delete pHeap;
 pHeap = new int;
 *pHeap = 9;
 cout <<“*pHeap:”<<*pHeap << “\n”;
 delete pHeap;
 return 0;
}
```

Dastur ishlashi natijasida ekranga quyidagi qiymatlar chiqadi:

```
local variable: 5
*pLocal:  5
*pHeap:  7
*pHeap:  9
```

Dinamik 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 massivni e’lon qilib, aniq masala yechilishida 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 obektga 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 obekt 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_t 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>
using namespace std;
#include <alloc.h>
int main()
{
 int * pVector;
 if ((pVector=(int*)malloc(10*sizeof(int)))==NULL) {
  cout<<”Xotira etarli emas!!!”;
  return 1;
 }
for(int i=0;i<10;i++) *(pVector+i)=i;  // ajratilgan xotira sohasini to‘ldirish
for(int i=0; i<10; i++) cout<<*(pVector+i)<<endl;    // vektorni chop etish
 free(pVector);        // ajratilgan xotira bo‘lagini qaytarish (o‘chirish)
 return 0;
}
```

Misol: n×n o‘lchamli haqiqiy sonlar massivining bosh diagonalidan yuqorida joylashgan elementlar yig‘indisini hisoblash masalasi yechilgan.

```
#include <iostream>
using namespace std;
```

```
#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 dinamik massivlar bilan ishlash

new operatori yordamida, massivga xotira ajratishda obekt turidan keyin kvadrat qavs ichida obektlar soni ko‘rsatiladi. Masalan, butun turdagi 10 ta sondan iborat massivga joy ajratish uchun

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

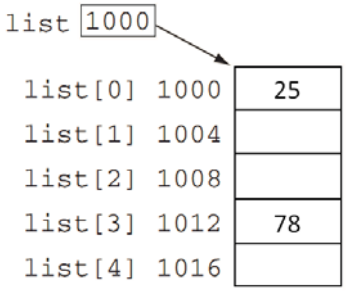
ifodasi yozilishi kerak. Bunga qarama-qarshi ravishda, bu usulda ajratilgan xotirani bo‘shatish uchun

```
delete [] pVector;
```

ko‘rsatmasini berish kerak bo‘ladi.

Qiymat berish indeksi orqali amalga oshiriladi.

```
int * list;
list=new int[5];
list[0] = 25;
list[3] = 78;
for (j = 0; j < 10; j++) list[j] = 0;
```



Ikki o‘lchamli dinamik massivni tashkil qilish uchun “int **a;”

```
int **a;
```

ko‘rinishidagi “*ko‘rsatkichga ko‘rsatkich*” ishlatiladi.

Birinchi navbatda massiv satrlari soniga qarab ko‘rsatkichlar massiviga dinamik xotiradan joy ajratish kerak:

```
a=new int *[n]       // bu yerda n 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<n;i++)
a[i]=new int[m];     //m ustunlar soni
```

Shuni qayd etish kerakki, dinamik massivning har bir satri xotiraning turli joylarida joylashishi mumkin.

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

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

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

```
void main ()
{
 int n,m, 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];
 float **a=new float *[n] ; // ko‘rsatkichlar massiviga xotira ajratish
 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;
}
```

Nazorat savollari

1. Formal va parametrlar deb nimaga aytiladi?
2. Massivlar nima maqsadda ishlatiladi?
3. new operatori natija sifatida nimani qaytaradi?
4. Dinamik xotirada new amali bilan joy ajratish?
5. Kerak bo‘lmagan xotirani qaysi operator yordamida bo‘shatish mumkin?
6. Dinamik massiv bilan statik massivning farqini aytib bering.
7. Qaysi operatorlar yordamida dinamik massiv bilash ishlash imkoni tug‘iladi?
8. «alloc.h» kutubxonasida aniqlangan funksiyalar va ularning vazifalarini aytib bering.
9. Dinamik massiv e’lementlari miqdorini qanday ko‘rsatish mumkin?
10. Bir o‘lchamli dinamik massiv e’lon qilinishi va qiymat olishiga misol keltiring.
11. Ko‘p o‘lchamli dinamik massiv e’lon qilinishi va qiymat olishiga misol keltiring.

§17. Funksiya va massivlar

Bir o‘lchamli massiv funksiya parametri sifatida

Funksiya 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>
using namespace std;
int len(char s[])      //massivni parametr sifatida ishlatish
{
```

```
 int m=0;
 while(s[m++]);
 return m-1;
}
void main()
{
 char z[]=”Ushbu satr uzunligi = ”;
 cout << z << len(z);
}
```

Funksiya parametri sifatida sonlar massivi jo‘natilganda massiv elementlari soni alohida jo‘natiladi. Massivning o‘zi o‘lchamisiz yozilgan holda funksiyada e’lon qilinadi.

```
int sumArray(const int list[], int listSize)
{
 int index, sum = 0;
 for(index = 0; index < listSize; index++) sum+ =list[index];
 return sum;
}
```

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 bumaqsadda global o‘zgaruvchidan foydalanishga to‘g‘ri keladi.

```
#include <iostream>
using namespace std;
float sum(int n,float *x)
{
 float s=0;
 for (int i=0;i<n;i++)s+=x[i];
```

```
 return s;
}
void 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>
using namespace std;
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;
}
void 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.

```
#include <iostream>
using namespace std;
```

```
\\butun turdagi massivga ko‘rsatkich qaytaradigan funksiya
int * massiv_ulash(int,int*,int,int*);
void 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 massiv funksiya parametri sifatida

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×n o‘lchamdagi haqiqiy turdagi x[n][n] massivga mos keluvchi parametrni

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

ko‘rinishda yozib bo‘lmaydi. Muammo yechimi – bu massiv o‘lchamini parameter 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‘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 *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;
}
void 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;
}
void 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;
}
```

Misol: Berilgan qiymatni massiv elementlari ichidan izlash funksiyasi qo’llanilgan dastur:

```
#include <iostream>
using namespace std;
const int ARRAY_SIZE = 10;
```

```
int seqSearch(const int list[], int listLength,int searchItem);
int main()
{
 int intList[ARRAY_SIZE], number;
 cout<<" Enter "<<ARRAY_SIZE<<" integers."<<endl;
 for (int index = 0; index < ARRAY_SIZE; index++) cin >> intList[index];
 cout << "/n Enter the number to be "<<"searched:"; cin >> number;
 cout << endl;
  int pos = seqSearch(intList, ARRAY_SIZE, number);
  if (pos!= -1) cout<<number<<" is found at position "<<pos<<endl;
   else cout<<number<<" is not in the list."<< endl;
  return 0;
}
int seqSearch(const int list[ ], int listLength, int searchItem)
{
 int loc;
 bool found = false;
 loc = 0;
 while (loc < listLength && !found)
   if (list[loc]== searchItem)  found = true;
    else   loc++;
 if(found) return loc;
   else return -1;
}
```

Dastur bajarilishi:

```
Enter 10 integers.
2 56 34 25 73 46 89 10 5 16↵
Enter the number to be searched: 25↵
```

25 is found at position 3

O‘zgaruvchan parametrli funksiyalar

C++ tilida parametrlar soni noma’lum bo‘lgan funksiyalarni ham ishlatish mumkin. Bundan tashqari ularning turlari ham noma’lum bo‘lishi mumkin. Parametrlar soni va turi funksiyani chaqirish-dagi argumentlar soni va ularning turiga qarab aniqlanadi. 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 umuman olganda ikkita.

1-usul. Parametrlar ro‘yxati oxirida yana bir maxsus parameter 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.

Misol:

```
#include <iostream>
using namespace std;
float Sonlar_kupaytmasi(float arg,...)
{
 float p=1.0;
 float *ptr=&arg;
 if(*prt==0.0) return 0.0;
 for(;*prt;prt++)p*=*prt;
 return p;
```

```
}
void main()
{
 cout<<Sonlar_kupaytmasi(2e0,3e0,4e0,0e0)<<'\n';
 cout<<Sonlar_kupaytmasi(1.0,2.0,3.0,10.0,8.0,0.0);
}
```

Natija:

```
24
480
```

2-usul. Birorta maxsus parametr sifatida noma'lum parametrlar soni kiritiladi va unga qarab parametrlar soni aniqlanadi.

Misol:

```
#include <iostream>
using namespace std;
int Yigindi(int,...);
void main()
{
 cout<<"\nYigindi(2,6,4)="<<Yigindi(2,6,4);
 cout<<"\nYigindi(6,1,2,3,4,5,6)="
 cout<<Yigindi(6,1,2,3,4,5,6);
}
int Yigindi(int k,...)
{
 int *ptr=&k
 int s=0;
 for(;k;k--)s+=*(++ptr);
 return s;
}
```

Natija:

Yigindi(2,6,4)=10

Yigindi(6,1,2,3,4,5,6)=21

Yuqorida keltirilgan ikkala misolda ham noma'lum parametrlar berilgan maxsus parametr turini qabul qilgan. Har xil turdagi parametrlarni ishlatish uchun turni aniqlaydigan parametr kiritish kerak bo'ladi:

```
#include <iostream>
using namespace std;
float Summa(char,int,…);
void main()
{
 cout<<Summa('i',3,10,20,30);
 cout<<Summa('f',3,10.0,20.0,5.0);
 cout<<Summa('d',3,10,20,30);
}
int Summa(char z,int k,…)
{
 switch(z){
  case 'i': {
   int *ptr=&k+1; int s=0;
   for (;k--;ptr++) s+=*(ptr);
   return (float)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;
 }
}
}
```

Yuqorida keltirilgan misolda noma'lum parametrlarni turini aniqlash masalasi kompilyator tomonidan emas, balki dastur tuzuvchisi tomonidan hal qilingan.

Nazorat savollari

1. Formal va parametrlar deb nimaga aytiladi?
2. Dinamik massiv bilan statik massivning farqini aytib bering.
3. O'zgaruvchi parametrli funksiyalar qanday e'lon qilinadi?
4. Dinamik massiv elementlari miqdorini qanday ko'rsatish mumkin?
5. Funksiyada bir o'lchamli statik massiv qanday ishlatiladi?
6. Funksiyada bir o'lchamli dinamik massiv qanday ishlatiladi?
7. Funksiyada ko'p o'lchamli statik massiv qanday ishlatiladi?
8. Funksiyada ko'p o'lchamli dinamik massiv qanday ishlatiladi?

§18. Satrlar. Satr ustida amallar. Satr funksiyalari

ASCIIZ-satrlar

Standart C++ tili ikki xildagi belgilar majmuasini qo'llab-quvvatlaydi. Birinchi toifaga, an'anaviy, "*tor*" belgilar deb nomlanuvchi 8-bitli belgilar majmuasi kiradi, ikkinchisiga 16-bitli "*keng*" belgilar kiradi. Til kutubxonasida har bir guruh belgilari uchun maxsus funksiyalar to'plami aniqlangan.

C++ tilida satr uchun maxsus tur aniqlanmagan. Satr char turidagi belgilar massivi sifatida qaraladi va bu belgilar ketma-ketligi *satr terminatori* deb nomlanuvchi 0 kodli belgi bilan tugaydi ('\0'). Odatda, nol-terminator bilan tugaydigan satrlarni *ASCIIZ-satrlar* deyiladi.

Quyidagi jadvalda C++ tilida belgi sifatida ishlatilishi mumkin bo‘lgan o‘zgarmaslar to‘plami keltirilgan.

C++ tilidagi belgi o‘zgarmaslar

Belgilar sinflari	Belgi o‘zgarmaslar
Katta harflar	‘A’ …’Z’, ‘A’…’Ya’
Kichik harflar	‘a’ …’z’, ‘a’…’ya’
Raqamlar	‘0’ …’9’
Bo‘sh joy	gorizontal tabulyasiya (ASCII kodi 9), satrni o‘tkazish (ASCII kodi 10), vertikal tabulyasiya (ASCII kodi 11), formani o‘tkazish (ASCII kodi 12), karetkani qaytarish (ASCII kodi 13)
Punktuatsiya belgilari (ajratuvchilar)	! ” # $ & ‘ ( ) *+ - , . / : ; < = > ? @ [\ ] ^ _ { | } ~
Boshqaruv belgilari	ASCII kodi 0...1Fh oralig‘ida va 7Fh bo‘lgan belgilar
Probel	ASCII kodi 32 bo‘lgan belgi
O‘n oltilik raqamlar	‘0’…’9’, ‘A’…’F’, ‘a’…’f’

Satr massivini e’lon qilinishida, satr oxiriga nol-terminator qo‘yilishi va natijada satrga qo‘shimcha bitta bayt bo‘lishini inobatga olinishi kerak:

```
char satr[10];
```

Ushbu e’londa satr satri uchun jami 10 bayt ajratiladi – 9 satr hosil qiluvchi belgilar uchun va 1 bayt terminator uchun.

Satr o‘zgaruvchilar e’lon qilinishida boshlang‘ich qiymatlarni qabul qilishi mumkin. Bunday holda kompilyator satr uzunligini avtomatik ravishda hisoblaydi va satr oxiriga terminatorni qo‘shib qo‘yadi:

```
char Hafta_kuni[]=”Juma”;
```

Ushbu e’lon quyidagi e’lon bilan ekvivalent:

```
char Hafta_kuni[]={’J’,’u’,’m’,’a’,’\0’};
```

Satr qiymatini o‘qishda oqimli o‘qish operatori “>>” o‘rniga getline() funksiyasini ishlatgan ma’qul hisoblanadi, chunki oqimli o‘qishda probellar

inkor qilinadi (garchi ular satr belgisi hisoblansa ham) va o‘qilayotgan belgilar ketma-ketligi satrdan “*oshib*” ketganda ham belgilarni kiritish davom etishi mumkin. Natijada satr o‘ziga ajratilgan o‘lchamdan ortiq belgilarni “*qabul*” qiladi. Shu sababli, getline() funksiyasi ikkita parametrga ega bo‘lib, birinchi parametr o‘qish amalga oshirilayotgan satrga ko‘rsatkich, ikkinchi parametrda esa o‘qilishi kerak bo‘lgan belgilar soni ko‘rsatiladi. Satrni getline() funksiyasi orqali o‘qishga misol ko‘raylik:

```
#include <iostream>
using namespace std;
int main()
{
 char satr[6];
 cout<<”Satrni kiriting: “<<’\n’;
 cin.getline(satr,6);
 cout<<”Siz kiritgan satr: “<<satr;
 return 0;
}
```

Dasturda ishlatilgan satr satri 5 ta belgini qabul qilishi mumkin, ortiqchalari tashlab yuboriladi. getline() funksiyasiga murojaatda ikkinchi parametr qiymati o‘qilayotgan satr uzunligidan katta bo‘lmasligi kerak.

Satr bilan ishlaydigan funksiyalarning aksariyati “*cstring*” (string.h) kutubxonasida jamlangan. Nisbatan ko‘p ishlatiladigan funksiyalarning tavsifini keltiramiz.

ASCIIZ-satrlar uzunligini aniqlash funksiyalari

Satrlar bilan ishlashda, aksariyat hollarda satr uzunligini bilish zarur bo‘ladi. Buning uchun “*string.h*” kutubxonasida strlen() funksiyasi aniqlangan bo‘lib, uning sintaksisi quyidagicha bo‘ladi:

```
size_t strlen(const char* string)
```

Bu funksiya uzunligi hisoblanishi kerak bo‘lgan satr boshiga ko‘rsatkich

bo‘lgan yagona parametrga ega va u natija sifatida ishorasiz butun sonni qaytaradi. strlen() funksiyasi satrning haqiqiy uzunligidan bitta kam qiymat qaytaradi, ya’ni nol-terminator o‘rni hisobga olinmaydi.

Xuddi shu maqsadda sizeof() funksiyasidan ham foydalanish mumkin va u strlen() funksiyasidan farqli ravishda satrning haqiqiy uzunligini qaytaradi. Quyida keltirilgan misolda satr uzunligini hisoblashning har ikkita varianti keltirilgan:

```
#include <iostream>
using namespace std;
#include <string.h>
int main()
{
 char Str[]="1234567890";
 cout <<"strlen(Str)="<<strlen(Str)<<endl;
 cout <<"sizeof(Str)="<<sizeof(Str)<<endl;
 return 0;
}
```

Dastur ishlashi natijasida ekranga

```
strlen(Str)=10
sizeof(Str)=11
```

xabarlari chiqadi.

Odatda sizeof() funksiyasidan getline() funksiyasining ikkinchi argumenti sifatida foydalaniladi va satr uzunligini yaqqol ko‘rsatmaslik imkonini beradi:

```
cin.getline(Satr, sizeof(Satr));
```

Masala. Faqat lotin harflaridan tashkil topgan satr berilgan. Undagi har xil harflar miqdori aniqlansin.

```
int main()
{
 const int n=80;
```

```
char Satr[n];
cout<<"Satrni kiriting:";
cin.getline(Satr,sizeof(Satr));
double s=0;
int k;
for(int i=0;i<strlen(Satr); i++)
 if(Satr[i]!=' '){
  k=0;
  for(int j=0;j<strlen(Satr); j++)
   if(Satr[i]==Satr[j]||abs(Satr[i]-Satr[j])==32) k++;
  s+=1./k;
 }
cout<<"Satrdagi turli harflar miqdori: "<<(int)s;
return 0;
}
```

Dasturda satr uchun 80 uzunligidagi Satr belgilar massivi e'lon qilingan va uning qiymati klaviaturadan kiritiladi. Masala quyidagicha yechiladi. Ichma-ich joylashgan takrorlash operatori yordamida Satr massivining har bir elementi Satr[i] massivning barcha elementlari Satr[j] bilan ustma-ust tushishi yoki ular bir-biridan 32 soniga farq qilishi (katta va kichik lotin harflarining kodlari o‘rtasidagi farq) holatlari k o‘zgaruvchisida sanaladi va s umumiy yig‘indiga 1/k qiymati bilan qo‘shiladi. Dastur oxirida s qiymati butun turga aylantirilgan holda chop etiladi. Satrdagi so‘zlarni bir-biridan ajratuvchi probel belgisi cheklab o‘tiladi.

Dasturga

Satrdagi turli harflar miqdori

satri kiritilsa, ekranga javob tariqasida

Satrdagi turli belgilar miqdori: 14

satri chop etiladi.

ASCIIZsatrlarni nusxalash

Satr qiymatini biridan ikkinchisiga nusxalash mumkin. Bu maqsadda bir qator standart funksiyalar aniqlangan bo‘lib, ularning ayrimlarining tavsiflarini keltiramiz.

strcpy() funksiyasi prototipi

char* strcpy(char* str1, const char* str2)

ko‘rinishga ega va bu funksiya str2 satrdagi belgilarni str1 satrga baytma-bayt nusxalaydi. Nusxalash str2 ko‘rsatib turgan satrdagi nol-terminal (‘\0’) uchraguncha davom etadi. Shu sababli, str2 satr uzunligi str1 satr uzunligidan katta emasligiga ishonch hosil qilish kerak, aks holda berilgan sohada (segmentda) str1 satrdan keyin joylashgan berilganlar “ustiga” str2 satrning “*ortib qolgan*” qismi yozilishi mumkin.

Navbatdagi dastur qismi “*Satrni nusxalash*!” satrini Str satrga nusxalaydi:

```
char Str[20];
strcpy(Str, ”Satrni nusxalash!”);
```

Zarur bo‘lganda satrning qaysidir joyidan boshlab, oxirigacha nusxalash mumkin. Masalan, “*Satrni nusxalash!*” satrini 8-belgisidan boshlab nusxa olish zarur bo‘lsa, uni quyidagicha yechish mumkin:

```
#include <iostream>
using namespace std;
#include <string.h>
int main()
{
 char Str1[20]=”Satrni nusxalash!”,Str2[20];
 char* kursatkich=Str1;
 kursatkich+=7;
 strcpy(Str2,kursatkich);
 cout<<Str2<<endl;
 return 0;
}
```

strncpy() funksiyasining strcpy() funksiyasidan farqli joyi shundaki, unda bir satrdan ikkinchisiga nusxalanadigan belgilar soni ko‘rsatiladi. Uning prototipi quyidagi ko‘rinishga ega:

char* strncpy(char*str1, const char*str2,size_t num);

Agar str1 satr uzunligi str2 satr uzunligidan kichik bo‘lsa, ortiqcha belgilar “*kesib*” tashlanadi. strncpy() funksiyasi ishlatilishiga misol ko‘raylik:

```
#include <iostream>
using namespace std;
#include <string.h>
int main()
{
 char Uzun_str[]="01234567890123456789";
 char Qisqa_str[]="ABCDEF";
 strncpy(Qisqa_str,Uzun_str,4);
 cout <<"Uzun_str= "<<Uzun_str<<endl;
 cout<<"Qisqa_str="<<Qisqa_str<<endl;
 return 0;
}
```

Dasturda Uzun_str satri boshidan 4 belgi Qisqa_str satriga, uning oldingi qiymatlari ustiga joylanadi va natijada ekranga

```
01234567890123456789
0123EF
```

satrlar chop etiladi.

strdup() funksiyasiga yagona parametr sifatida satr-manbaga ko‘rsatkich uzatiladi. Funksiya, satrga mos xotiradan joy ajratadi, unga satrni nusxalaydi va yuzaga kelgan satr nusxa adresini javob sifatida qaytaradi. strdup() funksiya sintaksisi:

char* strdup(const char* sourse)

Quyidagi dastur bo‘lagida satr1 satrining nusxasi xotiraning satr2 ko‘rsatgan joyida paydo bo‘ladi:

```
char* satr1=”Satr nusxasini olish.”; char* satr2;
satr2=strdup(satr1);
```

ASCIIZ satrlarni ulash

Satrlarni ulash (konkatenatsiya) amali yangi satrlarni hosil qilishda keng qo‘llaniladi. Bu maqsadda “string.h” kutubxonasida strcat() va strncat() funksiyalari aniqlangan. strcat() funksiyasi sintaksisi quyidagi ko‘rinishga ega:

```
char* strcat(char* str1, const char* str2)
```

Funksiya ishlashi natijasida str2 satr, funksiya qaytaruvchi satr – str1 satr oxiriga ulanadi. Funksiyani chaqirishdan oldin str1 satr uzunligi, unga str2 satri ulanishi uchun etarli bo‘lishi hisobga olingan bo‘lishi kerak.

Quyida keltirilgan amallar ketma-ketligining bajarilishi natijasida satr satriga qo‘shimcha satr ulanishi ko‘rsatilgan:

```
char satr[80];
strcpy(satr,”Bu satrga ”);
strcat(satr,”satr osti ulandi.”);
```

Amallar ketma-ketligini bajarilishi natijasida satr ko‘rsatayotgan joyda “*Bu satrga satr osti ulandi.*” satri paydo bo‘ladi.

strncat() funksiyasi strcat() funksiyadan farqli ravishda str1 satrga str2 satrning ko‘rsatilgan uzunlikdagi satr qismini ulaydi. Ulanadigan satr qismi uzunligi funksiyaning uchinchi parametri sifatida beriladi.

Funksiya sintaksisi

```
char* strncat(char* str1, const char* str2, size_t num)
```

Quyida keltirilgan dastur bo‘lagida str1 satrga str2 satrning boshlang‘ich 10 ta belgidan iborat satr qismini ulaydi:

```
char satr1[80]=”Dasturlash tillariga misol bu-“;
char satr2[80]=”C++, Pascal ,Basic”;
strncat(satr1,satr2,10);
cout << satr1;
```

Amallar bajarilishi natijasida ekranga

Dasturlash tillariga misol bu-C++, Pascal

satri chop etiladi.

ASCIIZ satrlarda izlash funksiyalari

Satrlar bilan ishlashda undagi birorta belgini izlash uchun "*string.h*" kutubxonasida bir qator standart funksiyalar mavjud.

Birorta belgini berilgan satrda bor yoki yo‘qligini aniqlab beruvchi strchr() funksiyasining prototipi

```
char* strchr(const char* string, int c);
```

ko‘rinishida bo‘lib, u c belginining string satrida izlaydi. Agar izlash muvofaqqiyatli bo‘lsa, funksiya shu belgining satrdagi o‘rnini (adresini) funksiya natijasi sifatida qaytaradi, aks holda, ya’ni belgi satrda uchramasa funksiya NULL qiymatini qaytaradi. Belgini izlash satr boshidan boshlanadi.

Quyida keltirilgan dastur bo‘lagi belgini satrdan izlash bilan bog‘liq.

```
char satr[]=”0123456789”;
char* pSatr;
pSatr=strchr(satr,’6’);
```

Dastur ishlashi natijasida pSatr ko‘rsatkichi satr satrining ‘6’ belgisi joylashgan o‘rni adresini ko‘rsatadi.

strrchr() funksiyasi berilgan belgini berilgan satr oxiridan boshlab izlaydi. Agar izlash muvaffaqiyatli bo‘lsa, belgini satrga oxirgi kirishining o‘rnini qaytaradi, aks holda NULL.

Misol uchun

```
char satr[]=”0123456789101112”;
char* pSatr;
pSatr=strrchr(satr,’0’);
```

amallarini bajarilishida pSatr ko‘rsatkichi satr satrining “01112” satr qismining boshlanishiga ko‘rsatadi.

strspn() funksiyasi ikkita satrni belgilarni solishtiradi. Funksiya quyidagi

size_t strspn(const char* str1, const char* str2);

ko‘rinishga ega bo‘lib, u str1 satrdagi str2 satrga kiruvchi birorta belgini izlaydi va agar bunday element topilsa, uning indeksi funksiya qiymati sifatida qaytariladi, aks holda funksiya satr uzunligidan bitta ortiq qiymatni qaytaradi.

Misol:

```
char satr1[]="0123ab6789012345678";
char satr2[]="a32156789012345678";
int farqli_belgi;
farqli_belgi=strspn(satr1,satr2);
cout<<"Satr1 satridagi Satr2 satrga kirmaydigan birinchi belgi
indeksi="<<farqli_belgi;
cout<<"va u '"<<satr1[farqli_belgi]<<"' belgisi.";
```

amallar bajarilishi natijasida ekranga

Satr1 satridagi Satr2 satrga kirmaydigan birinchi belgi indeksi=5 va u 'b'belgisi satri chop etiladi.

strcspn () funksiyasining prototipi

size_t strcspn(const char* str1, const char* str2);

ko‘rinishida bo‘lib, u str1 va str2 satrlarni solishtiradi va str1 satrining str2 satriga kirgan birinchi belgini indeksini qaytaradi. Masalan,

```
char satr[]=”Birinchi satr”;
int index;
index=strcspn(satr,”sanoq tizimi”);
```

amallari bajarilgandan keyin index o‘zgaruvchisi 1 qiymatini qabul qiladi, chunki birinchi satrning birinchi o‘rindagi belgisi ikkinchi satrda uchraydi.

strpbrk () funksiyasining prototipi

char* strpbrk(const char* str1, const char* str2);

ko‘rinishga ega bo‘lib, u str1 satrdagi str2 satrga kiruvchi birorta belgini izlaydi va agar bunday element topilsa, uning adresi funksiya qiymati sifatida

qaytariladi, aks holda funksiya NULL qiymati qaytaradi. Quyidagi misol funksiyani qanday ishlashini ko‘rsatadi.

