ЎЗБЕКИСТОН РЕСПУБЛИКАСИ
ОЛИЙ ВА МАХСУС ТАЪЛИМ ВАЗИРЛИГИ
МИРЗО УЛУҒБЕК НОМИДАГИ
ЎЗБЕКИСТОН МИЛЛИЙ УНИВЕРСИТЕТИ

Мадрахимов Ш.Ф., Гайназаров С.М.

C++ тилида программалаш асослари

Тошкент 2009

Аннотация

Қўлланмада C++ тилининг синтаксиси, семантикаси ва ундаги программалаш технологиялари келтирилган. Айниқса, C++ тили асосини ташкил этувчи берилганлар ва уларнинг турлари, операторлар, функциялар, кўрсаткичлар ва массивлар, ҳамда берилганлар оқимлари билан ишлаш тушунарли равишда баён қилинган ва содда мисолларда намуналар келтирилган.

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.

Тузувчилар: доцент Ш.Ф.Мадрахимов
доцент в.б. С.М.Гайназаров

Тақризчилар: ТДПУ Ахборот маркази директори, т.ф.н.,доц. Якубов А.Б.
ЎзМУ доценти Н.А.Игнатьев

Масъул муҳаррир: ЎзМУ профессори М.М.Арипов

Ушбу услубий қўлланма Мирзо Улуғбек номидаги Ўзбекистон Миллий университети Илмий кенгашининг 2008 йил 29 декабрда ўтказилган мажлисида нашрга тавсия этилган (5-сонли баённома)

Мундарижа

Кириш

Маълумки, программа машина кодларининг қандайдир кетма-кетлиги бўлиб, аниқ бир ҳисоблаш воситасини амал қилишини бошқаради. Программа таъминотини яратиш жараёнини осонлаштириш учун юзлаб программалаш тиллари яратилган. Барча программалаш тилларини икки тоифага ажратиш мумкин:

- қуйи даражадаги программалаш тиллари;
- юқори даражадаги программалаш тиллари.

Қуйи даражадаги программалаш тилларига Ассемблер туридаги тиллар киради. Бу тиллар нисбатан қисқа ва тезкор бажарилувчи кодларни яратиш имкониятини беради. Лекин, Асссемблер тилида программа тузиш заҳматли, нисбатан узоқ давом этадиган жараёндир. Бунга қарама-қарши равишда юқори босқич тиллари яратилганки, уларда табиий тилнинг чекланган кўринишидан фойдаланган ҳолда программа тузилади. Юқори босқич тилларидаги операторлар, берилганларнинг турлари, ўзгарувчилар ва программа ёзишнинг турли усуллари тилнинг ифодалаш имконияти оширади ва программани «ўқимишли» бўлишини таъминлайди. Юқори босқич тилларига Fortran, PL/1, Prolog, Lisp, Basic, Pascal, C ва бошқа тилларни мисол келтириш мумкин. Компьютер архитектурасини такомиллашуви, компьютер тармоғининг ривожланиши мос равишда юқори босқич тилларини янги вариантларини юзага келишига, янги тилларни пайдо бўлишига, айрим тилларни эса йўқолиб кетишига олиб келди. Ҳозирда кенг тарқалган тилларга Object Pascal, C++, C#, Php, Java, Asp тиллари ҳисобланади. Хусусан, C тилининг такоммиллашган варианти сифатида C++ тилини олишимиз мумкин. 1972 йилда Денис Ритч ва Брайан Кернеги томонидан C тили яратилди. 1980 йилда Бьярн Страустроп C тилининг авлоди C++ тилини яратдики, унда структурали ва объектга йўналтирилган программалаш технологиясига таянган ҳолда программа яратиш имконияти туғилди.

1- боб. C++ тили ва унинг лексик асоси

C++ тилидаги программа тузилиши ва унинг компиляцияси

C++ тилида программа яратиш бир нечта босқичлардан иборат бўлади. Дастлаб, матн таҳририда (одатда программалаш муҳитининг таҳририда) программа матни терилади, бу файлнинг кенгайтмаси «.cpp» бўлади, Кейинги босқичда программа матн ёзилган файл компиляторга узатилади, агарда программада хатоликлар бўлмаса, компилятор «.obj» кенгайтмали объект модул файлини ҳосил қилади. Охирги қадамда компоновка (йиғувчи) ёрдамида «.exe» кенгайтмали бажарилувчи файл - программа ҳосил бўлади. Босқичларда юзага келувчи файлларнинг номлари бошланғич матн файлининг номи билан бир хил бўлади.

Компиляция жараёнининг ўзи ҳам иккита босқичдан ташкил топади. Бошида препроцессор ишлайди, у матндаги компиляция директиваларини бажаради, хусусан #include директиваси бўйича кўрсатилган кутубхоналардан C++ тилида ёзилган модулларни программа таркибига киритади. Шундан сўнг кенгайтирилган программа матни компиляторга узатилади. Компилятор ўзи ҳам программа бўлиб, унинг учун кирувчи маълумот бўлиб, C++ тилида ёзилган программа матни ҳисобланади. Компилятор программа матнини лексема (атомар) элементларга ажратади ва уни лексик, кейинчалик синтаксик таҳлил қилади. Лексик таҳлил жараёнида у матнни лексемаларга ажратиш учун «пробел ажратувчисини» ишлатади. Пробел ажратувчисига - пробел белгиси (‘␣’), ‘\t’ - табуляция белгиси, ‘\n’- кейинги қаторга ўтиш белгиси, бошқа ажратувчилар ва изоҳлар ҳисобланади.

Программа матни тушунарли бўлиши учун изоҳлар ишлатилади. Изоҳлар компилятор томонидан «ўтказиб» юборилади ва улар программа амал қилишига ҳеч қандай таъсир қилмайди.

C++ тилида изоҳлар икки кўринишда ёзилиши мумкин.

Биринчисида “**/***” дан бошланиб, “***/**” белгилар оралиғида жойлашган барча белгилар кетма-кетлиги изоҳ ҳисобланади, иккинчиси «сатрий изоҳ» деб номланади ва у “**//**” белгилардан бошланган ва сатр охиригача ёзилган белгилар кетма-кетлиги бўлади. Изоҳнинг биринчи кўринишида ёзилган изоҳлар бир неча сатр бўлиши ва улардан кейин C++ операторлари давом этиши мумкин.

Мисол.

```
int main()
```

```
{
 // бу қатор изоҳ ҳисобланади
 int a=0; // int d;
 int c;
 /* int b=15 */
 /* - изоҳ бошланиши
 a=c;
 изоҳ тугаши */
 return 0;
}
```

Программада d, b ўзгарувчилар эълонлари инобатга олинмайди ва a=c амали бажарилмайди.

Қуйида C++ тилидаги содда программа матни келтирилган.

```
# include <iostream.h> // сарлавҳа файлни қўшиш
int main ()              // бош функция тавсифи
{                        // блок бошланиши
 cout << "Salom Olam!\n";// сатрни чоп этиш
 return 0;             // функция қайтарадиган қиймат
}                        // блок тугаши
```

Программа бажарилиши натижасида экранга “Salom Olam!” сатри чоп этилади.

Программанинг 1-сатрида #include.. препроцессор директиваси бўлиб, программа кодига оқимли ўқиш/ёзиш функциялари ва унинг ўзгарувчилари эълони жойлашган «iostream.h» сарлавҳа файлини қўшади. Кейинги қаторларда программанинг ягона, асосий функцияси - main() функцияси тавсифи келтирилган. Шуни қайд этиш керакки, C++ программасида албатта main() функцияси бўлиши шарт ва программа шу функцияни бажариш билан ўз ишини бошлайди.

Программа танасида консол режимида белгилар кетма-кетлигини оқимга чиқариш амали қўлланилган. Маълумотларни стандарт оқимга (экранга) чиқариш учун қуйидаги формат ишлатилган:

cout << <ифода>;

Бу ерда <ифода> сифатида ўзгарувчи ёки синтаксиси тўғри ёзилган ва қандайдир қиймат қабул қилувчи тил ифодаси келиши мумкин (*кейинчалик, бурчак қавс ичига олинган ўзбекча сатр остини тил таркибига кирмайдиган тушунча деб қабул қилиш керак).*

Масалан:

```
int uzg=324;
cout<<uzg;    // бутун сон чоп этилади
```

Берилганларни стандарт оқимдан (одатда клавиатурадан) ўқиш қуйидаги форматда амалга оширилади:

cin >> <ўзгарувчи>;

Бу ерда <ўзгарувчи> қиймат қабул қилувчи ўзгарувчининг номи.

Мисол:

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

Бутун турдаги Yosh ўзгарувчиси киритилган қийматни ўзлаштиради. Киритилган қийматни ўзгарувчи турига мос келишини текшириш масъулияти программа тузувчисининг зиммасига юкланади.

Бир пайтнинг ўзида пробел воситасида бир нечта ва ҳар хил турдаги қийматларни оқимдан киритиш мумкин. Қиймат киритиш <enter> тугмасини босиш билан тугайди. Агар киритилган қийматлар сони ўзгарувчилар сонидан кўп бўлса, «ортиқча» қийматлар буфер хотирада сақланиб қолади.

```
#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;
}
```

Ўзгарувчиларга қиймат киритиш учун клавиатура орқали

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

ҳаракати амалга оширилади. Шуни қайд этиш керакки, оқимга қиймат киритишда пробел ажратувчи ҳисобланади. Ҳақиқий соннинг бутун ва каср қисмлари '.' белгиси билан ажратилади.

C++ тили алфавити ва лексемалар

C++ тили алфавити ва лексемаларига қуйидагилар киради:

- катта ва кичик лотин алфавити ҳарфлари;
- рақамлар - 0,1,2,3,4,5,6,7,8,9;
- махсус белгилар:" { } | [ ] ( ) + - / % \ ; ' : ? < = > _ ! & ~ # ^ . *

Алфавит белгиларидан тилнинг лексемалари шакллантирилади: идентификаторлар; калит (хизматчи ёки заҳираланган) сўзлар; ўзгармаслар; амаллар белгиланишлари; ажратувчилар.

Идентификаторлар ва калит сўзлар

Программалаш тилининг муҳим таянч тушунчаларидан бири - идентификатор тушунчасидир. *Идентификатор* деганда катта ва кичик лотин ҳарфлари, рақамлар ва таг чизиқ (‘_’) белгиларидан ташкил топган ва рақамдан бошланмайдиган белгилар кетма-кетлиги тушунилади. Идентификаторларда ҳарфларнинг регистрлари (катта ёки кичиклиги) ҳисобга олинади. Масалан, RUN, run, Run - бу ҳар хил идентификаторлардир. Идентификатор узунлигига чегара қўйилмаган, лекин улар компилятор томонидан фақат бошидаги 32 белгиси билан фарқланади.

Идентификаторлар калит сўзлар, ўзгарувчилар, функциялар, нишонлар ва бошқа объектларни номлашда ишлатилади.

C++ тилининг калит сўзларига қуйидагилар киради:

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.

Юқорида келтирилган идентификаторларни бошқа мақсадда ишлатиш мумкин эмас.

Процессор регистрларини белгилаш учун қуйидаги сўзлар ишлатилади:

_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.

Булардан ташқари «__» (иккита тагчизиқ) белгиларидан бошланган идентификаторлар кутубхоналар учун заҳираланган. Шу сабабли ‘_’ ва «__» белгиларни идентификаторнинг биринчи белгиси сифатида ишлатмаган маъқул. Идентификатор белгилар орасида пробел ишлатиш мумкин эмас, зарур бўлганда унинг ўрнига ‘_’ ишлатиш мумкин: Cilindr_radiusi, ailana_diametiri.

2- боб. C++ тилида берилганлар ва уларнинг турлари

Ўзгармаслар

Ўзгармас (*литерал*) - бу фиксирланган сонни, сатрни ва белгини ифодаловчи лексемадир.

Ўзгармаслар бешта гуруҳга бўлинади - *бутун*, *ҳақиқий* (*сузувчи нуқтали*), *санаб ўтилувчи*, *белги* (*литерли*) ва *сатр* («*стринг*», *литерли сатр*).

Компилятор ўзгармасни лексема сифатида аниқлайди, унга хотирадан жой ажратади, кўриниши ва қийматига (турига) қараб мос гуруҳларга бўлади.

Бутун ўзгармаслар. Бутун ўзгармаслар қуйидаги форматларда бўлади:

- ўнлик сон;
- саккизлик сон;
- ўн олтилик сон.

Ўнлик ўзгармас 0 рақамидан фарқли рақамдан бошланувчи рақамлар кетма-кетлиги ва 0 ҳисобланади: `0; 123; 7987; 11.`

Манфий ўзгармас - бу ишорасиз ўзгармас бўлиб, унга фақат ишорани ўзгартириш амали қўлланилган деб ҳисобланади.

Саккизлик ўзгармас 0 рақамидан бошланувчи саккизлик саноқ системаси (0,1,..,7) рақамларидан ташкил топган рақамлар кетма-кетлиги:

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

Ўн олтилик ўзгармас 0x ёки 0X белгиларидан бошланадиган ўн олтилик саноқ системаси рақамларидан иборат кетма-кетлик ҳисобланади:

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

Ҳарф белгилар ихтиёрий регистрларда берилиши мумкин.

Компилятор соннинг қийматига қараб унга мос турни белгилайди. Агар тилда белгиланган турлар программа тузувчини қаноатлантирмаса, у ошкор равишда турни кўрсатиши мумкин. Бунинг учун бутун ўзгармас рақамлари охирига, пробелсиз l ёки L (long), u ёки U (unsigned) ёзилади. Зарур ҳолларда битта ўзгармас учун бу белгиларнинг иккитасини ҳам ишлатиш мумкин:

```
45lu, 012Ul, 0xA2L.
```

Ҳақиқий ўзгармаслар. Ҳақиқий ўзгармаслар - сузувчи нуқтали сон бўлиб, у икки хил форматда берилиши мумкин:

- ўнлик фиксирланган нуқтали форматда. Бу кўринишда сон нуқта орқали ажратилган бутун ва каср қисмлар кўринишида бўлади. Соннинг бутун ёки каср қисми бўлмаслиги мумкин, лекин нуқта албатта бўлиши керак. Фиксирланган нуқтали ўзгармасларга мисоллар: **`24.56; 13.0; 66.; .87`**;
- экспоненциал шаклда ҳақиқий ўзгармас 6 қисмдан иборат бўлади:
  1) бутун қисми (ўнли бутун сон);
  2) ўнли каср нуқта белгиси;
  3) каср қисми (ўнлик ишорасиз ўзгармас);
  4) экспонента белгиси ‘e’ ёки ‘E’;
  5) ўн даражаси кўрсаткичи (ўнли бутун сон);
  6) қўшимча белгиси (‘F’ ёки ‘f’ , ‘L’ ёки ‘l’).

