O‘ZBEKISTON RESPUBLIKASI
OLIY VA MAXSUS TA’LIM VAZIRLIGI
MIRZO ULUG‘BEK NOMIDAGI
O‘ZBEKISTON MILLIY UNIVERSITETI

Madraximov Sh.F

C++ tilida programmalash asoslari

Toshkent 2009

Annotatsiya

Qo'llanmada C++ tilining sintaksisi, semantikasi va undagi programmalash texnologiyalari keltirilgan. Ayniqsa, C++ tili asosini tashkil etuvchi berilganlar va ularning turlari, operatorlar, funksiyalar, ko'rsatkichlar va massivlar, hamda berilganlar oqimlari bilan ishlash tushunarli ravishda bayon qilingan va sodda misollarda namunalar keltirilgan.

В предлагаемой пособии приводится синтаксис и семантика языка C++, а также технологии программирования на данном языке. Последовательно и доступно объяснены основные понятия языка C++, такие как данные и их типы, операторы, функции, указатели и массивы, потоки ввода и вывода.

In the offered educational manual the syntax, semantics and programming technologies of C++ are given. The basic concepts of language C++ as data and its types, operators, functions, pointers and files, arrives, input and output data flows are explained in accessible form.

Tuzuvchilar: dotsent Sh.F.Madraximov

Taqrizchilar:

O'zMU dotsenti N.A.Ignatev

Mas'ul muharrir: O'zMU professori M.M.Aripov

Mundarija

Kirish

Ma'lumki, programma mashina kodlarining qandaydir ketma-ketligi bo'lib, aniq bir hisoblash vositasini amal qilishini boshqaradi. Programma ta'minotini yaratish jarayonini osonlash-tirish uchun yuzlab programmalash tillari yaratilgan. Barcha programmalash tillarini ikki toifaga ajratish mumkin:

- quyi darajadagi programmalash tillari;
- yuqori darajadagi programmalash tillari.

Quyi darajadagi programmalash tillariga Assembler turidagi tillar kiradi. Bu tillar nisbatan qisqa va tezkor bajariluvchi kodlarni yaratish imkoniyatini beradi. Lekin, Asssembler tilida programma tuzish zahmatli, nisbatan uzoq davom etadigan jarayondir. Bunga qarama-qarshi ravishda yuqori bosqich tillari yaratilganki, ularda tabiiy tilning cheklangan ko'rinishidan foydalangan holda programma tuziladi. Yuqori bosqich tillaridagi operatorlar, berilganlarning turlari, o'zgaruvchilar va programma yozishning turli usullari tilning ifodalash imkoniyati oshiradi va programmani «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#, Php, Java, Asp tillari hisoblanadi. Xususan, S tilining takommillashgan varianti sifatida C++ tilini olishimiz mumkin. 1972 yilda Denis Ritch va Brayan Kernegi tomonidan S tili yaratildi. 1980 yilda Byarn Straustrop S tilining avlodi C++ tilini yaratdiki, unda strukturali va ob'ektga yo'naltirilgan programmalash texnologiyasiga tayangan holda programma yaratish imkoniyati tug'ildi.

1- bob. C++ tili va uning leksik asosi

C++ tilidagi programma tuzilishi va uning kompilyasiyasi

C++ tilida programma yaratish bir nechta bosqichlardan iborat bo‘ladi. Dastlab, matn tahririda (odatda programmalash muhitining tahririda) programma matni teriladi, bu faylning kengaytmasi «.srr» bo‘ladi, Keyingi bosqichda programma matn yozilgan fayl kompilyatorga uzatiladi, agarda programmada xatoliklar bo‘lmasa, kompilyator «.obj» kengaytmali ob’ekt modul faylini hosil qiladi. Oxirgi qadamda komponovka (yig‘uvchi) yordamida «.exe» kengaytmali bajariluvchi fayl - programma hosil bo‘ladi. Bosqichlarda yuzaga keluvchi fayllarning nomlari boshlang‘ich matn faylining nomi bilan bir xil bo‘ladi.

Kompilyasiya jarayonining o‘zi ham ikkita bosqichdan tashkil topadi. Boshida preprotsessor ishlaydi, u matndagi kompilyasiya direktivalarini bajaradi, xususan #include direktivasi bo‘yicha ko‘rsatilgan kutubxonalardan C++ tilida yozilgan modullarni programma tarkibiga kiritadi. Shundan so‘ng kengaytirilgan programma matni kompilyatorga uzatiladi. Kompilyator o‘zi ham programma bo‘lib, uning uchun kiruvchi ma’lumot bo‘lib, C++ tilida yozilgan programma matni hisoblanadi. Kompilyator programma matnini leksema (atomar) elementlarga ajratadi va uni leksik, keyinchalik sintaksik tahlil qiladi. Leksik tahlil jarayonida u matnni leksemalarga ajratish uchun «probel ajratuvchisini» ishlatadi. Probel ajratuvchisiga - probel belgisi (‘␣’), ‘\t’ - tabulyasiya belgisi, ‘\n’- keyingi qatorga o‘tish belgisi, boshqa ajratuvchilar va izohlar hisoblanadi.

Programma matni tushunarli bo‘lishi uchun izohlar ishlatiladi. Izohlar kompilyator tomonidan «o‘tkazib» yuboriladi va ular programma amal qilishiga hech qanday ta’sir qilmaydi.

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

Birinchisida “/*” dan boshlanib, “*/” belgilar oralig‘ida joylashgan barcha belgilar ketma-ketligi izoh hisoblanadi, ikkinchisi «satriy 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;
}
```

Programmada d, b o‘zgaruvchilar e’lonlari inobatga olinmaydi va a=c amali bajarilmaydi.

Quyida C++ tilidagi sodda programma matni keltirilgan.

```
# include <iostream.h> // sarlavha faylni qo‘shish
int main ()            // bosh funksiya tavsifi
{                      // blok boshlanishi
 cout << "Salom Olam!\n";// satrni chop etish
 return 0;             //    funksiya    qaytaradigan
qiymat
}                      // blok tugashi
```

Programma bajarilishi natijasida ekranga “Salom Olam!” satri chop etiladi.

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

Programma tanasida konsol rejimida belgilar ketma-ketligini oqimga chiqarish amali qo‘llanilgan. Ma’lumotlarni standart oqimga (ekranga) chiqarish uchun quyidagi format ishlatilgan:

sout << <ifoda>;

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

Misol:

```
int Yosh;
cout<<"Yoshingizni kiriting_" ;
cin>>Yosh;
```

Butun turdagi Yosh o‘zgaruvchisi kiritilgan qiymatni o‘zlash-tiradi. Kiritilgan qiymatni o‘zgaruvchi turiga mos kelishini tekshirish mas’uliyati programma tuzuvchisining 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.h>
int main()
{
 int x,y;
 float z;
 cin>>x>>y>>z;
 cout<<"O'qilgan qiymatlar\n";
 cout<<x<<'\t'<<y<<'\t'<<z;
 return 0;
}
```

O‘zgaruvchilarga qiymat kiritish uchun klaviatura orqali

```
10 20 3.14 <enter>
```

harakati amalga oshiriladi. Shuni qayd etish kerakki, oqimga qiymat kiritishda probel ajratuvchi hisoblanadi. Haqiqiy sonning butun va kasr qismlari ‘.’ belgisi bilan ajratiladi.

C++ tili alfaviti va leksemalar

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; amallar belgilanishlari; ajratuvchilar.

Identifikatorlar va kalit so‘zlar

Programmalash tilining muhim tayanch tushunchalaridan biri - identifikator tushunchasidir. *Identifikator* deganda katta va kichik lotin harflari, raqamlar va tag chiziq (‘_’) belgilaridan tashkil topgan va

raqamdan boshlanmaydigan belgilar ketma-ketligi tushuniladi. Identifikatorlarda harflarning registrlari (katta yoki kichikligi) hisobga olinadi. Masalan, RUN, run, Run - bu har xil identifikatorlardir. Identifikator uzunligiga chegara qo‘yilma-gan, lekin ular kompilyator tomonidan faqat boshidagi 32 belgisi bilan farqlanadi.

Identifikatorlar kalit so‘zlar, o‘zgaruvchilar, funksiyalar, nishonlar va boshqa ob’ektlarni nomlashda ishlatiladi.

C++ tilining kalit so‘zlariga quyidagilar kiradi:

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

Yuqorida keltirilgan identifikatorlarni boshqa maqsadda ishlatish mumkin emas.

Protsessor registrlarini belgilash uchun quyidagi so‘zlar ishlatiladi:

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

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

2- bob. C++ tilida berilganlar va ularning turlari

O‘zgarmaslar

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

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

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

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

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

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

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

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

```
023; 0777; 0.
```

O‘n oltilik o‘zgarmas 0x yoki 0X belgilaridan boshlanadigan o‘n oltilik sanoq sistemasi raqamlaridan iborat ketma-ketlik hisob-lanadi:

```
0x1A; 0X9F2D; 0x23.
```

Harf belgilar ixtiyoriy registrlarda berilishi mumkin.

Kompilyator sonning qiymatiga qarab unga mos turni belgilaydi. Agar tilda belgilangan turlar programma 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.
```

Haqiqiy o‘zgarmaslar. 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` .

Belgi o‘zgarmaslar. Belgi o‘zgarmaslar qo‘shtirnoq (‘,’-apostroflar) ichiga olingan alohida belgilardan tashkil topadi va u char kalit so‘zi bilan aniqlanadi. 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','+','sh','*','a', 's'`.

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

Escape belgilari	Ichki kod (16 son)	Nomi	Amal
\\	0x5S	\	Teskari yon chiziqni chop etish
\’	0x27	‘	Apostrofni chop etish
\”	0x22	“	Qo‘shtirnoqni chop etish
\?	0x3F	?	So‘roq belgisi
\a	0x07	bel	Tovush signalini berish
\b	0x08	bs	Kursorni 1 belgi o‘rniga orqaga qaytarish
\f	0x0C	ff	Sahifani o‘tkazish
\n	0x0A	lf	Qatorni o‘tkazish
\r	0x0D	cr	Kursorni ayni qatorning boshiga qaytarish
\t	0x09	ht	Navbatdagi tabulyasiya joyiga o‘tish
\v	0x0D	vt	Vertikal tabulyasiya (pastga)
\000	000		Cakkizlik kodi
\xNN	0xNN		Belgi o‘n oltilik kodi bilan berilgan

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

C++ tilida qo‘shimcha ravishda wide harfli o‘zgarmaslar va ko‘p belgili o‘zgarmaslar aniqlangan.

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

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

Ko‘p belgili o‘zgarmas turi int bo‘lib, u to‘rtta belgidan iborat bo‘lishi mumkin:

```
'abc', '\001\002\003\004'.
```

Satr o‘zgarmaslar. Ikkita qo‘shtirnoq (“,”) ichiga olingan belgilar ketma-ketligi *satr o‘zgarmas* deyiladi:

```
"Bu satr o'zgarmas va uning nomi string\n"
```

Satr ichida escape ketma-ketligi ham ishlatilishi mumkin, faqat bu ketma-ketlik apostrofsiz yoziladi.

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

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

Bu yozuv

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

yozuvi bilan ekvivalent hisoblanadi.

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

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

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

Satr xotirada joylashganda uning oxiriga ‘\0’ (0 kodli belgi) qo‘shiladi va bu belgi satr tugaganligini bildiradi. Shu sababli satr uzunligi, uning «haqiqiy» qiymatidan bittaga ko‘p bo‘ladi.

Berilganlar turlari va o‘zgaruvchilar

Programma bajarilishi paytida qandaydir berilganlarni saqlab turish uchun o‘zgaruvchilar va o‘zgarmaslardan foydalaniladi. *O‘zgaruvchi* - programma ob’ekti 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 konkret turiga bog‘liq bo‘ladi. Shuni qayd etish zarurki, bitta turga turli apparat platformalarda turlicha joy ajratilishi mumkin.

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

C++ tilining tayanch turlari

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

Butun son turlari. Butun son qiymatlarni qabul qiladigan o‘zgaruvchilar int (butun), short (qisqa) va long (uzun) kalit so‘zlar bilan aniqlanadi. O‘zgaruvchi qiymatlari ishorali bo‘lishi yoki unsigned kalit so‘zi bilan ishorasiz son sifatida qaralishi mumkin (1-ilovaga qarang).

Belgi turi. Belgi turidagi o‘zgaruvchilar char kalit so‘zi bilan beriladi va ular o‘zida belgining ASCII kodini saqlaydi. Belgi turidagi qiymatlar nisbatan murakkab bo‘lgan tuzilmalar - satrlar, belgilar massivlari va hakozalarni hosil qilishda ishlatiladi (2-ilovaga qarang).

1.2-jadval. C++ tilining tayanch turlari

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

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

Mantiqiy tur. Bu turdagi o‘zgaruvchi bool kalit so‘zi bilan e’lon qilinadi. U turdagi o‘zgaruvchi 1 bayt joy egallaydi va 0 (false, yolg‘on) yoki 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 qonuniyatlapi haqidagi fan. Uning asosini mulohazalar hisobi tashkil qiladi. *Mulohaza* - bu ixtiyoriy jumla bo‘lib, unga nisbatan rost yoki yolg‘on fikrni bildirish mumkin. Masalan «3>2», «5 - juft son», «Moskva-Ukraina roytaxti» va hakozo. 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 yuqopidagi jumlani rost eki yolg‘onligi haqida fikp bildipish qiyin.

Mulohazalapning postligi holatlapga bog‘liq pavishda 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 Jopj Bul sharafiga bool so‘zi bilan ifodalangan. Mantiqiy amallar «Bul algebrasi» deyiladi.

Mantiqiy mulohazalar ustida uchta amal aniqlangan:

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

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

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

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

1.3-jadval. Mantiqiy amallar uchun rostlik jadvali

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

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

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

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

Misollar.

```
int a=0,A=1; float abc=17.5;
double Ildiz;
bool Ok=true;
char LETTER='z';
void Mening_Funktsiyam();/* funksiya qaytaradigan
                            qiymat inobatga olinmaydi */
```

Turlangan o'zgarmaslar

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

Turlangan o'zgarmas e'lonida const kalit so'zi, undan keyin o'zgarmas turi va nomi, xamda albatta initsializatsiya qismi bo'ladi.

Misol tariqasida turlangan va literal o‘zgarmaslardan foydalangan holda radius berilganda aylana yuzasini hisoblaydigan programmani keltiramiz.

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

Programma bosh funksiyasining boshlanishida ikkita - `pi` va `Radius` o‘zgarmaslari e’lon qilingan. Aylana yuzasini aniqlovchi `Square` o‘zgarmas deb e’lon qilinmagan, chunki u programma bajarilishida o‘zgaradi. Aylana radiusini programma ishlashida o‘zgartirish mo‘ljallanmagan. shu sababli u o‘zgarmas sifatida e’lon qilingan.

Sanab o‘tiluvchi tur

Ko‘p miqdordagi, mantiqan bog‘langan o‘zgarmaslardan foydalanilganda sanab o‘tiluvchi turdan foydalanilgani 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 programmaning boshqa joylarida nomlash uchun ishlatilishi mumkin emas. Sanab o‘tiluvchi tur qo‘yidagi ko‘rinishga ega:

enum <sanab o‘tiladigan tur nomi> { $<nom_1>$ =$<qiymat_1>$,
$<nom_2>$ =$<qiymat_2>$, … $<nom_n>$ =$<qiymat_n>$ };

Bu erda, enum - kalit so‘z (inglizcha enumerate - sanamoq); <sanab o‘tiladigan tur nomi>- o‘zgarmaslar ro‘yxatining nomi; $<nom_i>$ - butun qiymatli konstantalarning nomlari; $<qiymat_i>$- shart bo‘lmagan initsializatsiya qiymati (ifoda).

Misol uchun hafta kunlari bilan bog‘liq masala yechishda hafta kunlarini dush (dushanba), sesh (seshanba), chor (chorshanba), paysh (payshanba), juma (juma), shanba (shanba), yaksh (yakshanba) o‘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 qadamdagacha aniqlangan bo‘lishi kerak:

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

O‘zgarmasni qiymatlari manfiy son bo‘lishi xam mumkin:

```
enum {minus2=-2,minus1,nul,bir};
```

Turni boshqa turga keltirish

C++ tilida bir turni boshqa turga keltirishning oshkor va oshkormas yo‘llari mavjud.

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

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

```
#include <iostream.h>
int main()
{
 int Integer_1=54;
 int Integer_2;
 float Floating=15.854;
 Integer_1=(int)Floating; // oshkor keltirish;
 Integer_2=Floating;      // oshkormas keltirish;
 cout<<"Yangi Integer(Oshkor): "<<Integer_1<<"\n";
 cout<<"Yangi Integer(Oshkormas): "<<Integer_2<<"\n";
 return 0;
}
```

Programma 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 echiladi.

Programma matni:

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

Programmaning

```
Belgini kiriting:
```

so‘roviga

```
A <enter>
```

amali bajarilsa, ekranga

```
'A'-belgi ASCCI kodi=65
```

satri chop etiladi.

3- bob. Ifodalar va operatorlar

Arifmetik amallar. Qiymat berish operatori

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

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

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

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

Quyidagi programma matnida ayrim amallarga misollar keltirilgan.

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

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

```
4    188    2   12   9    1    482  2
```

Ifoda tushunchasi

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

C++ tilidagi til ko'rsatmasiga misol:

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

Inkrement va dekrement amallari

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

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

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

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

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

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 sizeof amali yordamida aniqlanadi. Bu amalni o'zgarmasga, turga va o'zgaruvchiga qo'llanishi mumkin.

Quyida keltirilgan programmada kompyuterning platformasiga mos ravishda tayanch turlarining o'lchamlari chop qilinadi.