```
char satr1[]=”0123456789ABCDEF”;
char satr2[]=”ZXYabcdefABC”;
char* element;
element = strpbrk(satr1,satr2);
cout<<element<<’\n’;
```

Dastur ishlashi natijasida ekranga str1 satrining

ABCDEF

satr ostisi chop etiladi.

Satrlar bilan ishlashda bir satrda ikkinchi bir satrning (yoki uning biror qismini) to‘liq kirishini aniqlash bilan bog‘liq masalalar nisbatan ko‘p uchraydi. Masalan, matn tahrirlaridagi satrdagi birorta satr qismini ikkinchi satr qismi bilan almashtirish masalasini misol keltirish mumkin (yuqorida xuddi shunday masala uchun dastur keltirilgan). Standart “*string.h*” kutubxonasi bu toifadagi masalalar uchun bir nechta funksiyalarni taklif etadi.

strstr () funksiyasi quyidagicha e’lon qilinadi:

```
char* strstr(const char* str, const char* substr);
```

Bu funksiya str satriga substr satr qismi kirishi tekshiradi, agar substr satr qismi str satriga to‘liq kirishi mavjud bo‘lsa, satrning chap tomonidan birinchi kirishdagi birinchi belgining adresi javob tariqasida qaytariladi, aks holda funksiya NULL qiymatini qaytaradi.

Quyidagi misol strstr() funksiyasini ishlatishni ko‘rsatadi.

```
char satr1[]=”Satrdan satr ostisi izlanmoqda, satr ostisi mavjud”;
char satr2[]=”satr ostisi”;
char* satr_osti;
satr_osti=strstr(satr1,satr2);
cout<<satr_osti<<’\n’;
```

Dastur buyruqlari bajarilishi natijasida ekranga

satr ostisi izlanmoqda, satr ostisi mavjud

satri chop etiladi.

Keyingi dastur bo‘lagida satrda boshqa bir satr qismi mavjud yoki yo‘qligini nazorat qilish holati ko‘rsatilgan:

```
char Ismlar[]=
”Alisher,Farxod, Munisa, Erkin, Akmal, Nodira”;
char Ism[10];
char* Satrdagi_ism;
cout<< “Ismni kiriting: ”; cin>>Ism;
Satrdagi_ism = strstr(Ismlar,Ism);
cout<< “Bunaqa ism ro\’yxatda “;
if(Satrdagi_ism==NULL) cout<< “yo\’q .”<<’\n’;
else cout<< “bor .”<<’\n’;
```

Dasturda foydalanuvchidan satr qismi sifatida birorta nomni kiritish talab qilinadi va bu qiymat Ism satriga o‘qiladi. Kiritilgan ism dasturda aniqlangan ro‘yxatda (Ismlar satrida) bor yoki yo‘qligi aniqlanadi va xabar beriladi.

strtok () funksiyasining sintaksisi

```
char* strtok(char* str, const char* delim);
```

ko‘rinishda bo‘lib, u str satrida delim satrro‘yxatida berilgan ajratuvchilar oralig‘iga olingan satr qismlarni ajratib olish imkonini beradi. Funksiya birinchi satrda ikkinchi satr-ro‘yxatdagi ajratuvchini uchratsa, undan keyin nol-terminatorni qo‘yish orqali str satrni ikkiga ajratadi. Satrning ikkinchi bo‘lagidan ajratuvchilar bilan “*o‘rab olingan*” satr qismlari topish uchun funksiyani keyingi chaqirilishida birinchi parametr o‘rniga NULL qiymatini qo‘yish kerak bo‘ladi. Quyidagi misolda satrni bo‘laklarga ajratish masalasi qaralgan:

```
#include <iostream>
using namespace std;
```

```
#include <string.h>
int main()
{
 char Ismlar[]=
 "Alisher,Farxod Munisa, Erkin? Akmal0, Nodira";
 char Ajratuvchi[]=" ,!?.0123456789";
 char* Satrdagi_ism;
 Satrdagi_ism=strtok(Ismlar,Ajratuvchi);
 if(Satrdagi_ism) cout<<Satrdagi_ism<<'\n';
 while(Satrdagi_ism){
  Satrdagi_ism=strtok(NULL,Ajratuvchi);
  if(Satrdagi_ism) cout<<Satrdagi_ism<<'\n';
 }
 return 0;
}
```

Dastur ishlashi natijasida ekranga Ismlar satridagi ‘␣’ (probel), ‘,’ (vergul), ‘?’ (so‘roq belgisi) va ‘0’ (raqam) bilan ajratilgan satr qismlari – ismlar chop qilinadi:

```
Alisher
Farxod
Munisa
Erkin
Akmal
Nodira
```

string turidagi satrlar

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”;
s1=”Qiymat berish 1-usul”;      //satrli o‘zgarmas qiymati berish
s2=str;                         // char turidagi satr yuklanmoqda
s3=’A’;                         // bitta belgi qiymat sifatida berish
s3=s3+s1+s2+”0123abc”;          //qiymat sifatida satr ifoda
```

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 (obektlar) o‘z funksiyalariga ega hisoblanadi va ularni chaqirish uchun oldin o‘zgaruvchi nomi, keyin ‘.’ (nuqta) va zarur funksiya nomi (argumentlari bilan) yoziladi.

Quyidagi jadvalda string turidagi satrlar ustida bajariladigan amallar keltirilgan.

string turidagi satrlar ustida amallar bajarish jadvali

Amal	Mazmuni	Misol
=,+=	Qiymat berish amali	s=”satr01234” s+=”2satr000”
+	Satrlarulash amali(konkantenatsiya)	s1+s2
==, !=, <, <=, >, >=	Satrlarni solishtirish amallari	s1==s2 || s1>s2&& s1!=s2
[]	Indeks berish	s[4]
<<	Oqimga chiqarish	cout << s
>>	Oqimdan o‘qish	cin >> s (probelgacha)

Satr qismini boshqa satrga nusxalash funksiyasi

Bir satr qismini boshqa satrga yuklash uchun kuyidagi funksiyalarni ishlatish mumkin, ularni prototipi kuyidagicha:

```
assign(const string &str);
assign(const string &str,unsigned int pos,unsigned int n);
assign(const char *str, int n);
```

Birinchi funksiya 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 funksiya 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 satrining pos o‘rnidan boshlab satr oxirigacha bo‘lgan belgilar nusxalanadi. Bu qoida barcha funksiyalar uchun tegishlidir.

Misol:

```
string s1,s2,s3;
s1=”0123456789”;
s2.assign(s1,4,5);          // s2=”45678”
s3.assign(s1,2,20);         // s3=”23456789”
```

Uchinchi funksiya argumentdagi char turidagi str satrni string turiga aylantirib, funksiyani chaqiruvchi satrga o‘zlashtiradi:

```
char * strold;
cin.getline(strold,100);    //      “0123456789” kiritilgan bo’lsin
string s1,s2;
s2.assign(strold,6);        // s2=”012345”
s3.assign(strold,20);       // s3=”0123456789”
```

Satr qismini boshqa satrga qo‘shish funksiyalari quyidagicha:

```
append(const string &str);
append(const string & str,unsigned int pos,unsigned int n);
append(const char *str, int n);
```

Bu funksiyalarni yuqorida keltirilgan mos assign funksiyalardan farqi - funksiyani chaqiruvchi satr oxiriga str satrni o‘zini yoki uning qismini qo‘shadi.

```
char * sc;
cin.getline(sc,100);        // ”0123456789” kiritilgan bo’lsin
string s1,s,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”
```

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‘rindan keyin joylashgan belgilar o‘nga suriladi.

Misol:

```
char * sc;
cin.getline (sc,100);                  // ”0123456789” satri kiritilgan bo’lsin
unsigned int i=3;
string s1,s,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 va almashtirish funksiyalari

Satr qismini o‘chirish uchun quyidagi funksiyani ishlatish mumkin:

erase(unsigned int pos=0,unsigned int n=npos);

Bu funksiya, uni chaqiruvchi satrningposo‘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="")
```

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,s,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 funksiyasiprototipi quyidagicha:

```
string substr(unsigned int pos=0,unsigned int n=npos) const;
```

Bu funksiya, uni chaqiruvchi satrningpos 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”
cout<<s1.substr(1,3)+s1.substr();// “30123456789”satr ekranga chiqadi
```

Satr qismini izlash va solishtirish 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 sonni qaytaradi (npos=4294967295), agar izlash o‘rni (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=npos)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=npos) 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 satrining 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 cbelgisini izlaydi;

```
unsigned int find_last_of(const string &str, unsigned int pos=npos)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=npos) const;
```

Bu funksiya oldingidan farqi –satrdan c 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 c belgisidan farqli birinchi belgini izlaydi.

```
unsigned int find_last_not_of(const string &str,
                                    unsigned int pos=npos)const;
```

Funksiya, uni chaqiruvchi satrning ko‘rsatilgan joydan boshlab str satrini tashkil etuvchi belgilar to‘plamiga kirmagan belgini qidiradi va o‘ng tomondan birinchi topilgan belgining indeksini, aks holda npos sonini qaytaradi.

```
unsigned int find_last_not_of(char c,unsigned int pos=npos)const;
```

Bu funksiyaning oldingidan farqi – satr oxiridan boshlab s belgisiga o‘xshamagan belgini izlaydi.

Izlash funksiyalarini qo‘llashga misol:

```
#include <iostream>
using namespace std;
```

```
#include <conio.h>
void 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
}
```

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 funksiyani 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>
using namespace std;
void 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;
 }
```

Satr xossalarini 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
```

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‘zgarmas ko‘rsatkichni qaytaradi:

```
char *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.

Nazorat savollari

1. C++ tilida qanday ko’rinishdagi satrlar mavjud?
2. Belgilarni o‘qish uchun qaysi funksiyalar ishlatiladi?
3. Belgilarni yozish uchun qaysi funksiyalar ishlatiladi?
4. Satrlarni o‘qish uchun qaysi funksiyalar ishlatiladi?
5. Satrlarni yozish uchun qaysi funksiyalar ishlatiladi?
6. Satr uzunligi qanday aniqlanadi?
7. Satrlarni qanday solishtirish mumkin?
8. Satr qismini izlash uchun qanday funksiyadan foydalanish mumkin?
9. Satr qismini qanday o’chirish mumkin?
10. Satrlarni ulash uchun nima qilish kerak?

§19. Tuzilmalar. Birlashmalar

Strukturalar

Ma’lumki, biror predmet sohasidagi masalani yechishda undagi obektlar bir nechta, har xil turdagi parametrlar bilan tavsiflanishi mumkin. Masalan, tekislikdagi nuqta haqiqiy turdagi x – absissa va y – ordinata juftligi – (x,y) ko‘rinishida beriladi. Talaba haqidagi ma’lumotlar: satr turidagi talaba familiya, ismi va sharifi, mutaxassislik yo‘nalish, talaba yashash adresi, butun turdagi tug‘ilgan yili, o‘quv bosqichi, haqiqiy turdagi reyting bali, satr turdagi talaba jinsi haqidagi ma’lumot va boshqalardan shakllanadi.

Dasturda holat yoki tushunchani tavsiflovchi har bir berilganlar uchun alohida o‘zgaruvchi aniqlab masalani yechish mumkin. Lekin bu holda obekt haqidagi ma’lumotlar “*tarqoq*” bo‘ladi, ularni qayta ishlash murakkablashadi, obekt haqidagi berilganlarni yaxlit holda ko‘rish qiyinlashadi.

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>
{
<tur_1><nom_1>;
<tur_2><nom_2>;
  …
<tur_n><nom_n>;
};
```

Bu yerda <struktura nomi>– struktura ko‘rinishida yaratilayotgan yangi turning nomi, “<tur_i><nom_i>;” – strukturaning i-maydonining (nom_i) e’loni.

Boshqacha aytganda, struktura e’lon qilingan o‘zgaruvchilardan (maydonlardan) tashkil topadi. Unga har xil turdagi berilganlarni o‘z ichiga oluvchi *qobiq* deb qarash mumkin. Qobiqdagi berilganlarni yaxlit holda ko‘chirish, tashqi qurilmalar (binar fayllarga) yozish, o‘qish mumkin bo‘ladi.

Talaba haqidagi berilganlarni o‘z ichiga oluvchi struktura turining e’lon qilinishini ko‘raylik.

```
struct Talaba {
 char FISh[30];
 unsigned int Tug_yil;
 unsigned int Kurs;
 char Yunalish[50];
 float Reyting;
 unsigned char Jinsi[5];
```

```
 char Adres[50];
 bool status;
};
```

Dasturda strukturalardan foydalanish, shu turdagi o‘zgaruvchilar e’lon qilish va ularni qayta ishlash orqali amalga oshiriladi:

```
Talaba talaba;
```

Struktura turini e’lonida turning nomi bo‘lmasligi mumkin, lekin bu holda struktura aniqlanishidan keyin albatta o‘zgaruvchilar nomlari yozilishi kerak:

```
struct {
  unsigned int x,y;
  unsigned char Rang;
 } Nuqta1, Nuqta2;
```

Keltirilgan misolda struktura turidagi Nuqta1, Nuqta2 o‘zgaruvchilari e’lon qilingan.

Struktura turidagi o‘zgaruvchilar bilan ishlash, uning maydonlari bilan ishlashni anglatadi. Struktura maydoniga murojaat qilish ‘.’ (nuqta) orqali amalga oshiriladi. Bunda struktura turidagi o‘zgaruvchi nomi, undan keyin nuqta qo‘yiladi va maydon o‘zgaruvchisining nomi yoziladi. Masalan, talaba haqidagi struktura maydonlariga murojaat quyidagicha bo‘ladi:

```
talaba.Kurs=2;
talaba.Tug_yil=1988;
strcpy(talaba.FISh, “Abdullaev A.A.”);
strcpy(talaba.Yunalish,
“Informatika va Axborot texnologiyalari”);
strcpy(talaba.Jinsi, “Erk”);
strcpy(talaba.Adres,
“Toshkent,Yunusobod 6-3-8, tel: 224-45-78”);
talaba.Reyting=123.52;
```

Keltirilgan misolda talaba strukturasining son turidagi maydonlariga

oddiy ko‘rinishda qiymatlar berilgan, satr turidagi maydonlar uchun strcpy funksiyasi orqali qiymat berish amalga oshirilgan.

Struktura turidagi obektning xotiradan qancha joy egallaganligini sizeof funksiyasi (operatori) orqali aniqlash mumkin:

```
int i=sizeof(Talaba);
```

Ayrim hollarda struktura maydonlari o‘lchamini bitlarda aniqlash orqali egallanadigan xotirani kamaytirish mumkin. Buning uchun struktura maydoni quyidagicha e’lon qilinadi:

<maydon nomi> : <o‘zgarmas ifoda>

Bu yerda <maydon nomi> – maydon turi va nomi, <o‘zgarmas ifoda> – maydonning bitlardagi uzunligi. Maydon turi butun turlar bo‘lishi kerak (int, long, unsigned, char).

Agar foydalanuvchi strukturaning maydoni faqat 0 va 1 qiymatini qabul qilishini bilsa, bu maydon uchun bir bit joy ajratishi mumkin (bir bayt yoki ikki bayt o‘rniga). Xotirani tejash evaziga maydon ustida amal bajarishda razryadli arifmetikani qo‘llash zarur bo‘ladi.

Misol uchun sana-vaqt bilan bog‘liq strukturani yaratishning ikkita variantini ko‘raylik. Struktura yil, oy, kun, soat, minut va sekund maydonlaridan iborat bo‘lsin va uni quyidagicha aniqlash mumkin:

```
struct Sana_vaqt {
 unsigned short Yil;
 unsigned short Oy;
 unsigned short Kun;
 unsigned short Soat;
 unsigned short Minut;
 unsigned short Sekund;
};
```

Bunday aniqlashda Sana_vaqt strukturasi xotirada 6 ta maydon*2 bayt=12 bayt joy egallaydi. Agar e’tibor berilsa strukturada ortiqcha joy egallangan

holatlar mavjud. Masalan, yil uchun qiymati 0 sonidan 99 sonigacha qiymat bilan aniqlanishi etarli (masalan, 2008 yilni 8 qiymati bilan ifodalash mumkin). Shuning uchun unga 2 bayt emas, balki 7 bit ajratish etarli. Xuddi shunday oy uchun 1..12 qiymatlarini ifodalashga 4 bit joy etarli va hakoza.

Yuqorida keltirilgan cheklovlardan keyin sana-vaqt strukturasini tejamli variantini aniqlash mumkin:

```
struct Sana_vaqt2 {
 unsigned Yil:7;
 unsigned Oy:4;
 unsigned Kun:5;
 unsigned Soat:6;
 unsigned Minut:6;
 unsigned Sekund:6;
};
```

Bu struktura xotiradan 5 bayt joy egallaydi.

Struktura funksiya argumenti sifatida

Strukturalar funksiya argumenti sifatida ishlatilishi mumkin. Buning uchun funksiya prototipida struktura turi ko‘rsatilishi kerak bo‘ladi. Masalan, talaba haqidagi berilganlarni o‘z ichiga oluvchi Talaba strukturasi turidagi berilganlarni Talaba_Adresi() funksiyasiga parametr sifatida berish uchun funksiya prototipi quyidagiko‘rinishda bo‘lishi kerak:

```
void Talaba_Adresi(Talaba);
```

Funksiyaga strukturani argument sifatida uzatishga misol sifatidagi dasturning matni:

```
#include <iostream>
#include <string.h>
using namespace std;
struct Talaba {
 char FISh[30];
```

```
 unsigned int Tug_yil;
 unsigned int Kurs;
 char Yunalish[50];
 float Reyting;
 unsigned char Jinsi[5];
 char Adres[50];
 bool status;
};
void Talaba_Adresi(Talaba);
int main()
{
 Talaba talaba;
 talaba.Kurs=2;
 talaba.Tug_yil=1988;
 strcpy(talaba.FISh,"Abdullaev A.A.");
 strcpy(talaba.Yunalish,"Informatika va Axborot  texnologiyalari");
 strcpy(talaba.Jinsi,"Erk");
 strcpy(talaba.Adres,"Toshkent, Yunusobod 6-3-8, tel: 224-45-78");
 talaba.Reyting=123.52;
 Talaba_Adresi(talaba);
return 0;
}
void Talaba_Adresi(Talaba t)
{
 cout<<"Talaba FIO: "<<t.FIO<<endl;
 cout<<"Adresi: "<<t.Adres<<endl;
}
```

Dastur bosh funksiyasida talaba strukturasi aniqlanib, uning maydonlariga qiymatlar beriladi. Keyin talaba strukturasi Talaba_Adresi() funksiyasiga argument sifatida uzatiladi. Dastur ishlashi natijasida ekranga quyidagi ma'lumotlar chop etiladi.

Talaba FIO: Abdullaev A.A.

Adresi: Toshkent, Yunusobod 6-3-8, tel: 224-45-78

Strukturalar massivi

O'z-o'zidan ma'lumki, struktura turidagi bitta berilgan bilan yechish mumkin bo'lgan masalalar doirasi juda tor va aksariyat holatlarda, qo'yilgan masala strukturalar majmuasini ishlatishni talab qiladi. Bu turdagi masalalarga berilganlar bazasini qayta ishlash masalalari deb qarash mumkin.

Strukturalar massivini e'lon qilish xuddi standart massivlarni e'lon qilishdek, farqi massiv turi o'rnida foydalanuvchi tomonidan aniqlangan struktura turining nomi yoziladi. Masalan, talabalar haqidagi berilganlarni o'z ichiga olgan massiv yaratish e'loni quyidagicha bo'ladi:

```
const int n=25;
Talaba talabalar[n];
```

Strukturalar massivining elementlariga murojaat odatdagi massiv elementlariga murojaat usullari orqali, har bir elementning maydonlariga murojaat esa '.' orqali amalga oshiriladi.

Quyidagi dasturda guruhidagi har bir talaba haqidagi berilganlarni klaviaturadan kiritish va guruh talabalarini familiya, ismi va sharifini chop qilinadi.

```
#include <iostream>
using namespace std;
#include <conio.h>
const int n=3;
struct Talaba {
char FISh[30];
```

```
unsigned int Tug_yil;
unsigned int Kurs;
char Yunalish[50];
float Reyting;
char Jinsi[6];
char Adres[50];
bool status;
};
void Talaba_Kiritish(Talaba t[]);
void Talabalar_FISh(Talaba t[]);
int main(int argc, char* argv[])
{
 Talaba talabalar[n];
 Talaba_Kiritish(talabalar);
 Talabalar_FISh(talabalar);
 return 0;
}
void Talabalar_FISh(Talaba t[])
{
  for(int i=0; i<n; i++)cout<<t[i].FISh<<endl;
}
void Talaba_Kiritish(Talaba t[])
{
 for(int i=0; i<n; i++) {
 cout<<i+1<<"-talaba malumotlarini kiriting:"<<endl;
 cout<<" Talaba FISh :"; cin.getline(t[i].FISh,30);
 cout<<" Kurs:"; cin>>t[i].Kurs;
 cout<<" Reyting bali:"; cin>>t[i].Reyting;
```

```
 cout<<" Tug''ilgan yili:"; cin>>t[i].Tug_yil;
 cout<<" Ta'lim yo'nalishi:"; cin.getline(t[i].Yunalish,50);
 cout<<" Jinsi(erkak,ayol):"; cin.getline(t[i].Jinsi,6);
 cout<<" Yashash adresi:"; cin.getline(t[i].Adres,50);
 }
}
```

Strukturalarga ko‘rsatkich

Struktura elementlariga ko‘rsatkichlar orqali murojaat qilish mumkin. Buning uchun strukturaga ko‘rsatkich o‘zgaruvchisi e’lon qilinishi kerak. Masalan, yuqorida keltirilgan misolda Talaba strukturasiga ko‘rsatkich quyidagicha e’lon qilinadi:

```
Talaba * k_talaba;
```

Ko‘rsatkich orqali aniqlangan struktura elementlariga murojaat “.” bilan emas, balki “->” vositasida amalga oshiriladi:

```
cout<<k_talaba ->FISh;
```

Strukturalarni ko‘rsatkich va adresni olish (&) vositasida funksiya argumenti sifatida uzatish mumkin. Quyida keltirilgan dastur bo‘lagida strukturani Talaba_Kiritish() funksiyasiga ko‘rsatkich orqali, Talabalar_FISh() funksiyasiga esa adresni olish vositasida uzatishga misol keltirilgan.