Экспоненциал шаклдаги ўзгармас сонларга мисоллар: **`1e2; 5e+3; .25e4; 31.4e-1`** .

Белги ўзгармаслар. Белги ўзгармаслар қўштирноқ (‘,’-апострофлар) ичига олинган алоҳида белгилардан ташкил топади ва у char калит сўзи билан аниқланади. Белги ўзгармас учун хотирада бир байт жой ажратилади ва унда бутун сон кўринишидаги белгининг ASCII коди жойлашади. Қуйидагилар белги ўзгармасларга мисол бўлади: **`'e', '@', '7', 'z','W','+','ш','*','a', 's'.`**

1.1-жадвал. C++ тилида escape -белгилар жадвали

Escape белгилари	**Ички код (16 сон)**	**Номи**	**Амал**
\\	0x5C	\	Тескари ён чизиқни чоп этиш
\’	0x27	‘	Апострофни чоп этиш
\”	0x22	“	Қўштирноқни чоп этиш
\?	0x3F	?	Сўроқ белгиси
\a	0x07	bel	Товуш сигналини бериш
\b	0x08	bs	Курсорни 1 белги ўрнига орқага қайтариш
\f	0x0C	ff	Саҳифани ўтказиш
\n	0x0A	lf	Қаторни ўтказиш
\r	0x0D	cr	Курсорни айни қаторнинг бошига қайтариш
\t	0x09	ht	Навбатдаги табуляция жойига ўтиш
\v	0x0D	vt	Вертикал табуляция (пастга)
\000	000		Саккизлик коди
\xNN	0xNN		Белги ўн олтилик коди билан берилган

Айрим белги ўзгармаслар ‘\’ белгисидан бошланади, бу белги биринчидан, график кўринишга эга бўлмаган ўзгармасларни белгилайди, иккинчидан, махсус вазифалар юкланган белгилар - апостроф

белгиси(‘), савол белгисини (‘?’), тескари ён чизиқ белгисини (‘\’) ва иккита қўштирноқ белгисини (‘“’) чоп қилиш учун ишлатилади. Ундан ташқари, бу белги орқали белгини кўринишини эмас, балки ошкор равишда унинг ASCII кодини саккизлик ёки ўн олтилик шаклда ёзиш мумкин. Бундай белгидан бошланган белгилар escape кетма-кетликлар дейилади (1.1-жадвал).

C++ тилида қўшимча равишда wide ҳарфли ўзгармаслар ва кўп белгили ўзгармаслар аниқланган.

wide ҳарфли ўзгармаслар тури миллий кодларни белгилаш учун киритилган бўлиб, у wchar_t калит сўзи билан берилади, ҳамда хотирада 2 байт жой эгаллайди. Бу ўзгармас L белгисидан бошланади:

```
L’\013\022’, L’cc’
```

Кўп белгили ўзгармас тури int бўлиб, у тўртта белгидан иборат бўлиши мумкин:

```
’abc’, ’\001\002\003\004’.
```

Сатр ўзгармаслар. Иккита қўштирноқ (“,”) ичига олинган белгилар кетма-кетлиги *сатр ўзгармас* дейилади:

```
”Bu satr o’zgarmas va uning nomi string\n”
```

Сатр ичида escape кетма-кетлиги ҳам ишлатилиши мумкин, фақат бу кетма-кетлик апострофсиз ёзилади.

Пробел билан ажратиб ёзилган сатрлар компилятор томонидан ягона сатрга уланади (конкантенация):

```
”Satr - bu belgilar massivi” /* бу сатр кейинги сатрга қўшилади */ ”, uning turi char[]”;
```

Бу ёзув

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

ёзуви билан эквивалент ҳисобланади.

Узун сатрни бир нечта қаторга ёзиш мумкин ва бунинг учун қатор охирида ‘\’ белгиси қўйилади:

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

Юқоридаги учта қаторда ёзилган сатр келтирилган. Тескари ён чизиқ (‘\’) белгиси кейинги қаторда ёзилган белгилар кетма-кетлигини юқоридаги сатрга қўшиш кераклигини билдиради. Агар қўшиладиган сатр бошланишида пробеллар бўлса, улар ҳам сатр таркибига киради.

Сатр хотирада жойлашганда унинг охирига ‘\0’ (0 кодли белги) қўшилади ва бу белги сатр тугаганлигини билдиради. Шу сабабли сатр узунлиги, унинг «ҳақиқий» қийматидан биттага кўп бўлади.

Берилганлар турлари ва ўзгарувчилар

Программа бажарилиши пайтида қандайдир берилганларни сақлаб туриш учун ўзгарувчилар ва ўзгармаслардан фойдаланилади. *Ўзгарувчи* - программа объекти бўлиб, хотирадаги бир нечта ячейкаларни эгаллайди ва берилганларни сақлаш учун хизмат қилади. Ўзгарувчи номга, ўлчамга ва бошқа атрибутларга - кўриниш соҳаси, амал қилиш вақти ва бошқа хусусиятларга эга бўлади. Ўзгарувчиларни ишлатиш учун улар албатта эълон қилиниши керак. Эълон натижасида ўзгарувчи учун хотирадан қандайдир соҳа заҳираланади, соҳа ўлчами эса ўзгарувчининг конкрет турига боғлиқ бўлади. Шуни қайд этиш зарурки, битта турга турли аппарат платформаларда турлича жой ажратилиши мумкин.

Ўзгарувчи эълони унинг турини аниқловчи калит сўзи билан бошланади ва ‘=’ белгиси орқали бошланғич қиймат берилади (шарт эмас). Битта калит сўз билан бир нечта ўзгарувчиларни эълон қилиш мумкин. Бунинг учун ўзгарувчилар бир-биридан ‘,’ белгиси билан ажратилади. Эълонлар ‘;’ белгиси билан тугайди. Ўзгарувчи номи 255 белгидан ошмаслиги керак.

C++ тилининг таянч турлари

C++ тилининг таянч турлари, уларнинг байтлардаги ўлчамлари ва қийматларининг чегаралари 1.2-жадвалда келтирилган.

Бутун сон турлари. Бутун сон қийматларни қабул қиладиган ўзгарувчилар int (бутун), short (қисқа) ва long (узун) калит сўзлар билан аниқланади. Ўзгарувчи қийматлари ишорали бўлиши ёки unsigned калит сўзи билан ишорасиз сон сифатида қаралиши мумкин (1-иловага қаранг).

Белги тури. Белги туридаги ўзгарувчилар char калит сўзи билан берилади ва улар ўзида белгининг ASCII кодини сақлайди. Белги туридаги қийматлар нисбатан мураккаб бўлган тузилмалар - сатрлар, белгилар массивлари ва ҳакозаларни ҳосил қилишда ишлатилади (2-иловага қаранг).

1.2-жадвал. C++ тилининг таянч турлари

Тур номи	Байтлардаги ўлчами	Қиймат чегараси
bool	1	true ёки 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 разрядли)	2	-32768..32767
int (32 разрядли)	4	-2147483648..2147483647
unsigned int (16 разрядли)	2	0..65535
unsigned int (32 разрядли)	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 разрядли)	10	3.4e-4932..-3.4e4932
void	2 ёки 4	-

Ҳақиқий сон тури. Ҳақиқий сонлар float калит сўзи билан эълон қилинади. Бу турдаги ўзгарувчи учун хотирада 4 байт жой ажратилади ва <ишора><тартиб><мантисса> қолипида сонни сақлайди(1-иловага қаранг). Агар касрли сон жуда катта (кичик) қийматларни қабул қиладиган бўлса, у хотиради 8 ёки 10 байтда иккиланган аниқлик кўринишида сақланади ва мос равишда double ва long double калит сўзлари билан эълон қилинади. Охирги ҳолат 32-разрядли платформалар учун ўринли.

Мантиқий тур. Бу турдаги ўзгарувчи bool калит сўзи билан эълон қилинади. У турдаги ўзгарувчи 1 байт жой эгаллайди ва 0 (false, ёлғон) ёки 0 қийматидан фарқли қиймат (true, рост) қабул қилади. Мантиқий турдаги ўзгарувчилар қийматлар ўртасидаги муносабатларни ифодалайдиган мулоҳазаларни рост ёки ёлғон эканлигини тавсифлашда қўлланилади ва улар қабул қиладиган қийматлар математик мантиқ қонуниятларига асосланади.

Математик мантиқ - фикрлашнинг шакли ва қонуниятлари ҳақидаги фан. Унинг асосини мулоҳазалар ҳисоби ташкил қилади. *Мулоҳаза* - бу ихтиёрий жумла бўлиб, унга нисбатан рост ёки ёлғон фикрни билдириш мумкин. Масалан «3>2», «5 - жуфт сон», «Москва-Украина пойтахти» ва ҳакозо. Лекин «0.000001 кичик сон» жумласи мулоҳаза ҳисобланмайди, чунки «кичик сон» тушунчаси жуда ҳам нисбий, яъни кичик сон деганда қандай сонни тушуниш кераклиги аниқ эмас. Шунинг учун юқоридаги жумлани рост еки ёлғонлиги ҳақида фикр билдириш қийин.

Мулоҳазаларнинг ростлиги ҳолатларга боғлиқ равишда ўзгариши мумкин. Масалан «бугун - чоршанба» жумласини рост ёки ёлғонлиги айни қаралаётган кунга боғлиқ. Худди шундай «x<0»

жумласи x ўзгарувчисининг айни пайтдаги қийматига мос равишда рост ёки ёлғон бўлади.

C++ тилида мантиқий тур номи англиялик математик Жорж Бул шарафига bool сўзи билан ифодаланган. Мантиқий амаллар «Бул алгебраси» дейилади.

Мантиқий мулоҳазалар устида учта амал аниқланган:

1) *инкор* - А мулоҳазани инкори деганда А рост бўлганда ёлғон ва ёлғон бўлганда рост қиймат қабул қилувчи мулоҳазага айтилади. C++ тилида инкор - '!' белгиси билан берилади. Масалан, А мулоҳаза инкори «!А» кўринишида ёзилади;

2) *коньюкция*- иккита А ва В мулоҳазалар конъюкцияси ёки мантиқий кўпайтмаси «А && В» кўринишга эга. Бу мулоҳаза фақат А ва В мулоҳазалар рост бўлгандагина рост бўлади, акс ҳолда ёлғон бўлади (одатда «&&» амали «ва» деб ўқилади). Масалан «бугун ойнинг 5 куни ва бугун чоршанба» мулоҳазаси ойнинг 5 куни чоршанба бўлган кунлар учунгина рост бўлади;

3) *дизъюнкция* - иккита А ва В мулоҳазалар дизъюнкцияси ёки мантиқий йиғиндиси «А || В» кўринишда ёзилади. Бу мулоҳаза рост бўлиши учун А ёки В мулоҳазалардан бири рост бўлиши етарли. Одатда «||» амали «ёки» деб ўқилади.

Юқорида келтирилган фикрлар асосида мантиқий амаллар учун ростлик жадвали аниқланган (1.3-жадвал).

1.3-жадвал. Мантиқий амаллар учун ростлик жадвали

Мулоҳазалар		**Мулоҳазалар устида амаллар**		
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

Мантиқий тур қийматлари устида мантиқий кўпайтириш, қўшиш ва инкор амалларини қўллаш орқали мураккаб мантиқий ифодаларни қуриш мумкин. Мисол учун, «x -мусбат ва у қиймати [1..3] сонлар оралиғига тегишли эмас» мулоҳазасини мантиқий ифода кўриниши қуйидашича бўлади:

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

void тури. void туридаги программа объекти ҳеч қандай қийматга эга бўлмайди ва бу турдан қурилманинг тил синтаксисига мос келишини таъминлаш учун ишлатилади. Масалан, C++ тили синтаксиси функция қиймат қайтаришини талаб қилади. Агар

функция қиймат қайтармайдиган бўлса, у void калит сўзи билан эълон қилинади.

Мисоллар.

```
int a=0,A=1; float abc=17.5;
double Ildiz;
bool Ok=true;
char LETTER='z';
void Mening_Funktsiyam();/* функция қайтарадиган
                          қиймат инобатга олинмайди */
```

Турланган ўзгармаслар

Турланган ўзгармаслар худди ўзгарувчилардек ишлатилади ва инициализация қилингандан (бошланғич қиймат берилгандан) кейин уларнинг қийматини ўзгартириб бўлмайди.

Турланган ўзгармас эълонида const калит сўзи, ундан кейин ўзгармас тури ва номи, хамда албатта инициализация қисми бўлади.

Мисол тариқасида турланган ва литерал ўзгармаслардан фойдаланган ҳолда радиус берилганда айлана юзасини ҳисоблайдиган программани келтирамиз.

```
#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;
}
```

Программа бош функциясининг бошланишида иккита - pi ва Radius ўзгармаслари эълон қилинган. Айлана юзасини аниқловчи Square ўзгармас деб эълон қилинмаган, чунки у программа бажарилишида ўзгаради. Айлана радиусини программа ишлашида ўзгартириш мўлжалланмаган, шу сабабли у ўзгармас сифатида эълон қилинган.

Санаб ўтилувчи тур

Кўп миқдордаги, мантиқан боғланган ўзгармаслардан фойдаланилганда санаб ўтилувчи турдан фойдаланилгани маъқул. Санаб ўтилувчи ўзгармаслар enum калит сўзи билан аниқланади. Мазмуни бўйича бу ўзгармаслар оддий бутун сонлардир. Санаб ўтилувчи ўзгармаслар C++ стандарти бўйича бутун турдаги ўзгармаслар ҳисоб–

ланади. Ҳар бир ўзгармасга (сонга) мазмунли ном берилади ва бу идентификаторни программанинг бошқа жойларида номлаш учун ишлатилиши мумкин эмас. Санаб ўтилувчи тур қўйидаги кўринишга эга:

enum <санаб ўтиладиган тур номи> { <ном$_1$> =<қиймат$_1$>,
<ном$_2$> =<қиймат$_2$>, … <ном$_n$> =<қиймат$_n$> };

Бу ерда, enum - калит сўз (инглизча enumerate - санамоқ); <санаб ўтиладиган тур номи>- ўзгармаслар рўйхатининг номи; <ном$_i$> - бутун қийматли константаларнинг номлари; <қиймат$_i$>- шарт бўлмаган инициализация қиймати (ифода).

Мисол учун ҳафта кунлари билан боғлиқ масала ечишда ҳафта кунларини dush (душанба), sesh (сешанба), chor (чоршанба), paysh (пайшанба), juma (жума), shanba (шанба), yaksh (якшанба) ўзгармасларини ишлатиш мумкин ва улар санаб ўтилувчи тур ёрдамида битта сатрда ёзилади:

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

Санаб ўтилувчи ўзгармаслар қуйидаги хоссага эга: агар ўзгармас қиймати кўрсатилмаган бўлса, у олдинги ўзгармас қийматидан биттага ортиқ бўлади. Келишув бўйича биринчи ўзгармас қиймати 0 бўлади.