```
int main()
{
 cout<<"int turi o'lchami:"<<sizeof(int)<<"\n";
 cout<<"float turi o'lchami:"<<sizeof(float)<<"\n";
 cout<<"double     turi     o'lchami:"<<sizeof(double)
<<"\n";
 cout<<"char turi o'lchami:"<<sizeof(char)<<"\n";
 return 0;}
```

Razryadli mantiqiy amallar

Programma 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 bayt-ning razryadlaridan foydalanish ham mumkin. CHunki razryadlar faqat ikkita qiymatni - 0 va 1 sonlarini qabul qiladi. Bir baytda 8 razryad bo‘lgani uchun unda 8 ta bayroqni kodlash imkoniyati mavjud.

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

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

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

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

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

3.1-jadval. Bayt razryadlari ustida mantiqiy amallar

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

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

3.2-jadval. Razryadli mantiqiy amallarning bajarish natijalari

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

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

```
char q_taxtasi=0;
```

Bu erda 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 ulanganligi 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‘pay-tirish (bo‘lish) amali bilan ekvivalent bo‘lib va nisbatan tez bajariladi. Shuni e’tiborga olish kerakki, operand ishorali son bo‘lsa, u holda chapga surishda eng chapdagi ishora razryadi takrorlanadi (ishora saqlanib qoladi) va manfiy sonlar ustida bu amal bajarilganda matematika nuqtai-nazardan xato natijalar yuzaga keladi:

```
char B=-120;            // B=10001000₂=0x88₁₆
B=B<<2;                 // B=00100000₂=0x20₁₆=32₁₀
B=-120;                 // B=10001000₂=0x88₁₆
B=B>>3;                 // B=11110001₂=0xF1₁₆=-15₁₀
```

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

Taqqoslash amallari

C++ tilida qiymatlarni solishtirish uchun taqqoslash amallari aniqlangan (3.3-jadval). Taqqoslash amali binar amal bo‘lib, quyidagi ko‘rinishga ega:

<operand$_1$> <taqqoslash amali> < operand$_2$>

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.

3.3-jadval. Taqqoslash amallari va ularning qo‘llanishi

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

«Vergul» amali

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

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

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

Amallarning ustunliklari va bajarilish yo'nalishlari

An'anaviy arifmetikadagidek C++ tilida ham amallar ma'lum bir tartib va yo'nalishda bajariladi. Ma'lumki, matematik ifoda-larda bir xil ustunlikdagi (prioritetdagi) amallar uchrasa (masalan, qo'shish va ayirish), ular chapdan o'ngga bajariladi. Bu tartib C++ tilidagi 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 (opera-torlar), ularning ustunlik koeffitsientlari va bajarilish yo'nalishlari ($\Leftarrow$ - o'ngdan chapga, $\Rightarrow$ - chapdan o'ngga) keltirilgan.

3.4-jadval. Amallarning ustunliklari va bajarilish yo'nalishlari

Operator	Tavsifi	Ustunlik	Yo'nalish
::	Ko'rinish sohasiga ruxsat berish	16	$\Rightarrow$
[ ]	Massiv indeksi	16	$\Rightarrow$
( )	Funksiyani chaqirish	16	$\Rightarrow$
. ->	Struktura yoki sinf elementini tanlash	16	$\Rightarrow$
++	Postfiks inkrement	15	$\Leftarrow$
- -	Postfiks dekrement	15	$\Leftarrow$
++	Prefiks inkrement	14	$\Leftarrow$
- -	Prefiks dekrement	14	$\Leftarrow$
sizeof	O'lchamni olish	14	$\Leftarrow$
(<tur>)	Turga akslantirish	14	
~	Razryadli mantiqiy INKOR	14	$\Leftarrow$
!	Mantiqiy inkor	14	$\Leftarrow$
-	Unar minus	14	$\Leftarrow$

+	Unar plyus	14	⇐
&	Adresni olish	14	⇐
*	Vositali murojaat	14	⇐
new	Dinamik ob'ektni yaratish	14	⇐
delete	Dinamik ob'ektni yo'q qilish	14	⇐
casting	Turga keltirish	14	
*	Ko'paytirish	13	⇒
/	Bo'lish	13	⇒
%	Bo'lish qoldig'i	13	⇒
+	Qo'shish	12	⇒
-	Ayirish	12	⇒
>>	Razryad bo'yicha o'ngga surish	11	⇒
<<	Razryad bo'yicha chapga surish	11	⇒
<	Kichik	10	⇒
<=	Kichik yoki teng	10	⇒
>	Katta	10	⇒
>=	Katta yoki teng	10	⇒
==	Teng	9	⇒
!=	Teng emas	9	⇒
&	Razryadli VA	8	⇒
^	Razryadli istisno qiluvchi YOKI	7	⇒
|	Razryadli YOKI	6	⇒
&&	Mantiqiy VA	5	⇒
||	Mantiqiy YOKI	4	⇒
?:	Shart amali	3	⇐
=	Qiymat berish	2	⇐
*=	Ko'paytirish qiymat berish bilan	2	⇐
/=	Bo'lish qiymat berish bilan	2	⇐
%=	Modulli bo'lish qiymat berish bilan	2	⇐
+=	Qo'shish qiymat berish bilan	2	⇐
-=	Ayirish qiymat berish bilan	2	⇐
<<=	CHapga surish qiymat berish bilan	2	⇐
>>=	O'ngga surish qiymat berish bilan	2	⇐
&=	Razryadli VA qiymat berish bilan	2	⇐
^=	Razryadli istisno kiluvchi YOKI qiymat berish bilan	2	⇐
|=	Razryadli YOKI qiymat berish bilan	2	⇐
throw	Istisno holatni yuzaga keltirish	2	⇐
,	Vergul	1	⇒

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

Quyidagi programmada qavs yordamida amallarni bajarish tartibini o‘zgartirish ko‘rsatilgan.

```
#include <iostream.h>
int main()
{int x=0, y=0;
 int a=3, b=34, c=82;
 x=a*b+c;
 y=(a*(b+c));
 cout<<"x= "<<x<<'\n'<<"y= "<<y<<'\n';}
```

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

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

satrlari chop etiladi.

4- bob. Programma bajarilishini boshqarish

Operator tushunchasi

Programmalash tili operatorlari echilayotgan masala algorit-mini amalga oshirish uchun ishlatiladi. Operatorlar *chiziqli* va *boshqaruv operatorlariga* bo'linadi. Aksariyat holatlarda operatorlar «nuqta-vergul» (';') belgisi bilan tugallanadi va u kompilyator tomonidan alohida operator deb qabul qilinadi (for operatorining qavs ichida turgan ifodalari bundan mustasno). Bunday operator *ifoda operatori* deyiladi. Qiymat berish amallari guruhi, xususan, qiymat berish operatorlari ifoda operatorlari hisoblanadi:

```
I++; --j; k+=I;
```

Programma tuzish amaliyotida bo'sh operator - ';' ishlatiladi. Garchi bu operator hech nima bajarmasa ham, hisoblash ifodalarini til qurilmalariga mos kelishini ta'minlaydi. Ayrim hollarda yuzaga kelgan «boshi berk» holatlardan chiqib ketish imkonini beradi.

O'zgaruvchilarni e'lon qilish ham operator hisoblanadi va ularga *e'lon operatori* deyiladi.

Shart operatorlari

Oldingi bobda misol tariqasida keltirilgan programmalarda amallar yozilish tartibida ketma-ket va faqat bir marta bajariladigan holatlar, ya'ni chiziqli algoritmlar keltirilgan. Amalda esa kamdan-kam masalalar shu tariqa echilishi mumkin. Aksariyat masalalar yuzaga keladigan turli holatlarga bog'liq ravishda mos qaror qabul qilishni (yyechimni) talab etadi. C++ tili programmaning alohida bo'laklarining bajarilish tartibini boshqarishga imkon beruvchi qurilmalarning etarlicha katta majmuasiga ega. Masalan, programma bajarilishining birorta qadamida qandaydir shartni tekshirish natijasiga ko'ra boshqaruvni programmaning u yoki bu bo'lagiga uzatish mumkin (tarmoqlanuvchi algoritm). Tarmoqlanishni amalga oshirish uchun shartli operatordan foydalaniladi.

if operatori

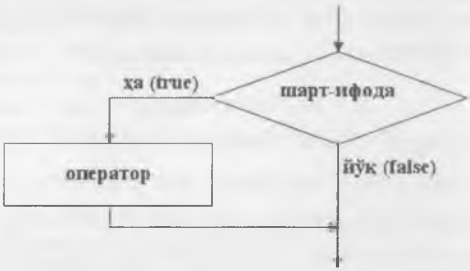
if operatori qandaydir shartni rostlikka tekshirish natijasiga ko'ra programmada tarmoqlanishni amalga oshiradi:

if (<shart>)<operator>;

Bu erda <shart> har qanday ifoda bo'lishi mumkin, odatda u taqqoslash amali bo'ladi.

Agar shart 0 qiymatidan farqli yoki rost (true) bo'lsa, <operator> bajariladi, aks holda, ya'ni shart 0 yoki yolg'on (false) bo'lsa, hech qanday

amal bajarilmaydi va boshqaruv if operatoridan keyingi operatorga o'tadi (agar u mavjud bo'lsa). Ushbu holat 4.1-rasmda ko'rsatilgan.



4.1-rasm. if() shart operatorining blok sxemasi

C++ tilining qurilmalari operatorlarni blok ko'rinishida tashkil qilishga imkon beradi. *Blok* - '{' va '}' belgi 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.

Quyida keltirilgan programmada if operatoridan foydalanish ko'rsatilgan.

```
#include <iostream.h>
int main()
{
 int b;
 cin>>b;
 if (b>0)
 {            // b>0 shart bajarilgan holat
 ...
 cout<<"b - musbat son";
 ...
 }
 if (b<0)
 cout<<"b - manfiy son"; // b<0 shart bajarilgan
holat
 return 0;
}
```

Programma bajarilishi jarayonida butun turdagi b o'zgaruvchi e'lon qilinadi va uning qiymati klaviaturadan o'qiladi. Keyin b qiymatini 0 sonidan kattaligi tekshiriladi, agar shart bajarilsa (true) , u holda '{' va '}' belgilar ichidagi operatorlar bajariladi va ekranga "b - musbat son" xabari

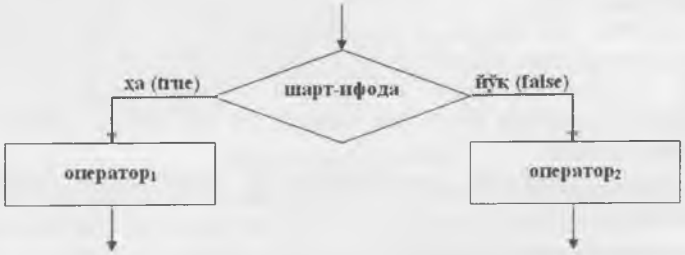
chiqadi. Agar shart bajarilmasa, bu operatorlar cheklab o‘tiladi. Navbatdagi shart operatori b o‘zgaruvchi qiymati manfiylikka tekshiradi, agar shart bajarilsa, yagona cout ko‘rsatmasi bajariladi va ekranga “b - manfiy son” xabari chiqadi.

if - else operatori

Shart operatorining if - else ko‘rinishi quyidagicha:

if (<shart-ifoda>) <operator$_1$>; else <operator$_2$>;

Bu erda <shart-ifoda> 0 qiymatidan farqli yoki true bo‘lsa, <operator$_1$>, aks holda <operator$_2$> bajariladi. if-else shart operatori mazmuniga ko‘ra algoritmning tarmoqlanuvchi blokini ifodalaydi: <shart-ifoda> - shart bloki (romb) va <operator$_1$> blokning «ha» shoxiga, <operator$_2$> esa blokning «yo‘q» shoxiga mos keluvchi amallar bloklari deb qarash mumkin (4.2-rasm).



4.1-rasm. if(); else shart operatorining blok sxemasi

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

```
#include <iostream.h>
#include <math.h>
int main()
{
 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;
  return 0;
  }
 else
 {cout << "Tenglama ikkita ildizga ega: ";
  x1=(-b+sqrt(D))/(2*a);
  x2=(-b-sqrt(D))/(2*a);
  cout<<"\nx1= "<<x1;
  cout<<"\nx2= "<<x2;
 }
 return 0;
}
```

Programma bajarilganda, birinchi navbatda tenglama koeffisientlari - 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 operator-lar bajariladi va ekranga "Tenglama haqiqiy ildizlarga ega emas" xabari chiqadi va programma 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 va programma shu erda o'z ishini tugatadi, 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 programma o'z ishini tugatadi.

O'z navbatida <operator$_1$> va <operator$_2$> ham shartli operator bo'lishi mumkin. Ifodadagi har bir else kalit so'zi, oldindagi 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 programma bo‘lagini keltirishimiz mumkin:

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

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
```

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.h>
int main()
{
 unsigned int n,a3,a2,a1,a0; // n=a3a2a1a0 ko‘rinishida
 cout<<"\nn - qiymatini kiriting: ";
 cin>>n;
 if(n<1000 || n>9999)
 {
  cout<<"Kiritilgan son 4 xonali emas!";
```

```
  return 1;
 }
 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;
}
```

Programma ishorasiz butun son kiritishni taklif qiladi. Agar kiritilgan son 4 xonali bo'lmasa (n<1000 yoki n>9999), bu haqda xabar beriladi va programma 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'qligi qarab mos javob chop qilinadi.

?: shart amali

Agar tekshirilayotgan shart nisbatan sodda bo'lsa, shart amalining «?:» ko'rinishini ishlatish mumkin:

<shart ifoda> ? <ifoda$_1$> : <ifoda$_2$>;

Shart amali if shart operatoriga o'xshash holda ishlaydi: agar <shart ifoda> 0 qiymatidan farqli yoki true bo'lsa, <ifoda$_1$>, aks holda <ifoda$_2$> bajariladi. Odatda ifodalar qiymatlari birorta o'zgaruvchiga o'zlashtiriladi.

Misol tariqasida ikkita butun son maksimumini topish masalasini ko'raylik.

```
#include <iostream.h>
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;
 s=a>b?a:b;
 cout<<"\nSonlar maksimumi: "<<c;
 return 0;
}
```

Programmadagi shart operatori qiymat berish operatorining tarkibiga kirgan bo'lib, a o'zgaruvchining qiymati b o'zgaruvchining qiymatidan kattaligi tekshiriladi. Agar shart rost bo'lsa, s o'zgaruvchisiga a

o‘zgaruvchi qiymatini, aks holda b o‘zgaruvchining qiymatini o‘zlashtiradi va s o‘zgaruvchisining qiymati chop etiladi.

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

```
#include <iostream.h>
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 tarmoqlanish 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 qiladi: 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 boshqa-ruv 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.

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 programma o‘z ishini

tarmoqlanuvchi operatordan keyingi operatorlarni bajarish bilan davom ettiradi, aks holda "Jarayon tugadi!" javobi beriladi va programma o'z ishini tugatadi. Bunda, foydalanuvchining 'u' yoki 'U' javoblari jarayonni davom ettirishni bildiradi, boshqa belgilar esa jarayonni tugatishni anglatadi.

```
#include <iostream.h>
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 tygadi!\n";
 return 0;
 }
 ... // jarayon
 return 0;
}
```

Umuman olganda, tarmoqlanuvchi operatorda 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 tarmoqlanuvchi 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 tygadi!
```

xabarlari chiqadi va programma o'z ishini tugatadi (return operatorining bajarilishi natijasida).

Tarmoqlanuvchi operator sanab o'tiluvchi turdagi o'zgarmaslar bilan birgalikda ishlatilganda samara beradi. Quyidagi programmada ranglar gammasini toifalash masalasi echilgan.

```
#include <iostream.h>
int main()
{
 enum Ranglar{Qizil,Tuq_sariq,Sariq,Yashil,
              Kuk,Zangori,Binafsha};
 Ranglar Rang;
 //...
 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;
}
```

Programma bajarilishida boshqaruv tarmoqlanuvchi operatorga kelganda, Rang qiymati Qizil yoki To'q_sariq yoki Sariq bo'lsa, "Issiq gamma tanlandi" xabari, agar Rang qiymati Yashil yoki Ko'k yoki Zangori yoki Binafsha bo'lsa, ekranga "Sovuq gamma tanlandi" xabari, agar Rang qiymati sanab o'tilgan qiymatlardan farqli bo'lsa, ekranga "Kamalak bunday rangga ega emas" xabari chop etiladi va programma 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 programma ishida xatolik ro'y berishi mumkin:

```
//...
int k=0,n=0;
cin >>n;
switch (n)
{
 int i=10;//xato,bu operator hech qachon 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. Quyida sanab o'tiluvchi turlar va shu turdagi o'zgaruvchilar e'lon qilingan:

```
enum
 Birlik {desimetr,kilometr,metr,millimetr,santimetr};
float x; Birlik r;
```

Berilgan r birlikda berilgan x o'zgaruvchisining qiymati metrlarda chop qilinsin.

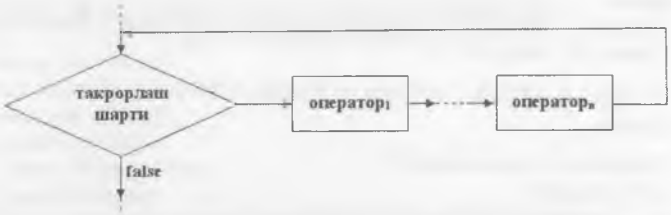
```
#include <iostream.h>
int main()
{
 enum
 Birlik{desimetr,kilometr,metr,millimetr, santimetr};
 float x,y;
 Birlik r;
 cout<<"Uzunlikni kiriting: x=";
 cin>>x;
 cout<<" Uzunlik birliklari\n";
 cout<<" 0- desimetr\n";
 cout<<" 1- kilometr\n";
 cout<<" 2- metr\n";
 cout<<" 3- millimetr\n";
 cout<<" 4- santimetr\n";
 cout<<" Uzunlikni birligini tanlang: r=";
 cin>>r;
 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;
}
```

Takrorlash operatorlari

Programma bajarilishini boshqarishning boshqa bir kuchli mexanizmlaridan biri - takrorlash operatorlari hisoblanadi.

Takrorlash operatori «takrorlash sharti» deb nomlanuvchi ifodaning rost qiymatida programmaning ma'lum bir qismidagi operatorlarni

(takrorlash tanasini) ko‘p marta takror ravishda bajaradi (itarativ jarayon) (4.2-rasm).



4.2-rasm. Takrorlash operatorining blok sxemasi

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 takrorlashlari) yoki takrorlash tanasidagi operatorlari bir marta bajarilgandan keyin tekshirilishi mumkin (do-while).

Takrorlash operatorlari ichma-ich joylashgan bo‘lishi mumkin.

for takrorlash operatori

for takrorlash operatorining sintaksisi qo‘yidagi ko‘rinishga ega:

for (<ifoda$_1$>; <ifoda$_2$>;<ifoda$_3$>) <operator yoki blok>;

Bu operator o‘z ishini <ifoda$_1$> ifodasini bajarishdan boshlaydi. Keyin takrorlash qadamlari boshlanadi. Har bir qadamda <ifoda$_2$> bajariladi, agar natija 0 qiymatidan farqli yoki true bo‘lsa, takrorlash tanasi - <operator yoki blok> bajariladi va oxirida <ifoda$_3$> bajariladi. Agar <ifoda$_2$> qiymati 0 (false) bo‘lsa, takrorlash jarayoni to‘xtaydi va boshqaruv takrorlash operatoridan keyingi operatorga o‘tadi. Shuni qayd qilish kerakki, <ifoda$_2$> ifodasi vergul bilan ajratilgan bir nechta ifodalar birlashmasidan iborat bo‘lishi mumkin, bu holda oxirgi ifoda qiymati takrorlash sharti hisoblanadi. Takrorlash tanasi sifatida bitta operator, jumladan bo‘sh operator bo‘lishi yoki operatorlar bloki kelishi mumkin.

Misol uchun 10 dan 20 gacha bo‘lgan butun sonlar yig‘indisini hisoblash masalasini ko‘raylik.

```
#include <iostream.h>
int main()
{
 int Summa=0;
 for (int i=10; i<=20; i++)
```

```
    Summa+=i;
    sout<<"Yig'indi=" <<Summa;
    return 0;
}
```

Programmadagi takrorlash operatori o‘z ishini, i takrorlash parametriga (takrorlash sanagichiga) boshlang‘ich qiymat - 10 sonini berishdan boshlaydi va har bir takrorlash qadamidan (itaratsiyadan) keyin qavs ichidagi uchinchi operator bajarilishi hisobiga uning qiymati bittaga oshadi. Har bir takrorlash qadamida takrorlash tanasidagi operator bajariladi, ya’ni Summa o‘zgaruvchisiga i qiymati qo‘shiladi. Takrorlash sanagichi i qiymati 21 bo‘lganda “i<=20” takrorlash sharti false (0-qiymati) bo‘ladi va takrorlash tugaydi. Natijada boshqaruv takrorlash operatoridan keyingi cout operatoriga o‘tadi va ekranga yig‘indi chop etiladi.

Yuqorida keltirilgan misolga qarab takrorlash operatorlarining qavs ichidagi ifodalariga izoh berish mumkin:

<ifoda$_1$> - 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» (C++ Builder kompilyatori uchun);

<ifoda$_2$> - 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;

<ifoda$_3$> - odatda takrorlash sanagichining qiymatini oshirish (kamaytirish) uchun xizmat qiladi yoki unda takrorlash shartiga ta’sir qiluvchi boshqa amallar bo‘lishi mumkin.

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

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

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

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

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

for operatorida takrorlash tanasi bo‘lmasligi ham mumkin. Masalan, programma bajarilishini ma’lum bir muddatga «to‘xtab» turish zarur

bo‘lsa, bunga takrorlashni hech qanday qo‘shimcha ishlarni bajarmasdan amal qilishi orqali erishish mumkin:

```
#include <iostream.h>
 int main()
 {int delay;
  ...
  for (delay=5000; delay>0; delay--); // bo‘sh operator
  ...
  return 0;}
```

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

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

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

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

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

Masala. 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‘rishimiz mumkin:

```
#include <iostream.h>
int main()
{
 unsigned char a2,a1,a0; // uch xonali son raqamlari
 for (a2='1';a2<='9';a2++)//sonning 2-raqami
  for(a1='0';a1<='9';a1++)//sonning 1-raqami
   for(a0='0';a0<='9';a0++)//sonning 0-raqami
```

```
// raqamlarni o'zaro teng emasligini tekshirish
    if(a0!=a1 && a1!=a2 && a0!=a2) //o'zaro teng emas
     cout<<a2<<a1<<a0<<'\n';
 return 0;
}
```

Programmada uch xonali sonning har bir raqami takrorlash operatorlarining parametrlari sifatida hosil qilinadi. Birinchi, tashqi takrorlash operatori bilan 2-xonadagi raqam (a2 takrorlash parametri) hosil qilinadi. Ikkinchi, ichki takrorlash operatorida (a1 takrorlash parametri) son ko'rinishining 1-xonasidagi raqam va nihoyat, unga nisbatan ichki bo'lgan a0 parametrli takrorlash operatorida 0-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.