```
...
void Talaba_Kiritish(Talaba *t);
void Talabalar_FISh(Talaba & t);
int main( )
{
 Talaba * k_talaba;
 k_talaba=(Talaba*)malloc(n*sizeof(Talaba));
 Talaba_Kiritish(k_talaba);
 Talabalar_FISh(*k_talaba);
 return 0;
```

```
}
void Talabalar_FISh(Talaba & t)
{
 for(int i=0; i<n; i++)cout<<(&t+i)->FISh<<endl;
}
void Talaba_Kiritish(Talaba *t)
{
 for(int i=0; i<n; i++){
  cout<<i+1<<"-talaba malumotlarini kiriting:"<<endl;
  cout<<" Talaba FISh :";  cin.getline((t+i)->FISh,30);
  cout<<" Kurs:";  cin>>(t+i)->Kurs;
  ...
  }
}
```

Shunga e’tibor berish kerakki, dinamik ravishda hosil qilingan strukturalar massivi elementi bo‘lgan strukturaning maydoniga murojaatda “*” belgisi qo‘llanilmaydi.

Masala. Futbol jamoalari haqidagi ma’lumotlar – jamoa nomi, ayni paytdagi yutuqlar, durang va mag‘lubiyatlar sonlari, hamda raqib darvozasiga kiritilgan va o‘z darvozasidan o‘tkazib yuborilgan to‘plar sonlari bilan berilgan. Futbol jamoalarining turnir jadvali chop qilinsin. Jamoalarni jadvalda tartiblashda quyidagi qoidalarga amal qilinsin:

1) jamoalar to‘plagan ochkolarini kamayishi bo‘yicha tartiblanishi kerak;

2) agar jamoalar to‘plagan ochkolari teng bo‘lsa, ulardan nisbatan ko‘p g‘alabaga erishgan jamoa jadvalda yuqori o‘rinni egallaydi;

3) agar ikkita jamoaning to‘plagan ochkolari va g‘alabalar soni teng bo‘lsa, ulardan nisbatan ko‘p to‘p kiritgan jamoa jadvalda yuqori o‘rinni egallaydi.

Jamoa haqidagi berilganlarstruktura ko‘rinishida, jadval esa struktura

massivi sifati aniqlanadi:

```
struct Jamoa
{
 string Nomi;
 int Yutuq, Durang, Maglub, Urgan_tup, Utkazgan_tup;
 int Uyin, Ochko;
};
```

Bu yerda Uyin maydoni Yutuq, Durang va Maglub maydonlar yig‘indisi, jamoa to‘plagan ochkolar –Ochko=3*Yutuq+1*Durang ko‘rinishida aniqlanadi. Jamoalar massivi Ochko, Yutuq va Urgan_tup maydonlari bo‘yicha tartiblanadi.

Dastur matni:

```
struct Jamoa
{
 string Nomi;
 int Yutuq, Durang, Maglub, Urgan_tup, Utkazgan_tup;
 int Uyin, Ochko;
};
const nom_uzunligi=10;
int jamoalar_soni;
Jamoa * Jamoalar_Jadvali()
{
 char *jm_nomi=(char*)malloc(nom_uzunligi+1);
 cout<<" Jamoalar soni: "; cin>>jamoalar_soni;
 Jamoa * jm=new Jamoa[jamoalar_soni];
 for(int i=0; i<jamoalar_soni; i++){
  cin.ignore();
  cout<<i+1<<"-jamoa ma'lumotlari:\n";
  cout<<" Nomi: ";cin.getline(jm_nomi,nom_uzunligi);
```

```
 while(strlen(jm_nomi)<nom_uzunligi)strcat(jm_nomi," ");
 jm[i].Nomi.assign(snomi);
 cout<<" Yutuqlar soni: ";cin>> jm[i].Yutuq;
 cout<<" Duranglar soni: ";cin>>jm[i].Durang;
 cout<<" Mag'lubiyatlar soni: ";cin>>jm[i].Maglub;
 cout<<" Raqib darvozasiga urilgan to'plar soni: ";cin>>jm[i].Urgan_tup;
 cout<<" O'z darvozasigan o'tkazgan to'plar soni: ";cin>>jm[i].Utkazgan_tup;
 jm[i].Uyin=jm[i].Yutuq+jm[i].Durang + jm[i].Maglub;
 jm[i].Ochko=jm[i].Yutuq*3 +jm[i].Durang;
 }
free(snomi);
return jm;
}
void Utkazish(Jamoa & jamoa1, const Jamoa & jamoa2) {
 jamoa1.Nomi=jamoa2.Nomi;
 jamoa1.Yutuq=jamoa2.Yutuq;
 jamoa1.Durang=jamoa2.Durang;
 jamoa1.Maglub=jamoa2.Maglub;
 jamoa1.Urgan_tup=jamoa2.Urgan_tup;
 jamoa1.Utkazgan_tup=jamoa2.Utkazgan_tup;
 jamoa1.Uyin=jamoa2.Uyin;
 jamoa1.Ochko=jamoa2.Ochko;
}
Jamoa * Jadvalni_Tartiblash(Jamoa * jm)
{
 bool urin_almashdi=true;
 for(int i=0;i<jamoalar_soni-1 && urin_almashdi; i++){
  Jamoa Vaqtincha;
```

```
 urin_almashdi=false;
 for(int j=i; j<jamoalar_soni-1; j++)  {
 //  j-jamoaning ochkosi (j+1)-jamoa ochkosidan kattabo‘lsa, takrorlashning
 //  keyingi qadamiga o‘tilsin.
  if(jm[j].Ochko>jm[j+1].Ochko) continue;
  // j va (j+1)-jamoalarning ochkolari teng va j-jamoayutuqlari (j+1)-jamoa
  // yutuqlaridan ko‘p bo‘lsa,takrorlashning keyingi qadamiga o‘tilsin.
  if(jm[j].Ochko==jm[j+1].Ochko && jm[j].Yutuq>jm[j+1].Yutuq) continue;
  // j va (j+1)-jamoalarning ochkolari va yutuqlar soniteng va j-jamoa urgan
  // to‘plar soni (j+1)- jamoaurgan to‘plardan ko‘p bo‘lsa, takrorlashning
  // keyingi qadamiga o‘tilsin
  if(jm[j].Ochko==jm[j+1].Ochko && jm[j].Yutuq==jm[j+1].Yutuq &&
     jm[j].Urgan_tup>jm[j+1].Urgan_tup) continue;
 //yuqoridagi shartlarning birortasi ham bajarilmasa,j va (j+1)-jamoalar o‘rinlari
// almashtirilsin.
  urin_almashdi=true;
  Utkazish(Vaqtincha,jm[j]);
  Utkazish(jm[j],jm[j+1]);
  Utkazish(jm[j+1], Vaqtincha);
 }
}
 return jm;
}
  void Jadavlni_Chop_Qilish(const Jamoa *jm)
  {
char probel=’ ’;
cout<<"    FUTBOL JAMOALARINING TURNIR JADVALI\n" ;
cout<<"---------------------------------------\n";
```

```
cout<<"|  JAMOA   | O | Y | D | M |UrT|O'T|OChKO|\n";
cout<<"----------------------------------------\n";
 for(int i=0; i<jamoalar_soni; i++){
  cout<<"|"<<jm[i].Nomi.substr(0,10);cout<<'|';
  if(jm[i].Uyin<10)cout<<probel;  cout<<jm[i].Uyin<<" |";
  if(jm[i].Yutuq<10)cout<<probel; cout<<jm[i].Yutuq<<" |";
  if(jm[i].Durang<10)cout<<probel;  cout<<jm[i].Durang<<" |";;
  if(jm[i].Maglub<10)cout<<probel; cout<<jm[i].Maglub<<" |";
  if(jm[i].Urgan_tup<10)cout<<prorbel; cout<<jm[i].Urgan_tup<<" |";
  if(jm[i].Utkazgan_tup<10)cout<<prpbel; cout<<jm[i].Utkazgan_tup<<" |";;
  if(jm[i].Ochko<10)cout<<probel;   cout<<jm[i].Ochko<<"  |"<<endl;
 }
 cout<<"----------------------------------------\n";
}
int main()
{
 Jamoa *jamoa;
 jamoa=Berilganlarni_kiritish();
 jamoa=Jadvalni_Tartiblash(jamoa);
 Jadvalni_Chop_Qilish(jamoa);
 return 0;
}
```

Dastur bosh funksiya va quyidagi vazifalarni bajaruvchi to‘rtta funksiyadan tashkil topgan:

1) Jamoa*Jamoalar_Jadvali()– jamoalar haqidagi berilganlarni saqlaydigan Jamoa strukturalaridan tashkil topgan dinamik massiv yaratadi va unga oqimdan har bir jamoa berilganlarni o‘qib joylashtiradi. Hosil bo‘lgan massivga ko‘rsatkichni funksiya natijasi sifatida qaytaradi;

2) Jamoa * Jadvalni_Tartiblash(Jamoa * jm) – argument orqali ko‘rsatilgan massivni masala sharti bo‘yicha tartiblaydi va shu massivga ko‘rsatkichni qaytaradi;

3) void Utkazish(Jamoa & jamoa1, const Jamoa & jamoa2) – jamoa2 strukturasidagi maydonlarni jamoa1 strukturasiga o‘tkazadi. Bu funksiya Jadvalni_Tartiblash() funksiyasidan massivdagi ikkita strukturani o‘zaro o‘rinlarini almashtirish uchun chaqiriladi;

4) void Jadavlni_Chop_Qilish(const Jamoa *jm) – argumentda berilgan massivni turnir jadvali qolipida chop qiladi.

Uchta jamoa haqida ma’lumot berilganda dastur ishlashining natijasi quyidagicha bo‘lishi mumkin:

```
FUTBOL JAMOALARINING TURNIR JADVALI
-----------------------------------------------------
| JAMOA   | O | Y | D | M |UrT|O‘T|Och|
-----------------------------------------------------
|Bunyodkor |20 |15 | 3 | 2 |30 |10 | 48  |
|Paxtakor    |20 |11 | 5 | 4 |20 |16 | 38  |
|Neftchi      |20 | 8 | 5 | 7 |22 |20 |  29  |
...
-----------------------------------------------------
```

Birlashmalar va ular ustida amallar

Birlashmalar xotiraning bitta sohasida (bitta adres bo‘yicha) har xil turdagi bir nechta berilganlarni saqlash imkonini beradi.

Birlashma e’loni union kalit so‘zi, undan keyin identifikator va blok ichida har xil turdagi elementlar e’lonidan iborat bo‘ladi, masalan:

```
union Birlashma
{
 int n;
 unsigned long N;
```

```
 char Satr[10];
 };
```

Birlashmaning bu e’lonida kompilyator tomonidan Birlashma uchun uning ichidagi eng ko‘p joy egallovchi elementning – Satr satrining o‘lchamida, ya’ni 10 bayt joy ajratiladi. Vaqtning har bir momentida birlashmada, e’lon qilingan maydonlarning faqat bittasining turidagi berilgan mavjud deb hisoblanadi. Yuqoridagi misolda Birlashma ustida amal bajarilishida uning uchun ajratilgan xotirada yoki int turidagi n yoki unsigned long turidagi N yoki Satr satr qiymati joylashgan deb hisoblanadi.

Birlashma maydonlariga xuddi struktura maydonlariga murojaat qilgandek ‘.’ orqali murojaat qilinadi.

Strukturalardan farqli ravishda birlashma e’lonida faqat uning birinchi elementiga boshlang‘ich qiymat berish mumkin:

```
union Birlashma
{
 int n;
 unsigned long N;
 char Satr[10];
}
birlashma={25};
```

Bu misolda birlashma birlashmasining n maydoni boshlang‘ich qiymat olgan hisoblanadi.

Birlashma elementi sifatida strukturalar kelishi mumkin va ular odatda berilganni “bo‘laklarga” ajratish yoki “bo‘laklardan” yaxlit berilganni hosil qilish uchun xizmat qiladi. Misol uchun so‘zni baytlarga, baytlarni tetradalarga (4 bitga) ajratish va qaytadan birlashtirish mumkin.

Quyida baytni katta va kichik yarim baytlarga ajratishda birlashma va strukturadan foydalanilgan dasturni matni keltirilgan.

```
#include <iostream>
using namespace std;
union BCD {
 unsigned char bayt;
 struct {
    unsigned char lo:4;
    unsigned char hi:4;
   } bin;
} bcd;
int main()
{
 bcd.bayt=127;
 cout<<"\n Katta yarim bayt : "<<(int)bcd.bin.hi;
 cout<<"\n Kichik yarim bayt: "<<(int)bcd.bin.lo;
 return 0;
}
```

Dastur bosh funksiyasida BCD birlashmasining bayt o‘lchamida bayt maydoniga 127 qiymati beriladi va uning katta va kichik yarim baytlari chop etiladi.

Dastur ishlashi natijasida ekranga quyidagi natijalar chiqadi:

```
Katta yarim bayt : 7
Kichik yarim bayt: 15
```

Masala. Haqiqiy turdagi sonning kompyuter xotirasidagi ichki ko‘rinishini chop qilish. Haqiqiy son float turida deb hisoblanadi va u xotirada 4 bayt joy egallaydi. Qo‘yilgan masalani yechish uchun birlashma xususiyatdan foydalaniladi, ya’ni xotiraning bitta adresiga haqiqiy son va belgilar massivi joylashtiriladi. Haqiqiy son xotiraga o‘qilib, belgilar massivining har bir elementining (baytining) ikkilik ko‘rinishi chop etiladi.

Dasturmatni:

```
#include <iostream>
using namespace std;
const unsigned char bitlar_soni=7;
const unsigned char format=sizeof(float);
void Belgi_2kodi(unsigned char blg);
union Son_va_Belgi {
 float son;
 unsigned char belgi[format];
};
int main()
{
 Son_va_Belgi son_va_belgi;
 cin>>son_va_belgi.son;
 for(int b=format-1; b>=0; b--) Belgi_2kodi(son_va_belgi.belgi[b]);
 return 0;
}
void Belgi_2kodi(unsigned char blg)
{
 unsigned char l0000000=128;
 for(int i=0;i<=bitlar_soni;i++){
  if(blg&l0000000)cout<<'1';else cout<<'0';
  blg=blg<<1;
 }
 cout<<' ';
}
```

Dasturda Son_va_Belgi birlashmasini e’lon qilish orqali float turidagi x o‘zgaruvchisini va float turi formatining baytlardagi uzunligidagi belgilardan

iborat belgi massivini xotiraning bitta joyiga joylashuviga erishiladi. Bosh funksiyada birlashma turidagi son_va_belgi o‘zgaruvchisi e’lon qilinadi va uning x maydoniga klaviaturadan haqiqiy son o‘qiladi. Keyin belgilar massividagi har bir elementning ikkilik kodi chop etiladi. Ikkilik kodni chop etish 8 marta baytni 7 – razryadidagi sonni chop etish va bayt razryadlarini bittaga chapga surish orqali amalga oshiriladi. Shunga e’tibor berish kerakki, belgilar massividagi elementlarning ikkilik kodlarini chop qilish o‘ngdan chap tomonga bajarilgan. Bunga sabab, son ichki formatidagi baytlarning xotirada “*kichik bayt - kichik adresda*” qoidasiga ko‘ra joylashuvidir.

Dasturga -8.5 soni kiritilsa, ekranda

11000001 00001000 00000000 00000000

ko‘rinishidagi ikkilik sonlari ketma-ketligi paydo bo‘ladi.

Nazorat savollari

1. Tuzilma deb nimaga aytiladi?
2. Birlashma deb nimaga aytiladi?
3. Birlashma va tuzilmaning farqi nimada?
4. Tuzilma maydonlari qanday turdarda bo‘lishi mumkin?
5. Tuzilma maydoni o‘lchamlari xajmini qanday ko‘rinishda aniq ko‘rsatish mumkin?
6. Tuzilmani funksiya argumenti sifatida ishlatishga misol keltiring.
7. Tuzilmalar massivi qanday e’lon qilinadi?
8. Tuzilma maydonlariga qanday murojaat qilish mumkin?
9. Tuzilmaga ko‘rsatkich qanday ishlatiladi?

§20. Identifikatorlarning amal qilish doirasi. Makroslarni aniqlash va joylashtirish

Identifikatorlarning amal qilish doirasi

O‘zgaruvchilar funksiya tanasida yoki undan tashqarida e’lon qilinishi mumkin. Funksiya ichida e’lon qilingan o‘zgaruvchilarga *lokal o‘zgaruvchilar*deyiladi. Bunday o‘zgaruvchilar xotiradagi dastur stekida

joylashadi va faqat o‘zi e’lon qilingan funksiya tanasida amal qiladi. Boshqaruv asosiy funksiyaga qaytishi bilan lokal o‘zgaruvchilar uchun ajratilgan xotira bo‘shatiladi (o‘chiriladi).

Har bir o‘zgaruvchi o‘zining amal qilish sohasi va yashash vaqti xususiyatlari bilan xarakterlanadi.

O‘zgaruvchi *amal qilish sohasi* deganda o‘zgaruvchini ishlatish mumkin bo‘lgan dastur sohasi (qismi) tushuniladi. Bu tushuncha bilan o‘zgaruvchining *ko‘rinish sohasi* uzviy bog‘langan. O‘zgaruvchi amal qilish sohasidan chiqqanda ko‘rinmay qoladi. Ikkinchi tomondan, o‘zgaruvchi amal qilish sohasida bo‘lishi, lekin ko‘rinmasligi mumkin. Bunda ko‘rinish sohasiga ruxsat berish amali “::” yordamida ko‘rinmas o‘zgaruvchiga murojat qilish mumkin bo‘ladi.

O‘zgaruvchining *yashash vaqti* deb, u mavjud bo‘lgan dastur bo‘lagining bajarilishiga ketgan vaqt intervaliga aytiladi.

Lokal o‘zgaruvchilar o‘zlari e’lon qilingan funksiya yoki blok chegarasida ko‘rinish sohasiga ega. Blokdagi ichki bloklarda xuddi shu nomdagi o‘zgaruvchi e’lon qilingan bo‘lsa, ichki bloklarda bu lokal o‘zgaruvchi ham amal qilmay qoladi. Lokal o‘zgaruvchi yashash vaqti – blok yoki funksiyani bajarish vaqti bilan aniqlanadi. Bu hol shuni anglatadiki, turli funksiyalarda bir-biriga umuman bog‘liq bo‘lmagan bir xil nomdagi lokal o‘zgaruvchilarni ishlatish mumkin.

Quyidagi dasturda main() va sum() funksiyalarida bir xil nomdagi o‘zgaruvchilarni ishlatish ko‘rsatilgan. Dasturda ikkita sonning yig‘indisi hisoblanadi va chop etiladi:

```
#include <iostream>
using namespace std;
int sum(int a, int b);        // funksiya prototipi
int main()
{
 int x=r, y=4;          // lokal o‘zgaruvchilar
cout << sum(x, y);
```

```
 return 0;
}
int sum(int a,int b)
{
int x=a+b;                    //  lokal o‘zgaruvchi
return x;
}
```

Global o‘zgaruvchilar dastur matnida funksiya aniqlanishidan tashqarida e’lon qilinadi va e’lon qilingan joyidan boshlab dastur oxirigacha amal qiladi.

```
#include <iostream>
using namespace std;
int f1();
int f2();
int main()
{
cout<<f1()<<" "<<f2()<<endl;
return 0;
}
int f1()
{
return x;              // kompilyatsiya xatosi ro‘y beradi
}
int x=10;              // global o‘zgaruvchi e’loni
int f2()
{
return x*x;
}
```

Yuqorida keltirilgan dasturda kompilyatsiya xatosi ro‘y beradi, chunki f1() funksiya uchun x o‘zgaruvchisi noma’lum hisoblanadi.

Dastur matnida global o‘zgaruvchilarni ular e’lonidan keyin yozilgan ixtiyoriy funksiyada ishlatish mumkin. Shu sababli, global o‘zgaruvchilar dastur matnining boshida yoziladi. Funksiya ichidan global o‘zgaruvchiga murojat qilish uchun funksiyada uning nomi bilan mos tushadigan lokal o‘zgaruvchilar bo‘lmasligi kerak. Agar global o‘zgaruvchi e’lonida unga boshlang‘ich qiymat berilmagan bo‘lsa, ularning qiymati 0 hisoblanadi. Global o‘zgaruvchining amal qilish sohasi uning ko‘rinish sohasi bilan ustma-ust tushadi.

Shuni qayd etish kerakki, tajribali dastur tuzuvchilar imkon qadar global o‘zgaruvchilarni ishlatmaslikka harakat qilishadi, chunki bunday o‘zgaruvchilar qiymatini dasturning ixtiyoriy joyidan o‘zgartirish xavfi mavjudligi sababli dastur ishlashida mazmunan xatolar yuzaga kelishi mumkin. Bu fikrni tasdiqlovchi dasturni ko‘raylik.

```
#include <iostream>
using namespace std;
int test = 100;        // global o‘zgaruvchi e’loni
void Chop_qilish(void);
int main()
{
 int test=10;          // lokal o‘zgaruvchi e’loni
 Chop_qilish();        // global o‘zgaruvchi chop qilish funksiyasini chaqirish
 cout << “Lokal o’zgaruvchi: ” << test << ‘\n’;
 return 0;
}
void Chop_qilish(void)
{
 cout<< “Global o’zgaruvchi: ” << test << ‘\n’;
}
```

Dastur boshida test global o‘zgaruvchisi 100 qiymati bilan e’lon qilinadi. Keyinchalik, main() funksiyasida test nomi bilan lokal o‘zgaruvchisi 10 qiymati

bilan e’lon qilinadi. Dasturda, Chop_qilish() funksiyasiga murojaat qilinganida, asosiy funksiya tanasidan vaqtincha chiqiladi va natijada main() funksiyasida e’lon qilingan barcha lokal o‘zgaruvchilarga murojaat qilish mumkin bo‘lmay qoladi. Shu sababli Chop_qilish() funksiyasida global test o‘zgaruvchisining qiymatini chop etiladi. Asosiy dasturga qaytilgandan keyin, main() funksiyasidagi lokal test o‘zgaruvchisi global test o‘zgaruvchisini “*yopadi*” va lokal test o‘zgaruvchini qiymati chop etiladi. Dastur ishlashi natijasida ekranga quyidagi natijalar chop etiladi:

```
Global o‘zgaruvchi: 100
Lokal o‘zgaruvchi: 10
```

“::”amali

Yuqorida qayd qilingandek, lokal o‘zgaruvchi e’loni xuddi shu nomdagi global o‘zgaruvchini “*yopadi*” va bu joydan global o‘zgaruvchiga murojat qilish imkoni bo‘lmay qoladi. C++ tilida bunday holatlarda ham global o‘zgaruvchiga murojat qilish imkoniyati saqlanib qolingan. Buning uchun “*ko‘rinish sohasiga ruxsat berish*” amalidan foydalanish mumkin va o‘zgaruvchi oldiga ikkita nuqta –“::” qo‘yish zarur bo‘ladi. Misol tariqasida quyidagi dastur keltirilgan.

```
#include <iostream>
using namespace std;
int uzg=5;                    // global o‘zgaruvchi e’loni
int main()
{
 int uzg=70;                  // lokal o‘zgaruvchi e’loni
 cout << uzg << ‘/n’;         // lokal o‘zgaruvchini chop etish
 cout << ::uzg << ‘/n’;       // global o‘zgaruvchini chop etish
 return 0;
}
```

Dastur ishlashi natijasida ekranga oldin 70 va keyin 5 sonlari chop etiladi.

Makroslarni aniqlash va joylashtirish

Makros – bu dastur (kod) bo‘lagi bo‘lib, ko‘rinishi va ishlashi xuddi funksiyadek. Biroq u funksiya emas. Funksiyalar va makroslar o‘rtasida bir nechta farqlar mavjud:

– dastur matnida uchragan makros ifodasi o‘z aniqlanishi (tanasi bilan) bilan preprotsessor ishlash paytida, ya’ni dastur kompilyatsiyasidan oldinalmashtiriladi. Shu sababli makros funksiyani chaqirish bilan bog‘liq qo‘shimcha vaqt sarfini talab qilmaydi;

– makroslardan foydalanish dasturning boshlang‘ich kodi (matnini) kattalashuviga olib keladi. Bunga qarama-qarshi holda funksiya kodi yagona nusxada bo‘ladi va u dastur kodini qisqarishiga olib keladi. Lekin funksiyani chaqirish uchun qo‘shimcha resurslar sarflanadi;

– kompilyator makrosdagi turlar mosligini tekshirmaydi. Shu sababli, makrosga argument jo‘natishda turlarning mosligi yoki argumentlar sonining to‘g‘ri kelishi yoki kelmasligi haqidagi xatolik xabarlari berilmaydi;

– makros boshlang‘ich kodga dastur bo‘lagini qo‘yish vositasi bo‘lganligi va bunday bo‘laklar matnning turli joylariga qo‘yish mumkinligi sababli makroslar bilan bog‘liq fiksirlangan, yagona adreslar bo‘lmaydi. Shu sababli makroslarda ko‘rsatkichlar e’lon qilish yoki makros adreslarini ishlatish imkoniyati yo‘q.

Makroslarni aniqlash uchun #define direktivasidan foydalaniladi. Funksiyaga o‘xshab makroslar ham parametrlarga ega bo‘lishi mumkin. Misol uchun ikkita sonni ko‘paytmasini hisoblovchi makros quyidagicha aniqlanadi:

```
#include <iostream.h>
#define KUPAYTMA(x,y)((x)+(y))
int main()
{
 int a=2, b=3;
 c=KUPAYTMA(a,b);
```

```
 cout<<c;
 return 0;
}
```

Misoldan ko‘rinib turibdiki, tashqi ko‘rinishi bo‘yicha mak-roslardan foydalanish funksiyalardan foydalanishga o‘xshash. Shuning uchun ularni ayrim hollarda ularga *psevdofunksiyalar* deb atashadi. Makroslar aniqlanishining yana bir o‘ziga xos tomoni shundaki, C++ tilida ularning nomlarini katta harflar bilan yozishga kelishilgan.

Yuqoridagi misolning o‘ziga xos ko‘rinishidan biri bu makros parametrlarini qavs ichida yozilishidir. Aks holda makros aniqlanishini (tanasini) matnga qo‘yishda mazmunan xatolik yuzaga kelishi mumkin. Masalan,

```
#define KVADRAT(x) x*x
```

Dastur matnida ushbu makros ishlatilgan satr mavjud bo‘lsin:

```
int y=KVADRAT(2);
```

u holda, makros aniqlanishini matnga qo‘yish natijasida dastur matnida yuqoridagi satr quyidagi ko‘rinishga keladi:

```
int y=2*2;
```

Lekin, dasturda makrosni ishlatish

```
int y=KVADRAT(x+1);
```

ko‘rinishida bo‘lsa, makros aniqlanishini matnga qo‘yish natijasida ushbu satr

```
int y=x+1*x+1;
```

ko‘rsatmasi bilan almashtiriladiki, bu albatta kutilgan almashtirish emas. Shu sababli, makros aniqlanishida umumiy qoida sifatida parametrlarni qavsga olish tavsiya etiladi:

```
#define KVADRAT(x)(x)*(x)
```

Agar makros chaqirilishida turga keltirish operatoridan foydalangan holat bo‘lsa, makros tanasini to‘liqligicha qavsga olish talab qilinadi. Misol uchun dastur matnida makrosga murojaat quyidagicha bo‘lsin:

```
double x=(double)KVADRAT(x+1);
```

Bu holda makros aniqlanishi

```
#define KVADRAT(x)((x)*(x))
```

ko‘rinishi to‘g‘ri hisoblanadi.

Makros aniqlanishida oxirgi eslatma sifatida shuni qayd etish kerakki, ortiqcha probellar makrosdan foydalanishda xatoliklarga olib kelishi mumkin. Masalan

```
#define ChOP_QILISh (x)cout<<x
```

makros aniqlanishida makros nomi ChOP_QILISh va parametrlar ro‘yxati (x) o‘rtasida ortiqcha probel qo‘yilgan. Preprotsessor bu makrosni parametrsiz makros deb qabul qiladi, hamda “(x)cout<<x” satr ostini makros tanasi deb hisoblaydi va makros almashtirishlarda shu satrni dastur matniga qo‘yilada. Natijada kompilyatsiya xatosi ro‘y beradi. Xatoni tuzatish uchun makros nomi va parametrlar ro‘yxati o‘rtasidagi probelni olib tashlash etarli:

```
#define ChOP_QILISh(x)cout<<x
```

Agar makros aniqlanishi bitta satrga sig‘masa, shu satr oxiriga ‘\’ belgisini qo‘yish orqali keyingi satrda davom ettirish mumkin:

```
#define BURChAK3(a,b,c)(unsigned int)a+(unsigned int)b\
>(unsigned int)c &&(unsigned int)a+(unsigned int)s>\ (unsigned int)b &&(unsigned int)b+(unsigned int)s>\ (unsigned int)a
```

Makros aniqlanishida boshqa makroslar ishtirok etishi mumkin.Quyidagi misolda ichma-ich joylashgan makros aniqlanishi ko‘rsatilgan.