Инициализация ёрдамида ўзгармас қийматини ўзгартириш мумкин:

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

Бу эълонда sesh қиймати 9, shanba эса 17 га тенг бўлади.

Санаб ўтилувчи ўзгармасларнинг номлари ҳар хил бўлиши керак, лекин уларнинг қийматлари бир хил бўлиши мумкин:

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

Ўзгармаснинг қиймати ифода кўринишда берилиши мумкин, фақат ифодадаги номларнинг қийматлари шу қадамдагача аниқланган бўлиши керак:

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

Ўзгармасни қийматлари манфий сон бўлиши хам мумкин:

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

Турни бошқа тургa келтириш

C++ тилида бир турни бошқа тургa келтиришнинг ошкор ва ошкормас йўллари мавжуд.

Умуман олганда, турни бошқа турга ошкормас келтириш ифодада ҳар хил турдаги ўзгарувчилар қатнашган ҳолларда амал қилади (аралаш турлар арифметикаси). Айрим ҳолларда, хусусан таянч турлар билан боғлиқ турга келтириш амалларида хатоликлар юзага келиши мумкин. Масалан, ҳисоблаш натижасидаги соннинг хотирадан вақтинча эгаллаган жойи узунлиги, уни ўзлаштирадиган ўзгарувчи учун ажратилган жой узунлигидан катта бўлса, қийматга эга разрядларни йўқотиш ҳолати юз беради.

Ошкор равишда турга келтиришда, ўзгарувчи олдига қавс ичида бошқа тур номи ёзилади:

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

Программа натижаси қуйидаги кўринишида бўлади:

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

Масала. Берилган белгининг ASCII коди чоп этилсин. Масала белги туридаги қийматни ошкор равишда бутун сон турига келтириб чоп қилиш орқали ечилади.

Программа матни:

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

Программанинг

```
Belgini kiriting:
```

сўровига

```
A <enter>
```

амали бажарилса, экранга

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

сатри чоп этилади.

3- боб. Ифодалар ва операторлар

Арифметик амаллар. Қиймат бериш оператори

Берилганларни қайта ишлаш учун C++ тилида амалларнинг жуда кенг мажмуаси аниқланган. *Амал* - бу қандайдир ҳаракат бўлиб, у битта (унар) ёки иккита (бинар) операндлар устида бажарилади, ҳисоб натижаси унинг қайтарувчи қиймати ҳисобланади.

Таянч арифметик амалларга қўшиш (+), айириш (-), кўпайтириш (*), бўлиш (/) ва бўлиш қолдиғини олиш (%) амалларини келтириш мумкин.

Амаллар қайтарадиган қийматларни ўзлаштириш учун қиймат бериш амали (=) ва унинг турли модификациялари ишлатилади: қўшиш, қиймат бериш билан (+=); айириш, қиймат бериш билан (−=); кўпайтириш, қиймат бериш билан (*=); бўлиш, қиймат бериш билан (/=); бўлиш қолдиғини олиш, қиймат бериш билан (%=) ва бошқалар. Бу ҳолатларнинг умумий кўриниши:

<ўзгарувчи><амал>=<ифода>;

Қуйидаги программа матнида айрим амалларга мисоллар келтирилган.

```
#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;
}
```

Программа бажарилиши натижасида экранга қуйидаги сонлар қатори пайдо бўлади:

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

Ифода тушунчаси

C++ тилида *ифода* - амаллар, операндлар ва пунктация белгиларининг кетма-кетлиги бўлиб, компилятор томонидан берилганлар

устида маълум бир амалларни бажаришга кўрсатма деб қабул қилинади. Ҳар қандай ‘;’ белги билан тугайдиган ифодага *тил кўрсатмаси* дейилади.

C++ тилидаги тил кўрсатмасига мисол:

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

Инкремент ва декремент амаллари

C++ тилида операнд қийматини бирга ошириш ва камайтиришнинг самарали воситалари мавжуд. Булар инкремент (++) ва декремент (−−) унар амаллардир.

Операндга нисбатан бу амалларнинг префикс ва постфикс кўринишлари бўлади. Префикс кўринишда амал тил кўрсатмаси бўйича иш бажарилишидан олдин операндга қўлланилади. Постфикс ҳолатда эса амал тил кўрсатмаси бўйича иш бажарилгандан кейин операндга қўлланилади.

Префикс ёки постфикс амал тушунчаси фақат қиймат бериш билан боғлиқ ифодаларда ўринли:

```
x=y++;       // постфикс
index =--i; // префикс
count++;     // унар амал, ”++count;” билан эквивалент
abc-- ;      // унар амал, ”--abc; ” билан эквивалент
```

Бу ерда у ўзгарувчининг қийматини х ўзгарувчисига ўзлаштирилади ва кейин биттага оширилади, і ўзгарувчининг қиймати биттага камайтириб, index ўзгарувчисига ўзлаштирилади.

sizeof амали

Ҳар хил турдаги ўзгарувчилар компьютер хотирасида турли сондаги байтларни эгаллайди. Бунда, ҳаттоки бир турдаги ўзгарувчилар ҳам қайси компьютерда ёки қайси операцион системада амал қилинишига қараб турли ўлчамдаги хотирани банд қилиши мумкин.

C++ тилида ихтиёрий (таянч ва ҳосилавий) турдаги ўзгарувчиларнинг ўлчамини sizeof амали ёрдамида аниқланади. Бу амални ўзгармасга, турга ва ўзгарувчига қўлланиши мумкин.

Қуйида келтирилган программада компьютернинг платформасига мос равишда таянч турларининг ўлчамлари чоп қилинади.

```
int main()
{
 cout<<”int тури ўлчами:”<<sizeof(int)<<”\n”;
 cout<<”float тури ўлчами:”<<sizeof(float)<<”\n”;
```

```
cout<<"double тури ўлчами:"<<sizeof(double) <<"\n";
cout<<"char тури ўлчами:"<<sizeof(char)<<"\n";
return 0;}
```

Разрядли мантиқий амаллар

Программа тузиш тажрибаси шуни кўрсатадики, одатда қўйилган масалани ечишда бирор ҳолат рўй берган ёки йўқлигини ифодалаш учун 0 ва 1 қиймат қабул қилувчи *байроқлардан* фойдаланилади. Бу мақсадда бир ёки ундан ортиқ байтли ўзгарувчилардан фойдаланиш мумкин. Масалан, bool туридаги ўзгарувчини шу мақсадда ишлатса бўлади. Бошқа томондан, байроқ сифатида байтнинг разрядларидан фойдаланиш ҳам мумкин. Чунки разрядлар фақат иккита қийматни - 0 ва 1 сонларини қабул қилади. Бир байтда 8 разряд бўлгани учун унда 8 та байроқни кодлаш имконияти мавжуд.

Фараз қилайлик, қўриқлаш тизимига 5 та хона уланган ва тизим тахтасида 5 та чироқча (индикатор) хоналар ҳолатини билдиради: хона қўриқлаш тизими назоратида эканлигини мос индикаторнинг ёниб туриши (разряднинг 1 қиймати) ва хонани тизимга уланмаганлигини индикатор ўчганлиги (разряднинг 0 қиймати) билдиради. Тизим ҳолатини ифодалаш учун бир байт етарли бўлади ва унинг кичик разрядидан бошлаб бештасини шу мақсадда ишлатиш мумкин:

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

Масалан, байтнинг қуйидаги ҳолати 1, 4 ва 5 хоналар қўриқлаш тизимига уланганлигини билдиради:

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

Қуйидаги жадвалда C++ тилида байт разрядлари устида мантиқий амаллар мажмуаси келтирилган.

3.1-жадвал. Байт разрядлари устида мантиқий амаллар

Амаллар	**Мазмуни**
&	Мантиқий ВА (кўпайтириш)
|	Мантиқий ЁКИ (қўшиш)
^	Истисно қилувчи ЁКИ
~	Мантиқий ИНКОР (инверсия)

Разрядли мантиқий амалларнинг бажариш натижаларини жадвал кўринишида кўрсатиш мумкин.

3.2-жадвал. Разрядли мантиқий амалларнинг бажариш натижалари

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

Юқоридаги келтирилган мисол учун қўриқлаш тизимини ифодаловчи бир байтли char туридаги ўзгарувчини эълон қилиш мумкин:

```
char q_taxtasi=0;
```

Бу ерда q_taxtasi ўзгарувчисига 0 қиймат бериш орқали барча хоналар қўриқлаш тизимига уланмаганлиги ифодаланади:

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

Агар 3-хонани тизимга улаш зарур бўлса

```
q_taxtasi=q_taxtasi|0x04;
```

амалини бажариш керак, чунки $0x04_{16}=00000100_2$ ва мантиқий ЁКИ амали натижасида q_taxtasi ўзгарувчиси байти қуйидаги кўринишда бўлади:

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

Худди шундай йўл билан бошқа хоналарни тизимга улаш мумкин, зарур бўлса бирданига иккитасини (зарур бўлса барчасини):

```
q_taxtasi=q_taxtasi|0x1F;
```

Мантиқий кўпайтириш орқали хоналарни қўриқлаш тизимидан чиқариш мумкин:

```
q_taxtasi=q_taxtasi&0xFD; // 0xFD16=11111101₂
```

Худди шу натижани ‘~’ амалидан фойдаланган ҳолда ҳам олиш мумкин. Иккинчи хона тизимга уланганлиги билдирувчи байт қиймати - 00000010_2, демак шу ҳолатни инкор қилган ҳолда мантиқий кўпайтиришни бажариш керак.

```
q_taxtasi=q_taxtasi&(~0x02);
```

Ва ниҳоят, агар 3-хона индикаторини, уни қандай қийматда бўлишидан қатъий назар қарама-қарши ҳолатга ўтказишни «инкор қилувчи ЁКИ» амали ёрдамида бажариш мумкин:

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

Разрядли мантиқий амалларни қиймат бериш оператори билан биргаликда бажарилишининг қуйидаги кўринишлари мавжуд:

&= - разрядли ВА қиймат бериш билан;

| = - разрядли ЁКИ қиймат бериш билан;

^= - разрядли истисно қилувчи ЁКИ қиймат бериш билан.

Чапга ва ўнгга суриш амаллари

Байтдаги битлар қийматини чапга ёки ўнгга суриш учун, мос равишда “<<” ва “>>” амаллари қўлланилади. Амалдан кейинги сон битлар нечта ўрин чапга ёки ўнга суриш кераклигини билдиради.

Масалан:

```
unsigned char A=12;     // A=00001100₂=0x0C₁₆
A=A<<2;                 // A=00110000₂=0x30₁₆=48₁₀
A=A>>3;                 // A=00000110₂=0x06₁₆=6₁₀
```

Разрядларни n та чапга (ўнга) суриш сонни 2^n сонига кўпайтириш (бўлиш) амали билан эквивалент бўлиб ва нисбатан тез бажарилади. Шуни эътиборга олиш керакки, операнд ишорали сон бўлса, у ҳолда чапга суришда энг чапдаги ишора разряди такрорланади (ишора сақланиб қолади) ва манфий сонлар устида бу амал бажарилганда математика нуқтаи-назардан хато натижалар юзага келади:

```
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
```

Шу сабабли, бу разрядли суриш амаллари ишорасиз (unsigned) турдаги қийматлар устида бажарилгани маъқул.

Таққослаш амаллари

С++ тилида қийматларни солиштириш учун таққослаш амаллари аниқланган (3.3-жадвал). Таққослаш амали бинар амал бўлиб, қуйидаги кўринишга эга:

<операнд$_1$> <таққослаш амали> < операнд$_2$>

Таққослаш амалларининг натижаси - таққослаш ўринли бўлса, true (рост), акс ҳолда false (ёлғон) қиймат бўлади. Агар таққослашда арифметик ифода қатнашса, унинг қиймати 0 қийматидан фарқли ҳолатлар учун 1 деб ҳисобланади.

3.3-жадвал. Таққослаш амаллари ва уларнинг қўлланиши

Амаллар	Қўлланиши	Мазмуни (ўқилиши)
<	A<b	“a кичик b”

<=	a<=b	“a кичик ёки тенг b”
>	a>b	“a катта b”
>=	a>=b	“a катта ёки тенг b”
==	a==b	“a тенг b”
!=	a!=b	“a тенг эмас b”

«Вергул» амали

Тил қурилмаларидаги бир нечта ифодаларни компилятор томонидан яхлит бир ифода деб қабул қилиши учун «вергул» амали қўлланилади. Бу амални қўллаш орқали программа ёзишда маълум бир самарадорликка эришиш мумкин. Одатда «вергул» амали if ва for операторларида кенг қўлланилади. Масалан, if оператори қўйидаги кўринишда бўлиши мумкин:

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

Бу ерда, олдин CallFunc() функцияси чақирилади ва унинг натижаси i ўзгарувчисига ўзлаштирилади, кейин i қиймати 7 билан солиштирилади.

Амалларнинг устунликлари ва бажарилиш йўналишлари

Анъанавий арифметикадагидек C++ тилида ҳам амаллар маълум бир тартиб ва йўналишда бажарилади. Маълумки, математик ифодаларда бир хил устунликдаги (приоритетдаги) амаллар учраса (масалан, қўшиш ва айириш), улар чапдан ўнгга бажарилади. Бу тартиб C++ тилидаги ҳам ўринли, бироқ айрим ҳолларда амал ўнгдан чапга бажарилиши мумкин (хусусан, қиймат бериш амалида).

Ифодалар қийматини ҳисоблашда амаллар устунлиги ҳисобга олинади. Биринчи навбатда энг юқори устунликка эга бўлган амал бажарилади.

Қуйидаги жадвалда C++ тилида ишлатиладиган амаллар (операторлар), уларнинг устунлик коэффициентлари ва бажарилиш йўналишлари ($\Leftarrow$ - ўнгдан чапга, $\Rightarrow$ - чапдан ўнгга) келтирилган.