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

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

```
while(1); // cheksiz takrorlash
```

Xuddi for operatoridek, ',' yordamida <ifoda> da bir nechta amallar sinxron ravishda bajarish mumkin. Masalan, son va uning kvadratlarini chop qiladigan programmada ushbu holat ko'rsatilgan:

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

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

```
#include <iosteam.h>
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)      //son10 & 00001000
   cout<<'1';
   else cout<<'0';
   son10=son10<<1;  //razryadlarni 1 o'rin chapga su-
rish
   sanagich--;
  }
  cout<<'\n';
  cout<<"Jarayonni to'xtasin(0),davom etsin(1):_ ";
  cin >> jarayon;
  sanagich=4;
 }
 return 0;
}
```

Programmada 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 natija 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 programma kodi yozishga yana bir misol bu - ikkita natural sonlarning eng katta umumiy bo‘luvchisini (EKUB) Evklid algoritmi bilan topish masalasini keltirishimiz mumkin:

```
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‘rsatma vositasida a o‘zgaruvchisining 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 - bu takrorlashni 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.h>
int main()
{
  char javob;
```

```
    do
    {
     ...    // programma tanasi
    cout<<"Jarayonni to'xtatish (N):_ ";
    cin>>javob;
    } while(javob !=N)
   return 0;
}
```

Programma toki "Jarayonni to'xtatish (N):_ " so'roviga 'N' javobi kiritilmaguncha davom etadi.

Bu operator ham cheksiz takrorlanishi mumkin:

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

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

```
#include <iostream.h>
int main()
{
 unsigned int Pul; //Pul- kiritiladigan pul miqdori
 unsigned n3,m5;   //n-3 so'mliklar,m-5 so'mliklar
soni
 bool   xato=false;   //Pul   qiymatini   kiritishdagi
xatolik
 do
 {
  if(xato)cout<<"Pul qiymati 7 dan kichik!";
  xato=true; // keyingi takrorlash xato hisoblanadi
  cout<<"\nPul qiymatini kiriting (>7): ";
  cin>>Pul;
 }
  while(Pul<=7);//toki 7 katta son kiritilguncha
 n3=0;          //birorta ham 3 so'mlik yo'q
 do
 {
  m5=0;         //birorta ham 5 so'mlik yo'q
  do
  {
   if (3*n3+5*m5==Pul)
   cout<<n3<<" ta 3 so'mlik+"<<m5<<" ta 5 so'mlik\n";
   m5++;      // 5 so'mliklar bittaga oshiriladi
  } while(3*n3+5*m5<=Pul);
  n3++;       //3 so'mliklar bittaga oshiriladi
```

```
    } while(3*n3<=Pul);
    return 0;
}
```

Programma pul qiymatini kiritishni so‘raydi (Pul o‘zgaruvchi-siga). 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 + 6 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 etkazmasdan 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 programmada ikkita ichma-ich joylashgan takrorlash operatoridan foydalangan holda foydalanuvchi tomonidan kiritil-gan qandaydir sonni 3 va 7 sonlariga nisbatan qanday oraliqqa tushishi aniqlanadi. Tashqi takrorlashda “Son kiriting (0- to’xtash):_” so‘rovi beriladi va javob 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, programma o‘z ishini tugatadi.

```
#include <iosteam.h>
```

```
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 cheksiz for 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.h>
int main()
```

```
{
 unsigned int Ai_1=0,Ai;
 cout<<"Sonlar ketma-ketligini kiriting"
 cout<<(0-tugash alomati):\n ";
 cin>>Ai;          // ketma-ketlikning birinchi hadi
 while(Ai)
 {
  Ai_1=Ai;
  cin>>Ai;         // navbatdagi had
  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;
}
```

Programma ishga tushganda, boshda 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 operatorlari 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-ketlikni 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 programma matnini keltiramiz.

```
#include <iosteam.h>
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 programmada qo'yilgan masala ichma-ich joylashgan ikkita takrorlash operatorlari yordamida echilgan. 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, programmaning 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 bajariladi. 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;
i++; if(i)goto m;
```

```
int j;
m:j+=i;
```

operatorlarining bajarilishi xatolikka olib keladi, chunki j e'lon qilinmay qoladi.

Shartsiz o'tish operatori programmani 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. Shu sababli, imkon qadar goto operatoridan foydalanmaslik kerak, ishlatilgan taqdirda ham quyidagi qoidaga amal qilish zarur: «*blok ichiga, if...else va switch operatorlari ichiga, hamda takrorlash operatorlari tanasiga tash-qaridan kirish mumkin emas*».

Garchi, nishon yordamida programmaning 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.

Masalan:

```
...
goto B;       \\ xato
float x=0.0;
goto  B;      \\ to'g'ri
{ int n=10;x=n*x+x; }
B: cout << "x="<<x;
...
```

Xususan, nishon yordamida ichki blokdan tashqi blokka va tashqi blokdan ichki blokka o'tishga C++ tili ruxsat beradi:

```
{...
 goto ABC:
 ...
 {int i=15;
   ...
   ABC:
   ...
   goto XYZ;
   int y=10;
   ...
   XYZ:
   ...
   goto KLM;
   ... }
...
 int k=0;
 ...
 KLM:
```

```
  ...
}
```

Lekin, yuqorida keltirilgan misoldagi barcha o‘tishlar mazmunan xato hisoblanadi.

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

```
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;
}
```

Programmadagi 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.

Shuni qayd etish kerakki, bu masalani takrorlash operatorlari yordamida bajarish ancha samarali hisoblanadi.

5-bob. Funksiyalar

Programma ta'minotini yaratish amalda murakkab jarayon hisoblanadi. Programma tuzuvchi programma kompleksini bir butunlikdagi va uning har bir bo'lagining ichki mazmunini va ularning sezilmas farqlarini hisobga olishi kerak bo'ladi.

Programmalashga tizimli yondoshuv shundan iboratki, program-ma tuzuvchi oldiga qo'yilgan masala oldindan ikkita, uchta va undan ortiq nisbatan kichik masala ostilarga bo'linadi. O'z navbatida bu masala ostilari ham yana kichik masala ostilariga bo'linishi mumkin. Bu jarayon toki mayda masalalarni oddiy standart amallar yordamida yechish mumkin bo'lguncha davom etadi. Shu yo'l bilan masalani dekompozitsiyalash amalga oshiriladi.

Ikkinchi tomondan, programmalashda shunday holatlar kuzatila-diki, unda programmaning turli joylarida mazmunan bir xil algoritmlarni bajarishga to'g'ri keladi. Algoritmning bu bo'laklari asosiy echilayotgan masaladan ajratib olingan qandaydir masala ostini yechishga mo'ljallangan bo'lib, etarlicha mustaqil qiymatga (natijaga) egadir. Misol uchun quyidagi masalani ko'raylik:

Berilgan $a_0,a_1,...,a_{30}$, $b_0,b_1,...,b_{30}$, $c_0,c_1,...,c_{30}$ 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 programmada har bir ifodani (masala osti) 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 s 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 sistemasini Gauss, Kramer, Zeydel usullarining birortasi bilan yechish talab qilinsin. U holda asosiy programmani quyidagi bo'laklarga bo'lish maqsadga muvofiq bo'lar edi: tenglama koeffitsentlarini kiritish bo'lagi, yechish usulini tanlash bo'lagi, Gauss, Kramer, Zeydel usullarini amalga oshirish uchun alohida bo'laklar, natijani chop qilish bo'lagi. Har bir bo'lak uchun o'z funksiyalar majmuasi yaratib, zarur bo'lganda ularga bosh funksiya ta-

nasidan murojaatni amalga oshirish orqali bosh masala yechish samarali hisoblanadi.

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

Funksiya bu - C++ tilida masala yechishdagi kalit elementlaridan biridir.

Funksiya parametrlari va argumentlari

Programmada ishlatiladigan har qanday funksiya e'lon qili-nishi kerak. Odatda funksiyalar e'loni sarlavha fayllarda e'lon qilinadi va #include direktivasi yordamida programma 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 erda <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 programma matnida funksiya aniqlanishi uni chaqiradigan funksiyalar matnidan oldin yozilgan bo'lsa. Lekin bu holat yaxshi uslub hisoblanmaydi, ayniqsa o'zaro bir-biriga murojaat qiluvchi funksiyalar-ni 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 programmada ishlatilmaydigan bo'lsa, funksiyadan chiqish uchun parametrsiz return operatori ishlatilishi mumkin yoki umuman return

ishlatilmaydi. Oxirgi holda funksiyadan chiqish - oxirgi yopiluvchi qavsga etib kelganda ro'y beradi.

Funksiya programmaning 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 programmada funksiya ma'lum bir ishni amalga oshirish uchun chaqiriladi. Funksiyaga murojaat qilganda, u qo'yilgan masalani echadi va o'z ishini tugatishida qandaydir qiymatni natija sifatida qaytaradi.

Funksiyani chaqirish uchun uning nomi va undan keyin qavs ichida argumentlar ro'yxati beriladi:

<funksiya nomi>(<argument$_1$>, <argument$_2$>,..., <argument$_n$ >);

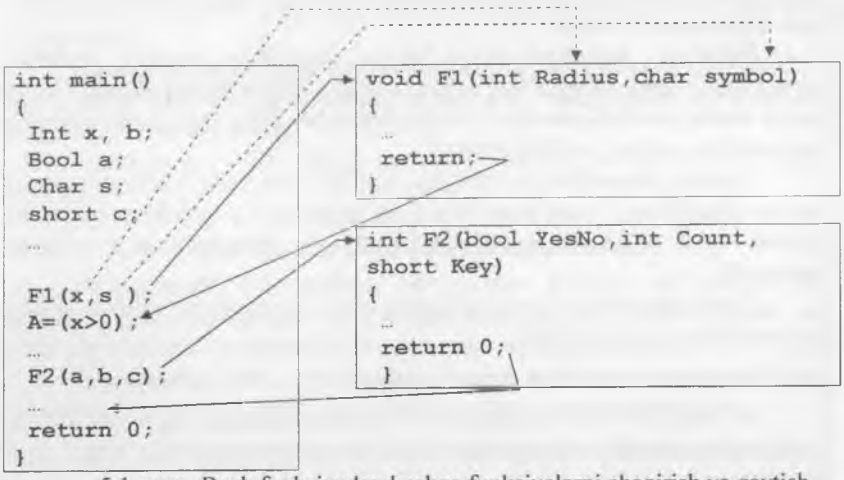
Bu erda har bir <argument> - funksiya tanasiga uzatiladigan va keyinchalik hisoblash jarayonida ishlatiladigan o'zgaruvchi, ifoda yoki o'zgarmasdir. Argumentlar ro'yxati bo'sh bo'lishi mumkin.

Funksiyalar ham o'z tanasida boshqa funksiyalarni, o'zini ham chaqirishi mumkin. O'z tanasida o'zini chaqiradigan funksiyalarga *rekursiv funksiyalar* deyiladi.

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

5.1- rasmda bosh funksiyadan boshqa funksiyalarni chaqirish va ulardan qaytish sxemasi ko'rsatilgan.

Programma main() funksiyasini bajarishdan boshlanadi va «f1(x,y);» - funksiya chaqirishgacha davom etadi va keyinchalik boshqaruv f1(x,y) funksiya tanasidagi amallarni bajarishga o'tadi. Bunda Radius parametrining qiymati sifatida funksiya x o'zgaruvchi qiymatini, Symbol parametri sifatida u o'zgaruvchisining qiymati ishlatiladi. Funksiya tanasi return operatorigacha bajariladi. return operatori boshqaruvni main() funksiyasi tanasidagi f1() funksiyasi chaqirilgan operatordan keyingi operatorga o'tishni ta'minlaydi, ya'ni funksiyadan qaytish ro'y beradi. Shundan keyin main() funksiya-si operatorlari bajariliShda davom etadi va «f2(a,b,c);» - funksiya chaqirishi orqali boshqaruv f2() funksiya tanasiga o'tadi va hisoblash jarayonida mos ravishda YesNo sifatida a o'zgaruvchisining, sount sifatida b o'zgaruvchicining va key sifatida s o'zgaruvchisining qiymatlari ishlatiladi. Funksiya tanasidagi return operatori yoki oxirgi operator bajargandan keyin avtomatik ravishda bosh funksiyaga qaytish amalga oshiriladi.



5.1-rasm. Bosh funksiyadan boshqa funksiyalarni chaqirish va qaytish

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

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

Bu erda 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, chaqirish 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);
// funksiyalarni chaqirish
result=Mening_funksiyam(Varb1,3.14);
symb=Belgini_uqish();
bitni_urnatish(3);
Amal_yoq(2,Smb1);
// 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 programmadagi o‘rnini yanada tushunarli bo‘lishi uchun son kvadratini hisoblash masalasida funksiyadan foydalanishni ko‘raylik.

Funksiya prototipini sarlavha.h sarlavha faylida joylashtiramiz:

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

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

```
#include <iostream.h>
#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 programma matniga yozish orqali ham hal qilish mumkin:

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

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

Masala. Ikkita tub son «egizak» deyiladi, agar ular bir-biridan 2 farq qilsa (masalan, 41 va 43 sonlari). Berilgan natural n uchun [n..2n] kesmadagi barcha «egizak» sonlar juftliklari chop etilsin. Masalani yechish uchun berilgan k conini tub son yoki yo‘qligi aniqlovchi 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.

Programma 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 coni uchun 100 kiritilsa, programma 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 funksiya chaqirilganda 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:

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

Birinchi chaqiruvda 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 programmani ko'raylik. Qo'yilgan masalani yechishda sonni darajaga oshirish funksiyasi - pow() va suzuvchi nuqtali uzun sondan modul olish fabsl() funksiyasidan foydala-niladi. Bu funksiyalar prototipi «math.h» sarlavha faylida joylashgan (3-ilova qarang):

```
#include <iostream.h>
#include <math.h>
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';
}
```

Programmada sonni turli aniqlikda (Aniqlik parametri qiymati orqali) chop etish uchun har xil variantlarda Chop_qilish( ) funksiyasi chaqirilgan. Programma ishlashi natijasida ekranda 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 programma 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 programma 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 programma 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'lma-gan bir xil nomdagi lokal o'zgaruvchilarni ishlatish mumkin.

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

```
#include <iostream.h>
// funksiya prototipi
int sum(int a;int b);
int main()
{
 // lokal o'zgaruvchilar
 int x=r;
```

```
 int y=4;
 cout<<sum(x, y);
 return 0;
}
int sum(int a,int b)
{
 // lokal o'zgaruvchi
 int x=a+b;
 return x;
}
```

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

```
#include <iostream.h>
int f1(); int f2();
int main()
{
 cout<<f1()<<" "<<f2()<<endl;
 return 0;
}
int f1()
{
 return x;// kompilyasiya xatosi ro'y beradi
}
int x=10; // global o'zgaruvchi e'loni
int f2(){ return x*x;}
```

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

Programma matnida global o‘zgaruvchilarni ular e’lonidan keyin yozilgan ixtiyoriy funksiyada ishlatish mumkin. Shu sababli, global o‘zgaruvchilar programma 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 programma tuzuvchilar imkon qadar global o‘zgaruvchilarni ishlatmaslikka harakat qilishadi, chunki bunday o‘zgaruvchilar qiymatini programmaning ixtiyoriy joyidan o‘zgartirish xavfi mavjudligi sababli programma ishlashida mazmunan xatolar yuzaga kelishi mumkin. Bu fikrimizni tasdiqlovchi programmani ko‘raylik.

```
# include <iostream.h>
// global o'zgaruvchi e'loni
```

```
int test=100;
void Chop_qilish(void );
int main()
{
 //lokal o'zgaruvchi e'loni
 int test=10;
 //global o'zgaruvchi chop qilish funksiyasini
chaqirish
 Chop_qilish();
 sout<<"Lokal o'zgaruvchi: "<<test<<'\n';
 return 0;
}
void Chop_qilish(void)
{
 cout<<"Global o'zgaruvchi: "<<test<<'\n';
}
```

Programma 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. Programmada, 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 programmaga qaytilgandan keyin, main() funksiyasidagi lokal test o‘zgaruvchisi global test o‘zgaruvchisini «berkitadi» va lokal test o‘zgaruvchini qiymati chop etiladi. Programma 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 «berkitadi» va bu joydan global o‘zgaruvchiga murojat qilish imkoni bo‘lmay qoladi. C++ tilida bunday holatlarda ham global o‘zgaruvchiga murojat qilish imko-niyati 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.h >
//global o'zgaruvchi e'loni
int uzg=5;
int main()
{
```

```
//lokal o'zgaruvchi e'loni
 int uzg=70;
 //lokal o'zgaruvchini chop etish
 cout<<uzg<<'/n';
 //global o'zgaruvchini chop etish
 cout<<::uzg <<'/n';
 return 0;
}
```

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

Xotira sinflari

O'zgaruvchilarning ko'rinish sohasi va amal qilish vaqtini aniqlovchi o'zgaruvchi modifikatorlari mavjud (5.1-jadval).

5.1-jadval. O'zgaruvchi modifikatorlari

Modifikator	Qo'llanishi	Amal qilish sohasi	YAshash davri
auto	lokal	blok	vaqtincha
register	lokal	blok	vaqtincha
exteru	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.h>
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 yo'q bo'lib ketadi.

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 programma 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.h >
int main()
{
 register int Reg;
 ...
 return 0;
}
```

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

Tashqi o‘zgaruvchilar. Agar programma 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 erda extern modifikatori bilan e’lon qilinishi kerak bo‘ladi. extern modifikatori o‘zgaruvchini boshqa faylda e’lon qilinganligini bildiradi. Tashqi o‘zgaruvchilar ishlatilgan prog-rammani ko‘raylik.

```
//Sarlavha.h faylida
void Bayroq_Almashsin(void);
// modul_1.cpp faylida
bool Bayroq;
void Bayroq_Almashsin(void){Bayroq=!Bayroq;}
// masala.cpp faylida
#include < iostream.h>
#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.srr 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 programma 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.h >
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 erda asosiy funksiyadan counter statik o'zgaruvchiga ega Sanagicht() funksiyasi 30 marta chaqiriladi. Funksiya birinchi marta chaqirilganda 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.

Programma matni:

```
#include<iostream.h>
#include<math.h>
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;
}
```

Programmada 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.

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

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

volatile sinfi o'zgaruvchilari. Agar programmada o'zgaruvchini birorta tashqi qurilma yoki boshqa programma bilan bog'lash uchun ishlatish zarur bo'ladigan bo'lsa, u volatile modifikatori bilan e'lon

qilinadi. 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;
```

Misoldan ko‘rinib turibdiki, volatile modifikatorli o‘zgarmas ham e’lon qilinishi mumkin.

Nomlar fazosi

Ma’lumki, programmaga qo‘shilgan sarlavha fayllarida e’lon qilingan identifikator va o‘zgarmaslar kompilyator tomonidan yagona global nomlar fazosiga kiritiladi. Agar programma ko‘p miqdordagi sarlavha fayllarni ishlatsa va undagi identifikatorlar (funksiya nomlari va o‘zgaruvchilar nomlari, sinflar nomlari va hakozalar) va o‘zgarmaslar nomlari turli programma 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 programmada 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;
 void Mening_funksiyam(char belgi);
}
```

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

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

Agar programma matnida konkret nomlar fazosiga nisbatan ko‘p murojaat qilinadigan bo‘lsa using namespace qurilmasini ishlatish orqali yozuvni soddalashtirish mumkin:

using namespace <nomlar fazosi nomi>;

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_functsiyam('A');
```

Programma 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 namespace std;
using namespace Shaxsiy_nomlar::x;
```

ko‘rsatmasi x 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 Functsiya(char Bayroq);
}

void Nomlar_fazosi::Functsiya(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;
```

Joylashtiriladigan (inline) funksiyalar

Kompilyator ishlashi natijasida har bir funksiya mashina kodi ko‘rinishida bo‘ladi. Agar programmada funksiyani chaqirish 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 chaqiriladigan funksiya hajmi unchalik katta bo‘lmagan hollarda, kompilyatorga funksiyani chaqirish kodi o‘rniga funksiya tanasini o‘zini 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 programma kodi yuzaga keladi.

Funksiya kodi joylashtiriladigan programmaga misol.