```
#define PI 3.14159
#define KVADRAT(x) ((x)*(x))
#define AYLANA_YuZI(r)(PI* KVADRAT(r))
```

Foydalanishga zarurati qolmagan makrosni dastur matnining ixtiyoriy joyida #undef direktivasi bilan bekor qilish mumkin, ya’ni shu satrdan keyin makros preprotsessor uchun noaniq hisoblanadi. Quyida aylana yuzasini hisoblaydigan dastur matni keltirilgan.

```
#include <iostream.h>
#define PI 3.14159
#define KVADRAT(x) ((x)*(x))
#define AYLANA_YuZI(r)(PI* KVADRAT(r))
int main()
{
 double r1=5,r2=10;
 double c1,c2;
 c1=AYLANA_YuZI(r1);
 #undef AYLANA_YuZI
 c2=AYLANA_YuZI(r2);
 cout<<c1<<endl;
 cout<<c2<<endl;
 return 0;
}
```

Dastur kompilyatsiyasida “c1=AYLANA_YuZI(r1);” satr normal qayti ishlangan holda “c2=AYLANA_YuZI(r2);” satri uchun AYLANA_YuZI funksiyasi aniqlanmaganligi haqida xatolik xabari chop etiladi.

Makroslarda ishlatiladigan amallar

Makroslar ishlatilishi mumkin bo‘lgan ikkita amal mavjud: ‘#’- satrni joylashtirish va ” ##” - satrni ulash amallari.

Agar makros parametri oldida ‘#’- satrni joylashtirish amali qo‘yilgan bo‘lsa, makros aniqlanishini matnga qo‘yish paytida shu o‘ringa mos argumentning (o‘zgaruvchining) nomi qo‘yiladi. Buni quyidagi misolda ko‘rish mumkin:

```
#include <iostream.h>
#define UZG_NOMI(uzg) cout<<#uzg<<'='<<uzg;
int main()
{
```

```
 int x=10;
 UZG_NOMI(x);
 return 0;
}
```

Dastur ishlashi natijasida ekranda

```
x=10
```

satri paydo bo‘ladi.

Satr ulash amali ikkita satrni bittaga birlashtirish uchun xizmat qiladi. Satrlarni birlashtirishdan oldin ularni ajratib turgan probellar o‘chiriladi. Agar hosil bo‘lgan satr nomidagi makros mavjud bo‘lsa, preprotsessor shu makros tanasini chaqiruv bo‘lgan joyga joylashtiradi.

Misoluchun,

```
#include <iostream.h>
#define MACRO_BIR cout<<"MACRO_1";
#define MACRO_IKKI cout<<"MACRO_2";
#define MACRO_BIRLAShMA(n) MACRO_##n
int main(int argc, char* argv[])
{
 int x=10;
 MACRO_BIRLAShMA(BIR);
 cin>>x;
 return 0;
}
```

dastur preprotsessor tomonidan qayta ishlangandan keyin uning oraliq matni kuyidagi ko‘rinishda bo‘ladi:

```
int main(int argc, char* argv[])
{
 int x=10;
 cout<<"MACRO_1";
```

```
 cin>>x;
 return 0;
}
```

Satrlarni ulash amalidan yangi o‘zgaruvchilarni hosil qilish uchun foydalanish mumkin.

```
#define UZG_ELONI(i) int var ## i
...
UZG_ELONI(1);
...
```

Yuqoridagi misolda makros o‘z aniqlanishi bilan almashtirish natijasida “UZG_ELONI(1);” satri o‘rnida “int var1;”ko‘rsatmasi paydo bo‘ladi.

Nazorat savollari

1. Lokal va global o’zgaruvchilarning farqi nimada?
2. :: amali nima uchun xizmat qiladi?
3. Makroslar nima uchun xizmat qiladi?
4. Makroslar qanday e’lon qilinadi?
5. Makroslar bilan funksiyalarning farqi nimada?
6. define direktivasi nima uchun xizmat qiladi?

§21. Standart oqimlar. Berilganlarni formatlash

O‘qish-yozish oqimlari

Oqim tushunchasi 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 chiqarish;
- oqimdagi berilganlar blokini yangilash;
- oqimdan yozuvni o‘qish;
- oqimga yozuvni chiqarish.

Oqim bilan ishlaydigan barcha funksiyalar buferli, formatlashgan yoki formatlashmagan o‘qish-yozishni ta’minlaydi.

Dastur ishga tushganda o‘qish-yozishning quyidagi standart oqimlar 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 (ekran), stdprn – printer bilan, hamda stdaux – kompyuter yordamchi portlariga bog‘langan 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. Masalan, belgilarni tashqi qurilmaga chiqarishda belgilar ketma-ketligi buferda jamlanadi va bufer to‘lgandan keyingina tashqi qurilmaga chiqariladi.

Hozirdagi operatsion sistemalarda klaviatura va displeylar matn fayllari sifatida qaraladi. Haqiqatdan ham berilganlarni klaviaturadan dasturga kiritish (o‘qish) mumkin, ekranga esa chiqarish (yozish) mumkin. Dastur ishga tushganda standart o‘qish va yozish oqimlari o‘rniga matn fayllarni tayinlash orqali bu oqimlarni qayta aniqlash mumkin. Bu holatni *o‘qishni (yozishni) qayta adreslashro‘y berdi* deyiladi. O‘qish uchun qayta adreslashda ‘<’ belgisidan, yozish uchun esa ‘>’ belgisidan foydalaniladi. Misol uchun gauss.exe bajariluvchi dastur berilganlarni o‘qishni klaviaturadan emas, balki massiv.txt faylidan amalga oshirish zarur bo‘lsa, u buyruq satrida quyidagi ko‘rinishda yuklanishi zarur bo‘ladi:

```
gauss.exe < massiv.txt
```

Agar dastur natijasini natija.txt fayliga chiqarish zarur bo‘lsa

```
gauss.exe > natija.txt
```

satri yoziladi.

Va nihoyat, agar berilganlarni massiv.txt faylidan o‘qish va natijani natija.txt fayliga yozish uchun

```
gauss.exe < massiv.txt > natija.txt
```

buyruq satri teriladi.

Umuman olganda, bir dasturning chiqish oqimini ikkinchi dasturning kirish oqimi bilan bog‘lash mumkin. Buni *konveyrli jo‘natish* deyiladi. Agar ikkita junat.exe dastursi qabul.exe dastursiga berilganlarni jo‘natishi kerak bo‘lsa, u holda ular o‘rtasiga ‘|’ belgi qo‘yib yoziladi:

```
junat.exe | qabul.exe
```

Bu ko‘rinishdagi dasturlar o‘rtasidagi konveyrli jo‘natishni operatsion sistemaning o‘zi ta’minlaydi.

Belgilarni o‘qish-yozish funksiyalari

Belgilarni o‘qish-yozish funksiyalari makros ko‘rinishida amalga oshirilgan.

getc() makrosi tayinlangan oqimdan navbatdagi belgini qaytaradi va kirish oqimi ko‘rsatkichini keyingi belgini o‘qishga moslagan holda oshiradi. Agar o‘qish muvaffaqiyatli bo‘lsa getc() funksiyasi ishorasiz int ko‘rinishidagi qiymatni, aks holda EOF qaytaradi. Ushbu funksiya prototipi quyidagicha:

```
int getc(FILE * stream)
```

EOF identifikator makrosi

```
#define EOF(-1)
```

ko‘rinishida aniqlangan va o‘qish-yozish amallarida fayl oxirini belgilash uchun xizmat qiladi. EOF qiymati ishorali char turida deb hisoblanadi. Shu sababli o‘qish-yozish jarayonida unsigned char turidagi belgilar ishlatilsa, EOF makrosini ishlatib bo‘lmaydi.

Navbatdagi misol getc() makrosini ishlatishni namoyon qiladi.

```
#include <iostream>
using namespace std;
#include <stdio.h>
```

```
int main()
{
 char ch;
 cout<<”Belgini kiriting: “;
 ch=getc(stdin);
 cout<<”Siz ”<<ch<<” belgisini kiritdingiz.\n”;
 return 0;
}
```

getc() makrosi aksariyat holatlarda stdin oqimi bilan ishlatilganligi sababli, uning getc(stdin) ko‘rinishiga ekvivalent bo‘lgan int getchar() makrosi aniqlangan. Yuqoridagi misolda “ch=getc(stdin);” qatorini “ch=getchar();” qatori bilan almashtirish mumkin.

Belgini oqimga chiqarish uchun putc() makrosi aniqlangan va uning prototipi

```
int putc (int s, FILE * stream);
```

ko‘rinishida aniqlangan. putc() funksiyasi stream nomi bilan berilgan oqimga s belgini chiqaradi. Funksiya qaytaruvchi qiymati sifatida int turiga aylantirilgan s belgi bo‘ladi. Agar belgini chiqarishda xatolik ro‘y bersa EOF qaytariladi.

putc()funksiyasini standart stdout oqimi bilan bog‘langan holati - putc(c,strout) uchun putchar(c) makrosi aniqlangan.

Satrlarni o‘qish - yozish funksiyalari

Oqimdan satrni o‘qishga mo‘ljallangan gets() funksiyasining prototipi

```
char * gets(char *s);
```

ko‘rinishida aniqlangan. gets() funksiyasi standart oqimdan satrni o‘qiydi va uni s o‘zgaruvchisiga joylashtiradi. Joylashtirish paytida oqimdagi ‘\n’ belgisi ‘\0’ belgisi bilan almashtiriladi. Bu funksiyani ishlatishda o‘qilayotgan satrning uzunligi s satr uchun ajratilgan joy uzunligidan oshib ketmasligini nazorat qilish kerak bo‘ladi.

puts() funksiyasi

int puts(const char *s);

ko‘rinishida bo‘lib, u standrat oqimga argumentda ko‘rsatilgan satrni chiqaradi. Bunda satr oxiriga yangi satrga o‘tish belgisi ‘\n’ qo‘shiladi. Agar satrni oqimga chiqarish muvaffaqiyatli bo‘lsa puts() funksiyasi manfiy bo‘lmagan sonni, aks holda EOF qaytaradi. Satrni o‘qish-yozish funksiyalarini ishlatishga misol tariqasida quyidagi dasturni keltirish mumkin.

```
#include <stdio.h>
int main()
{
 char *s;
 puts(”Satrni kiriting: “); gets(s);
 puts(”Kiritilgan satr: ”);
 puts(s);
 return 0;
}
```

Formatli o‘qish va yozish funksiyalari

Formatli o‘qish va yozish funksiyalari – scanf() va printf() funksiyalari C tilidan vorislik bilan olingan. Bu funksiyalarni ishlatish uchun “stdio.h” sarlavha faylini dasturga qo‘shish kerak bo‘ladi.

Formatli o‘qish funksiyasi scanf() quyidagi prototipga ega:

int scanf(const char * <format>[<adres>,...])

Bu funksiya standart oqimdan berilganlarni formatli o‘qishni amalga oshiradi. Funksiya, kirish oqimidagi maydonlar ketma-ketligi ko‘rinishidagi belgilarni birma-bir o‘qiydi va har bir maydonni <format> satrida keltirilgan format aniqlashtiruvchisiga mos ravishda formatlaydi. Oqimdagi har bir maydonga format aniqlashtiruvchisi va natija joylashadigan o‘zgaruvchining adresi bo‘lishi shart. Boshqacha aytganda, oqimdagi maydon (ajratilgan belgilar ketma-ketligi) ko‘rsatilgan formatdagi qiymatga akslantiriladi va o‘zgaruvchi

bilan nomlangan xotira bo‘lagiga joylashtiriladi (saqlanadi). Funksiya oqimdan berilganlarni o‘qish jarayonini “*to‘ldiruvchi belgini*” uchratganda yoki oqim tugashi natijasida to‘xtatishi mumkin. Oqimdan berilganlarni o‘qish muvafaqqiyatli bo‘lsa, funksiya muvafaqqiyatli aylantirilgan va xotiraga saqlangan maydonlar sonini qaytaradi. Agar hech bir maydonni saqlash imkoni bo‘lmagan bo‘lsa, funksiya 0 qiymatini qaytaradi. Oqim oxiriga kelib qolganda (fayl yoki satr oxiriga) o‘qishga harakat bo‘lsa, funksiya EOF qiymatini qaytaradi.

Formatlash satri –<format> belgilar satri bo‘lib, u uchta toifaga bo‘linadi:

– to‘ldiruvchi belgilar;
– to‘ldiruvchi belgilardan farqli belgilar;
– format aniqlashtiruvchilari.

To‘ldiruvchi-belgilar – bu probel, ‘\t’, ‘\n’ belgilari.Bu belgilar formatlash satridan o‘qiladi, lekin saqlanmaydi.

To‘ldiruvchi belgilardan farqli belgilar – bu qolgan barcha ASCII belgilari, ‘%’ belgisilan tashqari. Bu belgilar formatlash satridan o‘qiladi, lekin saqlanmaydi.

Format aniqlashtiruvchilari va ularning vazifasi

Komponenta	**Bo‘lishi shart yoki yo‘q**	**Vazifasi**
[*]	Yo‘q	Navbatdagi ko‘rib chiqilayotgan maydon qiymatini o‘zgaruvchiga o‘zlashtirmaslik belgisi. Kirish oqimidagi maydon ko‘rib chiqiladi, lekin o‘zgaruvchida saqlanmaydi.
[<kenglik>]	Yo‘q	Maydon kengligini aniqlashtiruvchisi. O‘qiladigan belgilarning maksimal sonini aniqlaydi. Agar oqimda to‘ldiruvchi belgi yoki almashtirilmaydigan belgi uchrasi funksiya nisbatan kam sondagi belgilarni o‘qishi mumkin.

[F|N]	Yo‘q	O‘zgaruvchi ko‘rsatkichining (adresining) modifikatori: F – far pointer; N- near pointer
[h|l|L]	Yo‘q	Argument turining modifikatori. <tur belgisi> bilan aniqlangan o‘zgaruvchining qisqa (short - h) yoki uzun (long – l, L) ko‘rinishini aniqlaydi.
<tur belgisi>	Ha	Oqimdagi belgilarni almashtiriladigan tur belgisi

Format aniqlashtiruvchilari – oqim maydonidagi belgilarni ko‘rib chiqish, o‘qish va adresi bilan berilgan o‘zgaruvchilar turiga mos ravishda almashtirish jarayonini boshqaradi. Har bir format aniqlashtiruvchisiga bitta o‘zgaruvchi adresi mos kelishi kerak. Agar format aniqlashtiruvchilari soni o‘zgaruvchilardan ko‘p bo‘lsa, natija nima bo‘lishini oldindan aytib bo‘lmaydi. Aks holda, ya’ni o‘zgaruvchilar soni ko‘p bo‘lsa, ortiqcha o‘zgaruvchilar inobatga olinmaydi.

Format aniqlashtiruvchisi quyidagi ko‘rinishga ega:

% [*][<kenglik>] [F|N] [h|l|L] <tur belgisi>

Format aniqlashtiruvchisi ‘%’ belgisidan boshlanadi va undan keyin 1–jadvalda keltirilgan shart yoki shart bo‘lmagan komponentlar keladi.

Oqimdagi belgilar bo‘lagini almashtiriladigan tur alomatining qabul qilishi mumkin bo‘lgan belgilar quyidagijadvalda keltirilgan.

Tur alomati	**Kutilayotgan qiymat**	**Argument turi**
Son turidagi argument		
d, D	O‘nlik butun	int * arg yoki long * arg
E,e	Suzuvchi nuqtali son	float * arg
F	Suzuvchi nuqtali son	float * arg
G,g	Suzuvchi nuqtali son	float * arg
O	Sakkizlik son	int * arg
O	Sakkizlik son	long* arg

l	O‘nlik, sakkizlik va o‘n oltilik butun son	int* arg
l	O‘nlik, sakkizlik va o‘n oltilik butun son	long* arg
U	Ishorasiz o‘nlik son	unsigned int * arg
U	Ishorasiz o‘nlik son	unsigned long * arg
x	O‘n oltilik son	int* arg
X	O‘n oltilik son	int* arg
Belgilar		
S	Satr	char * arg(belgilar massivi)
C	Belgi	char * arg(belgi uchun maydon kengligi berilishi mumkin (masalan, %4s). N belgidan tashkil topgan belgilar massiviga ko‘rsatkich: char arg [N])
%	‘%’ belgisi	Hech qanday almashtiri-shlar bajarilmaydi, ‘%’ belgisi saqlanadi.
Ko‘rsatkichlar		
N	int * arg	%n argumentigacha muvaffaqiyatli o‘qilgan belgilar soni, aynan shu int ko‘rsatkichi bo‘yicha adresda saqlanadi.
P	YYYY:ZZZZyoki ZZZZko‘rinishidagi o‘n oltilik	Obektga ko‘rsatkich (far* yoki near*).

Format aniqlashtiruvchilari va ularning vazifasi

Komponenta	Bo‘lishi shart yoki yo‘q	Vazifasi
[bayroq]	Yo‘q	Bayroq belgilari. Chiqarilayotgan qiymatni chapga yoki o‘nga tekislashni, sonning

		ishorasini, o‘nlik kasr nuqtasini, oxirdagi nollarni, sakkizlik va o‘n oltilik sonlarning alomatlarni chop etishni boshqaradi. Masalan, ‘-‘ bayrog‘i qiymatni ajratilgan o‘ringa nisbatan chapdan boshlab chiqarishni va kerak bo‘lsa o‘ngdan probel bilan to‘ldirishni bildiradi, aks holda chap tomondan probellar bilan to‘ldiradi va davomiga qiymat chiqariladi.
[<kenglik>]	Yo‘q	Maydon kengligini aniqlashtiruvchisi. Chiqariladigan belgilarning minimal sonini aniqlaydi. Zarur bo‘lsa qiymat yozilishidan ortgan joylar probel bilan to‘ldiriladi.
[.<xona>]	Yo‘q	Aniqlik. Chiqariladigan belgilarning maksimal sonini ko‘rsatadi. Sondagiraqamlarning minimal sonini.
[F|N|h|l|L]	Yo‘q	O‘lcham modifikatori. Argumentning qisqa (short - h) yoki uzun (long – l,L) ko‘rinishini, adres turini aniqlaydi.
<tur belgisi>	Ha	Argument qiymati almashtiriladigan tur alomati belgisi

Formatli yozish funksiyasi printf() quyidagi prototipga ega:

int printf(const char * <format>[,<argument>,...])

Bu funksiya standart oqimga formatlashgan chiqarishni amalga oshiradi. Funksiya argumentlar ketma-ketligidagi har bir argument qiymatini qabul qiladi va unga <format> satridagi mos format aniqlashtiruvchisini qo‘llaydi va oqimga chiqaradi.

printf() funksiyasining almashtiriladigan tur belgilari

Tur alomati	**Kutilayotgan qiymat**	**Chiqish formati**
Son qiymatlari		
D	Butun son	Ishorali o‘nlik butun son

I	Butun son	Ishorali o‘nlik butun son
O	Butun son	Ishorasiz sakkizlik butun son
U	Butun son	Ishorasiz o‘nlik butun son
x	Butun son	Ishorasiz o‘n oltilik butun son (a,b,c,d,e,f belgilari ishlatiladi)
X	Butun son	Ishorasiz o‘n oltilik butun son (A,B,C,D,E,F belgilari ishlatiladi)
F	Suzuvchi nuqtali son	[-]dddd.dddd ko‘rinishidagi suzuvchi nuqtali son
E	Suzuvchi nuqtali son	[-]d.ddddyoki e[+/-]ddd ko‘rinishidagi suzuvchi nuqtali son
G	Suzuvchi nuqtali son	Ko‘rsatilgan aniqlikka mos e yoki f shaklidagi suzuvchi nuqtali son
E, G	Suzuvchi nuqtali son	Ko‘rsatilgan aniqlikka mos e yoki f shaklidagi suzuvchi nuqtali son. E format uchun ‘E’ chop etiladi.
Belgilar		
S	Satrga ko‘rsatkich	0-belgisi uchramaguncha yoki ko‘rsatilgan aniqlikka erishilmaguncha belgilar oqimga chiqariladi.
C	Belgi	Bitta belgi chiqariladi
%	Hech nima	‘%’ belgisi oqimga chiqariladi.
Ko‘rsatkichlar		
N	intko‘rsatkich (int* arg)	%n argumentigacha muvaffaqiyatli chiqarilgan belgilar soni, aynan shu int ko‘rsatkichi bo‘yicha adresda saqlanadi.
P	Ko‘rsatkich	Argumentni YYYY:ZZZZ yoki ZZZZ ko‘rinishidagi o‘n oltilik songa aylantirib oqimga chiqaradi.

Har bir format aniqlashtiruvchisiga bitta o‘zgaruvchi adresi mos kelishi kerak. Agar format aniqlashtiruvchilari soni o‘zgaruvchilardan ko‘p bo‘lsa, natijada nima bo‘lishini oldindan aytib bo‘lmaydi. Aks holda, ya’ni o‘zgaruvchilar soni ko‘p bo‘lsa, ortiqcha o‘zgaruvchilar inobatga olinmaydi. Agar oqimga chiqarish muvaffaqiyatli bo‘lsa, funksiya chiqarilgan baytlar sonini qaytaradi, aks holda EOF.

printf() funksiyasining <format> satri argumentlarni almashtirish, formatlash va berilganlarni oqimga chiqarish jarayonini boshqaradi va u ikki turdagi obektlardan tashkil topadi:

– oqimga o‘zgarishsiz chiqariladigan oddiy belgilar;

– argumentlar ro‘yxatidagi tanlanadigan argumentga qo‘llani-ladigan format aniqlashtiruvchilari.

Format aniqlashtiruvchisi quyidagi ko‘rinishga ega:

% [<bayroq>][<.kenglik>] [.<xona>][F|N|h||L] <tur belgisi>

Format aniqlashtiruvchisi ‘%’ belgisidan boshlanadi va undan keyin 3–jadvalda keltirilgan shart yoki shart bo‘lmagan komponentalar keladi.

Almashtiriladigan tur belgisining qabul qilishi mumkin bo‘lgan belgilar 4-jadvalda keltirilgan.

Berilganlar qiymatlarini oqimdan o‘qish va oqimga chiqarishda scanf() va printf() funksiyalaridan foydalanishga misol:

```
#include <stdio.h>
int main()
{
 int bson, natija;
 float hson;
 char blg, satr[81];
 printf("\nButun va suzuvchi nuqtali sonlarni,");
 printf("\nbelgi hamda satrni kiriting\n");
 natija=scanf("%d %f %c %s", &bson, &hson,&blg,satr);
```

```
 printf("\nOqimdan %d ta qiymat o'qildi ",natija);
 printf("va ular quyidagilar:");
 printf("\n %d %f %c %s \n",bson, hson, blg, satr);
 return 0;
}
```

Dastur foydalanuvchidan butun va suzuvchi nuqtali sonlarni, belgi va satrni kiritishni so‘raydi. Bunga javoban foydalanuvchi tomonidan