3.4-жадвал. Амалларнинг устунликлари ва бажарилиш йўналишлари

Оператор	**Тавсифи**	**Устунлик**	**Йўналиш**
::	Кўриниш соҳасига рухсат бериш	16	$\Rightarrow$
[ ]	Массив индекси	16	$\Rightarrow$
( )	Функцияни чақириш	16	$\Rightarrow$
.	Структура ёки синф элементини танлаш	16	$\Rightarrow$
->			

++	Постфикс инкремент	15	⇐
- -	Постфикс декремент	15	⇐
++	Префикс инкремент	14	⇐
- -	Префикс декремент	14	⇐
sizeof	Ўлчамни олиш	14	⇐
(<тур>)	Турга акслантириш	14	
~	Разрядли мантиқий ИНКОР	14	⇐
!	Мантиқий инкор	14	⇐
-	Унар минус	14	⇐
+	Унар плюс	14	⇐
&	Адресни олиш	14	⇐
*	Воситали мурожаат	14	⇐
new	Динамик объектни яратиш	14	⇐
delete	Динамик объектни йўқ қилиш	14	⇐
casting	Турга келтириш	14	
*	Кўпайтириш	13	⇒
/	Бўлиш	13	⇒
%	Бўлиш қолдиғи	13	⇒
+	Қўшиш	12	⇒
-	Айириш	12	⇒
>>	Разряд бўйича ўнгга суриш	11	⇒
<<	Разряд бўйича чапга суриш	11	⇒
<	Кичик	10	⇒
<=	Кичик ёки тенг	10	⇒
>	Катта	10	⇒
>=	Катта ёки тенг	10	⇒
==	Тенг	9	⇒
!=	Тенг эмас	9	⇒
&	Разрядли ВА	8	⇒
^	Разрядли истисно қилувчи ЁКИ	7	⇒
|	Разрядли ЁКИ	6	⇒
&&	Мантиқий ВА	5	⇒
||	Мантиқий ЁКИ	4	⇒
?:	Шарт амали	3	⇐
=	Қиймат бериш	2	⇐
*=	Кўпайтириш қиймат бериш билан	2	⇐
/=	Бўлиш қиймат бериш билан	2	⇐
%=	Модулли бўлиш қиймат бериш билан	2	⇐
+=	Қўшиш қиймат бериш билан	2	⇐

- =	Айириш қиймат бериш билан	2	⇐
<<=	Чапга суриш қиймат бериш билан	2	⇐
>>=	Ўнгга суриш қиймат бериш билан	2	⇐
&=	Разрядли ВА қиймат бериш билан	2	⇐
^=	Разрядли истисно килувчи ЁКИ қиймат бериш билан	2	⇐
| =	Разрядли ЁКИ қиймат бериш билан	2	⇐
throw	Истисно ҳолатни юзага келтириш	2	⇐
,	Вергул	1	⇒

C++ тили программа тузувчисига амалларнинг бажарилиш тартибини ўзгартириш имкониятини беради. Худди математика-дагидек, амалларни қавслар ёрдамида гуруҳларга жамлаш мумкин. Қавс ишлатишга чеклов йўқ.

Қуйидаги программада қавс ёрдамида амалларни бажариш тартибини ўзгартириш кўрсатилган.

```
#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';}
```

Программада амаллар устунлигига кўра x қийматини ҳисоблашда олдин a ўзгарувчи b ўзгарувчига кўпайтирилади ва унга c ўзгарувчи қийматига қўшилади. Навбатдаги кўрсатмани бажаришда эса биринчи навбатда ички қавс ичидаги ифода - (b+c) қиймати ҳисобланади, кейин бу қиймат a кўпайтирилиб, y ўзгарувчисига ўзлаштирилади. Программа бажарилиши натижасида экранга

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

сатрлари чоп этилади.

4- боб. Программа бажарилишини бошқариш

Оператор тушунчаси

Программалаш тили операторлари ечилаётган масала алгоритмини амалга ошириш учун ишлатилади. Операторлар *чизиқли* ва *бошқарув операторларига* бўлинади. Аксарият ҳолатларда операторлар «нуқта-вергул» (‘;’) белгиси билан тугалланади ва у компилятор томонидан алоҳида оператор деб қабул қилинади (for операторининг қавс ичида турган ифодалари бундан мустасно). Бундай оператор *ифода оператори* дейилади. Қиймат бериш амаллари гуруҳи, хусусан, қиймат бериш операторлари ифода операторлари ҳисобланади:

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

Программа тузиш амалиётида бўш оператор - ‘;’ ишлатилади. Гарчи бу оператор ҳеч нима бажармаса ҳам, ҳисоблаш ифодаларини тил қурилмаларига мос келишини таъминлайди. Айрим ҳолларда юзага келган «боши берк» ҳолатлардан чиқиб кетиш имконини беради.

Ўзгарувчиларни эълон қилиш ҳам оператор ҳисобланади ва уларга *эълон оператори* дейилади.

Шарт операторлари

Олдинги бобда мисол тариқасида келтирилган программаларда амаллар ёзилиш тартибида кетма-кет ва фақат бир марта бажариладиган ҳолатлар, яъни чизиқли алгоритмлар келтирилган. Амалда эса камдан-кам масалалар шу тариқа ечилиши мумкин. Аксарият масалалар юзага келадиган турли ҳолатларга боғлиқ равишда мос қарор қабул қилишни (ечимни) талаб этади. C++ тили программанинг алоҳида бўлакларининг бажарилиш тартибини бошқаришга имкон берувчи қурилмаларнинг етарлича катта мажмуасига эга. Масалан, программа бажарилишининг бирорта қадамида қандайдир шартни текшириш натижасига кўра бошқарувни программанинг у ёки бу бўлагига узатиш мумкин (тармоқланувчи алгоритм). Тармоқланишни амалга ошириш учун шартли оператордан фойдаланилади.

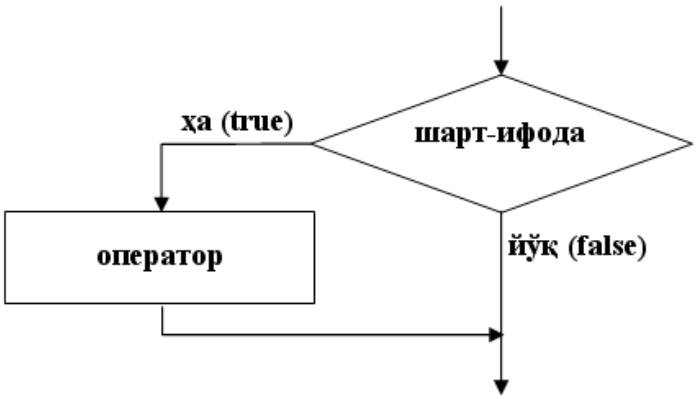
if оператори

if оператори қандайдир шартни ростликка текшириш натижасига кўра программада тармоқланишни амалга оширади:

if (<шарт>)<оператор>;

Бу ерда <шарт> ҳар қандай ифода бўлиши мумкин, одатда у таққослаш амали бўлади.

Агар шарт 0 қийматидан фарқли ёки рост (true) бўлса, <оператор> бажарилади, акс ҳолда, яъни шарт 0 ёки ёлғон (false) бўлса, ҳеч қандай амал бажарилмайди ва бошқарув if операторидан кейинги операторга ўтади (агар у мавжуд бўлса). Ушбу ҳолат 4.1-расмда кўрсатилган.



4.1-расм. if() шарт операторининг блок схемаси

C++ тилининг қурилмалари операторларни блок кўринишида ташкил қилишга имкон беради. *Блок* - ‘{‘ ва ‘}’ белги оралиғига олинган операторлар кетма-кетлиги бўлиб, у компилятор томонидан яхлит бир оператор деб қабул қилинади. Блок ичида эълон операторлари ҳам бўлиши мумкин ва уларда эълон қилинган ўзгарувчилар фақат шу блок ичида кўринади (амал қилади), блокдан ташқарида кўринмайди. Блокдан кейин ‘;’ белгиси қўйилмаслиги мумкин, лекин блок ичидаги ҳар бир ифода ‘;’ белгиси билан якунланиши шарт.

Қуйида келтирилган программада if операторидан фойдаланиш кўрсатилган.

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

```
}
```

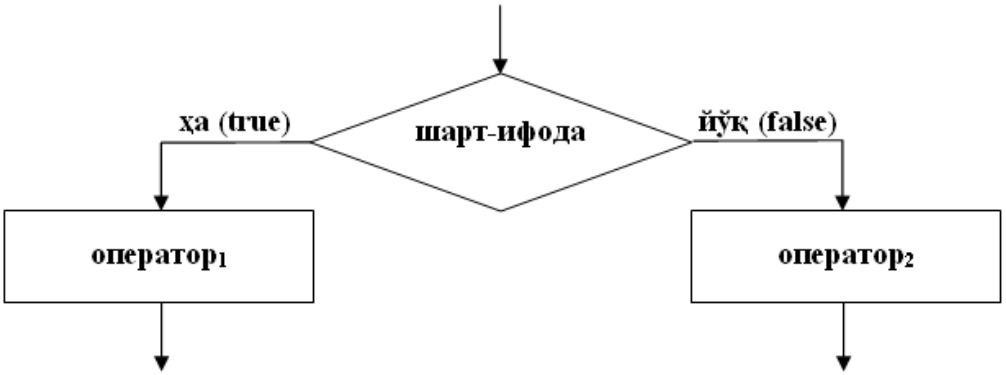
Программа бажарилиши жараёнида бутун турдаги b ўзгарувчи эълон қилинади ва унинг қиймати клавиатурадан ўқилади. Кейин b қийматини 0 сонидан катталиги текширилади, агар шарт бажарилса (true) , у ҳолда ‘{‘ ва ‘}’ белгилар ичидаги операторлар бажарилади ва экранга “b - мусбат сон” хабари чиқади. Агар шарт бажарилмаса, бу операторлар чеклаб ўтилади. Навбатдаги шарт оператори b ўзгарувчи қиймати манфийликка текширади, агар шарт бажарилса, ягона cout кўрсатмаси бажарилади ва экранга “b - манфий сон” хабари чиқади.

if - else оператори

Шарт операторининг if - else кўриниши қуйидагича:

if (<шарт-ифода>) <оператор$_1$>; else <оператор$_2$>;

Бу ерда <шарт-ифода> 0 қийматидан фарқли ёки true бўлса, <оператор$_1$>, акс ҳолда <оператор$_2$> бажарилади. if-else шарт оператори мазмунига кўра алгоритмнинг тармоқланувчи блокини ифодалайди: <шарт-ифода> - шарт блоки (ромб) ва <оператор$_1$> блокнинг «ҳа» шохига, <оператор$_2$> эса блокнинг «йўқ» шохига мос келувчи амаллар блоклари деб қараш мумкин (4.2-расм).



4.1-расм. if(); else шарт операторининг блок схемаси

Мисол тариқасида дискриминантни ҳисоблаш усули ёрдамида $ax^2+bx+c=0$ кўринишидаги квадрат тенглама илдизларини топиш масаласини кўрайлик:

```
#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;
}
```

Программа бажарилганда, биринчи навбатда тенглама коэффициентлари - a, b, c ўзгарувчилар қийматлари киритилади, кейин дискриминант - D ўзгарувчи қиймати ҳисобланади. Кейин D қийматининг манфий эканлиги текширилади. Агар шарт ўринли бўлса, яхлит оператор сифатида келувчи ‘{‘ ва ‘}’ белгилари орасидаги операторлар бажарилади ва экранга “Тенглама ҳақиқий илдизларга эга эмас” хабари чиқади ва программа ўз ишини тугатади (“return 0;” операторини бажариш орқали). Дискриминант нолдан кичик бўлмаса, навбатдаги шарт оператори уни нолга тенглигини текширади. Агар шарт ўринли бўлса, кейинги қаторлардаги операторлар блоки бажарилади - экранга “Тенглама ягона илдизга эга:” хабари, ҳамда x1 ўзгарувчи қиймати чоп этилади ва программа шу ерда ўз ишини тугатади, акс ҳолда, яъни D қиймати нолдан катта ҳолати учун else калит сўзидан кейинги операторлар блоки бажарилади ва экранга “Тенглама иккита илдизга эга: “ хабари, ҳамда x1 ва x2 ўзгарувчилар қийматлари чоп этилади. Шу билан шарт операторидан чиқилади ва асосий функциянинг return кўрсатмасини бажариш орқали программа ўз ишини тугатади.

Ўз навбатида <оператор$_1$> ва <оператор$_2$> ҳам шартли оператор бўлиши мумкин. Ифодадаги ҳар бир else калит сўзи, олдиндаги энг

яқин if калит сўзига тегишли ҳисобланади (худди очилувчи ва ёпилувчи қавслардек). Буни инобатга олмаслик мазмунан хатоликларга олиб келиши мумкин.

Масалан:

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

Бу мисолда "x<>1" хабари x қиймати 1 ва y қиймати 1 бўлмаган ҳолда ҳам чоп этилади. Қуйидаги вариантда ушбу мазмунан хатолик бартараф этилган:

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

Иккинчи мисол тариқасида учта бутун соннинг максимал қийматини топадиган программа бўлагини келтиришимиз мумкин:

```
...
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;
...
```

Шарт операторида эълон қилиш операторларини ишлатиш ман этилади, лекин ундаги блокларда ўзгарувчиларни эълон қилиш мумкин ва бу ўзгарувчилар фақат блок ичида амал қилади. Қуйидаги мисолда бу ҳолат билан боғлиқ хатолик кўрсатилган:

```
if(j>0){int i;i=2*j;}
else i=-j;//хато,чунки i блокдан ташқарида кўринмайди
```

Масала. Берилган тўрт хонали ишорасиз соннинг бошидаги иккита рақамининг йиғиндиси қолган рақамлар йиғиндисига тенг ёки йўқлиги аниқлансин (рақамлар йиғиндиси деганда уларга мос сон қийматларининг йиғиндиси тушунилади). Соннинг рақамларини ажратиб олиш учун бутун сонлар арифметикаси амалларидан фойдаланилади:

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

```
int main()
{
 unsigned int n,a3,a2,a1,a0; // n=a3a2a1a0 кўринишида
 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;
}
```

Программа ишорасиз бутун сон киритишни таклиф қилади. Агар киритилган сон 4 хонали бўлмаса (n<1000 ёки n>9999), бу ҳақда хабар берилади ва программа ўз ишини тугатади. Акс ҳолда n сонининг рақамлари ажратиб олинади, ҳамда бошидаги иккита рақамнинг йиғиндиси - (a3+a2) қолган иккита рақамлар йиғиндиси - (a1+a0) билан солиштирилади ва уларнинг тенг ёки йўқлиги қараб мос жавоб чоп қилинади.

?: шарт амали

Агар текширилаётган шарт нисбатан содда бўлса, шарт амалининг «?:» кўринишини ишлатиш мумкин:

<шарт ифода> ? <ифода$_1$> : <ифода$_2$>;

Шарт амали if шарт операторига ўхшаш ҳолда ишлайди: агар <шарт ифода> 0 қийматидан фарқли ёки true бўлса, <ифода$_1$>, акс ҳолда <ифода$_2$> бажарилади. Одатда ифодалар қийматлари бирорта ўзгарувчига ўзлаштирилади.

Мисол тариқасида иккита бутун сон максимумини топиш масаласини кўрайлик.

```
#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;
 c=a>b?a:b;
 cout<<"\nSonlar maksimumi: "<<c;
 return 0;
}
```

Программадаги шарт оператори қиймат бериш операторининг таркибига кирган бўлиб, a ўзгарувчининг қиймати b ўзгарувчининг қийматидан катталиги текширилади. Агар шарт рост бўлса, c ўзгарувчисига a ўзгарувчи қийматини, акс ҳолда b ўзгарувчининг қийматини ўзлаштиради ва c ўзгарувчисининг қиймати чоп этилади.