```
#include <iostream.h>
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 programma kodini hosil qilishda Summa() funksiyasi chaqirilgan joylarga uning tanasidagi buyruqlar joylashtiriladi.

Rekursiv funksiyalar

Yuqorida qayd qilingandek *rekursiya* deb funksiya tanasida shu funksiyaning o‘zini chaqirishiga aytiladi. Rekursiya ikki xil bo‘ladi:

1) *oddiy* - agar funksiya o‘z tanasida o‘zini chaqirsa;

2) *vositali* - agar birinchi funksiya ikkinchi funksiyani chaqirsa, ikkinchisi esa o‘z navbatida birinchi funksiyani chaqirsa.

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:

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

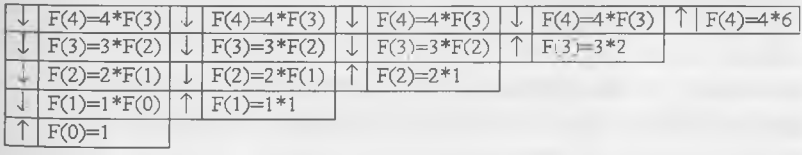
Ko‘rinib turibdiki, navbatdagi qiymatni hisoblash uchun funksiyaning «oldingi qiymati» ma’lum bo‘lishi kerak. C++ tilida rekursiya matematikadagi rekursiyaga o‘xshash. Buni yuqoridagi misollar uchun tuzilgan funksiyalarda ko‘rish mumkin. Faktorial uchun:

```
long F(int n)
{
 if(!n) 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) return 1;
 else return x*Butun_Daraja(x,n-1);
}
```

Agar faktorial funksiyasiga n>0 qiymat berilsa, quyidagi holat ro‘y beradi: shart operatorining else shoxidagi qiymati (n qiymati) stekda eslab qolinadi. Hozircha qiymati noma’lum n-1 faktorialni hisoblash uchun shu funksiyaning o‘zi n-1 qiymati bilan 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) 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 5.2-rasmda ko‘rish mumkin:



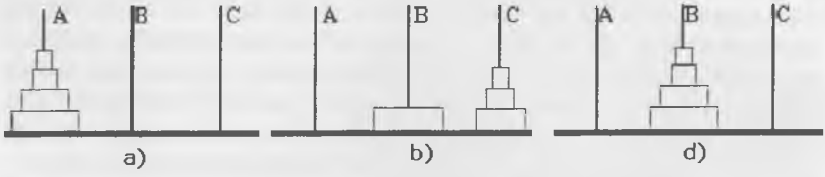
5.2-rasm. 4! hisoblash sxemasi

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).

Har bir rekursiv murojaat qo‘shimcha xotira talab qiladi - funksiyalarning lokal ob’ektlari (o‘zgaruvchilari) uchun har bir murojaatda stekdan yangidan joy ajratiladi. Masalan, rekursiv funksiyaga 100 marta murojaat bo‘lsa, jami 100 lokal ob’ektlarning majmuasi 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 programma o‘z ishini «Stek to‘lib ketdi» xabari bilan to‘xtadi.

Quyida, rekursiya bilan samarali echiladigan «Xanoy minorasi» masalasini ko‘raylik.

Masala. 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. Boshda barcha xalqalar A qoziqqa 5.3a -rasmdagidek joylashtirilgan. A qoziqdagi barcha xalqalarni B qoziqqa, yordamchi S 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.



5.3-rasm. Xanoy minorasi masalasini yechish jarayoni

Amallar ketma-ketligini chop etadigan («Xalqa q dan r ga o‘tkazilsin» ko‘rinishida, bunda q va r - 5.3-rasmdagi A,V yoki S xalqalar). Berilgan n ta xalqa uchun masala echilsin.

Ko‘rsatma: xalqalarni A dan B ga to‘g‘ri o‘tkazishda 5.3b –rasmlardagi holat yuzaga keladi, ya’ni n xalqani A dan B o‘tkazish masalasi n-1 xalqani A dan S ga o‘tkazish, hamda bitta xalqani A dan B o‘tkazish masalasiga keladi. Undan keyin S qoziqdagi n-1 xalqali A qoziq yordamida B qoziqqa o‘tkazish masalasi yuzaga keladi va hakoza.

```
#include <iostream.h>
void Hanoy(int n,char a='A',char b='B',char c='C')
{
 if(n)
 {
  Hanoy(n-1,a,s,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) programma ekranga xalqalarni ko'chirish bo'yicha amallar ketma-ketligini chop etadi:

```
Xalqa A dan B ga o'tkazilsin
Xalqa A dan C ga o'tkazilsin
Xalqa B dan C ga o'tkazilsin
Xalqa A dan B ga o'tkazilsin
Xalqa C dan A ga o'tkazilsin
Xalqa C dan B ga o'tkazilsin
Xalqa A dan B ga o'tkazilsin
```

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 ham o'ng'ay hisoblandi.

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 algoritmlar amalga oshirilgan funksiyalar 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 funksiyani chaqiradi. Bunday amalni «*hal qilish amali*» deyiladi va uning maqsadi

parametrlarga ko‘ra aynan (nisbatan) to‘g‘ri keladigan funksiyani chaqirishdir. Agar bunday funksiya topilmasa kompilyator xatolik haqida xabar beradi. Funksiyani aniqlashda funksiya qaytaruvchi qiymat turining ahamiyati yo‘q. Misol:

```
#include <iostream.h>
int max(int,int);
char max(char,char);
float max(float,float)
int max(int,int,int);
void main()
{
 int a,int b,char c,char d,int k,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 funksiya chaqirilishida 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 o‘tkazishga.

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.

6-bob. Ko‘rsatkichlar va adres oluvchi o‘zgaruvchilar

Ko‘rsatkichlar

Programma matnida o‘zgaruvchi e’lon qilinganda, kompilyator o‘zgaruvchiga xotiradan joy ajratadi. Boshqacha aytganda, programma kodi xotiraga yuklanganda berilganlar uchun, ular joylashadigan segmentning boshiga nisbatan siljishini, ya’ni nisbiy adresini aniqlaydi va ob’ekt kod hosil qilishda o‘zgaruvchi uchragan joyga uning adresini joylashtiradi.

Umuman olganda, programmadagi o‘zgarmaslar, o‘zgaruvchilar, funksiyalar va sinf ob’ektlar adreslarini xotiraning alohida joyida saqlash va ular ustidan amallar bajarish mumkin. Qiymatlari adres bo‘lgan o‘zgaruvchilarga *ko‘rsatkich o‘zgaruvchilar* deyiladi.

Ko‘rsatkich uch xil turda bo‘lishi mumkin:

- birorta ob’ektga, 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.

Funksiyaga ko‘rsatkich. Funksiyaga ko‘rsatkich programma 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 chaqirish 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 erda 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 erda $h = \frac{b-a}{n}$, $x_i = a + ih - h/2, i = 1..n$.

Programma 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 «math.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 regimini tanlang:\n";
  cout<<"1:f1(x)=5*sin(3*x)+x integralini\
  hisoblash\n";
  cout<<"2:f2(x)=cos(x) integralini hisoblash\n";
  cout<<"3:f3(x)=x^2+1 integralini hisoblash\n";
```

```
    cout<<"0:Programmadan chiqish\n";
    do
    {
     cout<<" Ish regimi-> ";
     cin>>menu;
    }
     while (menu<0 || menu>3);
    if(!menu)break;
    cout<<"Integral oralig'ining quyi chegarasi a=";
    cin>>a;
    cout<<"Integral oralig'ining yuqori chegarasi b=";
    cin>>b;
    cout<<"Funksiya integrali S=";
    switch (menu)
    {
     case 1 : cout<<Integral(f1,a,b)<<endl; break;
     case 2 : cout<<Integral(cos,a,b)<<endl; break;
     case 3 : cout<<Integral(f3,a,b)<<endl;
    }
   }
   return 0;
}
```

Programmaning ishi cheksiz takrorlash operatori tanasini bajarishdan iborat. Takrorlash tanasida foydalanuvchiga ish rejimini tanlash bo‘yicha menyu taklif qilinadi:

```
Ish regimini 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: Programmadan chiqish
Ish regimi->
```

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 programmadan 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.

Ob’ektga ko‘rsatkich. Biror ob’ektga ko‘rsatkich (shu jumladan o‘zgaruvchiga). Bunday ko‘rsatkichda ma’lum turdagi (tayanch yoki hosilaviy turdagi) berilganlarning xotiradagi adresi joylashadi. Ob’ektga ko‘rsatkich quyidagicha e’lon qilinadi:

<tur> *<nom>;

Bu erda <tur> - ko‘rsatkich aniqlaydigan adresdagi qiymatning turi, <nom> - ob’ekt nomi (identifikator). Agar bir turda bir nechta ko‘rsatkichlar e’lon qilinadigan bo‘lsa, har bir ko‘rsatkich uchun ‘*’ belgisi qo‘yilishi shart:

```
int *i, j,*k;
float x,*y,*z;
```

Keltirilgan misolda i va k - butun turdagi ko‘rsatkichlar va j - butun turdagi o‘zgaruvchi, ikkinchi operatorda x - haqiqiy o‘zgaruvchi va y,z - haqiqiy turdagi ko‘rsatkichlar e’lon qilingan.

void ko‘rsatkich. Bu ko‘rsatkich ob’ekt 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 kuyidagicha bo‘ladi:

void *<nom>;

Ko‘rsatkichning o‘zi o‘zgarmas yoki o‘zgaruvchan bo‘lishi va o‘zgarmas yoki o‘zgaruvchilar adresiga ko‘rsatishi mumkin, masalan:

```
int i;              // butun o`zgaruvchi
const int ci=1;     // butun o`zgarmas
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 qiymatni 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 programma ishlash jarayonida, kerakli paytida ajratib olinadi va zarurat qolmaganida qaytariladi (bo‘shatiladi). Keyinchalik, bu xotira bo‘lagi programma 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 programma 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 (initsializator) ko‘rsatkich nomidan so‘ng yoki qavs ichida yoki ‘=’ belgidan keyin beriladi. Boshlang‘ich qiymatlar quyidagi usullar bilan berilishi mumkin:

I. Ko‘rsatkichga mavjud bo‘lgan ob’ektning adresini berish:

a) adresni olish amal orqali:

```
int i=5,k=4; // butun o‘zgaruvchilar
int *p=&i;   // p ko‘rsatkichga i o‘zgaruvchining
             // adresi yoziladi
int *p1(&k); // p1 ko‘rsatkichga k o‘zgaruvchining
             // adresi yoziladi
```

b) boshqa, initsializatsiyalangan ko‘rsatkich qiymatini berish:

```
int * r=p; // p oldin e’lon qilingan va qiymatga ega
           // bo‘lgan ko‘rsatkich
```

d) massiv yoki funksiya nomini berish:

```
int b[10];  // massivni e’lon qilish
int  *t=b;    // massivning boshlang‘ich adresini
berish
void f(int a){/* … */} // funksiyani aniqlash
void  (*pf)(int);//  funksiyaga ko‘rsatkichni  e’lon
qilish
pf=f; // funksiya adresini ko‘rsatkichga berish
```

II. Oshkor ravishda xotiraning absolyut adresini berish:

```
char *vp = (char *)0xB8000000;
```

Bunda 0xB8000000 - o‘n oltilik o‘zgarmas son va (char*) - turga keltirish amali bo‘lib, u vp o‘zgaruvchisini xotiraning absolyut adresidagi baytlarni char sifatida qayta ishlovchi ko‘rsatkich turiga aylantirilishini anglatadi.

III. Bo‘sh qiymat berish:

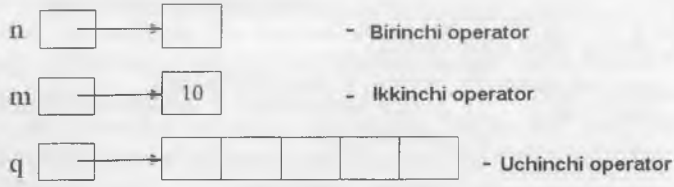
```
int *suxx=NULL;
int *r=0;
```

Birinchi satrda maxsus NULL o‘zgarmasi ishlatilgan, ikkinchi satrda 0 qiymat ishlatilgan. Ikkala holda ham ko‘rsatkich hech qanday ob’ektga murojaat qilmaydi. Bo‘sh ko‘rsatkich asosan ko‘rsatkichni aniq bir ob’ektga ko‘rsatayotgan yoki yo‘qligini aniqlash uchun ishlatiladi.

IV. 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 etarli joy ajratib olinib, uning adresi n ko‘rsatkichga yuklanadi. Ko‘rsatkichning o‘zi uchun joy kompilyasiya vaqtida ajratiladi.



6.1-rasm. Dinamik xotiradan joy ajratish

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.

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.

Ko‘rsatkich ustida amallar

Ko‘rsatkich ustida quyidagi amallar bajarilishi mumkin:

1) ob’ektga 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:

```
    shar a;              //    char    turidagi    o'zgaruvchi
e'loni.
    shar *p=new char; // Ko'rsatkichni e'lon qilib,unga
                      // dinamik xotiradan ajratilgan
                      // xotiraning adresini berish
    *p='b';  // p adresiga qiymat joylashtirish
    a=*p;    // a o'zgaruvchisiga p 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 berilgan-ning har xil turdagi qiymatlarini olish mumkin:

```
unsigned long int A=0Xcc77ffaa;
unsigned short int * pint=(unsigned short int*)&A;
unsigned char* pchar=(unsigned char*)&A;
cout<<hex<<A<<' '<<hex<<*pint<<' '<<hex<<(int)*pchar;
```

Ekranga har xil qiymatlar chop etiladi:

```
cc77ffaa ffaa aa
```

O'zgaruvchilar bitta adresda joylashgan holda yaxlit qiymatning turli bo'laklarini o'zlashtiradi. Bunda, bir baytdan katta joy egallagan son qiymatining xotirada «teskari» joylashishi inobatga olinishi kerak.

Agar har xil turdagi ko'rsatkichlarga qiymatlar berilsa, albatta turga keltirish amalidan foydalanish kerak:

```
int n=5;
float x=1.0;
int *pi=&n;
float *px=&x;
void *p;
int *r,*r1;
px=(float *)&n;
p=px;
r=(int *)p;
r1=pi;
```

Ko'rsatkich turini void turiga keltirish amalda ma'noga ega emas. Xuddi shunday, turlari bir xil bo'lgan ko'rsatkichlar uchun turni keltirish amalini bajarishga hojat yo'q.

Ko'rsatkich ustidan bajariladigan arifmetik amallarda avtomatik ravishda turlarning o'lchami hisobga olinadi.

Arifmetik amallar faqat bir xil turdagi ko'rsatkichlar ustidan bajariladi va ular asosan, massiv tuzilmalariga ko'rsatkichlar ustida bajariladi.

Inkrement amali ko'rsatkichni massivning keyingi elementiga, dekrement esa aksincha, bitta oldingi elementining adresiga ko'chiradi.

Bunda ko‘rsatkichning qiymati sizeof(<massiv 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 oshadi
q++;      // q qiymati 4 ga oshadi
q+=3;     // q qiymati 3*4=12 oshadi
```

Ko‘rsatkichlarning ayirmasi deb, ular ayirmasining tur o‘lchamiga bo‘linishiga aytiladi. Ko‘rsatkichlarni o‘zaro qo‘shish mumkin emas.

Adresni olish amali

Adresni olish quyidagicha e’lon qilinadi:

<tur> & <nom>;

Bu erda <tur> - adresi olinadigan qiymatning turi, <nom>- adres oluvchi o‘zgaruvchi nomi. O‘rtadagi ‘&’ belgisiga *adresni olish amali* deyiladi.

Bu ko‘rinishda e’lon qilingan o‘zgaruvchi shu turdagi o‘zgaruvchining sinonimi deb qaraladi. Adresni olish amali orqali bitta o‘zgaruvchiga har xil nom bilan murojaat qilish mumkin bo‘ladi.

Misol:

```
int kol;
int & pal=kol; // pal 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 qilingandan holatlardan tashqari barcha holatlarda boshlang‘ich qiymatga ega bo‘lishi kerak.

Adresni olish amali asosan funksiyalarda adres orqali uzatiluvchi parametrlar sifatida ishlatiladi.

Adres oluvchi o‘zgaruvchining ko‘rsatkichdan farqi shundaki, u alohida xotirani egallamaydi va faqat o‘z qiymati bo‘lgan o‘zgaruvchining boshqa nomi sifatida ishlatiladi.

Ko‘rsatkichlar va adres oluvchi o‘zgaruvchilar funksiya parametri sifatida

Funksiya prototipida yoki aniqlanish sarlavhasida ko‘rsatilgan parametrlar *formal parametrlar* deyiladi, funksiya chaqirishida ko‘rsatilgan argumentlarga *faktik parametrlar* deyiladi.

Funksiya chaqirilishida faktik parametrning turi mos o‘rindagi formal parametr turiga to‘g‘ri kelmasa yoki shu turga keltirishning iloji bo‘lmasa kompilyasiya xatosi ro‘y beradi.

Faktik parametrlarni funksiyaga ikki xil usul bilan uzatish mumkin: *qiymati* yoki *adresi* bilan.

Funksiya chaqirilishida argument qiymat bilan uzatilganda, argument yoki uning o‘rnidagi kelgan ifoda qiymati va boshqa argumentlarning nusxasi (qiymatlari) stek xotirasiga yoziladi. Funksiya faqat shu nusxalar bilan amal qiladi, kerak bo‘lsa bu nusxalarga o‘zgartirishlar qilinishi mumkin, lekin bu o‘zgarishlar argument-ning o‘ziga ta’sir qilmaydi, chunki funksiya o‘z ishini tugatishi bilan nusxalar o‘chiriladi (stek tozalanadi).

Agar parametr adres bilan uzatilsa, stekka adres nusxasi yoziladi va xuddi shu adres bo‘yicha qiymatlar o‘qiladi (yoziladi). Funksiya o‘z ishini tugatgandan keyin shu adres bo‘yicha qilingan o‘zgarishlar saqlanib qolinadi va bu qiymatlarni boshqa funksiyalar ishlatishi mumkin.

Argument qiymat bilan uzatilishi uchun mos formal parametr sifatida o‘zgaruvchini turi va nomi yoziladi. Funksiya chaqirilishida mos argument sifatida o‘zgaruvchining nomi yoki ifoda bo‘lishi mumkin.

Faktik parametr adres bilan uzatilganda unga mos keluvchi formal parametrni ikki xil usul bilan yozish mumkin: *ko‘rsatkich orqali* yoki *adresni oluvchi parametrlar orqali.* 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 funksiya chaqirilishida mos argument sifatida o‘zgaruvchi nomi keladi.

Misol:

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

Programma 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 funksiyani chaqirishda argumentlar faqat nomlari bilan berilgan bo'lsa, kelishuv bo'yicha massivlar va funksiyalar adresi bilan, qolgan turdagi parametrlar qiymatlari bilan uzatilgan deb hisoblanadi.

Misol tariqasida diskriminantni hisoblash usuli yordamida $ax^2+bx+c=0$ ko'rinishidagi kvadrat tenglama ildizlarini funksiya parametrlari vositasida olish masalasini ko'raylik.