```
10 12.35 A Satr
```

qiymatlari kiritilsa, ekranga

```
Oqimdan 4 ta qiymat o'qildi va ular quyidagilar:
10 12.35 A Satr
```

satrlari chop etiladi.

Nazorat savollari

1. Formatli o‘qish uchun qanday funksiya ishlatiladi va uning sintaksisi qanday?
2. Formatli yozish uchun qanday funksiya ishlatiladi va uning sintaksisi qanday?
3. Qanday o‘qish oqimlarini bilasiz?
4. Qanday yozish oqimlarini bilasiz?
5. Belgilarni o‘qish uchun qaysi funksiyalar ishlatiladi?
6. Belgilarni yozish uchun qaysi funksiyalar ishlatiladi?
7. Satrlarni o‘qish uchun qaysi funksiyalar ishlatiladi?
8. Satrlarni yozish uchun qaysi funksiyalar ishlatiladi?

§22. Fayllar.Matn va binar fayllar

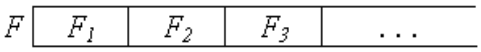
Fayl tushunchasi

C++ tilidagi standart va foydalanuvchi tomonidan aniqlangan turlarning muhim xususiyati shundan iboratki, ularning oldindan aniqlangan miqdordagi chekli elementlardan iboratligidir. Hatto berilganlar dinamik aniqlanganda ham, operativ xotiraning (uyumning) amalda cheklanganligi sababli, bu berilganlar

miqdori yuqoridan chegaralangan elementlardan iborat bo‘ladi. Ayrim bir tadbiqiy masalalar uchun oldindan berilganning komponentalari sonini aniqlash imkoni yo‘q. Ular masalani yechish jarayonida aniqlanadi va etarlicha katta hajmda bo‘lishi mumkin. Ikkinchi tomondan, dasturda e’lon qilingan o‘zgaruvchilarning qiymatlari sifatida aniqlangan berilganlar faqat dastur ishlash paytidagina mavjud bo‘ladi va dastur o‘z ishini tugatgandan keyin yo‘qolib ketadi. Agar dastur yangidan ishga tushirilsa, bu berilganlarni yangidan hosil qilish zarur bo‘ladi. Aksariyat tadbiqiy masalalar esa berilganlarni doimiy ravishda saqlab turishni talab qiladi. Masalan, korxona xodimlarinning oylik maoshini hisoblovchi dasturda xodimlar ro‘yxatini, shtat stavkalari va xodimlar tomonidan olingan maoshlar haqidagi ma’lumotlarni doimiy ravishda saqlab turish zarur. Bu talablarga fayl turidagi obektlar (o‘zgaruvchilar) javob beradi.

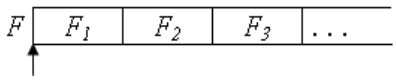
Fayl– bu bir xil turdagi qiymatlar joylashgan tashqi xotiradagi nomlangan sohadir.

Faylni, boshida ketma-ket ravishda joylashgan yozuvlar (masalan, musiqa) bilan to‘ldirilgan va oxiri bo‘sh bo‘lgan yetarlicha uzun magnit tasmasiga o‘xshatish mumkin.



Yuqorida rasmda F – fayl nomi, F_1, F_2, F_3 fayl elementlari (komponentalari). Xuddi yangi musiqani tasma oxiriga qo‘shish mumkin bo‘lgandek, yangi yozuvlar fayl oxiriga ko‘shilishi mumkin.

Yana bir muhim tushunchalardan biri fayl ko‘rsatkichi tushunchasidir. *Fayl ko‘rsatkichi* – ayni paytda fayldan o‘qilayotgan yoki unga yozilayotgan joyni (yozuv o‘rnini) ko‘rsatib turadi, ya’ni fayl ko‘rsatkichi ko‘rsatib turgan joydan bitta yozuvni o‘qish yoki shu joyga yangi yozuvni joylashtirish mumkin. Quyidagi rasmda fayl ko‘rsatkichi fayl boshini ko‘rsatmoqda.



Fayl yozuvlariga murojaat ketma-ket ravishda amalga oshiriladi: n-yozuvga murojaat kilish uchun n-1 yozuvni o‘qish zarur bo‘ladi. Shuni ta’kidlab o‘tish

zarurki, fayldan yozuvlarni o‘qish jarayoni qisman “*avtomatlashgan*”, unda i - yozuvni o‘qilgandan keyin, ko‘rsatkich navbatdagi i+1 yozuv boshiga ko‘rsatib turadi va shu tarzda o‘qishni davom ettirish mumkin (massivlardagidek indeksni oshirish shart emas).

Fayl bu berilganlarni saqlash joyidir va shu sababli uning yozuvlari ustida to‘g‘ridan-to‘g‘ri amal bajarib bo‘lmaydi. Fayl yozuvi ustida amal bajarish uchun yozuv qiymati operativ xotiraga mos turdagi o‘zgaruvchiga o‘qilishi kerak. Keyinchalik, zarur amallar shu o‘zgaruvchi ustida bajariladi va kerak bo‘lsa natijalar yana faylga yozilishi mumkin.

Operatsion tizim nuqtai-nazaridan fayl hisoblangan har qanday fayl C++ tili uchun *moddiy fayl* hisoblanadi. MS DOS uchun moddiy fayllar <fayl nomi>.<fayl kengaytmasi> ko‘rinishidagi «8.3» formatidagi satr (nom) orqali beriladi. Fayl nomlari satr o‘zgarmaslar yoki satr o‘zgaruvchilarida berilishi mumkin. MS DOS qoidalariga ko‘ra fayl nomi to‘liq bo‘lishi, ya’ni fayl nomining boshida adres qismi bo‘lishi mumkin: ” C:\\USER\\Misol.cpp”, “D:\\matn.txt”.

C++ tilida *mantiqiy fayl* tushunchasi bo‘lib, u fayl turidagi o‘zgaruvchini anglatadi. Fayl turidagi o‘zgaruvchilarga boshqa turdagi o‘zgaruvchilar kabi qiymat berish operatori orqali qiymat berib bo‘lmaydi. Boshqacha aytganda fayl turidagi o‘zgaruvchilar ustida hech qanday amal aniqlanmagan. Ular ustida bajariladigan barcha amallar funksiyalar vositasida bajariladi.

Fayllar bilan ishlash quyidagi bosqichlarni o‘z ichiga oladi:

– fayl o‘zgaruvchisi albatta diskdagi fayl bilan bog‘lanadi;

– fayl ochiladi;

– fayl ustida yozish yoki o‘qish amallari bajariladi;

– fayl yopiladi;

– fayl nomini o‘zgartirish yoki faylni diskdan o‘chirish amallarini bajarilishi mumkin.

Matn va binar fayllar

C++ tili C tilidan o‘qish-yozish amalini bajaruvchi standart funksiyalar kutubxonasini vorislik bo‘yicha olgan. Bu funksiyalar <stdio.h> sarlavha faylida e’lon qilingan. O‘qish-yozish amallari fayllar bilan bajariladi. Fayl matn yoki binar (ikkilik) bo‘lishi mumkin.

Matn fayl – ASCII kodidagi belgilar bilan berilganlar majmuasi. Belgilar ketma-ketligi satrlarga bo‘lingan bo‘ladi va satrning tugash alomati sifatida CR(karetkani qaytarish yoki ‘\r’) LF (satrni o‘tkazish yoki ‘\n’) belgilar juftligi hisoblanadi. Matn fayldan berilganlarni o‘qishda bu belgilar juftligi bitta CR belgisi bilan almashtiriladi va aksincha, yozishda CR belgisi ikkita CR va LF belgilariga almashtiriladi. Fayl oxiri #26 (^Z) belgisi bilan belgilanadi.

Matn faylga boshqacha ta’rif berish ham mumkin. Agar faylni matn tahririda ekranga chiqarish va o‘qish mumkin bo‘lsa, bu matn fayl. Klaviatura ham kompyuterga faqat matnlarni jo‘natadi. Boshqacha aytganda dastur tomonidan ekranga chiqariladigan barcha ma’lumotlarni stdout nomidagi matn fayliga chiqarilmoqda deb qarash mumkin. Xuddi shunday klaviaturadan o‘qilayotgan har qanday berilganlarni matn faylidan o‘qilmoqda deb hisoblanadi.

Matn fayllarining komponentalari *satrlar* deb nomlanadi. Satrlar uzluksiz joylashib, turli uzunlikda va bo‘sh bo‘lishi mumkin. Faraz qilaylik, T matn fayli 4 satrdan iborat bo‘lsin:

1-satr#13#10	2-satr uzunroq #13#10	#13#10	4-satr#13#10#26

Matnni ekranga chiqarishda satr oxiridagi #13#10 boshqaruv belgilari juftligi kursorni keyingi qatorga tushiradi va uni satr boshiga olib keladi. Bu matn fayl ekranga chop etilsa, uning ko‘rinishi quyidagicha bo‘ladi:

```
1- satr[13][10]
2- satr uzunroq[13][10]
[13][10]
4- satr[13][10]
[26]
```

Matndagi [n] – n-kodli boshqaruv belgisini bildiradi. Odatda matn tahrirlari bu belgilarni ko‘rsatmaydi.

Binar fayllar – bu oddiygina baytlar ketma-ketligi. Odatda binar fayllardan berilganlarni foydalanuvchi tomonidan bevosita “ko‘rish” zarur bo‘lmagan hollarda ishlatiladi. Binar fayllardan o‘qish-yozishda baytlar ustida hech qanday konvertatsiya amallari bajarilmaydi.

Fayllar oqimlari bilan ishlash

C++da fayllar oqimlari bilan ishlash uchun fstream kutubxonasi mavjud.

fstream kutubxonasi fayllarni o‘qib olish uchun javob beradigan ifstream sinfiga, hamda faylga ma’lumot yozishga imkon beradigan ofstream turlariga (sinfiga) ega.

```
#include <fstream>
```

Biron-bir faylni yozish yoki o‘qish uchun ochish uchun, ofstream turdagi yoki mos holda ifstream turdagi o‘zgaruvchini yaratish kerak.

```
ifstream inData;
ofstream outData;
```

Bunday o‘zgaruvchini initsiallashda fayl nomi o‘zgaruvchi nomidan keyin qavs ichida berilgan belgilar massivi ko‘rinishida uzatiladi. Masalan, C diskida joylashgan “text.txt” faylini ochish kerak. Buning uchun kodning quyidagi fragmenti qo‘llanadi:

```
ifstream inData (“C:\\text.txt”);
ofstream outData (“C:\\text.txt”);
```

yoki o‘zgaruvchi orqali quyidagicha initsiallash mumkin.

```
char s[20] = “C:\\text.txt”;
ifstream inData(s);
```

Shuningdek, faylni ochish uchun open() funksiyasini ham ishlatish mumkin.

```
ofstream outData;
outData.open(“cppfayl.txt”);
```

Agar fayl xam dasturning bajarilayotgan fayli joylashtirilgan papkada bo‘lsa, u holda faylning nomi to‘liq ko‘rsatilmasligi mumkin (faqat fayl nomi, unga borish yo‘lisiz). Bundan tashqari fayl nomini to‘g‘ridan-to‘g‘ri ko‘rsatish o‘rniga, uning nomidan iborat belgilar massivlarini ko‘rsatish mumkin.

Agar fayllar umumiy fayl oqimi turida yaratilsa, faylni ochish rejimi ko‘rsatilishi kerak. Qo‘shish tuzimida faylni ochish uchun ochilishda quyida ko‘rsatilgan ikkinchi parametrni ko‘rsatish lozim:

```
fstream output_file("cppfayl.txt", ios::app);
```

Bu holda ios::app parametri faylni ochish rejimini aniqlaydi.

Fayllarni ochish rejimlari

Ochish rejimi	Vazifasi
ios::app	Fayl ko‘rsatkichni faylni oxirida joylashtirib ko‘shish rejimida faylni ochadi.
ios::ate	Fayl ko‘rsatkichini faylni oxiriga joylashtiradi. O‘qish mumkin emas, chiqaruvchi ma’lumotlar faylni oxiriga yoziladi.
ios::in	Qiritish uchun faylni ochilishini ko‘rsatadi.
ios::nocreate	Agarda ko‘rsatilgan fayl mavjud bo‘lmasa, fayl yaratilmaydi va xato qaytariladi.
ios::noreplace	Agarda fayl mavjud bo‘lsa, ochish operatsiyasi to‘xtash va xatolikni qaytarish lozim.
ios::out	Chiqarish uchun faylni ochilishini ko‘rsatadi
ios::trunc	Mavjud bo‘lgan faylni ichidagini olib tashlaydi (ko‘chiradi).
ios::binary	Faylni binar fayl ko‘rinishida ochadi

Bir nechta rejimni birgalikda ishlatish imkoni ham mavjud. Bunda razryadli yoki amalidan foydalaniladi. Quyida faylni ochish hamda uni tozalab tashlashni oldini olish uchun ios::noreplace rejimlaridan foydalanilgan:

```
ifstream output_file("cppfayl.txt", ios::out | ios::noreplace);
```

Dasturni tugallash uchun operatsiya tizimi o‘zi ochgan fayllarni berkitadi. Biroq, odatga ko‘ra, agar dasturga fayl kerak bo‘lmay qolsa, uni berkitishi

kerak. Faylni berkitish uchun dastur, quyida ko‘rsatilganidek, close funksiyasidan foydalanishi kerak:

```
output_file.close();
```

Fayllar oqimlarida o‘qish-yozish funksiyalari

Axborotni faylga yozish uchun put() funksiyasidan foydalanish mumkin. Bu funksiya orqali standart turdagi yakka o‘zgaruvchi yoki biron-bir belgilar massivi uzatiladi. Belgilar massivi uzatilgan holda massivdagi yozilishi kerak bo‘lgan belgilar sonini uzatish kerak.

Bundan tashqari “<<” operatoridan foydalanish mumkin. Bu operatordan kodning bitta satrida turli turdagi qiymatlarni uzatgan holda ko‘p martalab foydalanish mumkin. Satr xaqida gap ketganda, chiqarish satr oxiri belgisi, ya’ni ‘\n’ paydo bo‘lishidan oldin amalga oshiriladi. Belgisiz turga ega bo‘lgan barcha o‘zgaruvchilar oldin belgilarga o‘zgartirib olinadi.

```
ofstream outData(“C:\\text.txt”);
char a=‘M’;
outData.put(s);
char s[21]=“Bir necha so’zli matn”;
outData.put(s,21);
outData<< “Oqim belgisi orqali yozish”;
int i=100;
outData<< “Bir nechta son ” << i << “ ” <<200;
```

Axborotni fayldan o‘qib olish uchun “>>” operatori va get() funksiyasidan foydalanish mumkin. put() funksiyasi kabi, get() funksiyasi ham har qanday o‘zgaruvchilarning standart turlari, belgilar massivlari bilan ishlay oladi. Shuningdek get ga har jihatdan ekvivalent bo‘lgan getline() funksiyasini ham ishlatish mumkin.

```
ifstream inData(“C:\\text.txt”);
char s;
char ss[9];
```

```
s=inData.get();
cout<<s;
inData.get(s);
cout<<s;
inData.getline(ss,9);
cout<<ss;
inData >> ss;
cout << ss;
```

Fayl oxirini aniqlash uchun, oqim obektining eof() funksiyasidan foydalanish mumkin. Agar fayl oxiri xali uchramagan bo‘lsa, bu funksiya 0 qiymatini qaytaradi, agar fayl oxiri uchrasa -1 qimatini qaytaradi. whiletakrorlash opratoridan foydalanib fayl oxiri topilmaguncha fayl bilan biror ish bajarish quyidagi kodda keltirilgan:

```
while ( ! inData.eof() )
{
// bajariladigan ishlar
}
```

Ushbu holda dastur eof() funksiyasi yolg‘on (0) qiymat qaytarguncha takrorlash davom etadi.

Massivlar va tuzilmalarni o‘qish va yozish kerak bo‘lsa, read() va write() funksiyalaridan foydalanishlari mumkin. Bu funksiyalaridan foydalanishda o‘qiladigan yoki yozib olinadigan berilganlar buferini, shuningdek buferning baytlarda o‘lchanadigan uzunligini ko‘rsatish lozim. Bu quyida ko‘rsatilganidek amalga oshiriladi:

```
input_file.read(buffer, sizeof(buffer));
output_file. write(buffer, sizeof(buffer));
```

Masalan, tuzilma ichidagi ma’lumotlarni “EMPLOYeE.DAT” fayliga chiqarish uchun, write() funksiyasidan foydalanadi:

```
struct employee {
 char name[64];
 int age;
 float salary;
 } worker = { "Djon Doy", 33, 25000.0 };
ofstream emp_file("EMPLOYeE.DAT") ;
emp_file.write((char *) &worker, sizeof(employee));
```

Odatda write() funksiyasi belgilar satriga ko‘rsatkich oladi. Xuddi shunday tarzda read() funksiyasi orqali xizmatchi xaqidagi axborotni fayldan o‘qib olish uchun foydalanadi:

```
ifstream emp_file("EMPLOYeE.DAT");
emp_file.read((char *) &worker, sizeof(employee));
cout << worker.name << endl;
cout << worker.age << endl;
cout << worker.salary << endl;
```

Nazorat savollari

1. Fayl nima?
2. Fayl ko‘rsatkichi deb nimaga aytiladi?
3. Matn fayl va binar faylning farqi nimada?
4. C++ tilida fayl bilan ishlovchi qanday turlar mavjud?
5. FILE* turi orqali faylni qanday ochish mumkin?
6. Faylni ochish rejimlari nima uchun kerak?
7. Qanday faylni ochish rejimlari mavjud?
8. Fayl oqimi turida faylni ochish rejimlarining qanday ko‘rinishlari mavjud.

§23. Fayldan o‘qish-yozish funksiyalari. Fayl ko‘rsatkichini boshqarish funksiyalari

Faylni ochish va yopish

Fayl oqimi bilan o‘qish-yozish amalini bajarish uchun fayl oqimini ochish zarur. Bu ishni, prototipi

FILE * fopen(const char* filename, const char *mode);

ko‘rinishida aniqlangan fopen() funksiyasi orqali amalga oshiriladi. Funksiya filename nomi bilan faylni ochadi, u bilan oqimni bog‘laydi va oqimni identifikatsiya qiluvchi ko‘rsatkichni javob tariqasida qaytaradi. Faylni ochish muvaffaqiyatsiz bo‘lganligini fopen() funksiyasining NULL qiymatli javobi bildiradi.

Parametrlar ro‘yxatidagi ikkinchi – mode parametri faylni ochish rejimini aniqlaydi. U qabul qilishi mumkin bo‘lgan qiymatlar quyidagi jadvalda keltirilgan.

Fayl ochish rejimlari

Qiymati	Fayl ochilish holati tavsifi
r	Fayl faqat o‘qish uchun ochiladi
w	Fayl yozish uchun ochiladi. Agar bunday fayl mavjud bo‘lsa, u qaytadan yoziladi (yangilanadi).
a	Faylga yozuvni qo‘shish rejimi. Agar fayl mavjud bo‘lsa, fayl uning oxiriga yozuvni yozish uchun ochiladi, aks holda yangi fayl yaratiladi va yozish rejimida ochiladi.
r+	Mavjud fayl o‘zgartirish (o‘qish va yozish) uchun ochiladi.
w+	Yangi fayl yaratilib, o‘zgartirish (o‘qish va yozish) uchun ochiladi. Agar fayl mavjud bo‘lsa, undagi oldingi yozuvlar o‘chiriladi va u qayta yozishga tayyorlanadi.
a+	Faylgayozuvni qo‘shish rejimi. Agar fayl mavjud bo‘lsa, uning oxiriga (EOF alomatidan keyin) yozuvni yozish (o‘qish) uchun ochiladi, aks holda yangi fayl yaratiladi va yozish rejimida ochiladi.

Matn fayli ochilayotganligini bildirish uchun fayl ochilish rejimi satriga ‘t’ belgisini qo‘shib yozish zarur bo‘ladi. Masalan, matn fayl o‘zgartirish (o‘qish va yozish) uchun ochilayotganligini bildirish uchun “rt+” satri yozish kerak bo‘ladi. Xuddi shunday binar fayllar ustida ishlash uchun ‘b’ belgisini ishlatish kerak. Misol uchun fayl ochilishining “wb+” rejimi binar fayl yangilanishini bildiradi.

Fayl o‘zgartirish (o‘qish-yozish) uchun ochilganda, berilganlarni oqimdan o‘qish, hamda oqimga yozish mumkin. Biroq yozish amalidan keyin darhol o‘qib bo‘lmaydi, buning uchun o‘qish amalidan oldin fseek() yoki rewind() funksiyalari chaqirilishi shart.

Faraz qilaylik “C:\\USER\\TALABA\\iat1kurs.txt” nomli matn faylni o‘qish uchun ochish zarur bo‘lsin. Bu talab

```
FILE *f=fopen(“C:\\USER\\TALABA\\iat1kurs.txt”, “r+”);
```

ko’rsatmasini yozish orqali amalga oshiriladi. Natijada diskda mavjud bo‘lgan fayl dasturda f o‘zgaruvchisi nomi bilan aynan bir narsa deb tushuniladi. Boshqacha aytganda, dasturda keyinchalik f ustida bajarilgan barcha amallar, diskdagi “iat1kurs.txt” fayli ustida ro‘y beradi.

Fayl oqimi bilan ishlash tugagandan keyin u albatta yopilishi kerak. Buning uchun fclose() funksiyasidan foydalaniladi. Funksiya prototipi quyidagi ko‘rinishga ega:

```
int fclose(FILE * stream);
```

fclose() funksiyasi oqim bilan bog‘liq buferlarni tozalaydi (masalan, faylga yozish ko‘rsatmalari berilishi natijasida buferda yig‘ilgan berilganlarni diskdagi faylga ko‘chiradi) va faylni yopadi. Agar faylni yopish xatolikka olib kelsa, funksiya EOF qiymatini, normal holatda 0 qiymatini qaytaradi.

Fayldan o‘qish-yozish funksiyalari

fgetc() funksiyasi prototipi

```
int fgetc(FILE *stream);
```

ko‘rinishida aniqlangan bo‘lib, fayl oqimidan belgini o‘qishni amalga oshiradi. Agar o‘qish muvafffaqiyatli bo‘lsa, funksiya o‘qilgan belgini int turidagi ishorasiz butun songa aylantiradi. Agar fayl oxirini o‘qishga harakat qilinsa yoki xatolik ro‘y bersa, funksiya EOF qiymatini qaytaradi.

Ko‘rinib turibdiki, getc() va fgetc() funksiyalari deyarli bir xil ishni bajaradi, farqi shundaki, getc() funksiyasi belgini standart oqimdan o‘qiydi. Boshqacha aytganda, getc() funksiyasi, fayl oqimi standart qurilma bo‘lgan

fgetc() funksiyasi bilan aniqlangan makrosdir.

fputs() funksiyasi

```
int fputc(int c, FILE *stream);
```

prototipi bilan aniqlangan. fputs() funksiyasi fayl oqimiga argumentda ko‘rsatilgan belgini yozadi (chiqaradi) va u amal qilishida puts() funksiyasi bilan bir xil.

Fayl oqimidan satr o‘qish uchun

```
char * fgets(char * s, int n, FILE *stream)
```

prototipi bilan fgets() aniqlangan. fgets() funksiyasi fayl oqimidan belgilar ketma-ketligini s satriga o‘qiydi. Funksiya o‘qish jarayonini oqimdan n-1 belgi o‘qilgandan keyin yoki keyingi satrga o‘tish belgisi (‘\n’) uchraganda to‘xtatadi. Oxirgi holda ‘\n’ belgisi ham s satrga qo‘shiladi. Belgilarni o‘qish tugagandan keyin s satr oxiriga, satr tugash alomati ‘\0’ belgisi qo‘shiladi. Agar satrni o‘qish muvaffaqiyatli bo‘lsa, funksiya s argument ko‘rsatadigan satrni qaytaradi, aks holda NULL.

Fayl oqimiga satrni fputs() funksiyasi yordamida chiqarish mumkin. Bu funksiya prototipi

```
int fputs (const char *s, FILE *stream);
```

ko‘rinishida aniqlangan. Satr oxiridagi yangi satrga o‘tish belgisi va terminatorlar oqimga chiqarilmaydi. Oqimga chiqarish muvaffaqiyatli bo‘lsa, funksiya nomanfiy son qaytaradi, aks holda EOF.

feof() funksiyasi aslida makros bo‘lib, fayl ustida o‘qish-yozish amallari bajarilayotganda fayl oxiri belgisi uchragan yoki yo‘qligini bildiradi. Funksiya

```
int feof(FILE *stream);
```

prototipiga ega bo‘lib u fayl oxiri belgisi uchrasa, noldan farqli sonni qaytaradi, boshqa holatlarda 0 qiymatini qaytaradi.

Quyida keltirilgan misolda faylga yozish va o‘qishga amallari ko‘rsatilgan.

```
#include <iostream>
```

```
using namespace std;
#include <stdio.h>
int main()
{
 char s;
 FILE *in,*out;
 if((in=fopen("D:\\USER\\TALABA.TXT","rt"))==NULL){
  cout<<"Talaba.txt faylini ochilmadi!!\n";
  return 1;
 }
 if((out=fopen("D:\\USER\\TALABA.DBL","wt+"))==NULL) {
  cout<<"Talaba.dbl faylini ochilmadi!!\n";
  return 1;
 }
 while (!feof(in)){
  char c=fgetc(in);
  cout<<c;
  fputc(c,out);
 }
 fclose(in);
 fclose(out);
 return 0;
}
```

Dasturda “talaba.txt” fayli matn fayli sifatida o‘qish uchun ochilgan va u in o‘zgaruvchisi bilan bog‘langan. Xuddi shunday, “talaba.dbl” matn fayli yozish uchun ochilgan va out bilan bog‘langan. Agar fayllarni ochish muvaffaqiyatsiz bo‘lsa, mos xabar beriladi va dastur o‘z ishini tugatadi. Keyinchalik, toki in fayli oxiriga etmaguncha, undan belgilar o‘qiladi va

ekranga, hamda out fayliga chiqariladi. Dastur oxirida ikkita fayl ham yopiladi.

Masala. G‘alvirli tartiblash usuli.

Berilgan x vektorini pufakcha usulida kamaymaydigan qilib tartiblash quyidagicha amalga oshiriladi: massivning qo‘shni elementlari x_k va $x_{k+1}(k=1,..,n-1)$ solishtiriladi. Agar $x_k>x_{k+1}$ bo‘lsa, u holda bu elementlar o‘zaro o‘rin almashadi. Shu yo‘l bilan birinchi o‘tishda eng katta element vektorning oxiriga joylashadi. Keyingi qadamda vektor boshidan n-1 o‘rindagi elementgacha yuqorida qayd qilingan yo‘l bilan qolgan elementlarning eng kattasi n-1 o‘ringa joylashtiriladi va h.k.

G‘alvirli tartiblash usuli pufakchali tartiblash usuliga o‘xshash, lekin x_k va $x_{k+1}(k=1,2,3,..,n-1)$ elementlar o‘rin almashgandan keyin “*g‘alvirdan*” o‘tkazish amali qo‘llaniladi: chap tomondagi kichik element imkon qadar chap tomonga tartiblash saqlangan holda ko‘chiriladi. Bu usul oddiy pufakchali tartiblash usuliga nisbatan tez ishlaydi.

Dastur matni:

```
#include <stdio.h>
#include <alloc.h>
int * Pufakchali_Tartiblash(int*,int);
int main()
{
 char fnomi[80];
 printf("Fayl nomini kiriting:");
 scanf("%s", &fnomi);
 int Ulcham,i=0,* Massiv;
 FILE * f1, *f2;
 if((f1=fopen(fnomi,"rt"))==NULL) {
  printf("Xato:%s fayli ochilmadi!",fnomi);
  return 1;
 }
```

```
 fscanf(f1,"%d",&Ulcham);
 Massiv=(int *)malloc(Ulcham*sizeof(int));
 while(!feof(f1)) fscanf(f1,"%d",&Massiv[i++]);
 fclose(f1);
 Massiv=Pufakchali_Tartiblash(Massiv,Ulcham);
 f2=fopen("natija.txt","wt");
 fprintf(f2,"%d%c",Ulcham,' ');
 for(i=0; i<Ulcham; i++)fprintf(f2,"%d%c",Massiv[i],' ');
 fclose(f2);
 return 0;
}
int * Pufakchali_Tartiblash(int M[],int n)
{
 int almashdi=1, vaqtincha;
for(int i=0; i<n-1 && almashdi;i++){
 almashdi=0;
  for(int j=0; j<n-i-1;j++)
if (M[j]>M[j+1]){
   almashdi=1;
      vaqtincha=M[j];
      M[j]=M[j+1];
      M[j+1]=vaqtincha;
   int k=j;
   if(k)
while(k && M[k]>M[k-1]){
           vaqtincha=M[k-1];
           M[k-1]=M[k];
           M[k]=vaqtincha;
```

```
        k--;
       }
 }
}
return M;
}
```

Dasturda berilganlarni oqimdan o‘qish yoki oqimga chiqarishda fayldan formatli o‘qish – fscanf() va yozish – fprintf() funksiyalaridan foydalanilgan. Bu funksiyalarning mos ravishda scanf() va printf() funksiyalaridan farqi – ular berilganlarni birinchi argument sifatida beriladigan matn fayldan o‘qiydi va yozadi.

Nomi foydalanuvchi tomonidan kiritiladigan f1 fayldan butun sonlar massivining uzunligi va qiymatlari o‘qiladi va tartiblangan massiv f2 faylga yoziladi.

Vektorni tartiblash Pufakchali_Tartiblash() funksiyasi tomonidan amalga oshiriladi. Unga vektor va uning uzunligi kiruvchi parametr bo‘ladi va tartiblangan vektor funksiya natijasi sifatida qaytariladi.

Navbatdagi ikkita funksiya fayl oqimidan formatlashmagan o‘qish-yozishni amalga oshirishga mo‘ljallangan.

fread() funksiyasi quyidagi prototipga ega:

size_t fread(void * ptr, size_t size, size_t n,FILE *stream);

Bu funksiya oqimdan ptr ko‘rsatib turgan buferga, har biri size bayt bo‘lgan n ta berilganlar blokini o‘qiydi. O‘qish muvaffaqiyatli bo‘lsa, funksiya o‘qilgan bloklar sonini qaytaradi. Agar o‘qish jarayonida fayl oxiri uchrab qolsa yoki xatolik ro‘y bersa, funksiya to‘liq o‘qilgan bloklar sonini yoki 0 qaytaradi.

fwrite() funksiyasi prototipi

size_t fwrite(const void*ptr,size_t size,size_t n,FILE *stream);

ko‘rinishi aniqlangan. Bu funksiya ptr ko‘rsatib turgan buferdan, har biri size bayt bo‘lgan n ta berilganlar blokini oqimga chiqaradi. Yozish muvaffaqiyatli

bo‘lsa, funksiya yozilgan bloklar sonini qaytaradi. Agar yozish jarayonida xatolik ro‘y bersa, funksiya to‘liq yozilgan bloklar sonini yoki 0 qaytaradi.

Fayl ko‘rsatkichini boshqarish funksiyalari

Fayl ochilganda, u bilan “stdio.h” sarlavha faylida aniqlangan FILE strukturasi bog‘lanadi. Bu struktura har bir ochilgan fayl uchun joriy yozuv o‘rnini ko‘rsatuvchi hisoblagichni – fayl ko‘rsatkichini mos qo‘yadi. Odatda fayl ochilganda ko‘rsatkich qiymati 0 bo‘ladi. Fayl ustida bajarilgan har bir amaldan keyin ko‘rsatkich qiymati o‘qilgan yoki yozilgan baytlar soniga oshadi. Fayl ko‘rsatkichini boshqarish funksiyalari –fseek(), ftell() va rewind() funksiyalari fayl ko‘rsatkichini o‘zgartirish, qiymatini olish imkonini beradi.

ftell() funksiyasining prototipi

```
long int ftell(FILE *stream);
```

ko‘rinishida aniqlangan bo‘lib, argumentda ko‘rsatilgan fayl bilan bog‘langan fayl ko‘rsatkichi qiymatini qaytaradi. Agar xatolik ro‘y bersa funksiya -1L qiymatini qaytaradi.