?: амалининг қиймат қайтариш хоссасидан фойдаланган ҳолда, уни бевосита cout кўрсатмасига ёзиш орқали ҳам қўйилган масалани ечиш мумкин:

```
#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 оператори

Шарт операторининг яна бир кўриниши switch тармоқланиш оператори бўлиб, унинг синтаксиси қуйидагича:

switch (<ифода>)
{
 case <ўзгармас ифода$_1$> : <операторлар гуруҳи$_1$>; break;
 case <ўзгармас ифода$_2$> : <операторлар гуруҳи$_2$>; break;
…
 case <ўзгармас ифода$_n$> : <операторлар гуруҳи$_n$>; break;
 default : <операторлар гуруҳи$_{n+1}$>;
}

Бу оператор қуйидагича амал қилади: биринчи навбатда <ифода> қиймати ҳисобланади, кейин бу қиймат case калит сўзи билан ажратилган <ўзгармас ифода$_i$> билан солиштирилади. Агар

улар устма-уст тушса, шу қатордаги ‘:’ белгисидан бошлаб, токи break калит сўзигача бўлган <операторлар гуруҳи$_i$> бажарилади ва бошқарув тармоқланувчи оператордан кейин жойлашган операторга ўтади. Агар <ифода> бирорта ҳам <ўзгармас ифода$_i$> билан мос келмаса, қурилманинг default қисмидаги <операторлар гуруҳи$_{n+1}$> бажарилади. Шуни қайд этиш керакки, қурилмада default калит сўзи фақат бир марта учраши мумкин.

Мисол учун, кириш оқимидан “Jarayon davom etilsinmi?” сўровига фойдаланувчи томонидан жавоб олинади. Агар ижобий жавоб олинса, экранга “Jarayon davom etadi!” хабари чоп этилади ва программа ўз ишини тармоқланувчи оператордан кейинги операторларни бажариш билан давом эттиради, акс ҳолда “Jarayon tugadi!” жавоби берилади ва программа ўз ишини тугатади. Бунда, фойдаланувчининг ‘y’ ёки ’У’ жавоблари жараённи давом эттиришни билдиради, бошқа белгилар эса жараённи тугатишни англатади.

```
#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;
 }
 ... // жараён
 return 0;
}
```

Умуман олганда, тармоқланувчи операторда break ва default калит сўзларини ишлатиш мажбурий эмас. Лекин бу ҳолатда оператор мазмуни бузилиши мумкин. Масалан, default қисми бўлмаган ҳолда, агар <ифода> бирорта <ўзгармас ифода$_i$> билан устма-уст тушмаса, оператор ҳеч қандай амал бажармасдан бошқарув тармоқланувчи оператордан кейинги операторга ўтади. Агар break бўлмаса, <ифода> бирорта <ўзгармас ифода$_i$> билан устма-уст тушган ҳолда, унга мос келувчи операторлар гуруҳини бажаради ва «тўхтамасдан» кейинги қатордаги операторлар гуруҳини ҳам бажаришда давом этади.

Масалан, юқоридаги мисолда break оператори бўлмаса ва жараённи давом эттиришни тасдиқловчи (‘Y’) жавоб бўлган тақдирда экранга

```
Jarayon davom etadi!
Jarayon tygadi!
```

хабарлари чиқади ва программа ўз ишини тугатади (return операторининг бажарилиши натижасида).

Тармоқланувчи оператор санаб ўтилувчи турдаги ўзгармаслар билан биргаликда ишлатилганда самара беради. Қуйидаги программада ранглар гаммасини тоифалаш масаласи ечилган.

```
#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;
}
```

Программа бажарилишида бошқарув тармоқланувчи операторга келганда, Rang қиймати Qizil ёки Tuq_sariq ёки Sariq бўлса, “Issiq gamma tanlandi” хабари, агар Rang қиймати Yashil ёки Kuk ёки Zangori ёки Binafsha бўлса, экранга “Sovuq gamma tanlandi” хабари, агар Rang қиймати санаб ўтилган қийматлардан фарқли бўлса, экранга “Kamalak bunday rangga ega emas” хабари чоп этилади ва программа ўз ишини тугатади.

switch операторида эълон операторлари ҳам учраши мумкин. Лекин switch оператори бажарилишида «сакраб ўтиш» ҳолатлари бўлиши ҳисобига блок ичидаги айрим эълонлар бажарилмаслиги ва бунинг оқибатида программа ишида хатолик рўй бериши мумкин:

```
//...
int k=0,n=0;
cin >>n;
```

```
switch (n)
{
 int i=10;//хато,бу оператор ҳеч қачон бажарилмайди
 case 1:
 int j=20;//агар n=2 бўлса,бу эълон бажарилмайди
 case 2:
 k+=i+j; //хато, чунки i,j ўзгарувчилар номаълум
}
cout<<k;
//...
```

Масала. Қуйида санаб ўтилувчи турлар ва шу турдаги ўзгарувчилар эълон қилинган:

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

Берилган r бирликда берилган x ўзгарувчисининг қиймати метрларда чоп қилинсин.

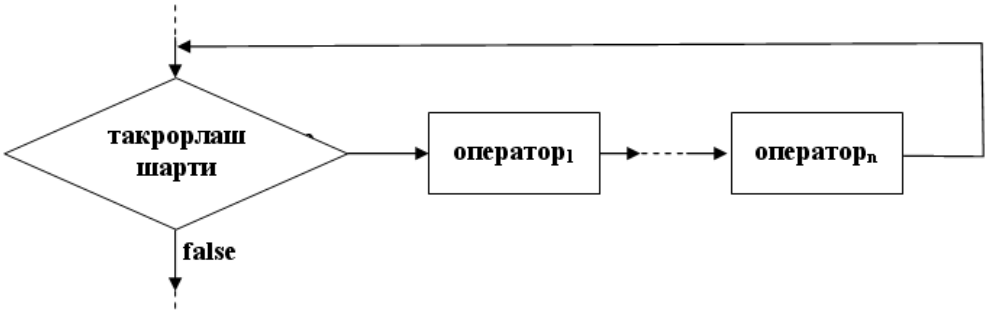
```
#include <iostream.h>
int main()
{
 enum
 Birlik{desimetr,kilometr,metr,millimetr, santimetr};
 float x,y;
 Birlik r;
 cout<<"Uzunlikni kiriting: x=";
 cin>>x;
 cout<<" Uzunlik birliklari\n";
 cout<<" 0- desimetr\n";
 cout<<" 1- kilometr\n";
 cout<<" 2- metr\n";
 cout<<" 3- millimetr\n";
 cout<<"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

Такрорлаш операторлари

Программа бажарилишини бошқаришнинг бошқа бир кучли механизмларидан бири - такрорлаш операторлари ҳисобланади.

Такрорлаш оператори «такрорлаш шарти» деб номланувчи ифоданинг рост қийматида программанинг маълум бир қисмидаги операторларни (такрорлаш танасини) кўп марта такрор равишда бажаради (итаратив жараён) (4.2-расм).



4.2-расм. Такрорлаш операторининг блок схемаси

Такрорлаш ўзининг кириш ва чиқиш нуқталарига эга, лекин чиқиш нуқтасининг бўлмаслиги мумкин. Бу ҳолда такрорлашга *чексиз такрорлаш* дейилади. Чексиз такрорлаш учун такрорлашни давом эттириш шарти доимо рост бўлади.

Такрорлаш шартини текшириш такрорлаш танасидаги операторларни бажаришдан олдин текширилиши мумкин (for, while такрорлашлари) ёки такрорлаш танасидаги операторлари бир марта бажарилгандан кейин текширилиши мумкин (do-while).

Такрорлаш операторлари ичма-ич жойлашган бўлиши мумкин.

for такрорлаш оператори

for такрорлаш операторининг синтаксиси қўйидаги кўринишга эга:

for (<ифода$_1$>; <ифода$_2$>;<ифода$_3$>) <оператор ёки блок>;

Бу оператор ўз ишини <ифода$_1$> ифодасини бажаришдан бошлайди. Кейин такрорлаш қадамлари бошланади. Ҳар бир қадамда <ифода$_2$> бажарилади, агар натижа 0 қийматидан фарқли ёки true бўлса, такрорлаш танаси - <оператор ёки блок> бажарилади ва охирида <ифода$_3$> бажарилади. Агар <ифода$_2$> қиймати 0 (false) бўлса, такрорлаш жараёни тўхтайди ва бошқарув такрорлаш операторидан кейинги операторга ўтади. Шуни қайд қилиш керакки, <ифода$_2$> ифодаси вергул билан ажратилган бир нечта ифодалар бирлашмасидан иборат

бўлиши мумкин, бу ҳолда охирги ифода қиймати такрорлаш шарти ҳисобланади. Такрорлаш танаси сифатида битта оператор, жумладан бўш оператор бўлиши ёки операторлар блоки келиши мумкин.

Мисол учун 10 дан 20 гача бўлган бутун сонлар йиғиндисини ҳисоблаш масаласини кўрайлик.

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

Программадаги такрорлаш оператори ўз ишини, і такрорлаш параметрига (такрорлаш санагичига) бошланғич қиймат - 10 сонини беришдан бошлайди ва ҳар бир такрорлаш қадамидан (итарациядан) кейин қавс ичидаги учинчи оператор бажарилиши ҳисобига унинг қиймати биттага ошади. Ҳар бир такрорлаш қадамида такрорлаш танасидаги оператор бажарилади, яъни Summa ўзгарувчисига і қиймати қўшилади. Такрорлаш санагичи і қиймати 21 бўлганда "i<=20" такрорлаш шарти false (0-қиймати) бўлади ва такрорлаш тугайди. Натижада бошқарув такрорлаш операторидан кейинги cout операторига ўтади ва экранга йиғинди чоп этилади.

Юқорида келтирилган мисолга қараб такрорлаш операторларининг қавс ичидаги ифодаларига изоҳ бериш мумкин:

<ифода$_1$> - такрорлаш санагичи вазифасини бажарувчи ўзгарувчига бошланғич қиймат беришга хизмат қилади ва у такрорлаш жараёни бошида фақат бир марта ҳисобланади. Ифодада ўзгарувчи эълони учраши мумкин ва бу ўзгарувчи такрорлаш оператори танасида амал қилади ва такрорлаш операторидан ташқарида «кўринмайди» (C++ Builder компилятори учун);

<ифода$_2$> - такрорлашни бажариш ёки йўқлигини аниқлаб берувчи мантиқий ифода, агар шарт рост бўлса, такрорлаш давом этади, акс ҳолда йўқ. Агар бу ифода бўш бўлса, шарт доимо рост деб ҳисобланади;

<ифода$_3$> - одатда такрорлаш санагичининг қийматини ошириш (камайтириш) учун хизмат қилади ёки унда такрорлаш шартига таъсир қилувчи бошқа амаллар бўлиши мумкин.

Такрорлаш операторида қавс ичидаги ифодалар бўлмаслиги мумкин, лекин синтаксис ';' бўлмаслигига рухсат бермайди. Шу

сабабли, энг содда кўринишдаги такрорлаш оператори қуйидагича бўлади:

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

Агар такрорлаш жараёнида бир нечта ўзгарувчиларнинг қийматлари синхрон равишда ўзгариши керак бўлса, такрорлаш ифодаларида зарур операторларни ‘,’ билан ёзиш орқали бунга эришиш мумкин:

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

Такрорлаш операторининг ҳар бир қадамида j ва i ўзгарувчиларнинг қийматлари мос равишда ўзгариб боради.

for операторида такрорлаш танаси бўлмаслиги ҳам мумкин. Масалан, программа бажарилишини маълум бир муддатга «тўхтаб» туриш зарур бўлса, бунга такрорлашни ҳеч қандай қўшимча ишларни бажармасдан амал қилиши орқали эришиш мумкин:

```
#include <iostream.h>
 int main()
 {int delay;
  ...
  for (delay=5000; delay>0; delay--); // бўш оператор
  ...
  return 0;}
```

Юқорида келтирилган 10 дан 20 гача бўлган сонлар йиғиндисини бўш танали такрорлаш оператори орқали ҳисоблаш мумкин:

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

Такрорлаш оператори танаси сифатида операторлар блоки ишлатишини факториални ҳисоблаш мисолида кўрсатиш мумкин:

```
#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;
```

```
}
```

Программа фойдаланувчи томонидан 0 дан 33 гача оралиқдаги сон киритилганда амал қилади, чунки 34! қиймати unsigned long учун ажратилган разрядларга сиғмайди.

Масала. Такрорлаш операторининг ичма-ич жойлашувига мисол сифатида рақамлари бир-бирига ўзаро тенг бўлмаган уч хонали натурал сонларни ўсиш тартибида чоп қилиш масаласини кўришимиз мумкин:

```
#include <iostream.h>
int main()
{
 unsigned char a2,a1,a0; // уч хонали сон рақамлари
 for (a2='1';a2<='9';a2++)//соннинг 2-рақами
  for(a1='0';a1<='9';a1++)//соннинг 1-рақами
   for(a0='0';a0<='9';a0++)//соннинг 0-рақами
// рақамларни ўзаро тенг эмаслигини текшириш
    if(a0!=a1 && a1!=a2 && a0!=a2) //ўзаро тенг эмас
     cout<<a2<<a1<<a0<<'\n';
 return 0;
}
```

Программада уч хонали соннинг ҳар бир рақами такрорлаш операторларининг параметрлари сифатида ҳосил қилинади. Биринчи, ташқи такрорлаш оператори билан 2-хонадаги рақам (a2 такрорлаш параметри) ҳосил қилинади. Иккинчи, ички такрорлаш операторида (a1 такрорлаш параметри) сон кўринишининг 1-хонасидаги рақам ва ниҳоят, унга нисбатан ички бўлган a0 параметрли такрорлаш операторида 0-хонадаги рақамлар ҳосил қилинади. Ҳар бир ташқи такрорлашнинг бир қадамига ички такрорлаш операторининг тўлиқ бажарилиши тўғри келиши ҳисобига барча уч хонали сонлар кўриниши ҳосил қилинади.

while такрорлаш оператори

while такрорлаш оператори, оператор ёки блокни такрорлаш шарти ёлғон (false ёки 0) бўлгунча такрор бажаради. У қуйидаги синтаксисга эга:

while (<ифода>) <оператор ёки блок>;

Агар <ифода> рост қийматли ўзгармас ифода бўлса, такрорлаш чексиз бўлади. Худди шундай, <ифода> такрорлаш бошланишида рост бўлиб, унинг қийматига такрорлаш танасидаги ҳисоблаш таъсир этмаса, яъни унинг қиймати ўзгармаса, такрорлаш чексиз бўлади.

while такрорлаш шартини олдиндан текширувчи такрорлаш оператори ҳисобланади. Агар такрорлаш бошида <ифода> ёлғон бўлса, while оператори таркибидаги <оператор ёки блок> қисми бажарилмасдан чеклаб ўтилади.