```
#include <iostream.h>
#include <math.h>
int Kvadrat_Ildiz(float a,float b,float c,
                  float & x1, float & x2)
{
 float D;
 D=b*b-4*a*c;
 if(D<0) return 0;
 if(D==0)
```

```
 {
   x1=x2=-b/(2*a);
  return 1;
 }
 else
 {
   x1=(-b+sqrt(D))/(2*a);
   x2=(-b-sqrt(D))/(2*a);
   return 2;
 }
}
int main()
{
 float a,b,c,D,x1,x2;
 cout<<"ax^2+bx+c=0 tenglama ildizini topish. ";
 cout<<"\n a - koeffisiyentni kiriting: "; cin>>a;
 cout<<"\n b - koeffisiyentni kiriting: "; cin>>b;
 cout<<"\n c - koeffisiyentni kiriting: "; cin>>c;
 switch (Kvadrat_Ildiz(a,b,c,x1,x2))
 {
  case 0: cout<<"Tenglama haqiqiy ildizga ega emas!";
          break;
  case 1: cout <<"Tenglama yagona ildizga ega: ";
          cout<<"\n x= "<<x1;
          break;
  default:cout<<"Tenglama ikkita ildizga ega: ";
          cout<<"\nx1= "<<x1;
          cout<<"\nx2= "<<x2;
 }
 return 0;
}
```

Programmadagi Kvadrat_Ildiz() funksiyasi kvadrat tenglama ildizini hisoblaydi. Uning qaytaradigan qiymati tenglamaning nechta ildizi borligini anglatadi. Agar tenglamaning haqiqiy ildizi mavjud bo‘lmasa (D<0), funksiya 0 qiymatini qaytaradi. Agar D=0 bo‘lsa, funksiya 1 qiymatini qaytaradi. Agar D>0 bo‘lsa funksiya 2 qiymatini qaytaradi. Mavjud ildizlar - x1 va x2 adres oluvchi parametrlarda qaytariladi.

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 erda <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 programma tuzuvchi zimmasiga yuklanadi.

O'zgaruvchan sondagi parametrlarni tashkil qilish usuli umuman olganda ikkita:

1-usul. Parametrlar ro'yxati oxirida yana bir maxsus parametr yoziladi va uning qiymati parametrlar tugaganligini bildiradi. Kompilyator tomonidan funksiya tanasida parametrlar birma-bir aniqlashtiriladi. Barcha parametrlar turi oxirgi maxsus parametr turi bilan ustma-ust tushadi deb hisoblanadi;

2-usul. Birorta maxsus parametr sifatida noma'lum parametrlar soni kiritiladi va unga qarab parametrlar soni aniqlanadi.

Ikkala usulda ham parametrlarga murojaat qilish uchun ko'rsatkichlar ishlatiladi. Misollar keltiramiz.

1 - usul:

```
#include <iostream.h>
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:

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

Ikkala misolda ham noma'lum parametrlar berilgan maxsus parametr turini qabul qilgan. Har xil turdagi parametrlarni ishlatish uchun turni aniqlaydigan parametr kiritish kerak:

```
#include <iostream.h>
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 programma tuzuvchisi tomonidan hal qilingan.

7-bob. Massivlar

Berilganlar massivi tushunchasi

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

Odatda massivlarga zarurat, katta hajmdagi, lekin cheklangan miqdordagi va tartiblangan 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, 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 erda 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 programmani tuzish katta qiyinchiliklarni yuzaga keltiradi. Ikkinchi yo'l - berilganlar ketma-ketligini yagona nom bilan aniqlab, uning qiymatlariga murojaatni, shu qiymatlarning ketma-ketlikda joylashgan o'rnining nomeri (indeksi) orqali amalga oshirishdir. Reytinglar ketma-ketligini Reyting deb nomlab, undagi qiymatlariga $Reyting_1$,...,$Reyting_N$ ko'rinishida murojaat qilish mumkin. Odatda berilganlarning bunday ko'rinishiga massivlar deyiladi. Massivlarni matematikadagi sonlar vektoriga o'xshatish mumkin, chunki vektor ham o'zining individual nomiga ega va u fiksirlangan miqdordagi bir turdagi qiymatlardan - sonlardan iboratdir.

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

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

<massiv nomi >[<indeks>]

Bu ko‘rinishga *xususiy o‘zgaruvchi* deyiladi, chunki uning qiymati massivning alohida elementidir. Bizning misolda Reyting massivining alohida komponentalariga Reyting[1],...,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 muro-jaat qilinayotgan massiv elementini aniqlovchi indeks ham o‘zgaradi. Shunday qilib, programmadagi bitta indeksli o‘zgaruvchi orqali massivning barcha elementlarini belgilash (aniqlash) mumkin bo‘ladi. Masalan, Reyting[I] o‘zgaruvchisi orqali I o‘zgaruvchining qiymatiga bog‘liq ravishda Reyting massivining ixtiyoriy elementiga murojaat qilish mavjud.

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

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

Massiv e’loni quyidagicha bo‘ladi:

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

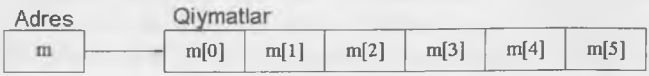
Bu erda <uzunlik> - o‘zgarmas ifoda. Misollar:

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

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

```
int m[6];
```

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



7.1-rasm. Bir o‘lchamli massivning xotiradagi joylashuvi

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

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

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

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

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

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

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

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

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

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

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

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

Massivni e'lon qilishga misollar:

```
shar ch[4]={'a','b','c','d'}; //belgilar massivi
int in[6] ={10,20,30,40};      // butun sonlar massivi
char str[]="abcd";
//satr uzunligi 5 ga teng, chunki uning oxiriga
// '\0' belgisi qo'shiladi
char str[]={'a','b','c','d'};
// yuqoridagi satrning boshqacha yozilishi
```

Masala. Bir oy ichidagi kundalik haroratlar berilgan. Oy uchun o'rtacha haroratni hisoblash programmasi tuzilsin.

Programma matni:

```
void main()
{const int n=30;
 int temp[n];
 int i,s,temp_urtacha;
 cout << "Kunlik haroratni kiriting:\n"
 for (i=0;i<n;i++)
 {cout << "\n temp["<<i<<"]=";
  cin >> temp[i]; }
 for (i=0,s=0; i<n;i++)s+=temp[i];
 temp_urtacha=s/n;
 cout << "Kunlik harorat :\n";
 for(i=0;i<n;i++)cout<< "\t temp["<<i<<"]="<<temp[i];
 cout<<"Oydagi o'rtacha harorat= "<<temp_urtacha;
 return;
}
```

Ko'p o'lchamli statik massivlar

C++ tilida massivlar elementining turiga cheklovlar qo'yilmaydi, lekin bu turlar chekli o'lchamdagi ob'ektlarning turi bo'lishi kerak. CHunki kompilyator massivning xotiradan qancha joy (bayt) egallashini hisoblay olishi kerak. Xususan, massiv komponentasi massiv bo'lishi mumkin («vektorlar-vektori»), natijada *matritsa* deb nomlanuvchi ikki o'lchamli massiv hosil bo'ladi.

Agar matritsaning elementi ham vektor bo'lsa, uch o'lchamli massivlar - *kub* hosil bo'ladi. Shu yo'l bilan echilayotgan masalaga bog'liq ravishda ixtiyoriy o'lchamdagi massivlarni yaratish mumkin.

Ikki o'lchamli massivning sintaksisi quyidagi ko'rinishda bo'ladi:

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

Masalan, 10×20 o'lchamli haqiqiy sonlar massivining e'loni:

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

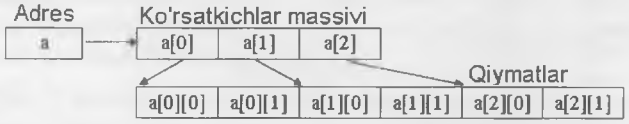
7.2-rasm. Ikki o'lchamli massivning xotiradagi joylashuvi

E'lon qilingan A matritsani ko'rinishi 7.2-rasmda keltirilgan.

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

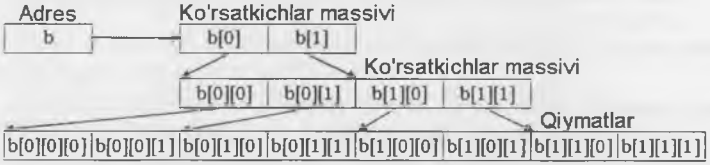
```
int a[3][2];
float b[2][2][2];
```

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



7.3-rasm. Ikki o'lchamli massiv elementlariga murojaat

Bu erda a[i] ko'rsatkichda i-chi satrning boshlang'ich adresi joylashadi, massiv elementiga a[i][j] ko'rinishidagi asosiy murojaatdan tashqari vositali murojaat qilish mumkin: *(*(a+i)+j) yoki *(a[i]+j).



7.3-rasm. Uch o'lchamli massivning xotirada tashkil bo'lishi

Massiv elementlariga murojaat qilish uchun nomdan keyin kvadrat qavsda har bir o'lcham uchun indeks yozilishi kerak, masalan b[i][j][k]. Bu elementga vositali murojaat ham qilish mumkin va uning variantlari:

((*(b+i)+j)+k) yoki *(*(b[i]+j)+k) yoki *(b[i][j] +k);

Ko'p o'lchamli 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}};
int c[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.

Misol uchun, matritsalar va vektor ko‘paytmasini - $C = A \times b$ hisoblash masalasini ko‘raylik. Bu erda $A = \{a_{ij}\}$, $b = \{b_j\}$, $c = \{c_i\}$, $0 \le i < m, 0 \le j < n$. Hisoblash formulasi - $c_i = \sum_{j=0}^{n-1} a_{ij} b_j$.

Mos programa matni:

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

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, etarlicha katta o‘lchamdagi massiv e’lon qilib, konkret masala echilishida ajratilgan xotira to‘liq ishlatilmasligi mumkin. Bu kamchiliklar dinamik massivlardan foydalanish orqali bartaraf etiladi, chunki ular programma ishlashi jarayonida kerak bo‘lgan o‘lchamdagi massivlarni yaratish va zarurat qolmaganda yo‘qotish imkoniyatini beradi.

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

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

malloc() funksiyasining sintaksisi

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

ko‘rinishida bo‘lib, u xotiraning uyum qismidan size bayt o‘lchamidagi uzluksiz sohani ajratadi. Agar xotira ajratish muvaffaqiyatli bo‘lsa,

malloc() funksiyasi ajratilgan sohaning boshlanish adresini qaytaradi. Talab qilingan xotirani ajratish muvaffaqiyatsiz bo‘lsa, funksiya NULL qiymatini qaytaradi.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
delete [] pVector;
```

ko'rsatmasini berish kerak bo'ladi.

Ikki o'lchamli dinamik massivni tashkil qilish uchun

```
int **a;
```

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

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

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

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

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

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

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

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

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

```
void main ()
{
 int n,m;
 int i,j; float s;
 cout<<"\n n="; cin>>n; // matritsa satrlari soni
 cout<<"\n m="; cin>>m; // matritsa ustunlari soni
 float *b=new float[m];
 float *c=new float[n];
 // ko'rsatkichlar massiviga xotira ajratish
 float **a=new float *[n] ;
 for(i=0;i<n;i++)       // har bir satr uchun
 a[i]=new float[m];    //dinamik xotira ajratish
 for(j=0;j<m;j++)cin>>b[j];
 for(i=0;i<n;i++)
  for(j=0;j<m;j++)cin>>a[i][j];
 for(i=0;i<n;i++)
 {
  for(j=0,s=0;j<m;j++)s+=a[i,j]*b[j];
  c[i]=s;
 }
 for(i=0;i<n;i++)cout<<"\t c["<<i<<"]="<<c[i];
 delete[]b;
 delete[]c;
 for (i=0;i<n;i++) delete[]a[i];
 delete[]a;
 return;
}
```

Funksiya va massivlar

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

Agar massiv parametr orqali funksiyaga uzatilsa, elementlar sonini aniqlash muammosi tug'iladi, chunki massiv nomidan uning uzunligini aniqlashning iloji yo'q. Ayrim hollarda, masalan, belgilar massivi sifatida

aniqlangan satr (ASCIIZ satrlar) bilan ishlaganda massiv uzunligini aniqlash mumkin, chunki satrlar '\0' belgisi bilan tugaydi.

Misol uchun:

```
#include <iostream.h>
int len(char s[])//massivni parametr sifatida
ishlatish
{
 int m=0;
 while(s[m++]);
 return m-1;
}
void main ()
{
 char z[]="Ushbu satr uzunligi = ";
 cout<<z<<len(z);
}
```

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

Misol:

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

Programma matni:

```
#include <iostream.h>
\\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 massivlar bilan ishlash ma'lum bir murakkablikka ega, chunki massivlar xotirada joylash tartibi turli variantda bo'lishi mumkin. Masalan, funksiya parametrlar ro'yxatida $n \times n$ o'lchamdagi haqiqiy turdagi x[n][n] massivga mos keluvchi parametrni

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

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

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

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

1-usul. Massivning ikkinchi o‘lchamini o‘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;
}
```

Navbatdagi programmada funksiya tomonidan natija sifatida ikki o'lchamli massivni qaytarishiga misol keltirilgan. Massiv elementlarning qiymatlari tasodifiy sonlardan tashkil topadi. Tasoddifiy sonlar «math.h» kutubxonasidagi random() funksiya yordamida hosil qilinadi:

```
#include <iostream.h>
#include <math.h>
int **rmatr(int n,int m)
{
 int ** ptr;
 ptr=new int *[n];
 for(int i=0;i<n;i++)
 {
   ptr[i]=new int[m];
  for(int j=0;j<m;j++) ptr[i][j]=random(100);
 }
 return ptr;
}
int sum(int n,int m,int **ix)
{
 float s=0;
 for(int i=0;i<n;i++)
 for(int j=0;j<m;j++) s+=ix[i][j];
 return s;
}
void main()
{
 int n,m;
 cin>>n>>m;
 int **matr;
 randomize();
 matr=rmatr(n,m);
 for(int i=0;i<n;i++)
 {
  cout<<endl<<i<<" - satr:"
  for (int j=0;j<m;j++) cout<<'\t'<<matr[i][j];
 }
 cout<<endl<<"Summa="<<sum(n,m,matr);
 for(int i=0;i<n;i++) delete matr[i];
 delete[]matr;
}
```

Masala. Eratosfen g'alviri - bu [1; N] oralig'idagi tub sonlarni topish algoritmidir. Algoritm g'oyasi juda sodda - 1..N oralig'idagi barcha butun sonlarni yozib chiqiladi, boshda ketma-ketlikdagi 2 sonidan boshqa 2 ga karrali sonlarni o'chiriladi, undan keyin 3 sonidan boshqa 3 ga karrali sonlarni o'chiriladi, undan keyin qolgan ketma-ketlikdagi 3 keyingi joylashgan songa - 5 soniga karralilarni o'chiramiz, 5 dan tashqari va hakoza. Jarayon ketma-ketlikda navbatdagi tub songa karrali sonlar qolmaguncha davom etadi. Shu tariqa ketma-ketlikda [1; N] oralig'idagi tub son qoladi.

Programma matni:

```
#include <iostream.h>
bool * Eratosfen(bool * k_k, int n)
{
 for (int i=2; i<=n; ++i)
  if (k_k[i])      // navbatdagi tub son
   for (int j=i+i; j<=n; j+=i)
    k_k[j]=false;  // karrali tub sonlarni o'chirish
 return k_k;
}
int main()
{
 int N;
 cout<<"[1..N]=> N=";
 cin>>N;
 bool *ketma_ketlik=new bool[N+1];
 // sonlar ketma-ketligini shaklantirish
 for(int i=1; i<=N; ++i) ketma_ketlik[i]=true;
 ketma_ketlik=Eratosfen(ketma_ketlik,N);
 cout<<"[1.."<<N<<"] oralig'idagi tub sonlar:"<<endl;
 for(int i=1; i<=N; ++i)
  if(ketma_ketlik[i])cout<<i<<' ';
 return 0;
}
```

Programmada kiritilgan interval chegarasi N uchun sonlar ketma-ketligi mantiqiy `ketma_ketlik` massivi ko'rinishida shakllantiriladi. Agar ayni paytda ketma-ketlikda `i` soni mavjud bo'lsa - `ketma_ketlik[i]` qiymati `true` bo'ladi, aks holda `false`. Algoritmni amalga oshiruvchi `Eratosfen()` funksiyasining natijasi bo'lgan massivning `true` qiymatlariga mos indekslar (tub sonlar) chop qilinadi.

8-bob. ASCIIZ satrlar va ular ustida amallar

Belgi va 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 turi-dagi 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.

8.1-jadval. 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 massivi e'lon qilinishida, satr oxiriga 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. Bu holda kompilyator avtomatik ravishda satr uzunligi 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.h>
int main()
{
 char satr[6];
 cout<<"Satrni kiriting: "<<'\n';
 cin.getline(satr,6);
 cout<<"Siz kiritgan satr: "<<satr;
 return 0;
}
```

Programmada 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 «string.h» kutubxonasida jamlangan. Nisbatan ko‘p ishlatiladigan funksiyalarning tavsifini keltiramiz.

Satr 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 real 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 real uzunligini qaytaradi. Quyida keltirilgan misolda satr uzunligini hisoblashning har ikkita varianti keltirilgan:

```
#include <iostream.h>
#include <string.h>
int main()
{
 char Str[]="1234567890";
 cout <<"strlen(Str)="<<strlen(Str)<<endl;
 cout<<"sizeof(Str)="<<sizeof(Str)<<endl;
 return 0;
}
```

Programma ishlashi natijasida ekranga

```
strlen(Str)=10
sizeof(Str)=11
```

xabarlari chiqadi.

Odatda sizeof() funksiyasidan getline() funksiyasining ikkinchi argumenti sifati ishlatiladi 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));
 float 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;
}
```

Programmada satr uchun 80 uzunligidagi Satr belgilar massivi e’lon qilingan va uning qiymati klaviaturadan kiritiladi. Masala quyidagicha echiladi. 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. Programma oxirida s qiymati butun turga aylantirilgan holda chop etiladi. Satrdagi so‘zlarni bir-biridan ajratuvchi probel belgisi cheklab o‘tiladi.

Programmaga

```
Satrdagi turli harflar miqdori
```

satri kiritilsa, ekranga javob tariqasida

```
Satrdagi turli belgilar miqdori: 13
```

satri chop etiladi.

Satrlarni 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 uchraguncha davom etadi. Shu sababli, str2 satr uzunligi str1 satr uzunligidan katta emasligiga ishonch hosil qilish kerak, aks holda berilgan sohasida (segmentda) str1 satrdan keyin joylashgan berilganlar «ustiga» str2 satrning «ortib qolgan» qismi yozilishi mumkin.

Navbatdagi programma 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.h>
#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.h>
#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;
}
```

Programmada 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 programma bo‘lagida satr1 satrining nusxasi xotiraning satr2 ko‘rsatgan joyida paydo bo‘ladi:

```
char* satr1="Satr nusxasini olish."; char* satr2;
satr2=strdup(satr1);
```

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)
```

Pastda keltirilgan programma bo'lagida str1 satrga str2 satrning boshlang'ich 10 ta belgidan iborat satr qismini ulaydi:

```
char satr1[80]="Programmalash tillariga misol bu-";
char satr2[80]="C++,Pascal,Basic";
strncat(satr1,satr2,10);
cout<<satr1;
```

Amallar bajarilishi natijasida ekranga

```
Programmalash tillariga misol bu-C++,Pascal
```

satri chop etiladi.

Masala. Nol-terminator bilan tugaydigan S,S1 va S2 satrlar berilgan. S satrdagi S1 satr ostilari S2 satr osti bilan almashtirilsin. Masalani yechish uchun quyidagi masala ostilarini yechish zarur bo'ladi:

1) S satrida S1 satr ostini kirish o'rnini aniqlash;
2) S satridan S1 satr ostini o'chirish;
3) S satrida S1 satr osti o'rniga S2 satr ostini o'rnatish.