```
int fseek(FILE *stream, long offset, int from);
```

prototipiga ega bo‘lgan fseek() funksiyasi stream fayli ko‘rsatkichini from joyiga nisbatan offset bayt masofaga surishni amalga oshiradi. Matn rejimidagi oqimlar uchun offset qiymati 0 yoki ftell() funksiyasi qaytargan qiymat bo‘lishi kerak. from parametri quyidagi qiymatlarni qabul qilishi mumkin:

SEEK_SET (=0) – fayl boshi;

SEEK_CUR (=1) – fayl ko‘rsatkichining ayni paytdagi qiymati;

SEEK_END (=2) – fayl oxiri.

Funksiya fayl ko‘rsatkichi qiymatini o‘zgartirish muvaffaqiyatli bo‘lsa, 0 qiymatini, aks holda noldan farqli qiymat qaytaradi.

rewind() funksiyasi

```
void rewind(FILE *stream);
```

prototipi bilan aniqlangan bo‘lib, fayl ko‘rsatkichini fayl boshlanishiga olib keladi.

Quyida keltirilgan dasturda binar fayl bilan ishlash ko‘rsatilgan.

```
#include <iostream>
using namespace std;
#include <stdio.h>
#include <string.h>
struct Shaxs
{
 char Familiya[20];
 char Ism[15];
 char Sharifi[20];};
int main()
{
 int n,k;
 cout<<"Talabalar sonini kiriting: "; cin>>n;
 FILE *oqim1,*oqim2;
 Shaxs *shaxs1, *shaxs2, shaxsk;
 shaxs1=new Shaxs[n];
 shaxs2=new Shaxs[n];
 if ((oqim1=fopen("Talaba.dat", "wb+"))==NULL){
  cout<<"Talaba.dat ochilmadi!!!";
  return 1;
 }
 for(int i=0; i<n; i++){
  cout<<i+1<<"- shaxs ma'lumotlarini kiriting:\n";
  cout<<"Familiysi: "; gets(shaxs1[i].Familiya);
  cout<<"Ismi: "; gets(shaxs1[i].Ism);
  cout<<"Sharifi: "; gets(shaxs1[i].Sharifi);
 }
```

```
if (n==fwrite(shaxs1,sizeof(Shaxs),n,oqim1))
 cout<<"Berilganlarni yozish amalga oshirildi!\n";
else {
 cout<<"Berilganlarni yozish amalga oshirilmadi!\n";
 return 3;
}
cout<<" Fayl uzunligi: "<<ftell(oqim1)<<'\n';
fclose(oqim1);
if((oqim2=fopen("Talaba.dat", "rb+"))==NULL){
 cout<<"Talaba.dat o'qishga ochilmadi!!!";
 return 2;
}
if (n==fread(shaxs2,sizeof(Shaxs),n,oqim2))
for(int i=0; i<n; i++){
 cout<<i+1<<"- shaxs ma'lumotlari:\n";
 cout<<"Familiysi: "<<shaxs2[i].Familiya<<'\n';
 cout<<"Ismi: "<<shaxs2[i].Ism<<'\n';
 cout<<"Sharifi: "<<shaxs2[i].Sharifi<<'\n';
 cout<<"**************************\n";
}
else {
 cout<<"Fayldan o'qish amalga oshirilmadi!\n" ;
 return 4;
}
do{
 cout<<"Yo'zuv nomerini kiriting (1.."<<n<<"):";
 cin>>k;
 }
```

```
while (k<0 && k>n);
 k--;
 cout<<"Oldingi Familiya: ";
 cout<<shaxs2[k].Familiya <<'\n';
 cout<<"Yangi Familiya: ";
 gets(shaxs2[k].Familiya);
 if (fseek(oqim2, k*sizeof(Shaxs),SEEK_SET)){
  cout<<"Faylda"<<k+1;
  cout<<"-yo'zuvga o'tishda xatolik ro'y berdi???\n";
  return 5;
 }
 fwrite(shaxs2+k,sizeof(Shaxs),1,oqim2);
 fseek(oqim2, k*sizeof(Shaxs),SEEK_SET);
 fread(&shaxsk,sizeof(Shaxs),1,oqim2);
 cout<<k+1<<"- shaxs ma'lumotlari:\n";
 cout<<"Familiysi: "<<shaxsk.Familiya<<'\n';
 cout<<"Ismi: "<<shaxsk.Ism<<'\n';
 cout<<"Sharifi: "<<shaxsk.Sharifi<<'\n';
 fclose(oqim2);
 delete shaxs1;
 delete shaxs2;
 return 0;
}
```

Yuqorida keltirilgan dasturda, oldin “Talaba.dat” fayli binar fayl sifatida yozish uchun ochiladi va u oqim1 o‘zgaruvchisi bilan bog‘lanadi. Shaxs haqidagi ma’lumotni saqlovchi n o‘lchamli dinamik shaxs1 strukturalar massivi oqim1 fayliga yoziladi, fayl uzunligi chop qilinib fayl yopiladi. Keyin, xuddi shu fayl oqim2 nomi bilan o‘qish uchun ochiladi va undagi berilganlar

shaxs2strukturalar massiviga o‘qiladi va ekranga chop qilinadi. Dasturda fayldagi yozuvni o‘zgartirish (qayta yozish) amalga oshirilgan. O‘zgartirish qilinishi kerak bo‘lgan yozuv tartib nomeri foydalanuvchi tomonidan kiritiladi (k o‘zgaruvchisi) va shaxs2 strukturalar massividagi mos o‘rindagi strukturaning Familiya maydoni klaviaturadan kiritilgan yangi satr bilan o‘zgartiriladi. oqim2 fayl ko‘rsatkichi fayl boshidan k*sizeof(Shaxs) baytga suriladi va shaxs2 massivning k– strukturasi (shaxs2+k) shu o‘rindan boshlab faylga yoziladi. Keyin oqim2 fayli ko‘rsatkichi o‘zgartirish kiritilgan yozuv boshiga qaytariladi va bu yozuv shaxsk strukturasiga o‘qiladi hamda ekranga chop etiladi.

Masala. Haqiqiy sonlar yozilgan f fayli berilgan. f fayldagi elementlarning o‘rta arifmetigidan kichik bo‘lgan elementlar miqdorini aniqlansin.

Masalani yechish uchun f faylini yaratish va qaytadan uni o‘qish uchun ochish zarur bo‘ladi. Yaratilgan faylning barcha elementlarining yig‘indisi s o‘zgaruvchisida hosil qilinadi va u fayl elementlari soniga bo‘linadi. Keyin f fayl ko‘rsatkichi fayl boshiga olib kelinadi va elementlar qayta o‘qiladi va s qiymatidan kichik elementlar soni –k sanab boriladi.

Faylni yaratish va undagi o‘rta arifmetikdan kichik sonlar miqdorini aniqlashni alohida funksiya ko‘rinishida aniqlash mumkin.

Dastur matni:

```
#include <iostream.h>
#include <stdio.h>
#include <string.h>
int Fayl_Yaratish()
{
 FILE * f;
 double x;
 // f fayli yangidan hosil qilish uchun ochiladi
```

```
 if ((f=fopen("Sonlar.dbl", "wb+"))==NULL)return 0;
 char *satr=new char[10];
 int n=1;
 do{
cout<<"Sonni kiriting(bo'sh satr tugatish): "; gets(satr);
  if(strlen(satr)) {
x=atof(satr);
 fwrite(&x,sizeof(double),n,f);
}
 } while(strlen(satr));        // satr bo‘sh bo‘lmasa,takrorlash
 fclose(f);
 return 1;
}
int OAdan_Kichiklar_Soni()
{
 FILE * f;
 double x;
 f=fopen("Sonlar.dbl", "rb+");
 double s=0;                   // s - f fayli 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 arifmetiki="<<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 elementlarsoni
```

```
fclose(f);
 return k;
}
int main()
{
 if(Fayl_Yaratish()){
  cout<<"Sonlar.dbl faylidagi\n";
  int OA_kichik=OAdan_Kichiklar_Soni();
  cout<<"O'rta arifmetikdan kichik sonlar miqdori=";
  cout<<OA_kichik;
 }
  else                          // f faylini yaratish muvaffaqiyatsiz bo‘ldi
cout<<"Faylini ochish imkoni bo'lmadi!!!";
  return 0;
}
```

Dasturda bosh funksiyadan tashqari ikkita funksiya aniqlangan:

int Fayl_Yaratish() – diskda “Sonlar.dbl” 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 songa aylantirilib, faylga yoziladi. Agar bo‘sh satr kiritilsa, sonlarni kiritish jarayoni to‘xtatiladi va fayl yopiladi;

int OAdan_Kichiklar_Soni() – diskdagi “Sonlar.dbl” nomli fayli o‘qish uchun ochiladi va fayl elementlarining s o‘rta arifmetigidan kichik elementlari soni k topiladi va funksiya natijasi sifatida qaytariladi.

Bosh funksiyada faylni yaratish muvaffaqiyatli kechganligi tekshiriladi va shunga mos xabar beriladi.

Nazorat savollari

1. Faylni qanday yopish mumkin?
2. Faylga ma’lumotlar qanday yoziladi?

3. Fayl boshiga qanday qaytish mumkin?
4. Fayl ko‘rsatkichini qanday boshqarish mumkin?
5. Fayl ko‘rsatkichi deb nimaga aytiladi?
6. Matn fayl va binar faylning farqi nimada?
7. C++ tilida fayl bilan ishlovchi qanday turlar mavjud?
8. Fayl oqimi turida faylni ochish rejimlarining qanday ko‘rinishlari mavjud.

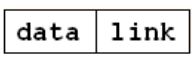
§24. Berilganlarning dinamik tuzilmalari

Berilganlarning dinamik tuzilmalari

Berilganlar ustida ishlashda ularning miqdori qancha bo‘lishi va ularga xotiradan qancha joy ajratish kerakligi oldindan noma’lum bo‘lishi mumkin. Dastur ishlash paytida berilganlar uchun zarurat bo‘yicha xotiradan joy ajratish va ularni ko‘rsatkichlar bilan bog‘lash orqali yagona struktura hosil qilish jarayoni xotiraning dinamik taqsimoti deyiladi. Bu usulda hosil bo‘lgan berilganlar majmuasiga *berilganlarning dinamik strukturasi* deyiladi, chunki ularning o‘lchami dastur bajarilishida o‘zgarib turadi. Dasturlashda dinamik strukturalardan chiziqli ro‘yxatlar, steklar, navbatlar va binar daraxtlar hosil qilishda nisbatan ko‘p ishlatiladi. Ular bir-biridan elementlar o‘rtasidagi bog‘lanishlari va ular ustida bajariladigan amallari bilan farqlanadi. Har qanday berilganlarning dinamik strukturasi maydonlardan tashkil topadi va ularning ayrimlari qo‘shni elementlar bilan bog‘lanish uchun xizmat qiladi.

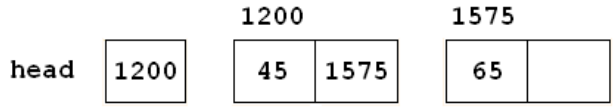
Chiziqli ro‘yxat

Chiziqli ro‘yxat tugunlardan (node) tashkil topadi. Zanjirning har bir tuguni ikkita maydondan iborat: qiymat maydoni (data) va keyingi element bilan bog‘lanishni ta’minlovchi ko‘rsatkich (link).



Birinchi tugun joylashgan xotiraning adresi (head) yoki birinchisi deyiladi.

Chiziqli ro‘yxatning har bir tuguni o‘zidan keyingi tugunning xotiradagi adresini ko‘rsatib turadi. So‘nggi tugundagi link o’zgaruvchi NULL qiymatga ega bo’ladi va undan keyin hech qanday tugun yo‘q.



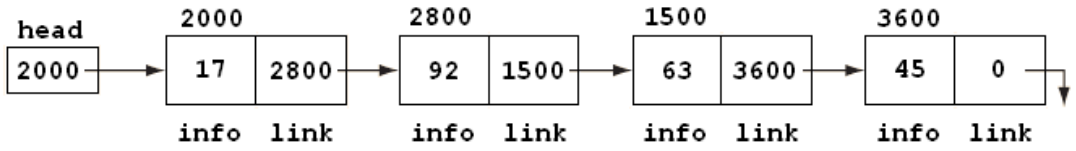
Yuqoridagi rasmda headko‘rsatib turgan adresda 1200 qiymati joylashgan. Ushbu adresda 45 qiymatiga ega tugun joylashgan. Bu tugun esa o‘z navbatida xotiraning 1575adresini ko‘rsatib turibdi.

Bunday chiziqli ro‘yxatni hosil qilishda strukturadan foydalaniladi:

```
struct nodeType
{
int info;
nodeType *link;
};
```

O‘zgaruvchi e’lon qilish quyidagicha amalga oshiriladi:

```
nodeType *head;
```



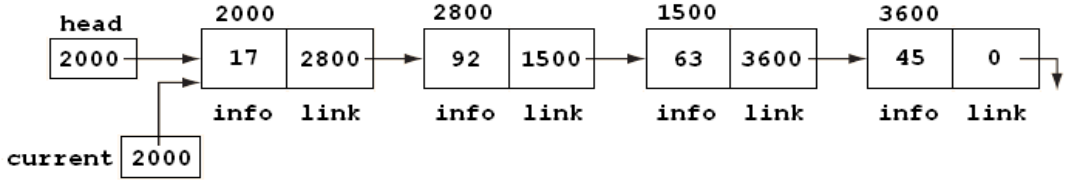
Ushbu ko‘rinishda chiziqli ro‘yxat qiymatlari quyidagicha bo‘ladi:

head	(2000)
head->info	(17)
head->link	(2800)
head->link->info	(92)

Agar, xuddi head e’lon qilgandek current ko‘rsatkichini e’lon qilib

```
current = head;
```

amalini bajarsak,current va head bitta adresni ko‘rsatib turadi:



Endi, current orqali tugunlar qiymatlarini olish mumkin:

current (2000)

current->info (17)

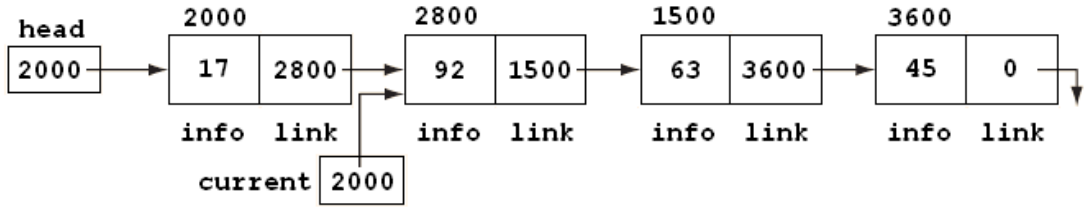
current->link (2800)

current->link->info (92)

Bunday yuklashning asosiy maqsadi, dastur ishlashi mobaynida chiziqli ro‘yxat boshini yo‘qotib qo‘ymaslik. Dastur ishlashi davomida headning qiymati o‘zgarmasligi kerak, aks holda zanjirning boshini topib bo‘lmaydi. Chiziqli ro‘yxat tugunlarida harakatlanishni current ko‘rsatkichi ta’minlaydi. Tugunlarda harakatlanish uchun quyidagicha qiymat yuklash amalga oshiriladi:

current = current->link;

Qiymat yuklanishi jarayonida current ko‘rsatkichining qiymati current->link qiymatini oladi, natijada current 2800-xotira adresini ko‘rsatib turadi. Ushbu adresda chiziqli ro‘yxatning ikkinchi tuguni joylashgan.



current (2800)

current->info (92)

current->link (1500)

current->link->info (63)

Chiziqli ro‘yxatning so‘nggi elementigacha borish uchun quyidagicha takrorlash ishlatish mumkin:

```
current = head;
while (current != NULL){
 // bu o’rinda qandaydir amallar ketma-ketligi
 current = current->link;
}
```

Masalan, chiziqli ro‘yxatning barcha elementlarini chop qilish kerak bo‘lsa, quyidagicha dastur matni ishlatilishi mumkin:

```
current = head;
while (current != NULL){
cout << current->info << " ";
current = current->link;
}
```

Odatda chiziqli ro‘yxat qurish uchun uchta ko‘rsatkich kerak bo‘ladi.

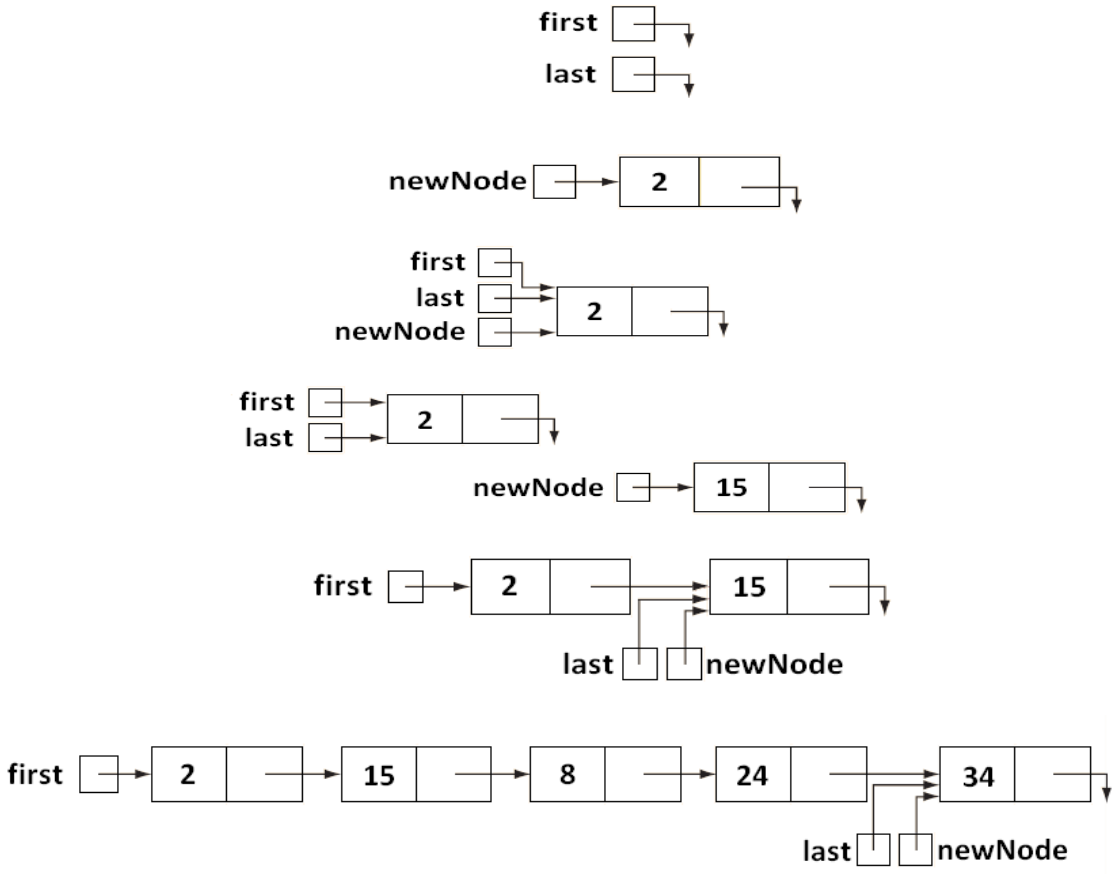
```
nodeType *first, *last, *newNode;
int num;
```

Bular chiziqli ro‘yxatning boshini, oxirini va yangi elementni ko‘rsatib turishi uchun kerak.

```
first = NULL;
last = NULL;
cin >> num;
newNode = new nodeType;
newNode->info = num;
newNode->link = NULL;
if (first == NULL){
 first = newNode;
 last = newNode;
 }
  else {
    last->link = newNode;
    last = newNode;
   }
```

Avval chiziqli ro‘yxat boshi va oxirigi NULL qiymat berildi. Keyin ro‘yxatga qo‘shish kerak bo‘lgan o‘zgaruvchi o‘qib olindi. Xotirada yangi tugun

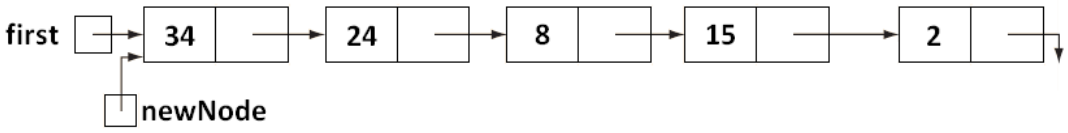
yaratildi va uning info maydoniga ekrandan kiritilgan son yuklandi. Yangi tugunning link ko‘rsatkichi NULL qiymatiniga ega. Sababi, ro‘yxat oxiridan to‘ldirilmoqda, ya’ni yangi element ro‘yxatning oxiriga joylashadi. Har qadamda “first==NULL” tekshirish amalga oshiriladi, agar tekshirish rost bo’lsa, ro‘yxat hali yaratilmagan bo‘ladi va yangi tugun ro‘yxatning boshi, ham oxiri hisoblanadi. Aks holda, ro‘yxatning oxirgi tuguniga yangi tugun ulanadi. Har doim ro‘yxatning oxirgi tuguniga last ko‘rsatib turadi.




```
nodeType* buildListForward()
{
 nodeType *first, *newNode, *last;
 int num;
 cout << "Sonni kiriting (tugash soni - 0): ";cin >> num;
 first = NULL;
 while (num != 0){
 newNode = new nodeType;
```

```
 newNode->info = num;
 newNode->link = NULL;
 if (first == NULL){
  first = newNode;
  last = newNode;
 }
  else{
   last->link = newNode;
   last = newNode;
  }
 cin >> num;
 }
 return first;
}
```

Chiziqli ro‘yxatni boshidan qurib kelish ham mumkin. Bunda oxirgi tugunni esda saqlab turuvchi o‘zgaruvchidan foydalanish zarurati yo‘qoladi.



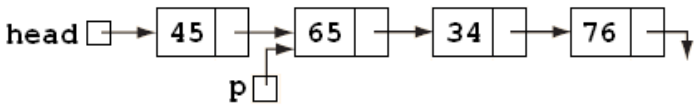

```
nodeType* buildListBackward()
{
 nodeType *first, *newNode;
 int num;
 cout << "Sonni kiriting (tugash soni - 0): ";cin >> num;
 first = NULL;
 while (num != 0){
   newNode = new nodeType;
   newNode->info = num;
```

```
  newNode->link = first;
  first = newNode;
  cin >> num;
 }
 return first;
}
```

Chiziqli ro‘yxatga element qo‘shishda qo‘shimcha ko‘rsatkich elementlaridan foydalanish mumkin.