Айрим ҳолларда <ифода> қиймат бериш оператори кўринишида келиши мумкин. Бунда қиймат бериш амали бажарилади ва натижа 0 билан солиштирилади. Натижа нолдан фарқли бўлса, такрорлаш давом эттирилади.

Агар рост ифоданинг қиймати нолдан фарқли ўзгармас бўлса, чексиз такрорлаш рўй беради. Масалан:

```
while(1); // чексиз такрорлаш
```

Худди for операторидек, ‘,’ ёрдамида <ифода> да бир нечта амаллар синхрон равишда бажариш мумкин. Масалан, сон ва унинг квадратларини чоп қиладиган программада ушбу ҳолат кўрсатилган:

```
#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;
}
```

Программадаги такрорлаш оператори бажарилишида n сони 1 гача камайиб боради. Ҳар бир қадамда n ва унинг квадрати чоп қилинади. Шунга эътибор бериш керакки, шарт ифодасида операторларни ёзилиш кетма-кетлигининг аҳамияти бор, чунки энг охирги оператор такрорлаш шарти сифатида қаралади ва n қиймати 0 бўлганда такрорлаш тугайди.

Кейинги программада берилган ўнлик соннинг иккилик кўринишини чоп қилиш масаласини ечишда while операторини қўллаш кўрсатилган.

```
#include <iosteam.h>
int main()
{
 int sanagich=4;
 short son10,jarayon=1;
 while (jarayon)     // чексиз такрорлаш
 {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;  //разрядларни 1 ўрин чапга суриш
   sanagich--;
  }
  cout<<'\n';
  cout<<"Jarayonni to'xtasin(0),davom etsin(1):_ ";
  cin >> jarayon;
  sanagich=4;
 }
 return 0;
}
```

Программада ичма-ич жойлашган такрорлаш операторлари ишлатилган. Биринчиси, соннинг иккилик кўринишини чоп қилиш жараёнини давом эттириш шарти бўйича амал қилади. Ички жойлашган иккинчи такрорлаш операторидаги амаллар - ҳар қандай, 0 дан 15 гача бўлган сонлар тўртта разрядли иккилик сон кўринишида бўлишига асосланган. Унда киритилган соннинг ички, иккилик кўринишида учинчи разрядида 0 ёки 1 турганлиги аниқланади ("son10&8"). Шарт натижаси натижа 1 (рост) бўлса, экранга '1', акс ҳолда '0' белгиси чоп этилади. Кейинги қадамда сон разрядлари чапга биттага сурилади ва яна учинчи разряддаги рақам чоп этилади. Такрорлаш sanagich қиймати 0 бўлгунча яъни тўрт марта бажарилади ва бошқарув ички такрорлаш операторидан чиқади.

while такрорлаш оператори ёрдамида самарали программа коди ёзишга яна бир мисол бу - иккита натурал сонларнинг энг катта умумий бўлувчисини (ЭКУБ) Эвклид алгоритми билан топиш масаласини келтиришимиз мумкин:

```
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;
}
```

Бутун турдаги a ва b қийматлари оқимдан ўқилгандан кейин токи уларнинг қийматлари ўзаро тенг бўлмагунча такрорлаш жараёни рўй беради. Такрорлашнинг ҳар бир қадамида a ва b сонларнинг

каттасидан кичиги айрилади. Такрорлашдан кейинги кўрсатма воситасида а ўзгарувчисининг қиймати натижа сифатида чоп этилади.

do-while такрорлаш оператори

do-while такрорлаш оператори while операторидан фарқли равишда олдин оператор ёки блокни бажаради, кейин такрорлаш шартини текширади. Бу қурилма такрорлаш танасини камида бир марта бажарилишини таъминлайди. do-while такрорлаш оператори қуйидаги синтаксисга эга:

do <оператор ёки блок>; while (<ифода>);

Бундай такрорлаш операторининг кенг қўлланиладиган ҳолатлари - бу такрорлашни бошламасдан туриб, такрорлаш шартини текширишнинг иложи бўлмаган ҳолатлар ҳисобланади. Масалан, бирорта жараённи давом эттириш ёки тўхтатиш ҳақидаги сўровга жавоб олиш ва уни текшириш зарур бўлсин. Кўриниб турибдики, жараённи бошламасдан олдин бу сўровни беришнинг маъноси йўқ. Ҳеч бўлмаганда такрорлаш жараёнининг битта қадами амалга оширилган бўлиши керак:

```
#include <iostream.h>
int main()
{
  char javob;
  do
  {
   ...   // программа танаси
  cout<<"Jarayonni to'xtatish (N):_ ";
  cin>>javob;
  } while(javob !=N)
 return 0;
}
```

Программа токи "Jarayonni to'xtatish (N):_ " сўровига 'N' жавоби киритилмагунча давом этади.

Бу оператор ҳам чексиз такрорланиши мумкин:

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

Масала. Ҳар қандай 7 катта бутун сондаги пул миқдорини 3 ва 5 сўмликларда бериш мумкинлиги исботлансин. Қўйилган масала p=3n+5m тенгламаси қаноатлантирувчи m, n сонлар жуфтликларини топиш масаласидир (p-пул миқдори). Бу шартнинг бажарилишини m ва n ўзгарувчиларининг мумкин бўлган қийматларининг барча комбинацияларида текшириш зарур бўлади.

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

```
int main()
{
 unsigned int Pul; //Pul- киритиладиган пул миқдори
 unsigned n3,m5;   //n-3 сўмликлар,m-5 сўмликлар сони
 bool xato=false; //Pul қийматини киритишдаги хатолик
 do
 {
  if(xato)cout<<"Pul qiymati 7 dan kichik!";
  xato=true; // кейинги такрорлаш хато ҳисобланади
  cout<<"\nPul qiymatini kiriting (>7): ";
  cin>>Pul;
 }
  while(Pul<=7);//токи 7 катта сон киритилгунча
 n3=0;           //бирорта ҳам 3 сўмлик йўқ
 do
 {
  m5=0;         //бирорта ҳам 5 сўмлик йўқ
  do
  {
   if (3*n3+5*m5==Pul)
   cout<<n3<<" ta 3 so'mlik+"<<m5<<" ta 5 so'mlik\n";
   m5++;      // 5 сўмликлар биттага оширилади
  } while(3*n3+5*m5<=Pul);
  n3++;       //3 сўмликлар биттага оширилади
 } while(3*n3<=Pul);
 return 0;
}
```

Программа пул қийматини киритишни сўрайди (Pul ўзгарувчисига). Агар пул қиймати 7 сонидан кичик бўлса, бу ҳақда хабар берилади ва такрор равишда қиймат киритиш талаб қилинади. Пул қиймати 7 дан катта бўлганда, 3 ва 5 сўмликларнинг мумкин бўлган тўла комбинациясини амалга ошириш учун ичма-ич такрорлашлар амалга оширилади. Ташқи такрорлаш n3 (3 сўмликлар миқдори) бўйича, ички такрорлаш эса m5 (5 сўмликлар миқдори) бўйича, токи бу миқдордаги пуллар қиймати Pul қийматидан ошиб кетмагунча давом этади. Ички такрорлашда m5 ўзгарувчисининг ҳар бир қийматида «3*n3+5*m5==Pul» шарти текширилади, агар у ўринли бўлса, ечим варианти сифатида n3 ва m5 ўзгарувчилар қийматлари чоп этилади.

Пул қиймати 30 сўм киритилганда (Pul=30), экранга

```
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
```

ечим вариантлари чоп этилади.

break оператори

Такрорлаш операторларининг бажарилишида шундай ҳолатлар юзага келиши мумкинки, унда қайсидир қадамда, такрорлашни якунига етказмасдан такрорлашдан чиқиш зарурати бўлиши мумкин. Бошқача айтганда, такрорлашни «узиш» керак бўлиши мумкин. Бунда break операторидан фойдаланилади. break операторини такрорлаш оператори танасининг ихтиёрий (зарур) жойларига қўйиш орқали шу жойлардан такрорлашдан чиқишни амалга ошириш мумкин. Эътибор берадиган бўлсак, switch-case операторининг туб моҳиятига ҳам break операторини қўллаш орқали эришилган.

Ичма - ич жойлашган такрорлаш ва switch операторларида break оператори фақат ўзи жойлашган блокдан чиқиш имкониятини беради.

Қуйидаги программада иккита ичма-ич жойлашган такрорлаш операторидан фойдаланган ҳолда фойдаланувчи томонидан киритилган қандайдир сонни 3 ва 7 сонларига нисбатан қандай ораликқа тушиши аниқланади. Ташқи такрорлашда “Son kiriting (0- to’xtash):_” сўрови берилади ва жавоб javob_son ўзгарувчисига ўқилади. Агар сон нолдан фарқли бўлса, ички такрорлаш операторида бу соннинг қандайдир ораликқа тушиши аниқланиб, шу ҳақида хабар берилади ва ички такрорлаш операторидан чиқилади. Ташқи такрорлашдаги сўровга жавоб тариқасида 0 киритилса, программа ўз ишини тугатади.

```
#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;
}
```

Амалиётда break операторидан чексиз такрорлашдан чиқишда фойдаланилади.

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

Бу мисолда чексиз for такрорлашидан 1 ёки 2 - шарт бажарилганда чиқилади.

Масала. Ишорасиз бутун сонлар кетма-кетлиги 0 қиймати билан тугайди, 0 кетма-кетлик ҳади ҳисобланмайди. Кетма-кетликни камаймайдиган ҳолда тартибланган ёки йўқлиги аниқлансин.

```
#include <iostream.h>
int main()
{
 unsigned int Ai_1=0,Ai;
 cout<<"Sonlar ketma-ketligini kiriting”
 cout<<(0-tugash alomati):\n ";
 cin>>Ai;          // кетма-кетликнинг биринчи ҳади
 while(Ai)
 {
  Ai_1=Ai;
  cin>>Ai;         // навбатдаги ҳад
  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;
}
```

Программа ишга тушганда, бошда кетма-кетликнинг биринчи ҳади алоҳида ўқиб олинади (Ai ўзгарувчисига). Кейин Ai қиймати нолга тенг бўлмагунча такрорлаш оператори амал қилади. Такрорлаш танасида Ai қиймати олдинги қиймат сифатида Ai_1 ўзгарувчисида эслаб қолинади ва навбатдаги ҳад Ai ўзгарувчисига ўқилади. Агар олдинги ҳад навбатдаги ҳаддан катта бўлса, break оператори ёрдамида такрорлаш жараёни узилади ва бошқарув такрорлашдан кейинги шарт операторига ўтади. Бу ердаги шарт операторлари мазмуни қуйидагича: агар Ai_1 нолдан фарқли бўлса, кетма-кетликнинг камида битта ҳади киритилган бўлади (кетма-кетлик мавжуд) ва охирги киритилган ҳад текширилади. Ўз навбатида агар Ai нолдан фарқли бўлса, бу ҳолат ҳадлар ўртасида камаймаслик шарти бажарилмаганлиги сабабли ҳадларни киритиш жараёни узилганини билдиради ва бу ҳақда хабар чоп этилади. Акс ҳолда кетма-кетликни камаймайдиган ҳолда тартибланган бўлади.

continue оператори

continue оператори худди break операторидек такрорлаш операторини танасини бажаришни тўхтатади, лекин такрорлашдан чиқиб кетмасдан кейинги қадамига «сакраб» ўтишини тайинлайди.

continue операторини қўлланишига мисол тариқасида 2 ва 50 сонлар оралиғидаги туб сонларни топадиган программа матнини келтирамиз.

```
#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 бажарилганда бошқарув ўтадиган жой
  if (!bulinadi) cout <<i<<" ";
  bulinadi=false;
 }
  return 0;
}
```

Келтирилган программада қўйилган масала ичма-ич жойлашган иккита такрорлаш операторлари ёрдамида ечилган. Биринчи такрор-

лаш оператори 2 дан 50 гача сонларни ҳосил қилишга хизмат қилади. Ички такрорлаш эса ҳар бир ҳосил қилинаётган сонни 2 сонидан токи шу соннинг ярмигача бўлган сонларга бўлиб, қолдиғини текширади, агар қолдиқ 0 сонидан фарқли бўлса, навбатдаги сонга бўлиш давом этади, акс ҳолда bulinadi ўзгарувчисига true қиймат бериб, ички такрорлаш узилади (сон ўзининг ярмигача бўлган қандайдир сонга бўлинар экан, демак у туб эмас ва кейинги сонларга бўлиб текширишга ҳожат йўқ). Ички j бўйича такрорлашдан чиққандан кейин bulinadi қиймати false бўлса (!bulinadi), i сони туб бўлади ва у чоп қилинади.

goto оператори ва нишонлар

Нишон - бу давомида иккита нуқта (‘:’) қўйилган идентификатор. Нишон билан қандайдир оператор белгиланади ва кейинчалик, программанинг бошқа бир қисмидан унга шартсиз ўтиш амалга оширилади. Нишон билан ҳар қандай оператор белгиланиши мумкин, шу жумладан эълон оператори ва бўш оператори ҳам. Нишон фақат функциялар ичида амал қилади.

Нишонга шартсиз ўтиш goto оператори ёрдамида бажарилади. goto оператори орқали фақат унинг ўзи жойлашган функция ичидаги операторларга ўтиш мумкин. goto операторининг синтаксиси қуйидагича:

goto <нишон>;

Айрим ҳолларда, goto операторининг «сакраб ўтиши» ҳисобига хатоликлар юзага келиши мумкин. Масалан,

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

операторларининг бажарилиши хатоликка олиб келади, чунки j эълон қилинмай қолади.

Шартсиз ўтиш оператори программани тузишдаги кучли ва шу билан биргаликда хавфли воситалардан бири ҳисобланади. Кучлилиги шундаки, унинг ёрдамида алгоритмнинг «боши берк» жойларидан чиқиб кетиш мумкин. Иккинчи томондан, блокларнинг ичига ўтиш, масалан, такрорлаш операторларини ичига «сакраб» кириш кутилмаган ҳолатларни юзага келтириши мумкин. Шу сабабли, имкон қадар goto операторидан фойдаланмаслик керак, ишлатилган тақдирда ҳам қуйидаги қоидага амал қилиш зарур: «*блок ичига, if…else ва switch операторлари ичига, ҳамда такрорлаш операторлари танасига ташқаридан кириш мумкин эмас*».

Гарчи, нишон ёрдамида программанинг ихтиёрий жойига ўтиш мумкин бўлса ҳам, бошланғич қиймат бериш эълонларидан сакраб ўтиш ман этилади, лекин блоклардан сакраб ўтиш мумкин.