Garchi bu masala ostilarining yechimlari C++ tilning standart kutubxonalarida funksiyalar ko'rinishida mavjud bo'lsa ham, ular kodini qayta yozish foydalanuvchiga bu amallarning ichki mohiyatini tushunishga imkon beradi. Quyida masala yechimining programma matni keltirilgan:

```
#include <iostream.h>
#include <string.h>
const int n=80;
int Izlash(char *,char *);
void Qirqish(char *, int, int);
```

```
void Joylash(char *,char *, int);
int main()
{
 char Satr[n], Satr1[n], Satr2[n];
 cout<<"Satrni kiriting: ";
 cin.getline(Satr,n);
 cout<<"Almashtiriladigan satr ostini kiriting: ";
 cin.getline(Satr1,n);
 cout<<Satr1<<"Qo'yiladigan satrni kiriting:";
 cin.getline(Satr2,n);
 int Satr1_uzunligi=strlen(Satr1);
 int Satr_osti_joyi;
 do
 {
  Satr_osti_joyi=Izlash(Satr,Satr1);
  if(Satr_osti_joyi!=-1)
  {
   Qirqish(Satr,Satr_osti_joyi,Satr1_uzunligi);
   Joylash(Satr,Satr2,Satr_osti_joyi);
  }
 } while (Satr_osti_joyi!=-1);
 cout<<"Almashtirish natijasi: "<<Satr;
 return 0;
}
int Izlash(char satr[],char satr_osti[])
{
 int satr_farqi=strlen(satr)-strlen(satr_osti);
 if(satr_farqi>=0)
 {
  for(int i=0; i<=satr_farqi; i++)
  {
   bool ustma_ust=true;
   for(int j=0; satr_osti[j]!='\0' && ustma_ust; j++)
    if(satr[i+j]!=satr_osti[j]) ustma_ust=false;
   if (ustma_ust) return i;
  }
 }
 return -1;
}

void Qirqish(char satr[],int joy,int qirqish_soni)
{
 int satr_uzunligi=strlen(satr);
 if (joy<satr_uzunligi)
  {
   if(joy+qirqish_soni>=satr_uzunligi)satr[joy]='\0';
    else
     for (int i=0; satr[joy+i-1]!='\0'; i++)
```

```
            satr[joy+i]=satr[joy+qirqish_soni+i];
        }
    }
    void Joylash(char satr[],char satr_osti[],int joy)
    {
     char vaqtincha[n];
     strcpy(vaqtincha, satr+joy);
     satr[joy]='\0';
     strcat(satr,satr_osti);
     strcat(satr,vaqtincha);
    }
```

Programmada har bir masala ostiga mos funksiyalar tuzilgan:

1) int Izlash(char satr[],char satr_osti[]) - funksiyasi satr satriga satr_osti satrining chap tomondan birinchi kirishining o'rnini qaytaradi. Agar satr satrida satr_osti uchramasa -1 qiymatini qaytaradi.

2) void Qirqish(char satr[],int joy,int qirqish_soni) - funksiyasi satr satrining joy o'rnidan boshlab qirqish_soni sondagi belgilarni qirqib tashlaydi. Funksiya natijasi satr satrida hosil bo'ladi;

3) void Joylash(char satr[],char satr_osti[],int joy) - funksiyasi satr satriga, uning joy o'rnidan boshlab satr_osti satrini joylashtiradi.

Bosh funksiyada satr (S), unda almashtiriladigan satr (S1) va S1 o'rniga joylashtiriladigan satr (S2) oqimdan o'qiladi. Takrorlash operatori bajarilishining har bir qadamida S satrining chap tomonidan boshlab S1 satri izlanadi. Agar S satrida S1 mavjud bo'lsa, u qirqiladi va shu o'ringa S2 satri joylashtiriladi. Takrorlash jarayoni Izlash() funksiyasi -1 qiymatini qaytarguncha davom etadi.

Satrlarni solishtirish

Satrlarni solishtirish ulardagi mos o'rinda joylashgan belgilar kodlarini o'zaro solishtirish bilan aniqlanadi. Buning uchun «string.h» kutubxonasida standart funksiyalar mavjud.

strcmp() funksiyasi sintaksisi

```
int strcmp(const char* str1, const char* str2)
```

ko'rinishiga ega bo'lib, funksiya str1 va str2 solishtirish natijasi sifatida son qiymatni qaytaradi (masalan, butun i o'zgaruvchisida) va ular quyidagicha izohlanadi:

a) i<0 - agar str1 satri str2 satridan kichik bo'lsa;

b) i=0 - agar str1 satri str2 satriga teng bo'lsa;

c) i>0 - agar str1 satri str2 satridan katta bo'lsa.

Funksiya harflarning registrini farqlaydi. Buni misolda ko'rishimiz mumkin:

```
char satr1[80]="Programmalash tillari:C++,pascal.";
char satr2[80]="Programmalash tillari:C++,Pascal.";
int i;
i=strcmp(satr1,satr2);
```

Natijada i o'zgaruvchisi musbat qiymat qabul qiladi, chunki solishtirilayotgan satrlardagi «pascal» va «Pascal» satr qismlarida birinchi harflar farq qiladi. Keltirilgan misolda i qiymati 32 bo'ladi. Bu farqlanuvchi harflar kodlarining ayirmasi. Agar funksiyaga

```
i= strcmp(satr2,satr1);
```

ko'rinishida murojaat qilinsa i qiymati manfiy son -32 bo'ladi.

Agar satrlardagi bosh yoki kichik harflarni farqlamasdan solishtirish amalini bajarish zarur bo'lsa, buning uchun strcmpi() funksiyasidan foydalanish mumkin. Yuqorida keltirilgan misoldagi satrlar uchun

```
i=strcmpi(satr2,satr1);
```

amali bajarilganda i qiymati 0 bo'ladi.

strncmp( ) funksiyasi sintaksisi

```
int strncmp(const char*str1,const char*str2,size_t num);
```

ko'rinishida bo'lib, str1 va str2 satrlarni boshlang'ich num sonidagi belgilarini solishtiradi. Funksiya harflar registrini inobatga oladi. Yuqorida misolda aniqlangan satr1 va satr2 satrlar uchun

```
i=strncmp(satr1,satr2,31);
```

amali bajarilishida i qiymati 0 bo'ladi, chunki satrlar boshidagi 31 belgilar bir xil.

strncmpi() funksiyasi strncmp() funksiyasidek amal qiladi, farqli tomoni shundaki, solishtirishda harflarning registrini hisobga olinmaydi. Xuddi shu satrlar uchun

```
i=strncmpi(satr1,satr2,32);
```

amali bajarilishi natijasida i o'zgaruvchi qiymati 0 bo'ladi.

Satrdagi harflar registrini almashtirish

Berilgan satrdagi kichik harflarni bosh harflarga yoki teskarisiga almashtirishga mos ravishda _strupr() va _strlwr() funksiyalar yordamida amalga oshirish mumkin. Kompilyatorlarning ayrim variantlarida funksiyalar nomidagi tagchiziq ('_') bo'lmasligi mumkin.

_strlwr() funksiyasi sintaksisi

```
char* _strlwr(char* str);
```

ko'rinishida bo'lib, argument sifatida berilgan satrdagi bosh harflarni kichik harflarga almashtiradi va hosil bo'lgan satr adresini funksiya natijasi

sifatida qaytaradi. Quyidagi programma bo‘lagi _strlwr() funksiyasidan foydalnishga misol bo‘ladi.

```
char str[]="10 TA KATTA HARFLAR";
_strlwr(str);
cout<<str;
```

Natijada ekranga

```
10 ta katta harflar
```

satri chop etiladi.

_strupr() funksiyasi xuddi _strlwr() funksiyasidek amal qiladi, lekin satrdagi kichik harflarni bosh harflarga almashtiradi:

```
char str[]="10 ta katta harflar";
_strupr(str);
cout<<str;
```

Natijada ekranga

```
10 TA KATTA HARFLAR
```

satri chop etiladi.

Programmalash 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 yoki harf (true) yoki yo‘qligini (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.h>
#include <ctype.h>
#include <string.h>
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 kiritdildi!";
    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);
}
```

Programmada 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 punktuatsiya belgisi bo‘lsa, shu haqda xabar beriladi va tug‘ilgan yilni qayta kiritish taklif etiladi. Programma tug‘ilgan yil (to‘rtta raqam) to‘g‘ri kiritilganda “Sizni tug'ilgan yilingiz: XXXX” satrini chop qilish bilan o‘z ishini tugatadi.

Satrni teskari tartiblash

Satrni teskari tartiblashni uchun strrev() funksiyasidan foydalanish mumkin. Bu funksiya quyidagicha prototipga ega:

```
char* strrev(char* str);
```

Satr reversini hosil etishga misol:

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

```
cout<<strrev(str);
```

amallar bajarilishi natijasida ekranga

```
nofelet
```

satri chop etiladi.

Satrda belgini 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 s 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 programma bo‘lagi belgini satrdan izlash bilan bog‘liq.

```
char satr[]="0123456789";
char* pSatr;
pSatr=strchr(satr,'6');
```

Programma 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 indexsi = "<<farqli_belgi;
cout<<"va u '"<<satr1[farqli_belgi]<<"' belgisi.";
```

amallar bajarilishi natijasida ekranga

```
Satrlardagi mos tushmagan belgi indexsi = 5
```

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';
```

Programma ishlashi natijasida ekranga str1 satrining

```
ABCDEF
```

satr ostisi chop etiladi.

Satr qismlarini izlash funksiyalari

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 programma 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';
```

Programma buyruqlari bajarilishi natijasida ekranga

```
satr ostisi izlanmoqda, satr ostisi mavjud
```

satri chop etiladi.

Keyingi programma 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 ru\'yxatda ";
if(Satrdagi_ism==NULL) cout<<"yo\'q ."<<'\n';
else cout<<"bor ."<'\n';
```

Programmada foydalanuvchidan satr qismi sifatida birorta nomni kiritish talab qilinadi va bu qiymat Ism satriga o'qiladi. Kiritilgan ism programmada 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 satr-ro'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.h>
#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;
}
```

Programma 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
```

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 «strlib.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;
 }
```

programmasining 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. Konvertatsiya qilinuvchi satrda xato bo‘lgan holatni ko‘rsatuvchi misol:

```
#include <stdlib.h>
#include <iostream.h>
int main(int argc, char* argv[])
{
 char satr[]="3.14D15E+2";
```

```
 char **kursatkich;
 double x= strtod(satr, kursatkich);
 cout<<"Konvertatsiya qilinuvchi satr: "<<satr<<endl;
 cout<<"Konvertatsiya qilingan x soni: "<<x<<endl;
 cout<<"Konvertatsiya uzilgan satr ostisi: "
 cout<<*kursatkich;
 return 0;
}
```

Programma bajarilishida x o‘zgaruvchi 3.14 sonini qabul qiladi, kursatkich o‘zgaruvchisi satrdagi ‘D’ belgisining adresini ko‘rsatadi. Ekranga quyidagi satrlar ketma-ketligi chop etiladi:

```
Konvertatsiya qilinuvchi satr: 3.14D15E+2
Konvertatsiya qilingan x soni: 3.14
Konvertatsiya uzilgan satr ostisi: D15E+2
```

Bir qator funksiyalar teskari amalni, ya’ni berilgan sonni satrga aylantirish amallarini bajaradi.

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. Misol uchun 12345 sonini turli sanoq sistemadagi satr ko‘rinishini hosil qilish masalasini ko‘raylik:

```
int main()
{
 char satr2[20],satr8[15],satr10[10],satr16[5];
 int son=12345;
 itoa(son,satr2,2);
 itoa(son,satr8,8);
 itoa(son,satr10,10);
 itoa(son,satr16,16);
 cout<<"Son ko'rinishlari"<<endl;
 cout<<"2 sanoq sistemasida : "<<satr2<<endl;
 cout<<"8 sanoq sistemasida : "<<satr8<<endl;
 cout<<"10 sanoq sistemasida: "<<satr10<<endl;
 cout<<"16 sanoq sistemasida: "<<satr16<<endl;
 return 0;
}
```

Programma ekranga quyidagi satrlarni chiqaradi:

```
Son ko'rinishlari
```

```
2 sanoq sistemasida : 11000000111001
8 sanoq sistemasida : 30071
10 sanoq sistemasida: 12345
16 sanoq sistemasida: 3039
```

gcvt( ) funksiyasi

```
char* gcvt(double val, int ndec, char *buf);
```

ko‘rinishdagi prototipga ega bo‘lib, double turidagi val sonini buf ko‘rsatuvchi ASCIIZ satrga aylantiradi. Ikkinchi argument sifatida beriladigan ndec qiymati son ko‘rinishida raqamlar miqdorini ko‘rsatadi. Agar raqamlar soni ndec qiymatidan ko‘p bo‘lsa, imkon bo‘lsa sonning kasr qismidan ortiqcha raqamlar qirqib tashlanadi (yaxlitlangan holda), aks holda son eksponensial ko‘rinishda hosil qilinadi. Quyidagi keltirilgan programmada gcvt() funksiyasidan foydalanishning turli variantlari ko‘rsatilgan.

```
int main()
{
 char satr[10]; double son; int raqamlar_soni=4;
 cout<<"Son ko\'rinishidagi raqamlat son: ";
 cout<<raqamlar_soni<<endl;
 son=3.154;
 gcvt(son,raqamlar_soni,satr);
 cout<<"3.154 sonining satr ko'rinishi: "<<satr;
 cout<<endl;
 son=-312.456;
 gcvt(son,raqamlar_soni,satr);
 cout<<"-312.456 sonining satr ko'rinishi: "
 cout<<satr<<endl;
 son=0.123E+4;
 gcvt(son,raqamlar_soni,satr);
 cout<<"0.123E+4 sonining satr ko'rinishi: "
 cout<<satr<<endl;
 son=12345.456;
 gcvt(son,raqamlar_soni,satr);
 cout<<"12345.456 sonining satr ko'rinishi: "
 cout<<satr<<endl;
 return 0;
}
```

Programma ekranga ketma-ket ravishda son ko‘rinishlarini chop etadi:

```
Son ko'rinishidagi raqamlat son: 4
3.154 sonining satr ko'rinishi: 3.154
-312.456 sonining satr ko'rinishi: -312.5
0.123E+4 sonining satr ko'rinishi: 1230
12345.456 sonining satr ko'rinishi: 1.235e+04
```

9-bob. 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";
//satrli o`zgarmas qiymati berish
s1="Qiymat berish 1-usul";
s2=str;       // char turidagi satr yuklanmoqda
s3='A';       // bitta belgi qiymat sifatida berish
s3=s3+s1+s2+"0123abc"; //qiymat sifatida satr ifoda
```

8.2-jadvalida string turidagi satrlar ustidan amallar keltirilgan.

Satr elementiga indeks vositasidan tashqari at() funksiyasi orqali murojaat qilish mumkin:

```
string s1="satr misoli";
cout<<s.at(3) // natijada 'r' belgisi ekranga chiqadi
```

Shuni aytib o‘tish kerakki, string sinfda shu turdagi o‘zgaruvchilar bilan ishlaydigan funksiyalar aniqlangan. Boshqacha aytganda, string turida e’lon qilingan o‘zgaruvchilar (ob’ektlar) o‘z funksiyalariga ega hisoblanadi va ularni chaqirish uchun oldin o‘zgaruvchi nomi, keyin ‘.’ (nuqta) va zarur funksiya nomi (argumentlari bilan) yoziladi.

8.2-jadval. string turidagi satrlar ustidan amallar

Amal	Mazmuni	Misol
=, +=	Qiymat berish amali	s="satr01234" s+="2satr000"
+	Satrlar ulash amali (konkantenatsiya)	s1+s2
==, !=, <, <=, >, >=	Satrlarni solishtirish amallari	s1==s2 || s1>s2 && s1!=s2
[]	Indeks berish	s[4]
<<	Oqimga chiqarish	sout << s
>>	Oqimdan o'qish	sin >> s (probelgacha)

Satr qismini boshqa satrga nusxalash funksiyasi

Bir satr qismini boshqa satrga yuklash uchun 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" kiritiladi
string s1,s2;
```

```
s2.assign(strold,6);   // s2="012345"
s3.assign(strold,20);  // s3="0123456789"
```

Satr qismini boshqa satrga qo‘shish funksiyasi

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" kiritiladi
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"
```

Satr qismini boshqa satr ichiga joylashtirish funksiyasi

Bir satrga ikkinchi satr qismini joylashtirish uchun quyidagi funksiyalar ishlatiladi:

```
insert(unsigned int pos1,const string &str);
insert(unsigned int pos1,const string & str,
       unsigned int pos2,unsigned int n);
insert(unsigned int pos1,const char *str, int n);
```

Bu fuksiyalar 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 kiritiladi
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 funksiyasi

Satr qismini o‘chirish uchun quyidagi funksiyani ishlatish mumkin:

```
erase(unsigned int pos=0,unsigned int n=npos);
```

Bu funksiya, uni chaqiruvchi satrning pos o‘rnidan boshlab n ta belgini o‘chiradi. Agarda pos ko‘rsatilmasa, satr boshidan boshlab o‘chiriladi. Agar n ko‘rsatilmasa, satrni oxirigacha bo‘lgan belgilar o‘chiriladi:

```
string s1,s2,s3;
s1="0123456789";
s2=s1;s3=s1;
s1.erase(4,5); // s1="01239"
s2.erase(3);   // s2="012"
s3.erase();    // s3=""
```

void clear() funksiyasi, uni chaqiruvchi satrni to‘liq tozalaydi.

Masalan:

```
s1.clear(); //satr bo‘sh hisoblanadi (s1="")
```

Satr qismini almashtirish funksiyasi

Bir satr qismining o‘rniga boshqa satr qismini qo‘yish uchun quyidagi funksiyalardan foydalanish mumkin:

```
replace(unsigned int pos1,unsigned int n1,
        const string & str);
replace(unsigned int pos1,unsigned int n1,
        const string & str,unsigned int pos2,
        unsigned int n2);
replace(unsigned int pos1,unsigned int n1,
       const char *str, int n);
```

Bu funksiyalar insert kabi ishlaydi, undan farqli ravishda amal chaqiruvchi satrning ko‘rsatilgan o‘rnidan (pos1) n1 belgilar o‘rniga str satrini yoki uning pos2 o‘rindan boshlangan n2 belgidan iborat qismini qo‘yadi (almashtiradi).

Misol:

```
char * sc="0123456789";
unsigned int i=3,j=2;
string s1,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 funksiyasi

Funksiya prototipi kuyidagicha:

```
string substr(unsigned int pos=0,
              unsigned int n=npos)const;
```

Bu funksiya, uni chaqiruvchi satrning pos o'rnidan boshlab n belgini natija sifatida qaytaradi. Agarda pos ko'rsatilmasa, satr boshidan boshlab ajratib olinadi, agar n ko'rsatilmasa, satr oxirigacha bo'lgan belgilar natija sifatida qaytariladi:

```
string s1,s2,s3;
s1="0123456789";
s2=s1; s3=s1;
s2=s1.substr(4,5); // s2="45678"
s3=s1.substr(3);    // s3="3456789"
// "30123456789" satr ekranga chiqadi
cout<<s1.substr(1,3)+s1.substr();
```

string turidagi satrni char turiga o'tkazish

string turidagi satrni char turiga o'tkazish uchun

```
const char * c_str()const;
```

funksiyani ishlatish kerak. Bu funksiya char turdagi '\0' belgisi bilan tugaydigan satrga o'zgarmas ko'rsatkichni qaytaradi:

```
shar *s1; string s2="0123456789";
s1=s2.c_str();
```

Xuddi shu maqsadda

```
const char * data()const;
```

funksiyasidan ham foydalanish mumkin. Lekin bu funksiya satr oxiriga '\0' belgisini qo'shmaydi.

Satr qismini izlash funksiyalari

string sinfida satr qismini izlash uchun har xil variantdagi funksiyalar aniqlangan. Quyida ulardan asosiylarining tavsifini keltiramiz.

```
unsigned int find(const string &str,
                  unsigned int pos=0)const;
```

Funksiya, uni chaqirgan satrning ko'rsatilgan joydan (pos) boshlab str satrni qidiradi va birinchi mos keluvchi satr qismining boshlanish indeksini

javob sifatida qaytaradi, aks holda maksimal musbat butun npos 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 s belgisini 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 s belgisini izlaydi;

```
unsigned int find_first_not_of(const string &str,
                           unsigned int pos=0)const;
```

Funksiya, uni chaqirgan satrning ko‘rsatilgan (pos) joydan boshlab str satrning birorta ham belgisi kirmaydigan satr qismini qidiradi va chap tomondan birinchi uchraganining indeksini, aks holda npos sonini qaytariladi.

```
unsigned int find_first_not_of(char c,
                           unsigned int pos=0)const;
```

Bu funksiyaning oldingidan farqi - satrdan s belgisidan farqli birinchi belgini izlaydi;

```
unsigned int find_last_not_of(const string &str,
                        unsigned int pos=npos)const;
```

Funksiya, uni chaqiruvchi satrning ko‘rsatilgan joydan boshlab str satrini tashkil etuvchi belgilar to‘plamiga kirmagan belgini qidiradi va eng o‘ng tomondan birinchi topilgan belgining indeksini, aks holda npos sonini qaytaradi.

```
unsigned int find_last_not_of(char c,
                          unsigned int pos=npos)const;
```

Bu funksiyaning oldingidan farqi - satr oxiridan boshlab s belgisiga o‘xshamagan belgini izlaydi.