```
struct nodeType {
 int info;
 nodeType *link;
};
nodeType *head, *p, *q, *newNode;
```

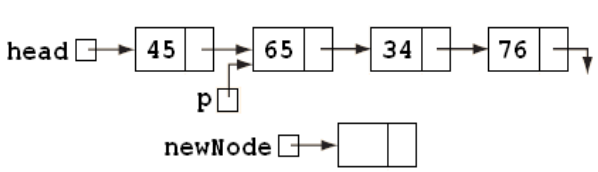
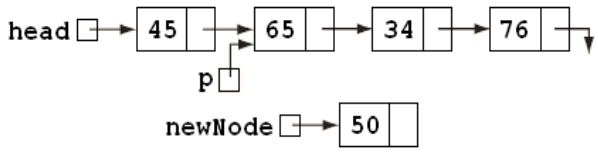
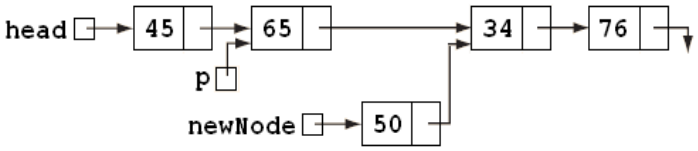
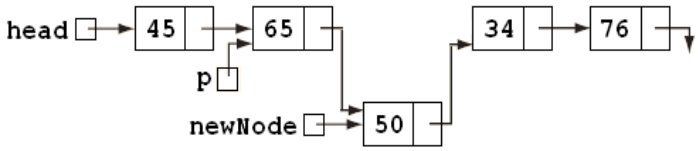
Element qo‘shishdan oldin chiziqli ro‘yxat quyidagicha ko‘rinishda bo‘lsin:



Yangi elementni p ko‘rsatkichdan keyin qo‘shish kerak bo‘lsa, quyidagicha amallar ketma-ketligi bajariladi:

```
newNode = new nodeType;       //yangi tugun
newNode->info = 50;           //yangi tugun info maydoni qiymati
newNode->link = p->link;
p->link = newNode;
```

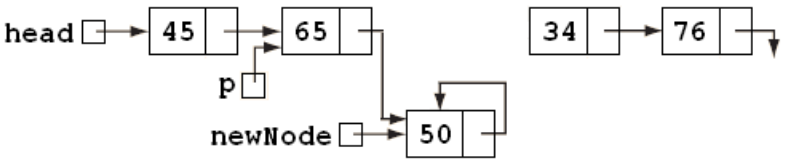
Har bir qator bajarilishida xotirada elementlar quyidagicha joylashadi:

`newNode = new nodeType;`	
`newNode->info = 50;`	
`newNode->link = p->link;`	
`p->link = newNode;`	

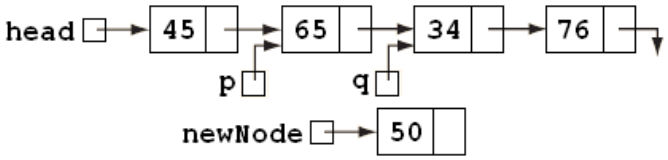
Dastur qismining aynan shu ko‘rinishda yozilishi muhim ahamiyatga ega. Agar qatorlar o‘rni almashib qolsa, chiziqli ro‘yxatning elementlari yo‘qolib qolishi yuzaga keladi. Masalan:

p->link = newNode;

newNode->link = p->link;



Ushbu holatda, newNode tuguni keyingi tugun sifatida o‘zini ko‘rsatib turibdi. Natijada oxirga ikkita tugunga bog‘lanish yo‘qolib qoldi. Dastur kodining bunday ketma-ketlikda yozilishida so‘nggi ikkita tugunni yo‘qotib qo‘ymaslik uchun qo‘shimcha ko‘rsatkichdan foydalanish kerak bo’ladi.



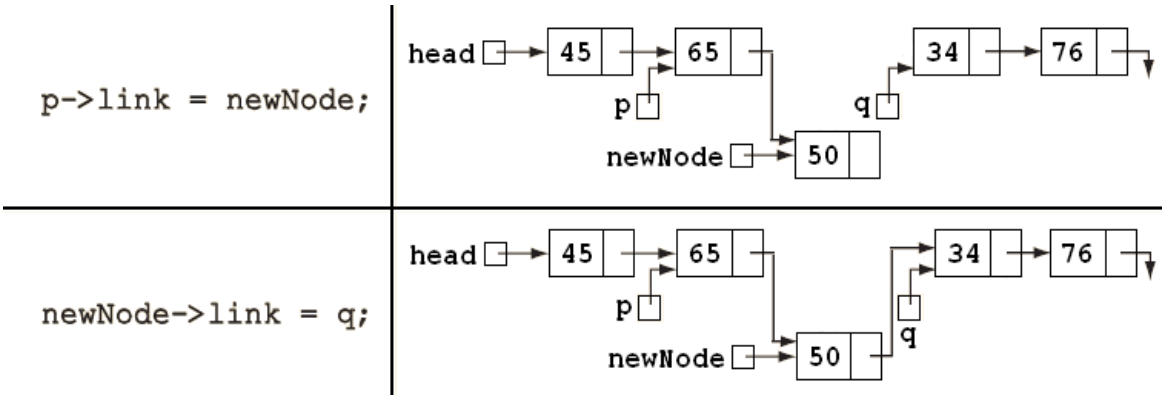
Bunda q ko‘rsatkichi uchinchi tugunni ko‘rsatib turibdi. Ushbu holatda yangi tugunni qo‘shishda qanday ketma-ketlikda yozish muhim emas, chunki q ko‘rsatkichi orqali oxirgi ikkita tugunni ixtiyoriy payt qo‘shib olish mumkin:

```
newNode->link = q;
p->link = newNode;
```

Shuningdek, quyidagi kodni ham yozish mumkin:

```
p->link = newNode;
newNode->link = q;
```

Ko’rsatmalar bajarilishida xotirada ro’yxat elementlar quyidagicha joylashadi:



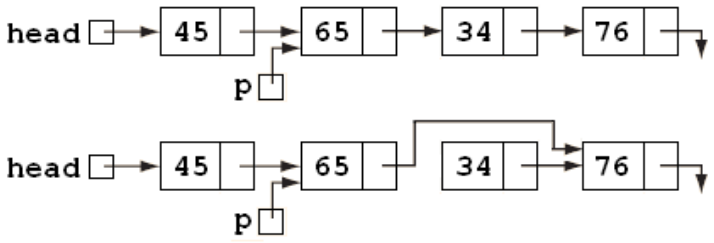
Biror tugunni o‘chirish kerak bo‘lsa ikki xil usuldan foydalanish mumkin:

1) ko‘rsatkichning o‘zi orqali

2) qo‘shimcha ko‘rsatkich orqali.

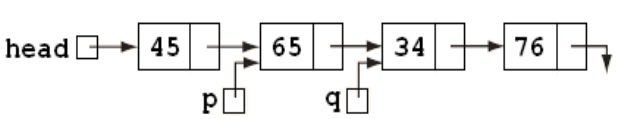
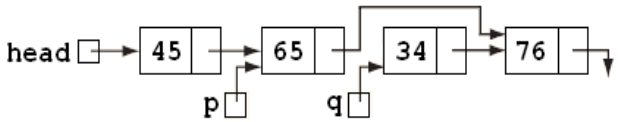
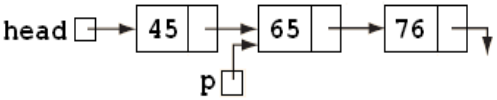
Birinchi usulda ko‘rsatkichga ko‘rsatkich ko‘rchatkichini yuklash orqali amalga oshiriladi.

```
p->link = p->link->link;
```



Ikkinchi usulda qo‘shimcha ko‘rsatkichni qaysi tugungacha o‘chirish kerakligini ko‘rsatgan holda, bir biriga yuklash orqali o‘chiriladi.

```
q = p->link;
p->link = q->link;
delete q;
```

`q = p->link;`	
`p->link = q->link;`	
`delete q;`	

Masala. Noldan farqli butun sonlardan iborat chiziqli ro‘yxat yaratilsin va undan ko‘rsatilgan songa teng element o‘chirilsin.

Butun sonlarning chiziqli ro‘yxat ko‘rinishidagi dinamik strukturasi quyidagi maydonlardan tashkil topadi:

```
struct Zanjir {
 int element;
 Zanjir * keyingi;
};
```

Dastur matni:

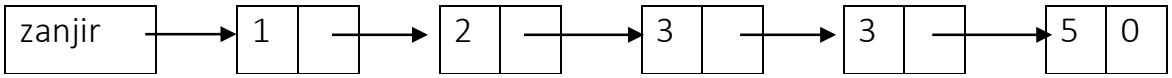
```
#include <iostream>
using namespace std;
struct Zanjir {
 int element;
 Zanjir * keyingi;
 };
Zanjir * Element_Joylash(Zanjir * z, int yangi_elem);
{
 Zanjir * yangi=new Zanjir;
 yangi->element=yangi_elem;
 yangi->keyingi=0;
 if(z) {                    // ro’yxat bo’sh emas
```

```
 Zanjir * temp=z;
 while(temp->keyingi)temp=temp->keyingi;  // ro'yxat oxirgi elementini topish
 temp=temp->yangi;       // elementni ro'yxat oxiriga qo'shish
}
 else z=yangi;          //ro'yxat bo'sh
return z;                          // ro'yxat boshi adresini qaytarish
}
Zanjir * Element_Uchirish(Zanjir * z,int del_elem)
{
 if(z){
  Zanjir * temp=z;
  Zanjir * oldingi=0;     // joriy elementdan oldingielementga ko'rsatkich
  while(temp){
    if(temp->element==del_elem) {
      if(oldingi) {       //o'chiriladigan element birinchi emas va  o'chiriladigan
                          // elementdan oldingi elementnikeyingi elementga ulash
       oldingi->keyingi = temp->keyingi;
       delete temp;     //elementni o'chirish
       temp=oldingi->keyingi;
     }
    else {                // o'chiriladigan element ro'yxat boshida
      z=z->keyingi;
     delete temp;
     temp=z;
   }
 }
else {           //element qiymati o'chiriladigan songa teng emas
  oldingi=temp;
```

```
   temp=temp->keyingi;
   }
  }
 }
 return z;
}
void Zanjir_Ekranga(Zanjir * z)
{
 cout<<"Zanjir elementlari:"<<end1;
Zanjir * temp=z;
while(temp){
  cout<<temp->element<<' ';
  temp=temp->keyingi;
  }
 cout<<end1;
}
Zanjir * Zanjirni_Uchirish(Zanjir * z);
{
 Zanjir * temp=z;
  while(z){
    z=z->keyingi;
   delete temp;
 }
return z;
}
int main()
{
 Zanjir * zanjir=0;
```

```
 int son,del_element;
do{
 cout<<"\n Sonni kiriting (0-jarayonni tugatish): ";cin>>son;
  if(son)zanjir=Element_Joylash(zanjir,son);
 }
 while(son);
Zanjir_Ekranga(zanjir);
cout<<"\n O'chiriladigan elementni kiriting: "; cin>>del_element;
zanjir=Element_Uchirish(zanjir,del_element);
Zanjir_Ekranga(zanjir);
Zanjir = Zanjirni_Uchirish(zanjir);
return 0;
}
```

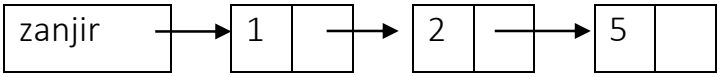
Dasturning bosh funksiyasida chiziqli ro‘yxat hosil qilish uchun Zanjir turidagi zanjir o‘zgaruvchisi aniqlangan bo‘lib, unga 0 qiymati berilgan (bo‘sh ko‘rsatkich qiymati, uning ekvivalenti –NULL). Keyin takrorlash operatori tanasida klaviaturadan butun son o‘kiladi va Element_Joylash() funksiyasini chaqirish orqali bu son ro‘yxatga oxiriga qo‘shiladi. Funksiya yangi hosil bo‘lgan ro‘yxat boshining adresini yana zanjir o‘zgaruvchisiga qaytaradi. Agar klaviaturadan 0 soni kiritilsa ro‘yxatni hosil kilish jarayoni tugaydi. Faraz qilaylik, quyidagi sonlar ketma-ketligi kiritilgan bo‘lsin: 1,2,3,3,5,0. U holdahosil bo‘lgan ro‘yxat quyidagi ko‘rinishda bo‘ladi:



Beshta sondan tashkil topgan chiziqli ro‘yxat

Hosil bo‘lgan ro‘yxatni ko‘rish uchun Zanjir_Ekranga() funksiyasi chaqiriladi va ekranda ro‘yxat elementlari chop etiladi. Ro‘yxat ustida amal sifatida berilgan son bilan ustma-ust tushadigan elementlarni o‘chirish masalasi qaralgan. Buning uchun o‘chiriladigan son del_element o‘zgaruvchiga o‘qiladi

va u Element_Uchirish() funksiyasi chaqirilishida argument sifatida uzatiladi. Funksiya bu son bilan ustma-ust tushadigan ro‘yxat elementlarini o‘chiradi (agar bunday element mavjud bo‘lsa) va o‘zgargan ro‘yxat boshining adresini zanjir o‘zgaruvchisiga qaytarib beradi. Masalan, ro‘yxatdan 3 soni bilan ustma-ust tushadigan elementlar o‘chirilgandan keyin u quyidagi ko‘rinishga ega bo‘ladi. Ro‘yxatdan 3 sonini o‘chirilgandan keyingi ko‘rinish:

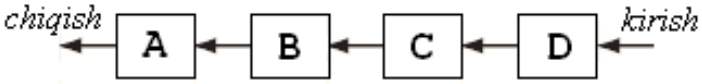


O‘zgargan ro‘xat elementlari ekranga chop etiladi. Dastur oxirida, Zanjirni_Uchirish() funksiyasini chaqirish orqali ro‘yxat uchun dinamik ravishda ajratilgan xotira bo‘shatiladi (garchi bu ishning dastur tugashi paytida bajarilishining ma’nosi yo‘q).

Navbat

Kundalik hayotda deyarli har kuni har bir inson navbat tushunchasi bilan duch keladi. Umuman olganda navbat elementi qandaydir xizmat ko‘rsatishga buyurtma bo‘lib hisoblanadi: masalan, ma’lumotlar byurosidan kerakli ma’lumotni olish, kinoteatrlarda chipta olish, do‘konda xarid qilib olingan mahsulotlarga kassada pul to‘lash va boshqa.

Dasturlashda shunday berilganlarningstrukturasi mavjudki, unga*navbat* deyiladi va u real navbatlarni modellashtirishda katta ahamiyatga ega. Bunda xizmat ko‘rsatishga kelib tushgan talab, uning ijrosi, ya’ni xizmat ko‘rsatish tartibini aniqlashda zarur bo‘ladi. Kundalik hayotimizdan barchamizga ma’lum bo‘lgan navbat turi, dasturlashda FIFO (“*first input – first output*”, ya’ni “*birinchi kelgan - birinchi ketadi*”) deb nomlanadi. Quyida 4 ta elementdan iborat navbat keltirilgan.



Bu yerdan ko‘rinib turibdiki, xizmat ko‘rsatilish birinchi kelgan elementga birinchi bo‘lib xizmat ko‘rsatiladi.Navbatda elementni olish ro‘yxat boshidan, yozish esa oxiridan amalga oshiriladi.

Kompyuter xotirasida elementlari soni chekli bo‘lgan bir o‘lchamli massiv ko‘rinishida yaratiladi. Albatta, bunda navbat elementi turini ko‘rsatish va navbat bilan ishlashni ko‘rsatuvchi o‘zgaruvchi zarur bo‘ladi.

Navbat uchun uchta asosiy amal aniqlangan:

1. Navbatga yangi element joylashtirish (insert(q, x), bu yerda q – navbat, x – yangi joylanadigan element);

2. Navbat boshidan elementni o‘chirish (remove(q));

3. Navbatni bo‘sh yoki bo‘sh emasligini aniqlash (empty(q)).

Bundan tashqari, navbat bir o‘lchamli massiv ko‘rinishida ifodalanganligi uchun massivni to‘la yoki to‘la emasligini kuzatib turish lozim bo‘ladi. Shu maqsadda full(q)funksiyasini kiritish mumkin.

Umuman olganda, insert() funksiyasini har doim bajarish mumkin. Sababi, navbatni tashkil qiluvchi elementlar soniga cheklanishlar qo‘yilmagan. remove() funksiyasi esa faqatgina navbat bo‘sh bo‘lmagandagina ishlaydi. Navbat bo’shligini tekshirish (empty()) esa har doim o‘rinli.

Stek

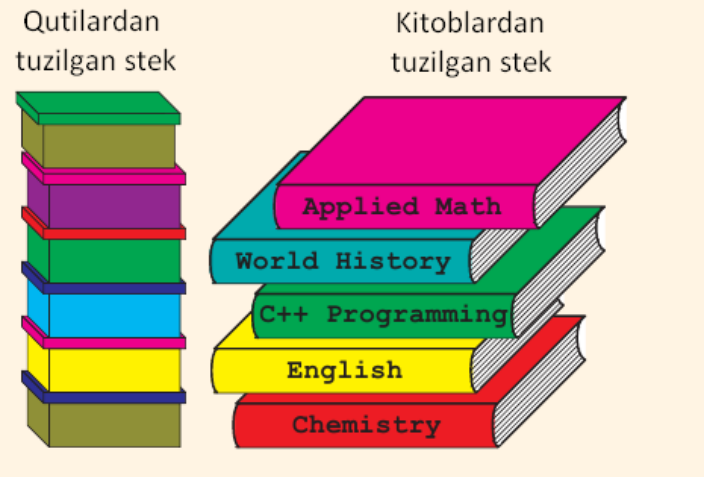
LIFO (“*last in – first out”*, “*oxirgi kelgan – birinchi ketadi*”), ya’ni navbatning oxirgi bo‘lib kirgan elementiga birinchi bo‘lib xizmat ko‘rsatiladi. Bu eng ko‘p ishlatiladigan ma’lumotlar tuzilmalaridan biri bo‘lib, turli xil masalalarni hal qilishda ancha qulay va samarali xisoblanadi.

Xizmat ko‘rsatishni keltirilgan tartibiga ko‘ra, stekda faqatgina bitta pozitsiyaga murojaat qilish mumkin. Bu pozitsiya stekning uchi deyilib unda stekka vaqt bo‘yicha eng oxirgi kelib tushgan element nazarda tutiladi. Biz stekga yangi element kiritsak, bu element oldingi stek uchida turgan element ustiga joylashtiriladi va u stekni uchiga joylashib qoladi. Elementni faqatgina stek uchidan tanlash (olish) mumkin; bunda tanlangan element stekdan chiqarib tashlanadi va stek uchini esa chiqarib tashlangan elementdan bitta oldin kelib tushgan element tashkil qilib qoladi.

Shuni qayd etish kerakki, stek va navbatdan element faqat bir marta

olinadi. Olingan element royxatdan “*o’chiriladi*”.

Stekni grafik ko‘rinishida quyidagicha tasvirlash mumkin:



Struktura orqali stekning ko‘rinishi, unga element qo‘shish va o‘chirish quyidagicha amalga oshirilishi mumkin:

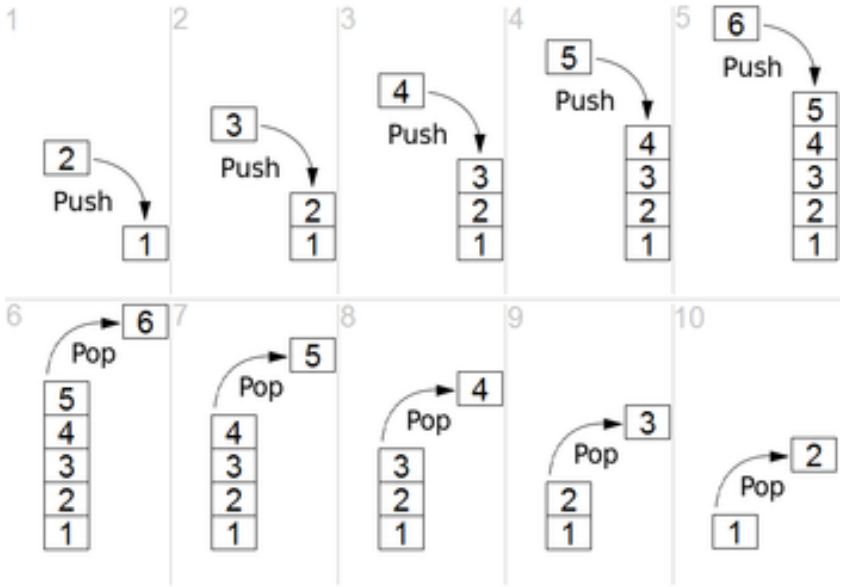
```
struct stack {
   char *data;
   struct stack *next;
};
void push( STACK *ps, int x )      // stekka yangi element qo‘shish funksiyasi
{
   if ( ps->size == STACKSIZE ) {
      fputs( "Error: stack overflow\n", stderr );
     abort();
 }
 elseps->items[ps->size++] = x;
}
int pop( STACK *ps )        // stekdan element o‘chirish funksiyasi
{
if (ps->size == 0) {
  fputs( "Error: stack underflow\n", stderr );
abort();
```

}

elsereturn ps->items[--ps->size];

}

Quyidagi rasmda stek bilan ishlash ko’rsatilgan



Nazorat savollari

1. Berilganlarning qanday dinamik tuziilmalari mavjud?
2. Chiziqli ro‘yxat qanday hosil qilinadi?
3. Stek qanday ko‘rinishda ishlaydi?
4. Navbat qanday ko‘rinishda ishlaydi?
5. Navbat ustida qanday amallar bajariladi?
6. Dek nima va u qanday prinsipda ishlaydi?
7. Stek ko‘rinishiga misollar keltiring.
8. Chiziqli ro‘yxat tugunlari qanday qilib bir-biri bilan bog‘lanadi?
9. Qanday hollarda chiziqli ro‘yxat bilan ishlash xatolari yuzaga keladi?
10. Stekka element joylash va o‘chirish amallari qanday ko‘rinishda bajarladi?

Glossariy

Termin	Terminology	O‘zbek tilidagi sharhi
break		takrorlashni to‘xtatish operatori
case		konstantalar bilan tekshirish operatori
char		belgi ko‘rinishidagi berilganlarning turi
cin		ekrandan kiritish oqimi
continue		takrorlash keyingi qadamiga o‘tkazish operatori
cout		ekranga chiqarish oqimi
define		Makrosni e’lon qiluvchi direktiva
delete		xotiradan ajratilgan joyni tozalash operatori
double		haqiqiy son ko‘rinishidagi berilganlarning turi
do-while		sharti keyin tekshiriladigan takrorlash operatori
else		shart yolg‘onligini aniqlovchi operator
enum		sanab o‘tiluvchi tur
EOF		#define EOF(-1) ko‘rinishida aniqlangan makros
for		takrorlash qadami bilan beriladigan takrorlash operatori
fstream		Fayl oqimi
if		shart operatori
ifstream		O‘qish fayli oqimi
include		preprotsessor direktivasi, kutubxona fayllarni dasturga ulash uchun ishlatiladi
int		butun son ko‘rinishidagi berilganlarning turi
namespace		nomlar fazosini e’lonini aniqlovchi kalit so‘z
new		xotiradan yangi joy ajratish operatori
NULL		mavjud bo‘lmagan qiymat

ofstream		Yozish fayli oqimi
setw		o‘zgaruvchining belgi bilan to‘ldirib chiqarish
sizeof		o‘zgaruvchi turining xotiradagi xajmini aniqlash
switch		bir nechta konstanta bilan tekshirish operatori
typedef		turlarni yangi nom bilan ishlatish imkonini beradi
using		nomlar fazosini dasturga ulash uchun ishlatiladigan kalit so‘z
while		sharti oldin tekshiriladigan takrorlash operatori
adres	adress	o‘zgaruvchi xotirada joylashadigan adres
amal qilish sohasi		o‘zgaruvchini ishlatish mumkin bo‘lgan dastur sohasi
argument	argument	funksiyaga parametriga jo‘natiladigan qiymat
bayt		kompyuter xotirasi o‘lchov birligi
berilganlar	data	dastur ishlashi uchun kerakli qiymatlar
binar amal	binary	ikkita operand ustida bajariluvchi amal
birlashma	union	maydonlariga umumiy joy ajratiladigan tuzilma
bit	bit	eng kichik o‘lchov birligi
dekrement	decrement	o‘zgaruvchining qiymatini bittaga kamaytirish
identifikator	identifier	katta va kichik lotin harflari, raqamlar va tag chiziq (‘_’) belgilaridan tashkil topgan va raqamdan boshlanmaydigan belgilar ketma-ketligi
inkrement	increment	o‘zgaruvchining qiymatini bittaga oshirish
kengaytma	extension	fayllarning turli dasturlarga tegishliligini aniqlovchi fayl ko‘rinishining qismi
kompilyatsiya	compilation	bajariluvchi fayl xosil bo‘lish jarayoni

konstanta	const	dastur davomida qiymati o‘zgarmaydigan berilgan
kutubxona	library	dasturga include direktivasi yordamida qo‘shiladigan fayllar
ko‘rsatkich	pointer	qiymatlari adres bo‘lgan o‘zgaruvchilar
leksema	lexeme	tilning ajralmaydigan qismlari
parametr	parametr	funksiya ishlashi uchun kerak berilganlar
postfiks	postfix	operatorning o‘zgaruvchidan keyin joylashgan ko‘rinishi
prefiks	prefix	operatorning o‘zgaruvchidan oldin joylashgan ko‘rinishi
razryad	discharge	bitlardan (0 yoki 1) tashkil topgan indikator
sarlavha fayli	header file	funksiyalar e’loni yozilgan fayl
semantika	semantics	tilning ma’nosini beruvchi qoidalar to‘plami
sintaktik qoidalar	sintaktik rules	grammatik qoidalarga o‘xshash qoidalar to‘plami
struktura	struct	bir yoki har xil turdagi berilganlarni jamlanmasi
unar amal	unar	bitta operand ustida bajariluvchi amal
o‘zgarmas	constant	dastur ishlashi davomida qiymatini o‘zgartirmaydigan berilgan
o‘zgaruvchi	variable	berilganlarni saqlab turish uchun ishlatiluvchi til birligi
fayl	file	bu bir xil turdagi qiymatlar joylashgan tashqi xotiradagi nomlangan sohadir
fayl ko‘rsatkichi	file pointer	ayni paytda fayldan o‘qilayotgan yoki unga yozilayotgan joyni (yozuv o‘rnini) ko‘rsatib turadi
funksiya	function	dastur alohida bo‘lagi, asosiy qism tomonidan chaqirib ishlatiladi
cheksiz takrorlash	endless loop	takrorlashni to‘xtatish shartining mavjud emasligi

Foydalanilgan adabiyotlar ro‘yxati

1. Bjarne Stroustrup. The C++ Programming Language (3th Edition). Addison-Wesley, 1997.
2. D.S. Malik. C++ Programming: From Problem Analysis to Program Design. Fifth Edition. Course Technology, 2011.
3. Мадрахимов Ш.Ф., Гайназаров С.М. С++ тилида дастурлаш асослари// Тошкент, ЎзМУ, 2009, 196 бет.
4. Madraximov Sh.F., Ikramov A.M., Babajanov M.R. C++ tilida dasturlash bo’yicha masalalar to’plami. O’quv qo’llanma // Toshkent, O’zbekiston Milliy Universiteti, “Universitet” nashriyoti, 2014. - 160 bet.
5. Ivor Horton. Beginning Visual C++2005. Wiley Publishing, 2005. 1182 page
6. T.A.Maxarov, Q.T.Maxarov. Visual Studio muhitida dasturlash asoslari. Bakalavrlar uchun uslubiy qo‘llanma. Toshkent, O‘zMU. 2017. – 90 b
7. Bjarne Stroustrup. The C++ Programming Language (4th Edition). Addison-Wesley, 2013. 1363 page.
8. Bjarne Stroustrup. Programming: Principles and Practice using C++ (Second Edition)" Addison-Wesley,2014, 1305page.
9. Павловская Т.А. С++. Программирование на языке высокого уровня – СПб.: Питер. 2005.- 461 с.
10. Walter Savitch. Absolute C++, 5th edition. Addison-Wesley/Pearson, 2012. 984 page.
11. Walter Savitch. Problem Solving with C++, 9th edition. Addison-Wesley/Pearson, 2015. 1088 page.
12. Павловская Т.С. Щупак Ю.С. С/С++. Структурное программирование. Практикум.-СПб.: Питер,2002-240с
13. Глушаков С.В., Коваль А.В., Смирнов С.В. Язык программирования С++: Учебный курс.- Харьков: Фолио; М.: ООО «Издательство АСТ», 2001.-500с.

14. Scheinerman Edwant C++ for Mathematicifns. AnIntroduction for Students and Professionals. Chapman&Hall/CRC,Taylor&Francis Group, LLC, Boca Raton, London, New York, 2006.
15. Герберд Шилдт. C++: базовый курс, 3-е издание. Перевод с английского. –М: Изд.дом “Вильямс”, 2010.
16. О.Е.Степаненко. Visual C++.NET. Классика программирования.: - М: Научная книга, К.; Букинист, 2010.
17. Пахомов Б.И. C/C++ и MS Visual Studio 2010 для начинающих. С-Пб. БХВ-Петербург, 2011.
18. Культин Н.Б. C++Builder в задачах и примерах.-СПб.: БХВ-Петербург, 2005.-336с.
19. Абрамов С.А., Гнездела Капустина Е.Н. и др. Задачи по программированию. - М.: Наука, 1988.

Internet manbalar

1. http://cppstudio.com – C++ tilida dasturlash bo‘yicha namunalar izohlari bilan keltirilgan.
2. http://cplusplus.com – C++ tilida mavjud konstruksiyalar ta’rifi, ishlatish namunalari bilan keltirilgan.
3. http://www.compteacher.ru/programming – dasturlash bo‘yicha video darsliklar mavjud.
4. http://www.intuit.ru – internet universitet, dasturlash bo‘yicha yozma va video ma’ruzalar o‘qish, test sinovlaridan o‘tish va sertifikat olish imkoniyati mavjud.
5. http://www.ziyonet.uz – dasturlash asoslari bo‘yicha referatlar topish mumkin.

Ilovalar

1-ilova. Dasturni sozlash texnologiyalari

Dasturdagi xatolar

Dasturdagi xatolar *baglar* deb nomlanadi va debug (debag) jarayoni ularni topish va sozlash uchun ishlatiladi. Dasturdagi xatolar asosan ikki xil ko‘rinishda bo‘ladi:

1. Sintaktik xato – dasturni yozishda yo‘l qo‘yilgan xotalar. Bunga nuqta, vergul, nuqtali vergulni noto‘g‘ri qo‘yilishi, dasturdagi kalit so‘zlarni noto‘g‘ri yozilishi misol bo‘ladi. Kompilyator bunday xatolarni ko‘rsatadi va qanday xatoligi to‘g‘risida izoh beradi.

2. Semantik xato – dasturning noto‘g‘ri ishlashi. Ya’ni, semantik xato bo‘lganda sintaktik xatolar bo‘lmaydi, dastur ishlaydi, lekin kutilgan natijaga erishib bo‘lmaydi. Masalan, quyidagi ikki qator sintaktik to‘g‘ri yozilgan, lekin ma’nolari turlicha (turlicha qiymat hosil bo‘ladi):