Масалан:

```
...
goto B;       \\ хато
float x=0.0;
goto  B;      \\ тўғри
{ int n=10;x=n*x+x; }
B: cout << "x="<<x;
...
```

Хусусан, нишон ёрдамида ички блокдан ташқи блокка ва ташқи блокдан ички блокка ўтишга C++ тили рухсат беради:

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

Лекин, юқорида келтирилган мисолдаги барча ўтишлар мазмунан хато ҳисобланади.

Қуйидаги программада иккита натурал сонлар энг катта умумий бщлувчини (ЭКУБ) топиш масаласидаги такрорлаш жараёнини нишон ва goto оператори воситасида амалга ошириш кўрсатилган:

```
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;
}
```

Программадаги нишон билан белгиланган операторда a ва b сонларни тенглиги текширилади. Агар улар тенг бўлса, ихтиёрий битташи, масалан a сони ЭКУБ бўлади ва функциядан чиқилади. Акс ҳолда, бу сонларнинг каттасидан кичиги айрилади ва goto орқали уларнинг тенглиги текширилади. Такрорлаш жараёни a ва b сонлар ўзаро тенг бўлгунча давом этади.

Шуни қайд этиш керакки, бу масалани такрорлаш операторлари ёрдамида бажариш анча самарали ҳисобланади.

5-боб. Функциялар

Программа таъминотини яратиш амалда мураккаб жараён ҳисобланади. Программа тузувчи программа комплексини бир бутунликдаги ва унинг ҳар бир бўлагининг ички мазмунини ва уларнинг сезилмас фарқларини ҳисобга олиши керак бўлади.

Программалашга тизимли ёндошув шундан иборатки, программа тузувчи олдига қўйилган масала олдиндан иккита, учта ва ундан ортиқ нисбатан кичик масала остиларга бўлинади. Ўз навбатида бу масала остилари ҳам яна кичик масала остиларига бўлиниши мумкин. Бу жараён токи майда масалаларни оддий стандарт амаллар ёрдамида ечиш мумкин бўлгунча давом этади. Шу йўл билан масалани декомпозициялаш амалга оширилади.

Иккинчи томондан, программалашда шундай ҳолатлар кузатиладики, унда программанинг турли жойларида мазмунан бир хил алгоритмларни бажаришга тўғри келади. Алгоритмнинг бу бўлаклари асосий ечилаётган масаладан ажратиб олинган қандайдир масала остини ечишга мўлжалланган бўлиб, етарлича мустақил қийматга (натижага) эгадир. Мисол учун қуйидаги масалани кўрайлик:

Берилган $a_0,a_1,...,a_{30}$, $b_0,b_1,...,b_{30}$, $c_0,c_1,...,c_{30}$ ва x,y,z ҳақиқий сонлар учун

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

ифоданинг қиймати ҳисоблансин.

Бу мисолни ечишда касрнинг сурат ва махражидаги ифодалар бир хил алгоритм билан ҳисобланади ва программада ҳар бир ифодани (масала ости) ҳисоблаш учун бу алгоритмни 3 марта ёзишга тўғри келади. Масаладаги 30-даражали кўпҳадни ҳисоблаш алгоритмини, масалан, Горнер алгоритмини алоҳида, битта нусхада ёзиб, унга турли параметрлар - бир сафар a вектор ва x қийматини, иккинчи сафар b вектор ва y қийматини, ҳамда c вектор ва (x+z) қийматлари билан мурожаат қилиш орқали асосий масалани ечиш мумкин бўлади. Функциялар қўлланишининг яна бир сабабини қуйидаги масалада кўришимиз мумкин - берилган чизиқли тенгламалар системасини Гаусс, Крамер, Зейдел усулларининг бирортаси билан ечиш талаб қилинсин. У ҳолда асосий программани қуйидаги бўлакларга бўлиш мақсадга мувофиқ бўлар эди: тенглама коэффицентларини киритиш бўлаги, ечиш усулини танлаш бўлаги, Гаусс, Крамер, Зейдел усулларини амалга ошириш учун алоҳида бўлаклар, натижани чоп қилиш бўлаги. Ҳар бир бўлак учун ўз функциялар мажмуаси яратиб, зарур

бўлганда уларга бош функция танасидан мурожаатни амалга ошириш орқали бош масала ечиш самарали ҳисобланади.

Бундай ҳолларда программани ихчам ва самарали қилиш учун C++ тилида программа бўлагини алоҳида ажратиб олиб, уни функция кўринишида аниқлаш имкони мавжуд.

Функция бу - C++ тилида масала ечишдаги калит элементларидан биридир.

Функция параметрлари ва аргументлари

Программада ишлатиладиган ҳар қандай функция эълон қилиниши керак. Одатда функциялар эълони сарлавҳа файлларда эълон қилинади ва #include директиваси ёрдамида программа матнига қўшилади.

Функция эълонини *функция прототипи* тавсифлайди (айрим ҳолларда *сигнатура* дейилади). Функция прототипи қуйидаги кўринишда бўлади:

<қайтарувчи қиймат тури> <функция номи>(<параметрлар рўйхати >);

Бу ерда <қайтарувчи қиймат тури> - функция ишлаши натижасида у томонидан қайтарадиган қийматнинг тури. Агар қайтариладиган қиймат тури кўрсатилмаган бўлса, келишув бўйича функция қайтарадиган қиймат тури int деб ҳисобланади, <параметрлар рўйхати>- вергул билан ажратилган функция параметрларининг тури ва номлари рўйхати. Параметр номини ёзмаса ҳам бўлади. Рўйхат бўш бўлиши ҳам мумкин. Функция прототипларига мисоллар:

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

Функция прототипи тушириб қолдирилиши мумкин, агар программа матнида функция аниқланиши уни чақирадиган функциялар матнидан олдин ёзилган бўлса. Лекин бу ҳолат яхши услуб ҳисобланмайди, айниқса ўзаро бир-бирига мурожаат қилувчи функцияларни эълон қилишда муаммолар юзага келиши мумкин.

Функция аниқланиши - функция сарлавҳаси ва фигурали қавсга (‘{‘,’}’) олинган қандайдир амалий мазмунга эга танадан иборат бўлади. Агар функция қайтарувчи тури void туридан фарқли бўлса, унинг танасида албатта мос турдаги параметрга эга return оператори бўлиши шарт. Функция танасида биттадан ортиқ return оператори бўлиши мумкин. Уларнинг ихтиёрий бирортасини бажариш орқали функциядан чиқиб кетилади. Агар функциянинг қиймати програм-

мада ишлатилмайдиган бўлса, функциядан чиқиш учун параметрсиз return оператори ишлатилиши мумкин ёки умуман return ишлатилмайди. Охирги ҳолда функциядан чиқиш - охирги ёпилувчи қавсга етиб келганда рўй беради.

Функция программанинг бирорта модулида ягона равишда аниқланиши керак, унинг эълони эса функцияни ишлатадиган модулларда бир неча марта ёзилиши мумкин. Функция аниқланишида сарлавҳадаги барча параметрлар номлари ёзилиши шарт.

Одатда программада функция маълум бир ишни амалга ошириш учун чақирилади. Функцияга мурожаат қилганда, у қўйилган масалани ечади ва ўз ишини тугатишида қандайдир қийматни натижа сифатида қайтаради.

Функцияни чақириш учун унинг номи ва ундан кейин қавс ичида аргументлар рўйхати берилади:

<функция номи>(<аргумент$_1$>, <аргумент$_2$>,…, <аргумент$_n$ >);

Бу ерда ҳар бир <аргумент> - функция танасига узатиладиган ва кейинчалик ҳисоблаш жараёнида ишлатиладиган ўзгарувчи, ифода ёки ўзгармасдир. Аргументлар рўйхати бўш бўлиши мумкин.

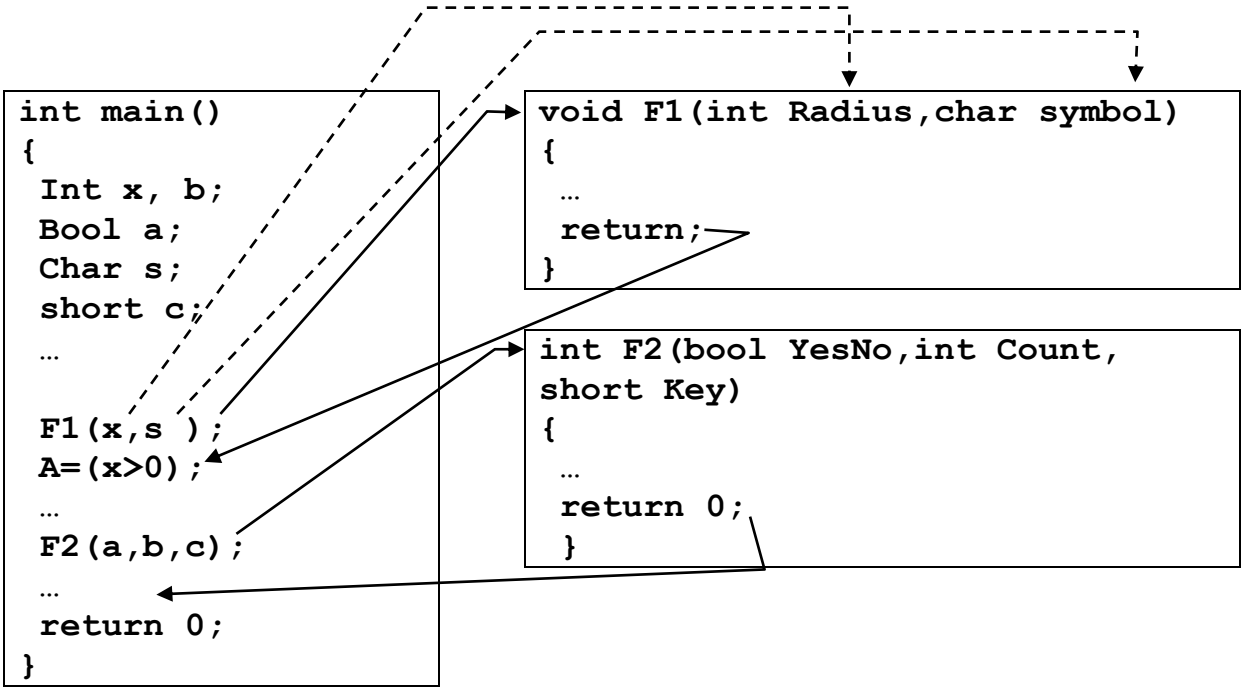
Функциялар ҳам ўз танасида бошқа функцияларни, ўзини ҳам чақириши мумкин. Ўз танасида ўзини чақирадиган функцияларга *рекурсив функциялар* дейилади.

Олдинги бобларда таъкидлаб ўтилганидек, C++ тилидаги ҳар қандай программада албатта main() бош функцияси бўлиши керак. Айни шу функцияни юклагич томонидан чақирилиши билан программа бажарилиши бошланади.

5.1- расмда бош функциядан бошқа функцияларни чақириш ва улардан қайтиш схемаси кўрсатилган.

Программа main() функциясини бажаришдан бошланади ва «f1(x,y);» - функция чақиришгача давом этади ва кейинчалик бошқарув f1(x,y) функция танасидаги амалларни бажаришга ўтади. Бунда Radius параметрининг қиймати сифатида функция x ўзгарувчи қийматини, Symbol параметри сифатида y ўзгарувчисининг қиймати ишлатилади. Функция танаси return операторигача бажарилади. return оператори бошқарувни main() функцияси танасидаги f1() функцияси чақирилган оператордан кейинги операторга ўтишни таъминлайди, яъни функциядан қайтиш рўй беради. Шундан кейин main() функцияси операторлари бажарилишда давом этади ва «f2(a,b,c);» - функция чақириши орқали бошқарув f2() функция танасига ўтади ва ҳисоблаш жараёнида мос равишда YesNo сифатида a ўзгарувчисининг, count сифатида b ўзгарувчисининг ва key сифатида c ўзгарувчисининг

қийматлари ишлатилади. Функция танасидаги return оператори ёки охирги оператор бажаргандан кейин автоматик равишда бош функцияга қайтиш амалга оширилади.



5.1-расм. Бош функциядан бошқа функцияларни чақириш ва қайтиш

Аксарият ҳолларда main() функциясининг параметрлар рўйхати бўш бўлади. Агар юкланувчи программани ишга туширишда,буйруқ сатри орқали юкланувчи программа ишга туширилганда, унга параметрларни узатиш (бериш) зарур бўлса, main() программаси функциясининг синтаксиси ўзгаради:

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

Бу ерда argc - узатиладиган параметрлар сони, argv[]- бир-биридан пунктуация белгилари (ва пробел) билан ажратилган параметрлар рўйхатини ўз ичига олган массивга кўрсаткич.

Қуйида функцияларни эълон қилиш, чақириш ва аниқлашга мисоллар келтирилган:

```
// функциялар эълони
int Mening_funksiyam(int Number, float Point);
char Belgini_uqish();
void bitni_urnatish(short Num);
void Amal_yoq(int,char);
// функцияларни чақириш
result=Mening_funksiyam(Varb1,3.14);
symb=Belgini_uqish();
bitni_urnatish(3);
Amal_yoq(2,Smbl);
// функцияларни аниқлаш
```

```
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){};
```

Функциянинг программадаги ўрнини янада тушунарли бўлиши учун сон квадратини ҳисоблаш масаласида функциядан фойдаланишни кўрайлик.

Функция прототипини sarlavha.h сарлавҳа файлида жойлаштирамиз:

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

Асосий программага ушбу сарлавҳа файлини қўшиш орқали Son_Kvadrati() функция эълони программа матнига киритилади:

```
#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;}
```

Худди шу масалани сарлавҳа файлидан фойдаланмаган ҳолда, функция эълонини программа матнига ёзиш орқали ҳам ҳал қилиш мумкин:

```
#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;}
```

Программа ишлашида ўзгариш бўлмайди ва натижа сифатида экранга 25 сонини чоп этади.

Масала. Иккита туб сон «эгизак» дейилади, агар улар бир-биридан 2 фарқ қилса (масалан, 41 ва 43 сонлари). Берилган натурал n

учун [n..2n] кесмадаги барча «эгизак» сонлар жуфтликлари чоп этилсин. Масалани ечиш учун берилган k сонини туб сон ёки йўқлиги аниқловчи мантиқий функцияни тузиш зарур бўлади. Функцияда k сони 2..k/2 гача сонларга бўлинади, агар k бу сонларнинг бирортасига ҳам бўлинмаса, у туб сон ҳисобланади ва функция true қийматини қайтаради. Бош функцияда, берилган n учун [n..2n] оралиқдаги (n, n+2), (n+1,n+3),..,(2n-2, 2n) сон жуфтликларини туб сонлар эканлиги текширилади ва шартни қаноатлантирган жуфтликлар чоп этилади.