Izlash funksiyalarini qo‘llashga misol:

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

Satrlarni solishtirish

Satrlar qismlarini solishtirish uchun compare funksiyasi ishlatiladi:

```
int compare(const string &str)const;
int compare(unsigned int pos1,unsigned int n1,
            const string & str)const;
int compare(unsigned int pos1,unsigned int n1,
            const string & str,unsigned int pos2,
            unsigned int n2)const;
```

Funksiyaning birinchi shaklida ikkita satrlar to‘la solishtiriladi: funksiya manfiy son qaytaradi, agar 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 ros2 o‘rnidan boshlab n2 ta belgili satr qismlari o‘zaro solishtiriladi.

Misol:

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

Masala. Familiya, ismi va shariflari bilan talabalar ro‘yxati berilgan. Ro‘yxat alfavit bo‘yicha tartiblansin.

Programma matni:

```
#include <iostream.h>
#include <alloc.h>
int main(int argc, char* argv[])
{const int FISh_uzunligi=50;
 string * Talaba;
 char * Satr=(char*)malloc(FISh_uzunligi);
 unsigned int talabalar_soni;
 char son[3];
 do
 {cout<<"Talabalar sonini kiriting: ";
  cin>>son;} while((talabalar_soni=atoi(son))<=0);
 Talaba =new string[talabalar_soni];
 cin.ignore();
 for(int i=0; i<talabalar_soni; i++)
 {
  cout<<i+1<<"-talabaning Familya ismi sharifi: ";
  cin.getline(Satr,50);
  Talaba[i].assign(Satr);
```

```
}
bool almashdi=true;
for(int i=0; i<talabalar_soni-1 && almashdi; i++)
{almashdi=false;
 for(int j=i; j<talabalar_soni-1; j++)
  if(Talaba[j].compare(Talaba[j+1])>0)
  {
   almashdi=true;
   strcpy(Satr,Talaba[j].data());
   Talaba[j].assign(Talaba[j+1]);
   Talaba[j+1].assign(Satr);
  }
 }
cout<<"Alfavit bo'yicha tartiblangan ro'yxat:\n";
for(int i=0;i<talabalar_soni;i++)
 cout<<Talaba[i]<<endl;
delete [] Talaba; free(Satr);
return 0;
}
```

Programmada talabalar ro‘yxati string turidagi Talaba dinamik massiv ko‘rinishida e’lon qilingan va uning o‘lchami foydalanuvchi tomonidan kiritilgan talabar_soni bilan aniqlanadi. Talabalar sonini kiritishda nazorat qilinadi: klaviaturadan satr o‘qiladi va u atoi() funksiyasi yordamida songa aylantiriladi. Agar hosil bo‘lgan son noldan katta son bo‘lmasa, sonni kiritish jarayoni takrorlanadi. Talabalar soni aniq bo‘lgandan keyin har bir talabaning familiya, ismi va sharifi bitta satr sifatida oqimdan o‘qiladi. Keyin, string turida aniqlangan compare() funksiyasi yordamida massivdagi satr-lar o‘zaro solishtiriladi va mos o‘rindagi belgilar kodlarini o‘sishi bo‘yicha «pufakchali saralash» orqali tartiblanadi. Programma oxirida hosil bo‘lgan massiv chop etiladi, hamda dinamik massivlar yo‘qotiladi.

Satr 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
```

10-bob. Strukturalar va birlashmalar

Strukturalar

Ma'lumki, biror predmet sohasidagi masalani yechishda undagi ob'ektlar bir nechta, har xil turdagi parametrlar bilan aniqlanishi mumkin. Masalan, tekislikdagi nuqta haqiqiy turdagi x- absissa va y-ordinata juftligi - (x,y) ko'rinishida beri-ladi. 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, mantiqiy turdagi talaba jinsi haqidagi ma'lumot va boshqalardan shakllanadi.

Programmada holat yoki tushunchani tavsiflovchi har bir berilganlar uchun alohida o'zgaruvchi aniqlab masalani yechish mumkin. Lekin bu holda ob'ekt haqidagi ma'lumotlar «tarqoq» bo'ladi, ularni qayta ishlash murakkablashadi, ob'ekt 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>
{
  <tur1 > <nom1>;
  <tur2 > <nom2>;
  ...
  <turn> <nomn>;
};
```

Bu erda <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 Manzil[50];
 bool status;
};
```

Programmada 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.Manzil,
"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 ob’ektning 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 erda <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 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_Manzili() funksiyasiga parametr sifatida berish uchun funksiya prototipi quyidagi ko‘rinishda bo‘lishi kerak:

```
void Talaba_Manzili(Talaba);
```

Funksiyaga strukturani argument sifatida uzatishga misol sifatidagi programmaning matni:

```
#include <iostream.h>
#include <string.h>
struct Talaba
{
 char FISh[30];
 unsigned int Tug_yil;
 unsigned int Kurs;
 char Yunalish[50];
 float Reyting;
 unsigned char Jinsi[5];
 char Manzil[50];
 bool status;
};
void Talaba_Manzili(Talaba);
int main(int argc,char* argv[])
{
 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.Manzil,
 "Toshkent, Yunusobod 6-3-8, tel: 224-45-78");
 talaba.Reyting=123.52;
 Talaba_Manzili(talaba);
 return 0;
}
void Talaba_Manzili(Talaba t)
{
 cout<<"Talaba FIO: "<<t.FIO<<endl;
 cout<<"Manzili: "<<t.Manzil<<endl;
}
```

Programma bosh funksiyasida talaba strukturasi aniqlanib, uning maydonlariga qiymatlar beriladi. Keyin talaba strukturasi Talaba_Manzili() funksiyasiga argument sifatida uzatiladi. Programma ishlashi natijasida ekranga quyidagi ma'lumotlar chop etiladi.

```
Talaba FIO: Abdullaev A.A.
Manzili: Toshkent, Yunusobod 6-3-8, tel: 224-45-78
```

Strukturalar massivi

O'z-o'zidan ma'lumki, struktura turidagi yagona 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 programmada guruhidagi har bir talaba haqidagi berilganlarni klaviaturadan kiritish va guruh talabalarini familiya, ismi va sharifini chop qilinadi.

```
#include <iostream.h>
#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 Manzil[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 manzili:";
  cin.getline(t[i].Manzil,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 programma 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 tartibla-nishi 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 berilganlar struktura ko‘rinishida, jadval esa struktura massivi sifati aniqlanadi:

```
struct Jamoa
{
 string Nomi;
 int Yutuq, Durang, Maglub, Urgan_tup, Utkazgan_tup;
 int Uyin, Ochko;
};
```

Bu erda 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.

Programma 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 katta
  // bo'lsa, takrorlashning keyingi qadamiga o'tilsin.
    if(jm[j].Ochko>jm[j+1].Ochko) continue;
  //j va (j+1)-jamoalarning ochkolari teng va j-jamoa
  // yutuqlari (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 soni
  // teng va j-jamoa urgan to'plar soni (j+1)- jamoa
  //urgan 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 pr=' ';
  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<<pr;cout<<jm[i].Uyin<<" |";
    if(jm[i].Yutuq<10)cout<<pr;cout<<jm[i].Yutuq<<" |";
    if(jm[i].Durang<10)cout<<pr;
    cout<<jm[i].Durang<<" |";;
    if(jm[i].Maglub<10)cout<<pr;
    cout<<jm[i].Maglub<<" |";
    if(jm[i].Urgan_tup<10)cout<<pr;
    cout<<jm[i].Urgan_tup<<" |";
    if(jm[i].Utkazgan_tup<10)cout<<pr;
    cout<<jm[i].Utkazgan_tup<<" |";;
    if(jm[i].Ochko<10)cout<<pr;
    cout<<jm[i].Ochko<<"  |"<<endl;
   }
  cout<<"-----------------------------------------\n";
  }
  int main()
  {
   Jamoa *jamoa;
   jamoa=Berilganlarni_kiritish();
   jamoa=Jadvalni_Tartiblash(jamoa);
   Jadvalni_Chop_Qilish(jamoa);
   return 0;
  }
```

Programma 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 sifa-tida 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, 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 programma ishlashining natijasi quyidagicha bo'lishi mumkin:

```
      FUTBOL JAMOALARINING TURNIR JADVALI
-----------------------------------------
|  JAMOA    | O | Y | D | M |UrT|O'T|OCHKO|
-----------------------------------------
|Bunyodkor |20 |15 | 3 | 2 |30 |10 | 48  |
|Paxtakor  |20 |11 | 5 | 4 |20 |16 | 38  |
|Neftchi   |20 | 8 | 5 | 7 |22 |20 | 29  |
-----------------------------------------
```

Dinamik strukturalar

Berilganlar ustida ishlashda ularning miqdori qancha bo'lishi va ularga xotiradan qancha joy ajratish kerakligi oldindan noma'lum bo'lishi mumkin. Programma 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 programma bajarilishida o'zgarib turadi. Programmalashda dinamik strukturalardan chiziqli ro'yxatlar (zanjirlar), steklar, navbatlar va binar daraxtlar nisbatan ko'p ishlatiladi. Ular bir - biridan elementlar o'rtasidagi bog'lanishlari va ular ustida bajariladigan amallari bilan farqlanadi. Programma ishlashida strukturaga yangi elementlar qo'shilishi yoki o'chirilishi mumkin.

Har qanday berilganlarning dinamik strukturasi maydonlardan tashkil topadi va ularning ayrimlari qo'shni elementlar bilan bog'lanish uchun xizmat qiladi.

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:

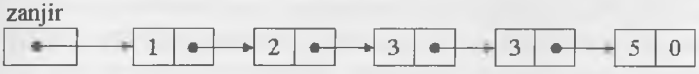
```
struct Zanjir
{
 int element;
 Zanjir * keyingi;
};
```

Programma matni:

```
#include <iostream.h>
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'yxatning oxirgi
                      // elementini topish
  temp->keyingi=yangi;// 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 oldingi
                       // elementga ko'rsatkich
  while (temp)
  {
   if (temp->element==del_elem)
   {
    if(oldingi)//o'chiriladigan element birinchi emas
    {
     // o'chiriladigan elementdan oldingi elementni
     // keyingi 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 o'chiriladigan songa teng emas
    {
     oldingi=temp;
     temp=temp->keyingi;
    }
   }
  }
 return z;
}
void Zanjir_Ekranga(Zanjir * z)
{
 cout<<"Zanjir elementlari:"<<endl;
 Zanjir * temp=z;
 while(temp)
 {
  cout<<temp->element<<' ';
  temp=temp->keyingi;
  }
 cout<<endl;
}
Zanjir * Zanjirni_Uchirish(Zanjir * z)
{
 Zanjir * temp=z;
 while(z)
 {
  temp=z;
  z=z->keyingi;
  delete temp;
 }
 return z;
}
int main()
{
 Zanjir * zanjir=0;
 int son, del_element;
 do
 {cout<<"\nSonni kiriting (0-jaryon tugatish): ";
  cin>>son;
  if(son) zanjir=Element_Joylash(zanjir,son);
 } while (son);
 Zanjir_Ekranga(zanjir);
 cout<<"\nO'chiriladigan elementni kiriting: ";
 cin>>del_element;
 zanjir= Element_Uchirish(zanjir,del_element);
 Zanjir_Ekranga(zanjir);
 Zanjir = Zanjirni_Uchirish(zanjir);
 return 0;
}
```

Programmaning bosh funksiyasida chiziqli ro‘yxat hosil qilish uchun Zanjir turidagi zanjir o‘zgaruvchisi aniqlangan bo‘lib, unga bo‘sh ko‘rsatkich qiymati 0 berilgan (uning ekvivalenti - NULL). Keyin takrorlash operatori tanasida klaviaturadan butun son o‘qiladi 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 qilish jarayoni tugaydi. Faraz qilaylik quyidagi sonlar ketma-ketligi kiritilgan bo‘lsin: 1,2,3,3,5,0. U holda hosil bo‘lgan ro‘yxat quyidagi ko‘rinishda bo‘ladi (10.1-rasm):



10.1-rasm. 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 adresi-ni 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 (10.2-rasm):

10.2-rasm. Ro‘yxatdan 3 sonini o‘chirilgandan keyingi ko‘rinish

O‘zgargan ro‘yxat elementlari ekranga chop etiladi. Programma oxirida, Zanjirni_Uchirish() funksiyasini chaqirish orqali ro‘yxat uchun dinamik ravishda ajratilgan xotira bo‘shatiladi (garchi bu ishning programma tugashi paytida bajarilishining ma’nosi yo‘q).

Dinamik strukturalarda o‘zgartirishlar (ro‘yxatga element qo‘shish yoki o‘chirish) nisbatan kam amallarda bajarilishi, ular vositasida masalalarni samarali yechishning asoslaridan biri hisoblanadi.

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 programmani matni keltirilgan.

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

Programma bosh funksiyasida bcd birlashmasining bayt o'lchamida bayt maydoniga 127 qiymati beriladi va uning katta va kichik yarim baytlari chop etiladi.

Programma 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 (1-ilovaga qarang). 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.

Programma matni:

```
#include <iostream.h>
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 10000000=128;
 for(int i=0;i<=bitlar_soni;i++)
```

```
 {
  if(blg&10000000)cout<<'1';
  else cout<<'0';
  blg=blg<<1;
 }
 cout<<' ';
}
```

Programmada 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.

Programmaga -8.5 soni kiritilsa, ekranda

```
11000001 00001000 00000000 00000000
```

ko'rinishidagi ikkilik sonlari ketma-ketligi paydo bo'ladi.

Foydalanuvchi tomonidan aniqlangan berilganlar turi

C++ tilida foydalanuvchi tomonidan nafaqat struktura yoki birlashma turlari, balki ayni paytda mavjud (aniqlangan) turlar asosida yangi turlarni yaratishi mumkin.

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, programmada 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 echilayotgan 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;
```

yozish mumkin va uning maydonlariga murojaat qilish mumkin:

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

11-bob. Makroslar

Makroslarni aniqlash va joylashtirish

Makros - bu programma (kod) bo‘lagi bo‘lib, ko‘rinishi va ishlashi xuddi funksiyadek. Biroq u funksiya emas. Funksiyalar va makroslar o‘rtasida bir nechta farqlar mavjud:

– programma matnida uchragan makros ifodasi o‘z aniqlanishi (tanasi bilan) bilan preprotsessor ishlash paytida, ya’ni programma kompilyasiyasidan oldin almashtiriladi. Shu sababli makros funksiyani chaqirish bilan bog‘liq qo‘shimcha vaqt sarfini talab qilmaydi;

– makroslardan foydalanish programmaning boshlang‘ich kodi (matnini) kattalashuviga olib keladi. Bunga qarama-qarshi holda funksiya kodi yagona nusxada bo‘ladi va u programma 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 programma 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 makroslardan 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
```

Programma matnida ushbu makros ishlatilgan satr mavjud bo‘lsin:

```
int y=KVADRAT(2);
```

u holda, makros aniqlanishini matnga qo‘yish natijasida programma matnida yuqoridagi satr quyidagi ko‘rinishga keladi:

```
int y=2*2;
```

Lekin, programmada 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 programma 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 programma matniga qo‘yilada. Natijada kompilyasiya 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 programma matnining ixtiyoriy joyida #undef direktivasi bilan bekor qilish mumkin, ya’ni shu satrdan keyin makros preprotsessor uchun noaniq hisoblanadi. Quyida aylana yuzasini hisoblaydigan programma 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;
}
```

Programma kompilyasiyasida “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;
}
```

Programma 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.

Misol uchun,

```
#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;
}
```

programmmasi 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.

12-bob. O‘qish - yozish funksiyalari

Fayl tushunchasi

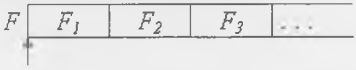
C++ tilidagi standart va foydalanuvchi tomonidan aniqlangan turlarning muhim xususiyati shundan iboratki, ularning oldindan aniqlangan miqdordagi chekli elementlardan iboratligidir. Hatto berilganlar dinamik aniqlanganda ham, operativ xotiraning (uyumning) amalda cheklanganligi sababli, bu berilganlar 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, programmada e’lon qilingan o‘zgaruvchilarning qiymatlari sifatida aniqlangan berilganlar faqat programma ishlash paytidagina mavjud bo‘ladi va programma o‘z ishini tugatgandan keyin yo‘qolib ketadi. Agar programma 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 programmada xodimlar ro‘yxatini, shtat stavkalari va xodimlar tomonidan olingan maoshlar haqidagi ma’lumotlarni doimiy ravishda saqlab turish zarur. Bu talablarga fayl turidagi ob’ektlar (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 etarlicha uzun magnit tasmasiga o‘xshatish mumkin.

12.1-rasm. Fayl tasviri

12.1-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.



12.2-rasm. Fayl ko‘rsatkichi

Yana bir muhim tushunchalardan biri fayl ko‘rsatkichi tushunchasidir. *Fayl ko‘rsatkichi* - ayni paytda fayldan o‘qilayotgan yoki unga yozilayotgan joy (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. 12.2-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 sistema 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.spp”, “A:\\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 S 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 programma 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

12.3-rasm. To‘rtta satrdan tashkil topgan matn fayli

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.

O‘qish-yozish oqimlari. Standart oqimlar

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.

Programma 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 programmaga kiritish (o‘qish) mumkin, ekranga esa chiqarish (yozish) mumkin. Programma 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 adreslash ro‘y berdi* deyiladi. O‘qish uchun qayta adreslashda ‘<’ belgisidan, yozish uchun esa ‘>’ belgisidan foydalaniladi. Misol uchun gauss.exe bajariluvchi programma 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 programma 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 programmaning chiqish oqimini ikkinchi programmaning kirish oqimi bilan bog'lash mumkin. Buni *konveyrli jo'natish* deyiladi. Agar ikkita junat.exe programmasi qabul.exe programmasiga berilganlarni jo'natishi kerak bo'lsa, u holda ular o'rtasiga '|' belgi qo'yib yoziladi:

```
junat.exe | qabul.exe
```

Bu ko'rinishdagi programmalar 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.h>
#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 c, 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 qo‘yidagi programmani 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 S tilidan vorislik bilan olingan. Bu funksiyalarni ishlatish uchun «stdio.h» sarlavha faylini programmaga 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 akslanti-riladi va o‘zgaruvchi bilan nomlangan xotira bo‘lagiga joylashtiriladi (saqlanadi). Funksiya oqimdan berilganlarni o‘qish jarayo-nini «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.

12.1–jadval. 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‘ri–nishini 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, nati-ja 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 12.1–jadvalda keltirilgan shart yoki shart bo‘lmagan komponentalar keladi.

12.2–jadval. Almashtiriladigan tur alomati belgilari

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
I	O‘nlik, sakkizlik va o‘n oltilik butun son	int * arg
I	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 almashtirishlar baja-rilmaydi, ‘%’ 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:ZZZZ yoki ZZZZ ko‘rinishidagi o‘n oltilik	Ob’ektga ko‘rsatkich (far* yoki near*).

Oqimdagi belgilar bo‘lagini almashtiriladigan tur alomatining qabul qilishi mumkin bo‘lgan belgilar 12.2-jadvalda keltirilgan.

12.3–jadval. 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. Sondagi

		raqamlarning 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.

12.4–jadval. 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.dddd yoki 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	int ko‘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 ob’ektlardan tashkil topadi:

– oqimga o‘zgarishsiz chiqariladigan oddiy belgilar;

– argumentlar ro‘yxatidagi tanlanadigan argumentga qo‘llaniladigan format aniqlashtiruvchilari.

Format aniqlashtiruvchisi quyidagi ko‘rinishga ega:

% [<bayroq>][<.kenglik>] [.<xona>][F|N|h|l|L] <tur belgisi>

Format aniqlashtiruvchisi ‘%’ belgisidan boshlanadi va undan keyin 12.3–jadvalda keltirilgan shart yoki shart bo‘lmagan komponentalar keladi.

Almashtiriladigan tur belgisining qabul qilishi mumkin bo‘lgan belgilar 12.4- jadvalda keltirilgan.

Berilganlar qiymatlarini oqimdan o‘qish va oqimga chiqarishda scanf() va prinf() 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;
}
```

Programma 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.