```
2 + 3 * 5          // 1-qator
```

va

```
(2 + 3) * 5        // 2-qator
```

Birinchi qatorda arifmetik operatorlar bajarilish ketma-ketligi qoidasiga ko‘ra ko‘paytirish amali bajariladi, so‘ngra qo‘shish amali ishga tushadi. Ikkinchi qatorda esa, avval qavs ichi bajariladi, sonlar qo‘shiladi, keyin ko‘paytirish amali ishlaydi.

Shuningdek, dastur tuzilayotgan dasturlash muhitining xatolari ham bo‘lishi mumkin. Lekin dasturning xatolari ichidan muxitning xatolarini qidirish eng so‘nggi xato qidirish sifatida qaralsa maqsadga muvofiq. Xatolarni tez topishning oson usullaridan biri – ularni alomati bo‘yicha guruhlab olish. Eng ko‘p tarqalgan xatolarning beshtasi quyidagi jadvalda keltirilgan.

Xatolik alomati	Sabablari
Berilganlarning buzilishi	1. O‘zgaruvchini initsializatsiya qilinmaganligi; 2. Son turi qiymatlari chegarasidan chiqib ketish; 3. Noto‘g‘ri qo‘rsatkich;

	4. Massivning indeksidagi ifodaning xatoligi; 5. Takrorlash shartining xatoligi; 6. Dinamik yaratilayotgan massivning o‘lchami xatoligi.
Qayta ishlanmagan uzilishlar	Noto‘g‘ri ko‘rsatkich yoki murojaat qilingan catch konstruksiyasining mavjud emasligi
Dasturning qotib qolishi	1. O‘zgaruvchini initsializatsiya qilinmaganligi; 2. Cheksiz takrorlash; 3. Noto‘g‘ri ko‘rsatkich; 4. Bo‘shatib bo‘lingan xotira bo‘lagining qaytadan bo‘shatilishi; 5. Foydalanuvchi berilganlarni kiritishida kutilmagan kiritishning oldini oluvchi qayta ishlashning mavjudmasligi.
Berilganlarning noto‘g‘ri kiritilishi	cin, scanf(), getline() funksiyalari bilan kiritishda noto‘g‘ri ishlatish
Noto‘g‘ri natijalar	1. Yozishdagi xatolik: “== “ amali o‘rniga “=” amalini ishlatilishi, i ning o‘rniga j ni ishlatilishi; 2. O‘zgaruvchini initsializatsiya qilinmaganligi; 3. Son turi qiymatlari chegarasidan chiqib ketish; 4. Noto‘g‘ri ko‘rsatkich; 5. switch qurilmasida break ni mavjudmasligi.

Sozlovchi va uni ishlatish

Sozlovchi – dasturni qadamma-qadam bajarilishini boshqaruvchi dastur. Shuningdek, dasturni ma’lum bir belgilangan joyigacha ham ishlatish mumkin. Har bir sozlovchi to‘xtagan dastur qismida o‘zgaruvchilar qiymatini ko‘rish, o‘zgartirish imkoni mavjud. Dastur kodini o‘zgartirish, qayta kompilyatsiya qilish va dasturni boshidan ishga tushirish mumkin.

Misol uchun quyidagi dasturni sozlovchi orqali ishlatish ko‘rilsin.

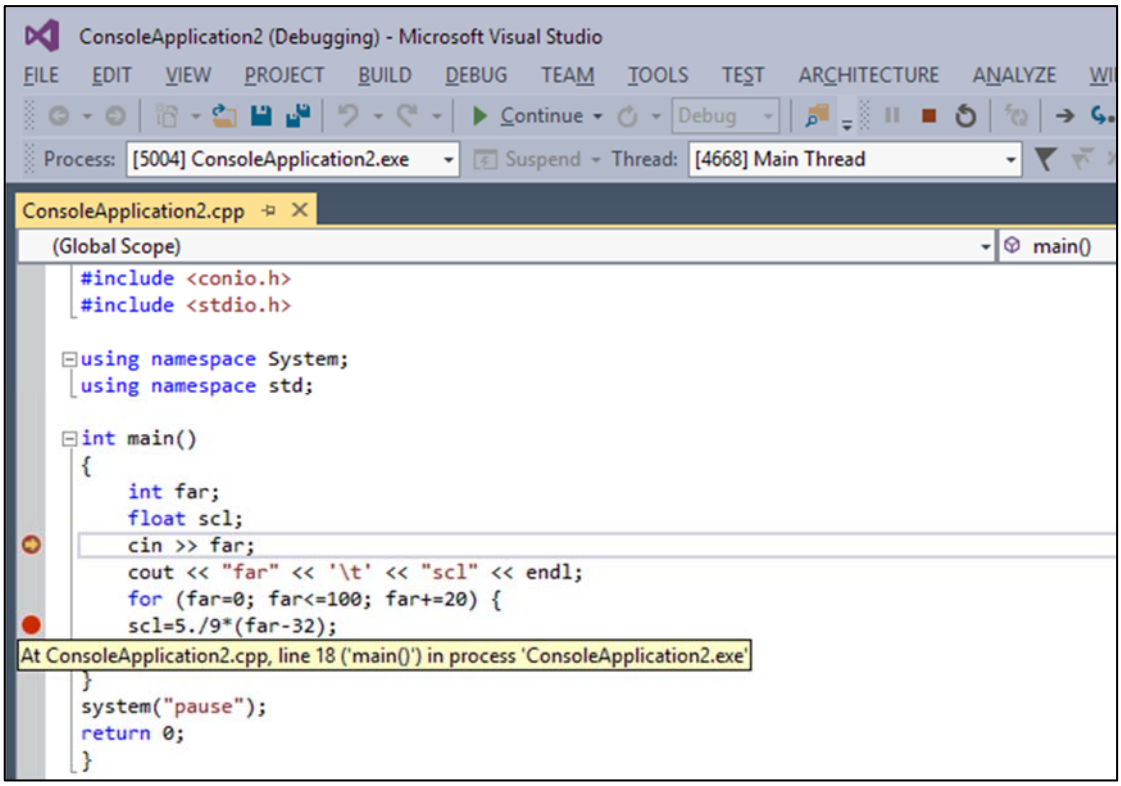
```
#include<iostream>
using namespace std;
int main()
```

```
{
long* pnumber = NULL;
long number1 = 55, number2 = 99;
pnumber = &number1;
*pnumber += 11;            // number1 qiymaini 11 ga oshirish
cout<< endl<<"number1="<<number1<<"&number1="<<hex<< pnumber;
pnumber = &number2;
number1 = *pnumber * 10;
cout << endl<< "number1 =" << dec << number1<<"pnumber ="<< hex<<
pnumber<< " *pnumber = " << dec << *pnumber;
cout << endl;
system("pause");
}
```

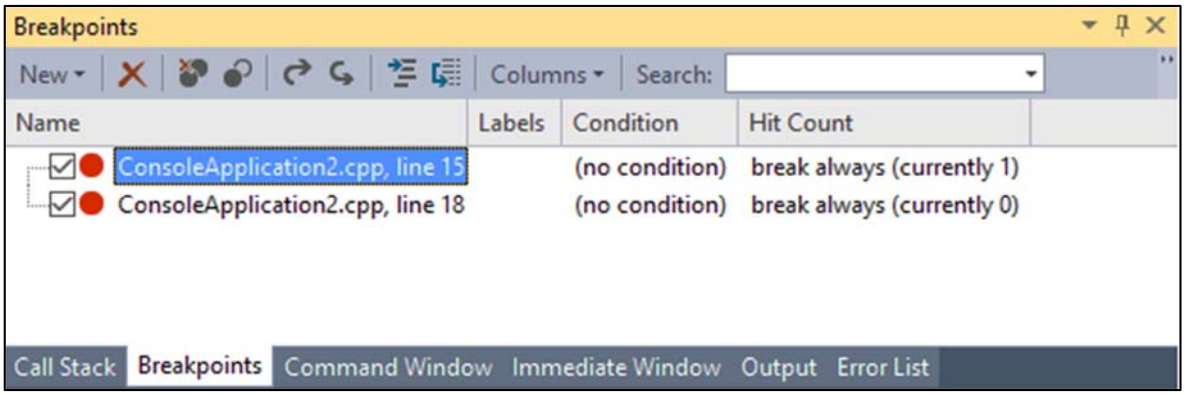
Avval, kompilyatsiya qilishdan oldin uning konfiguryasiyasi Win32 Release holatidan Win32 Debug holatiga o‘tkazilganiga ishonch hosil qilish kerak.

Debug konfiguryasiyasi dastur ishlashi jarayonida sozlash imkonini beruvchi qo‘shimcha ma’lumotlarni saqlab turadi. Qo‘shimcha ma’lumotlar loyixaning Debug papkasi ichidagi .pdb kengaytmali faylda saqlanadi. Debug instrumentlar paneli quyidagi ko‘rinishga ega:

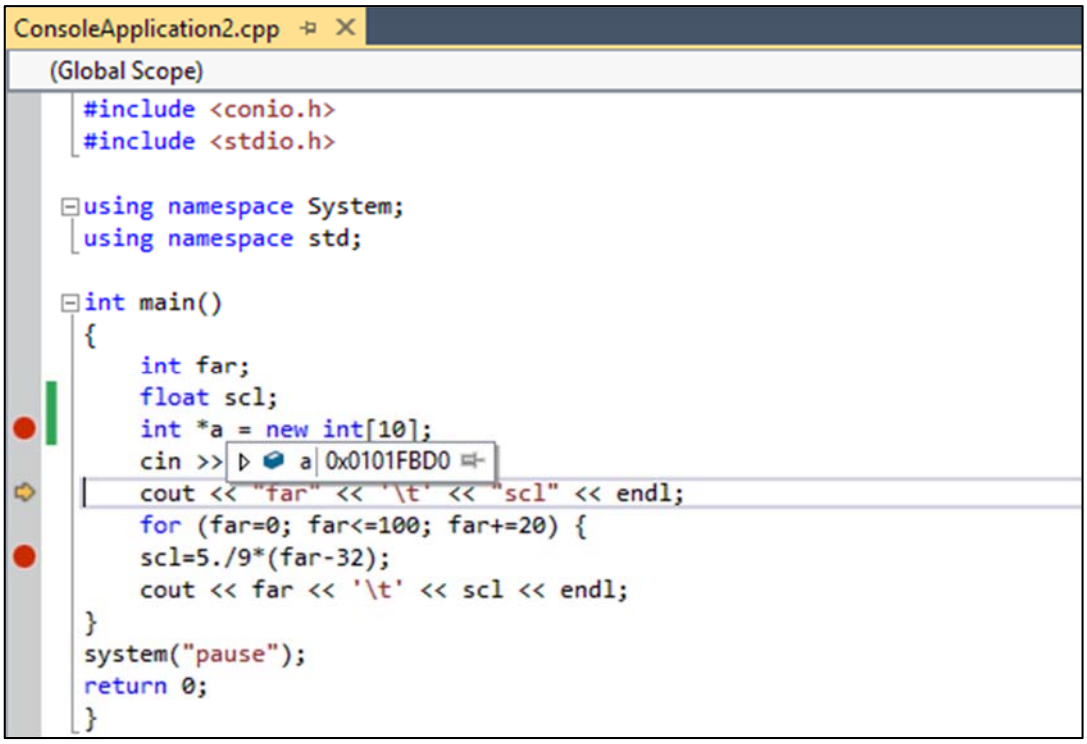
Sozlash jarayonini kerakli kod qismida to‘xtatish uchun to‘xtash nuqtalari ishlatiladi. Bir dasturda bir nechta to‘xtash nuqtalari bo‘lishi mumkin. Odatda to‘xtash nuqtalari xato bo‘lishi mumkin bo‘lgan qatorlarda qo‘yiladi. Ularni o‘rnatish uchun kerak kod qismida sichqonchaning o‘ng tugmasini bosib Breakpoint->Insert breakpoint menyusini tanlash yoki qatorning to‘g‘risida chap panelda sichqonchaning chap tugmasini ikki marta bosish kerak.



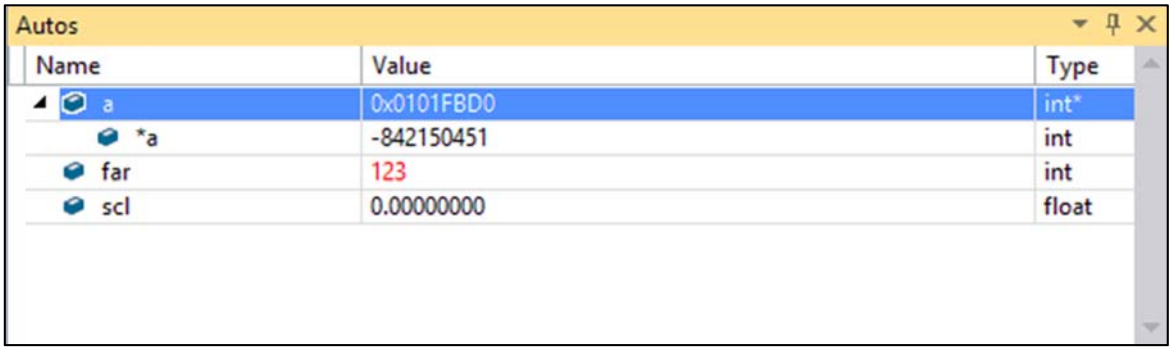
Alt+F9 tugmalar kombinatsiyasini bosish orqali to‘xtash nuqtalari oynasini ochish va ularni ko‘rish, qayta ishlash imkoni mavjud.



Debug holatida dasturni ishga tushirish uchun F5 tugmaini bosish kerak. Dastur doimgidek ishlaydi, ammo to‘xtash nuqtalari berilgan joyda to‘xtaydi. Dasturlash muxiti orqali shu kod qismigacha bajarilgan holatdagi o‘zgaruvchilar qiymatlarini ko‘rish uchun kerakli o‘zgaruvchi ustida sichqonchani ushlab turish yetarli. O‘zgaruvchiga ko‘rsatkich bo‘lsa uning xotiradagi adresini, aks holda xotiradagi qiymatini ko‘rish mumkin.



Shuningdek, “Autos” darchasi orqali dasturdagi to‘xtash qismigacha bo‘lgan barcha o‘zgaruvchi, ko‘rsatkichlarning turi va qiymatlarini ko‘rish mumkin.



Ushbu oyna orqali o‘zgaruvchining qiymatini o‘zgartirib dasturni ishlashini davom ettirish mumkin. Buning uchun mos ustundagi qiymat ustida sichqoncha tugmasini ikki marta bosish etarli. Keyin kerakli qiymatni kiritish mumkin. Dasturni ishlashini davom ettirish uchun yana F5 tugmasini bosish kerak.

2-ilova. Belgilarning ASCII kodlari jadvali

Boshqaruv belgilar kodlari (0-31)

Mnemonik nomi	10 s.s. kodi	16 s.s. kodi	Klaviatura tugmasi	Mazmuni
nul	0	00	^@	Nol
soh	1	01	^A	Sarlavha boshlanishi
stx	2	02	^B	Matn boshlanishi
etx	3	03	^C	Matn tugashi
eot	4	04	^D	Uzatishning tugashi
enq	5	05	^E	So‘rov
ack	6	06	^F	Taqiqlash
bel	7	07	^G	Signal (tovush)
bs	8	08	^H	Orqaga qadam
ht	9	09	^I	Gorizontal tabulyasiya
lf	10	0A	^J	Yangi satrga o‘tish
vt	11	0B	^K	Vertikal tabulyasiya
ff	12	0C	^L	Yangi sahifaga o‘tish
cr	13	0D	^M	Karetkani qaytarish
soh	14	0E	^N	Surishni man etish
si	15	0F	^O	Surishga ruxsat berish
dle	16	10	^P	Berilganlar bog‘lash kaliti
dc1	17	11	^Q	1-qurilmani boshqarish
dc2	18	12	^R	2-qurilmani boshqarish
dc3	19	13	^S	3-qurilmani boshqarish
dc4	20	14	^T	4-qurilmani boshqarish
nak	21	15	^U	Taqqoslash inkori
syn	22	16	^V	Sinxronizatsiya
etb	23	17	^W	Uzatilgan blok oxiri
can	24	18	^X	Rad qilish
em	25	19	^Y	Soha tugashi
sub	26	1A	^Z	Almashtirish
esc	27	1B	^[	Kalit
fs	28	1C	^\	Fayllar ajratuvchisi

qs	29	1D	^]	Guruh ajratuvchi
rs	30	1E	^^	Yozuvlar ajratuvchisi
us	31	1F	^_	Modullar ajratuvchisi

Akslanuvchi belgilar (32-127)

Belgi	10 s.s. kodi	16 s.s. kodi	Belgi	10 s.s. kodi	16 s.s. kodi	Belgi	10 s.s. kodi	16 s.s. kodi
	32	20	@	64	40	‘	96	60
!	33	21	A	65	41	a	97	61
“	34	22	B	66	42	b	98	62
#	35	23	C	67	43	c	99	63
$	36	24	D	68	44	d	100	64
%	37	25	E	69	45	e	101	65
&	38	26	F	70	46	f	102	66
`	39	27	G	71	47	g	103	67
(	40	28	H	72	48	h	104	68
)	41	29	I	73	49	i	105	69
*	42	2A	J	74	4A	j	106	6A
+	43	2B	K	75	4B	k	107	6B
,	44	2C	L	76	4C	l	108	6C
-	45	2D	M	77	4D	m	109	6D
.	46	2E	N	78	4E	n	110	6E
/	47	2F	O	79	4F	o	111	6F
0	48	30	P	80	50	p	112	70
1	49	31	Q	81	51	q	113	71
2	50	32	R	82	52	r	114	72
3	51	33	S	83	53	s	115	73
4	52	34	T	84	54	t	116	74
5	53	35	U	85	55	u	117	75
6	54	36	V	86	56	v	118	76
7	55	37	W	87	57	w	119	77
8	56	38	X	88	58	x	120	78
9	57	39	Y	89	59	y	121	79
:	58	3A	Z	90	5A	z	122	7A

;	59	3B	[	91	5B	{	123	7B
<	60	3C	\	92	5C	|	124	7C
=	61	3D	]	93	5D	}	125	7D
>	62	3E	^	94	5E	~	126	7E
&	63	3F	_	95	5F	del	127	7F

Akslanuvchi belgilar (128-255)(Windows-1251)

Belgi	**10 s.s. kodi**	**16 s.s. kodi**	**Belgi**	**10 s.s. kodi**	**16 s.s. kodi**	**Belgi**	**10 s.s. kodi**	**16 s.s. kodi**
Ђ	128	80	«	171	AB	S	214	D6
Ѓ	129	81	¬	172	AC	CH	215	D7
,	130	82	-	173	AD	Sh	216	D8
ѓ	131	83	®	174	AE	Щ	217	D9
„	132	84	Ї	175	AF	‘	218	DA
…	133	85	°	176	B0	Ьі	219	DB
†	134	86	±	177	B1		220	DC
‡	135	87	I	178	B2	E	221	DD
€	136	88	i	179	B3	Yu	222	DE
‰	137	89	ґ	180	B4	YA	223	DF
Љ	138	8A	µ	181	B5	a	224	E0
‹	139	8B	¶	182	B6	b	225	E1
Њ	140	8C	·	183	B7	v	226	E2
Ќ	141	8D	yo	184	B8	g	227	E3
Ћ	142	8E	№	185	B9	d	228	E4
Џ	143	8F	є	186	BA	e	229	E5
ђ	144	90	»	187	BB	j	230	E6
‘	145	91	j	188	BC	z	231	E7
’	146	92	S	189	BD	i	232	E8
“	147	93	s	190	BE	y	233	E9
”	148	94	ї	191	BF	k	234	EA
•	149	95	A	192	C0	l	235	EB
-	150	96	B	193	C1	m	236	EC
—	151	97	V	194	C2	n	237	ED
	152	98	G	195	C3	o	238	EE

™	153	99	D	196	C4	p	239	EF
љ	154	9A	E	197	C5	r	240	F0
›	155	9B	J	198	C6	s	241	F1
њ	156	9C	Z	199	C7	t	242	F2
ќ	157	9D	I	200	C8	u	243	F3
ћ	158	9E	Y	201	C9	f	244	F4
џ	159	9F	K	202	CA	x	245	F5
	160	A0	L	203	CB	s	246	F6
O‘	161	A1	M	204	CC	ch	247	F7
o‘	162	A2	N	205	CD	sh	248	F8
J	163	A3	O	206	CE	щ	249	F9
¤	164	A4	P	207	CF	‘	250	FA
Ґ	165	A5	R	208	D0	ы	251	FB
¦	166	A6	S	209	D1		251	FC
§	167	A7	T	210	D2	e	253	FD
YO	168	A8	U	211	D3	yu	254	FE
©	169	A9	F	212	D4	ya	255	FF
Є	170	AA	X	213	D5			