Fayldan o‘qish-yozish funksiyalari

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 12.5-jadvalda keltirilgan.

12.5-jadval. Fayl ochish rejimlari

Mode 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+	Faylga yozuvni 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\\iat1kuz.txt» nomli matn faylni o‘qish uchun ochish zarur bo‘lsin. Bu talab

```
FILE *f=fopen("C:\\USER\\TALABA\\iat1kuz.txt","r+");
```

ifodasini yozish orqali amalga oshiraladi. Natijada diskda mavjud bo‘lgan fayl programmada f o‘zgaruvchisi nomi bilan aynan bir narsa deb tushuniladi. Boshqacha aytganda, programmada keyinchalik f ustida bajarilgan barcha amallar, diskdagi «iat1kuz.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.

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.

fputc() funksiyasi

```
int fputc(int c, FILE *stream);
```

prototipi bilan aniqlangan. fputc() funksiyasi fayl oqimiga argumentda ko‘rsatilgan belgini yozadi (chiqaradi) va u amal qilishida putc() 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.

foef() 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.h>
#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;
}
```

Programmada «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 programma o‘z ishini tugatadi. Keyinchalik, toki in fayli oxiriga etmaguncha, undan belgilar o‘qiladi va ekranga, hamda out fayliga chiqariladi. Programma 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.

Programma 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;
}
```

Programmada 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 tomoni-dan 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 programmada binar fayl bilan ishlash ko‘rsatilgan.

```
#include <iostream.h>
#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 programmada, 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 shaxs2 strukturalar massiviga o‘qiladi va ekranga chop qilinadi. Programmada 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 klavia-turadan 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.

Programma 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 fayl elementlari yig'indisi
 while (!feof(f))
 {
  if (fread(&x,sizeof(double),1,f)) s+=x;
 }
 long sonlar_miqdori=ftell(f)/sizeof(double);
 s/=sonlar_miqdori; // s- o'rta arifmetik
 cout<<"Fayldagi sonlar o'rta 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 elementlar
            //soni
 }
 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;
}
```

Programmada 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‘xta-tiladi 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.

Adabiyotlar

1. Б. Страуструп. Язык программирования С++. Специальное издание.-М.:ООО «Бином-Пресс», 2006.-1104 с.
2. Глушаков С.В., Коваль А.В., Смирнов С.В. Язык программирования С++: Учебный курс.- Харьков: Фолио; М.: ООО «Издательство АСТ», 2001.-500с.
3. Павловская Т.А. С++. Программирование на языке высокого уровня - СПб.: Питер. 2005.- 461 с.
4. Подбельский В.В. Язык СИ++.- М.; Финансы и статистика- 2003 562с.
5. Павловская Т.С. Щупак Ю.С. С/С++. Структурное программирование. Практикум.-СПб.: Питер,2002-240с
6. Павловская Т.С. Щупак Ю.С. С++. Объектно-ориентированное программирование. Практикум.-СПб.: Питер,2005-265с
7. Юров В., Хорошенко С. Assembler: Учебный курс- СПб, “Питер”,2000.-672с.
8. Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн М.И. Задачи по программированию.-М.: Наука, 1988.-224с.
9. А.А. Абдукодиров, У.М.Мирзаев СИ тилида программалаш асослари. Ўкув қўлланма, Тошкент, «Университет», 1994.-52 бет.
10. A.A.Xaldjigitov, Sh.F.Madraximov, U.E.Adamboev Informatika va programmalash. O’quv qo’llanma, O’zMU, 2005 yil, 145 bet.
11. A.A.Xaldjigitov, Sh.F.Madraximov, A.M.Ikromov, S.I.Rasulov Pascal tilida programmalash bo’yicha masalalar to’plami. O’quv qo’llanma, O’zMU, 2005 yil, 94 bet.

Ilovalar

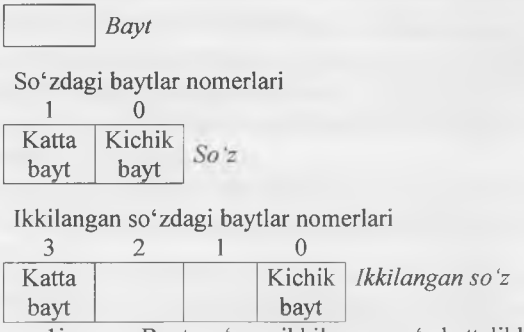
1-ilova

Berilganlarning kompyuter xotirasidagi ichki ko‘rinishi

Kompyuterning joriy (operativ) xotirasi katta sondagi, ikkita holatlarni eslab qolish elementlaridan va ularni boshqarish sxemalaridan iborat bo‘lgan elektron qurilmadir. Xotiradagi murojaat qilish mumkin bo‘lgan eng kichik ma’lumot birligi bayt (8 ikkilik razryad, yoki bitlar).

Ayrim berilganlarni xotirada saqlash uchun bir bayt etarlidir, masalan belgilar kodlarini, boshqalari uchun 2, 4, 8 baytlar talab qilinishi mumkin. Shu sababli berilganlarni xotirada saqlash uchun *so‘z* (2 bayt), *ikkilangan so‘z* (4 bayt) tushunchalari kiritilgan.

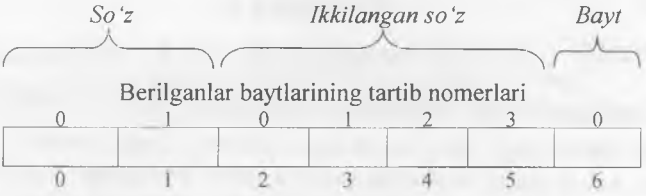
Ko‘p baytli berilganlarni qayta ishlashda ularning ichki baytlariga murojaat qilishga to‘g‘ri keladi: bu baytlar shartli ravishda noldan boshlab nomerlanadi va o‘ngdan chapga joylashadi (ularning qog‘ozdagi ko‘rinishida). O‘ngdagi (nolinchi) bayt - *kichik bayt*, chapdagi oxirgi bayt - *katta bayt* deb nomlanadi (1i-rasm).



1i- rasm. Bayt, so‘z va ikkilangan so‘z kattaliklari

Umuman olganda xotirada faqat butun ikkilik sonlarni saqlash mumkin. Boshqa turdagi berilganlar uchun, masalan belgi va kasr sonlar uchun kodlash qoidasi ko‘zda tutilgan.

Shuni ta’kidlab o‘tish kerakki, berilganlar baytlari xotirada joylashishi quyidagicha: har bir so‘z yoki ikkilangan so‘z xotirada kichik baytdan boshlanadi va katta bayt bilan tugaydi (2i-rasm).



Xotiradagi baytlar ketma-ketliklarining nomerlari

2i- rasm. Ko'pbaytli berilganlarning baytlarining nomerlari

Kompyuterning raqamli elektron qurilmalari amal qiladigan ikkilik sanoq sistemasi bilan ishlash foydalanuvchi uchun noqulay. Xotiradagi, registrlardagi berilganlarini ifodalash uchun ayrim hollarda 8 sanoq sistemasi, asosan 16 sanoq sistemasi ishlatiladi. Bunda bayt qiymat ikkita 16 sanoq sistemasidagi raqam bo'lgan ifodalanadi: 00h sonidan FFh sonigacha, bu erda h- sonning 16 sanoq sistemasida tasvirlanganini bildiradi. So'z to'rtta 16 sanoq sistemasidagi raqam bilan ifodalanadi (0000h...FFFFh oralig'idagi sonlar, 10 sanoq sistemasida 0...65535).

Ishorasiz butun sonlar xotirada ikkilik sanoq sistemasida yozilib, bayt, so'z, ikkilik so'z, to'rtlik so'z ko'rinishida yozilishi mumkin. Masalan, $98_{10}=62_{16}=01100010_2$. Bu erda indeks sanoq sistemasining asosi. Ushbu son bitta baytdagi ko'rinishida quyidagicha:

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

3i- rasm. 98 sonining baytdagi ikkilik ko'rinishi

Odatda bitta baytdagi son ikkita o'n oltilik raqam bilan ko'rsatiladi (62_{16}). Agar, 1100010_2 sonini ikki baytda (so'zda) tasvirlash zarur bo'lsa, uning katta razryadlari 0 bilan to'ldiriladi.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0

4i- rasm. 98 sonining so'zdagi ikkilik ko'rinishi

Quyidagi jadvalda egallagan bayt o'lchamiga mos butun ishorasiz sonlarning qiymat chegaralari ko'rsatilgan (1i-jadval).

1i-jadval. Ishorasiz butun son turlari

Bitlar soni	O'lchami	Tur	Qiymat chegarasi
8	Bayt	unsigned char	0 .. 255
16	So'z	unsigned int	0 .. 65535

32	Ikkilik so‘z	unsigned long	0 .. 4294967295
64	To‘rtlik so‘z	unsigned int64	0..18446744973709551615

O‘lchami baytdan katta turlarda ishorasiz son teskari ko‘rinishda saqlanadi, ya’ni, oldin kichik baytlar keyin katta baytlar joylashadi. Masalan, so‘z ko‘rinishidagi 0062_{16} sonining kompyuter xotirasidagi joylashuvi quyidagicha bo‘ladi:

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0
A - adresli bayt								A+1 adresli bayt							

5i- rasm. So‘zdagi 98 sonining xotirada joylashuvi

Ishorali butun sonlar kompyuter xotirasida qo‘shimcha kod ko‘rinishida saqlanadi. Musbat butun sonlar ishorasiz sonlar kabi yoziladi. Manfiy x soni esa 2^k -|x| ishorasiz son ko‘rinishida yoziladi, bu erda k - ajratilgan o‘lchamdagi bitlar soni.

Masalan, bir baytda joylashgan -98_{10} (-62_{16}) sonini qo‘shimcha kodining ko‘rinishi:

- 10 sanoq sistemasida: $2^8 - |-98_{10}| = 256 - 98 = 158$;
- 16 sanoq sistemasida: $100_{16} - |-62_{16}| = 9E_{16}$;
- 2 sanoq sistemasida: $100000000_2 - 01100010_2 = 10011110_2$.

Ishorali butun son yozilgan baytning katta razryadi (7 razryadi) son ishorasining alomati hisoblanadi. Agar 7-razryadda 1 bo‘lsa baytda qo‘shimcha koddagi manfiy son saqlanayapti, aks holda baytda musbat son joylashgan deb hisoblanadi.

Agar -62_{16} soni so‘z kattaligida bo‘lsa, u $(10000_{16}-62_{16})=FF9E_{16}$ soniga teng bo‘ladi (6i.a-rasm) va xotirada teskari ko‘rinishda saqlanadi (6i.b- rasm).

a)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0

b)

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1
A- adresli bayt								A+1 adresli bayt							

6i- rasm. So‘zdagi 98 sonining xotirada joylashuvi

Qo‘shimcha kodni topish boshqa usuli ham mavjud: oldin manfiy sonni ishorasiz ko‘rinishi 2 sanoq sistemasida yoziladi, keyin har bir razryaddagi 0 raqam 1 raqamiga, 1 raqami esa 0 almashtiriladi. Hosil bo‘lgan songa 1 qo‘shiladi. Misol uchun shu usulda 98_{10} sonini qo‘shimcha kodi quyidagicha topiladi:

$98_{10} = 62_{16} = 01100010_2 \rightarrow 10011101_2 + 1_2 = 10011110_2 = 9E_{16}$.

Quyidagi jadvalda bayt o‘lchamidagi sonlarning kompyuter xotirasidagi ichki ko‘rinishiga misollar keltirilgan (2i-jadval).

2i-jadval. Bayt o‘lchamidagi sonlarning ichki ko‘rinishi

10 s/s son	2 s/s kod	10 s/s son	2 s/s qo‘shimcha kod
0	00000000	-1	11111111
1	00000001	-2	11111110
2	00000010	-3	11111101
3	00000011	-126	10000010
126	01111110	-127	10000001
127	01111111	-128	10000000

Ishorali butun son turiga mos ravishda manfiy va musbat qiymatlar chegarasi mavjud (3i-jadval).

3i-jadval. Ishorali son qiymatlar chegarasi

Bitlar soni	O‘lchami	Turi	CHegarasi
8	Bayt	char	-128 ... +127
16	So‘z	int	-32768 ... +32767
32	ikkilik so‘z	long int	-2147483648... +2147483647
64	to‘rtlik so‘z	int64	-4294967296... +4294967295

Haqiqiy sonlar xotirada ikkilik sanoq sistemasida normallashgan eksponensial shaklda saqlanadi.

Ikkilik sanoq sistemasidagi normallashgan son deb, butun qismi doimo 1 teng, kasr qismi - mantissa (M) va eksponenta deb nomlanuvchi darajasi (tartibi p) bilan tasvirlangan songa aytiladi. Masalan 111.01_2 soning normal ko‘rinishi $1.1101*10^{10}_2$ teng. Bu erda $M=0.1101_2$ $p=10_2$ qiymatiga teng.

Intel protsessorlari uchun normallashgan son

$$A=(-1)^s*M*N^q$$

ko‘rinishda bo‘ladi.

Bu erda:

s - son ishorasi aniqlovchi razryad qiymati. Agar s=0 bo‘lsa, son musbat, s=1 holda son manfiy ekanligini bildiradi;

M mantissa va u $0 \le M < 1$ shartni qanoatlantiradi;

N - sanoq sistema asosi (N=2);

q - xarakteristika.

Xarakteristika son tartibi p bilan quyidagi munosabatda bo‘ladi: q=p+*fiksirlangan siljish*. Yuqorida qayd qilingan uchta formatning har biri uchun *fiksirlangan siljish* tur-licha bo‘ladi. Odatda u 2^{k-1}-1 qiymatiga teng bo‘ladi. Bu erda k - xarakteristika uchun ajratilgan razryadlar soni. Normallashgan sonning butun qismidagi raqam doimiy ravishda 1 bo‘lgani

uchun u katakka yozil-maydi va bu holat son ustida amal bajarishda apparat darajasida inobatga olinadi.

IEEE 754 standarti bo'yicha haqiqiy sonlar ikkilik sanoq sistemasida uchta formatda saqlanadi: qisqa format; uzun format () va kengaytirilgan format (80 bitlik):

- qisqa format, 32 bitlik, siljish - $127_{10}=7F_{16}$ (7i.a-rasm);
- uzun format, 64 bitlik, siljish - $1023_{10}=3FF_{16}$(7i.b-rasm);
- kengaytirilgan format, 80 bitlik, siljish -$16383_{10}=3FFF_{16}$(7i.d-rasm).

1 bit	8 bit	23 bit
Ishora (s)	Xarakteristika (q)	Mantissa (M)
31	30 23	22 0

a)

1 bit	11 bit	52 bit
Ishora (s)	Xarakteristika (q)	Mantissa (M)
63	62 52	51 0

b)

1 bit	15 bit	64 bit
Ishora (s)	Xarakteristika (q)	Mantissa (M)
79	78 64	63 0

d)

7i- rasm. Haqiqiy sonning ichki formatlari

Quyidagi jadvalda haqiqiy son formatlarining chegaralari berilgan (4i-jadval).

4i-jadval. Haqiqiy son formatlarining chegaralari

Berilgan formati	**Qiymat chegarasi**	**Aniqligi (o'nlik raqamda)**
Qisqa format	3.4×10^{-38} ... $3.4\times10^{+38}$	7
Ikkilangan aniqlik	1.7×10^{-308} ... $1.7\times10^{+308}$	16
Kengaytirilgan format	3.4×10^{-4932} ... $3.4\times10^{+4932}$	19

Misol sifatida 0.5_{10} va -8.5_{10} sonlarining xotiradagi ichki ko'rinishi aniqlaylik:

1) $0.5_{10}=0.1_2=1.0*10^{-1}{}_2$: s=0, $M=1.0_2$, $p=-1_{10}$, $q=p+127_{10}=126_{10}=1111110_2=7E_{16}$.

Ushbu sonning qisqa formatdagi ko'rinishi quyidagicha bo'ladi:

S	Q	M
0	0 1 1 1 1 1 1 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
31	30 23	22 0

8i- rasm. 0.5 soning qisqa formatdagi ichki ko'rinishi

2) $-8.5_{10}=-1.0001*10^{111}{}_2$: s=1, $M=1.0001_2$, $p=3_{10}$, $q=p+127_{10}=130_{10}=10000010_2=82_{16}$.

Ushbu sonning qisqa formatdagi ko‘rinishi quyidagicha bo‘ladi:

s	q	M
1	1 0 0 0 0 0 1 0	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
31	30 23	22 0

9i- rasm. -8.5 soning qisqa formatdagi ichki ko‘rinishi

Belgilar xotirada bir bayt joy egallaydi va har bir belgi o‘zini ikkilik ASCII (2-ilova) kodi bilan yoziladi. Unda maksimal ravishda 256 belgi aniqlanishi mumkin. Windows tizimida ikki bayt ajratilgan Unicode kod tizimi kiritilgan. Bu tizimda har milliy alfavitlar uchun 256 belgidan iborat bo‘lgan va maxsus nomerlangan kodlashlar kiritilgan.

Satr bu - belgilar ketma-ketligi va u xotirada ham xuddi shunday ketma-ketlikdagi baytlarda joylashadi. Masalan, «Bu satr» satri xotirada quyidagicha yoziladi:

B	u		s	a	t	r	\0
A	A+1	A+2	A+3	A+4	A+5	A+6	A+7

9i- rasm. ASCIIZ catrning xotiradagi ichki ko‘rinishi

Bu erda A - satr boshlanishining (uning kichik baytining) adresi.

2-ilova

ASCII kodlar jadvallari

5i-jadval. 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
gs	29	1D	^]	Guruh ajratuvchi
rs	30	1E	^^	Yozuvlar ajratuvchisi
us	31	1F	^_	Modullar ajratuvchisi

6i-jadval. 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

7i-jadval. 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
Г	129	81	¬	172	AC	CH	215	D7
’	130	82	-	173	AD	Sh	216	D8
г	131	83	®	174	AE	Щ	217	D9
„	132	84	Ї	175	AF	‘	218	DA
…	133	85	°	176	B0	Ы	219	DB
†	134	86	±	177	B1		220	DC
‡	135	87	І	178	B2	E	221	DD
€	136	88	і	179	B3	Yu	222	DE
‰	137	89	г	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	ѓ	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	ј	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
Oʻ	161	A1	M	204	CC	ch	247	F7
oʻ	162	A2	N	205	CD	sh	248	F8
J	163	A3	O	206	CE	щ	249	F9
¤	164	A4	P	207	CF	‘	250	FA
Г	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			

3-ilova

8i-jadval. Matematik funksiyalar kutubxonasi (math.h)

Funksiya prototipi	Bajaradigan amali
int abs(int i)	i sonni absolyut qiymatini qaytaradi
double acos(double x)	Radianda berilgan x argumentni arkko-sinus qiymatini qaytaradi
double asin(double x)	Radianda berilgan x argumentni arksinus qiymatini qaytaradi
double atan(double x)	Radianda berilgan x argumentni ark-tangens qiymatini qaytaradi
double atan2(double x, double y)	Radianda berilgan x/y nisbatning ark-tangensi 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 u soniga bo‘lish natijasidagi qoldiqni qaytaradi. % amaliga o‘xshagan, faqat haqiqiy son qaytaradi
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 gipotenuzani 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
double log10(double x)	x soning 10 asosli logarifmini qaytaradi
double modf(double x, double *intptr)	x sonining kasr qismini qaytaradi va butun qismini intptr adresga joylaydi
double poly(double x, int n, double c[])	$c[n]x^n+c[n-1]x^{n-1}+\ldots+c[1]x+c[0]$ poli-nomni qiymatini hisoblaydi

double pow(double x, double y)	x^y hisoblaydi
double pow10(int p)	10^p hisoblaydi
double sin(double x)	x radianga teng bo‘lgan burchakni sinusini qaytaradi
double sinh(double x)	x radianga teng bo‘lgan burchakni giperbolik sinusini qaytaradi
double sqrt(double x)	x sonining kvadrat ildizini qaytaradi
double tan(double x)	x radianga teng bo‘lgan burchakni giperbolik kosinusini qaytaradi
double tanh(double x)	x radianga teng bo‘lgan burchakni giperbolik kosinusini qaytaradi