Программа матни:

```
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;
}
```

Натурал n сони учун 100 киритилса, программа қуйидаги сонлар жуфтликларини чоп қилади:

```
[100..200] oralig'idagi egizak tub sonlar:
{101,103}; {107,109}; {137,139}; {149,151};
{179,181}; {191,193}; {197,199}.
```

Келишув бўйича аргументлар

C++ тилида функция чақирилганда айрим аргументларни тушириб қолдириш мумкин. Бунга функция прототипида ушбу параметрларни келишув бўйича қийматини кўрсатиш орқали эришиш мумкин. Масалан, қуйида прототипи келтирилган функция турли чақиришга эга бўлиши мумкин:

```
//функция прототипи
void Butun_Son(int I,bool Bayroq=true,char Blg='\n');
//функцияни чақириш вариантлари
Butun_Son(1,false,'a');
Butun_Son(2,false);
Butun_Son(3);
```

Биринчи чақирувда барча параметрлар мос аргументлар орқали қийматларини қабул қилади, иккинчи ҳолда I параметри 2 қийматини, bayroq параметри false қийматини ва Blg ўзгарувчиси келишув бўйича '\n' қийматини қабул қилади.

Келишув бўйича қиймат беришнинг битта шарти бор - параметрлар рўйхатида келишув бўйича қиймат берилган параметрлардан кейинги параметрлар ҳам келишув бўйича қийматга эга бўлишлари шарт. Юқоридаги мисолда I параметри келишув бўйича қиймат қабул қилинган ҳолда, Bayroq ёки Blg параметрлари қийматсиз бўлиши мумкин эмас. Мисол тариқасида берилган сонни кўрсатилган аниқликда чоп этувчи программани кўрайлик. Қўйилган масалани ечишда сонни даражага ошириш функцияси - pow() ва сузувчи нуқтали узун сондан модул олиш fabsl() функциясидан фойдаланилади. Бу функциялар прототипи «math.h» сарлавҳа файлида жойлашган (3-илова қаранг):

```
#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';
}
```

Программада сонни турли аниқликда (Aniqlik параметри қиймати орқали) чоп этиш учун ҳар хил вариантларда Chop_qilish( ) функцияси чақирилган. Программа ишлаши натижасида экранда қуйидаги сонлар чоп этилади:

```
3.1415
-3.14
-3.1
```

Параметрнинг келишув бўйича бериладиган қиймати ўзгармас, глобал ўзгарувчи ёки қандайдир функция томонидан қайтарадиган қиймат бўлиши мумкин.

Кўриниш соҳаси. Локал ва глобал ўзгарувчилар

Ўзгарувчилар функция танасида ёки ундан ташқарида эълон қилиниши мумкин. Функция ичида эълон қилинган ўзгарувчиларга *локал ўзгарувчилар* дейилади. Бундай ўзгарувчилар хотирадаги программа стекида жойлашади ва фақат ўзи эълон қилинган функция танасида амал қилади. Бошқарув асосий функцияга қайтиши билан локал ўзгарувчилар учун ажратилган хотира бўшатилади (ўчирилади).

Ҳар бир ўзгарувчи ўзининг амал қилиш соҳаси ва яшаш вақти хусусиятлари билан характерланади.

Ўзгарувчи *амал қилиш соҳаси* деганда ўзгарувчини ишлатиш мумкин бўлган программа соҳаси (қисми) тушунилади. Бу тушунча билан ўзгарувчининг *кўриниш соҳаси* узвий боғланган. Ўзгарувчи амал қилиш соҳасидан чиққанда кўринмай қолади. Иккинчи томондан, ўзгарувчи амал қилиш соҳасида бўлиши, лекин кўринмаслиги мумкин. Бунда кўриниш соҳасига рухсат бериш амали «::» ёрдамида кўринмас ўзгарувчига мурожат қилиш мумкин бўлади.

Ўзгарувчининг *яшаш вақти* деб, у мавжуд бўлган программа бўлагининг бажарилишига кетган вақт интервалига айтилади.

Локал ўзгарувчилар ўзлари эълон қилинган функция ёки блок чегарасида кўриниш соҳасига эга. Блокдаги ички блокларда худди шу номдаги ўзгарувчи эълон қилинган бўлса, ички блокларда бу локал ўзгарувчи ҳам амал қилмай қолади. Локал ўзгарувчи яшаш вақти - блок ёки функцияни бажариш вақти билан аниқланади. Бу ҳол шуни англатадики, турли функцияларда бир-бирига умуман боғлиқ бўлмаган бир хил номдаги локал ўзгарувчиларни ишлатиш мумкин.

Қуйидаги программада main() ва sum() функцияларида бир хил номдаги ўзгарувчиларни ишлатиш кўрсатилган. Программада иккита соннинг йиғиндиси ҳисобланади ва чоп этилади:

```
#include <iostream.h>
// функция прототипи
int sum(int a;int b);
int main()
{
 // локал ўзгарувчилар
 int x=r;
 int y=4;
 cout<<sum(x, y);
 return 0;
}
int sum(int a,int b)
{
 // локал ўзгарувчи
 int x=a+b;
 return x;
}
```

Глобал ўзгарувчилар программа матнида функция аниқланишидан ташқарида эълон қилинади ва эълон қилинган жойидан бошлаб программа охиригача амал қилади.

```
#include <iostream.h>
int f1(); int f2();
int main()
{
 cout<<f1()<<" "<<f2()<<endl;
 return 0;
}
int f1()
{
 return x;// компиляция хатоси рўй беради
}
int x=10; // глобал ўзгарувчи эълони
int f2(){ return x*x;}
```

Юқорида келтирилган программада компиляция хатоси рўй беради, чунки f1() функция учун x ўзгарувчиси номаълум ҳисобланади.

Программа матнида глобал ўзгарувчиларни улар эълонидан кейин ёзилган ихтиёрий функцияда ишлатиш мумкин. Шу сабабли, глобал ўзгарувчилар программа матнининг бошида ёзилади. Функция ичидан глобал ўзгарувчига мурожат қилиш учун функцияда унинг номи билан мос тушадиган локал ўзгарувчилар бўлмаслиги керак.

Агар глобал ўзгарувчи эълонида унга бошланғич қиймат берилмаган бўлса, уларнинг қиймати 0 ҳисобланади. Глобал ўзгарувчининг амал қилиш соҳаси унинг кўриниш соҳаси билан устма-уст тушади.

Шуни қайд этиш керакки, тажрибали программа тузувчилар имкон қадар глобал ўзгарувчиларни ишлатмасликка ҳаракат қилишади, чунки бундай ўзгарувчилар қийматини программанинг ихтиёрий жойидан ўзгартириш хавфи мавжудлиги сабабли программа ишлашида мазмунан хатолар юзага келиши мумкин. Бу фикримизни тасдиқловчи программани кўрайлик.

```
# include <iostream.h>
// глобал ўзгарувчи эълони
int test=100;
void Chop_qilish(void );
int main()
{
 //локал ўзгарувчи эълони
 int test=10;
 //глобал ўзгарувчи чоп қилиш функциясини чақириш
 Chop_qilish();
 cout<<"Lokal o'zgaruvchi: "<<test<<'\n';
 return 0;
}
void Chop_qilish(void)
{
 cout<<"Global o'zgaruvchi: "<<test<<'\n';
}
```

Программа бошида test глобал ўзгарувчиси 100 қиймати билан эълон қилинади. Кейинчалик, main() функциясида test номи билан локал ўзгарувчиси 10 қиймати билан эълон қилинади. Программада, Chop_qilish() функциясига мурожаат қилинганида, асосий функция танасидан вақтинча чиқилади ва натижада main() функциясида эълон қилинган барча локал ўзгарувчиларга мурожаат қилиш мумкин бўлмай қолади. Шу сабабли Chop_qilish() функциясида глобал test ўзгарувчисининг қийматини чоп этилади. Асосий программага қайтилгандан кейин, main() функциясидаги локал test ўзгарувчиси глобал test ўзгарувчисини «беркитади» ва локал test ўзгарувчини қиймати чоп этилади. Программа ишлаши натижасида экранга қуйидаги натижалар чоп этилади:

```
Глобал ўзгарувчи: 100
Локал ўзгарувчи: 10
```

:: амали

Юқорида қайд қилингандек, локал ўзгарувчи эълони худди шу номдаги глобал ўзгарувчини «беркитади» ва бу жойдан глобал ўзгарувчига мурожат қилиш имкони бўлмай қолади. C++ тилида бундай ҳолатларда ҳам глобал ўзгарувчига мурожат қилиш имконияти сақланиб қолинган. Бунинг учун «кўриниш соҳасига рухсат бериш» амалидан фойдаланиш мумкин ва ўзгарувчи олдига иккита нуқта - «::» қўйиш зарур бўлади. Мисол тариқасида қуйидаги програмани келтирамиз:

```
#include <iostream.h >
//глобал ўзгарувчи эълони
int uzg=5;
int main()
{
//локал ўзгарувчи эълони
 int uzg=70;
 //локал ўзгарувчини чоп этиш
 cout<<uzg<<'/n';
 //глобал ўзгарувчини чоп этиш
 cout<<::uzg <<'/n';
 return 0;
}
```

Программа ишлаши натижасида экранга олдин 70 ва кейин 5 сонлари чоп этилади.

Хотира синфлари

Ўзгарувчиларнинг кўриниш соҳаси ва амал қилиш вақтини аниқловчи ўзгарувчи модификаторлари мавжуд (5.1-жадвал).

5.1-жадвал. Ўзгарувчи модификаторлари

Модификатор	Қўлланиши	Амал қилиш соҳаси	Яшаш даври
auto	локал	блок	вақтинча
register	локал	блок	вақтинча
exteru	глобал	блок	вақтинча
static	локал	блок	доимий
	глобал	файл	доимий
volatile	глобал	файл	доимий

Автомат ўзгарувчилар. auto модификатори локал ўзгарувчилар эълонида ишлатилади. Одатда локал ўзгарувчилар эълонида бу модификатор келишув бўйича қўлланилади ва шу сабабли амалда уни ёзишмайди:

```
#include <iostream.h>
int main()
{
 auto int X=2; // int X=2; билан эквивалент
 cout<<X;
 returu 0;
}
```

auto модификатори блок ичида эълон қилинган локал ўзгарувчиларга қўлланилади. Бу ўзгарувчилар блокдан чиқиши билан автоматик равишда йўқ бўлиб кетади.

Регистр ўзгарувчилар. register модификатори компиляторга, кўрсатилган ўзгарувчини процессор регистрларига жойлаштиришга ҳаракат қилишни тайинлайди. Агар бу ҳаракат натижа бермаса ўзгарувчи auto туридаги локал ўзгарувчи сифатида амал қилади.

Ўзгарувчиларни регистрларда жойлаштириш программа кодини бажариш тезлиги бўйича оптималлаштиради, чунки процессор хотирадаги берилганларга нисбатан регистрдаги қийматлар билан анча тез ишлайди. Лекин регистрлар сони чекланганлиги учун ҳар доим ҳам ўзгарувчиларни регистрларда жойлаштиришнинг иложи бўлмайди.

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

register модификатори фақат локал ўзгарувчиларига нисбатан қўлланилади, глобал ўзгарувчиларга қўллаш компиляция хатосига олиб келади.

Ташқи ўзгарувчилар. Агар программа бир нечта модулдан иборат бўлса, улар қандайдир ўзгарувчи орқали ўзаро қиймат алмашишлари мумкин (файллар орасида). Бунинг учун ўзгарувчи бирорта модулда глобал тарзда эълон қилинади ва у бошқа файлда (модулда) кўриниши учун у ерда extern модификатори билан эълон қилиниши керак бўлади. extern модификатори ўзгарувчини бошқа файлда эълон қилинганлигини билдиради. Ташқи ўзгарувчилар ишлатилган программани кўрайлик.

```
//Sarlavha.h файлида
void Bayroq_Almashsin(void);
// modul_1.cpp файлида
bool Bayroq;
```

```
void Bayroq_Almashsin(void){Bayroq=!Bayroq;}
// masala.cpp файлида
#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;
}
```

Олдин sarlavha.h файлида Bayroq_Almashsin( ) функция сарлавҳаси эълон қилинади, кейин modul_1.cpp файлида ташқи ўзгарувчи эълон қилинади ва Bayroq_Almashsin() функциясининг танаси аниқланади ва ниҳоят, masala.cpp файлида Bayroq ўзгарувчиси ташқи деб эълон қилинади.

Статик ўзгарувчилар. Статик ўзгарувчилар static модификатори билан эълон қилинади ва ўз хусусиятига кўра глобал ўзгарувчиларга ўхшайди. Агар бу турдаги ўзгарувчи глобал бўлса, унинг амал қилиш соҳаси - эълон қилинган жойдан программа матнининг охиригача бўлади. Агар статик ўзгарувчи функция ёки блок ичида эълон қилинадиган бўлса, у функция ёки блокка биринчи киришда инициализация қилинади. Ўзгарувчининг бу қиймати функция кейинги чақирилганида ёки блокка қайта киришда сақланиб қолади ва бу қийматни ўзгартириш мумкин. Статик ўзгарувчиларни ташқи деб эълон қилиб бўлмайди.

Агар статик ўзгарувчи инициализация қилинмаган бўлса, унинг биринчи мурожатдаги қиймати 0 ҳисобланади.

Мисол тариқасида бирорта функцияни неча маротаба чақирилганлигини аниқлаш масаласини кўрайлик:

```
#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;
}
```

Бу ерда асосий функциядан counter статик ўзгарувчига эга Sanagicht() функцияси 30 марта чақирилади. Функция биринчи марта чақирилганда sanagich ўзгарувчига 0 қийматини қабул қилади ва унинг қиймати бирга ортган ҳолда функция қиймати сифатида қайтарилади. Статик ўзгарувчилар қийматларини функцияни бир чақирилишидан иккинчисига сақланиб қолиниши сабабли, кейинги ҳар бир чақиришларда sanagich қиймати биттага ортиб боради.

Масала. Берилган ишорасиз бутун соннинг барча туб бўлувчилари аниқлансин. Масалани ечиш алгоритми қуйидаги такрорланувчи жараёндан иборат бўлади: берилган сон туб сонга (1-қадамда 2 га) бўлинади. Агар қолдиқ 0 бўлса, туб сон чоп қилинади ва бўлинувчи сифатида бўлинма олинади, акс ҳолда навбатдаги туб сон олинади. Такрорлаш навбатдаги туб сон бўлинувчига тенг бўлгунча давом этади.

Программа матни:

```
#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;
}
```

Программада навбатдаги туб сонни ҳосил қилиш функция кўринишида амалга оширилган. Navb_tub() функциясининг ҳар чақирилишида охирги туб сондан кейинги туб сон топилади. Охирги туб сонни «эслаб» қолиш учун tub ўзгарувчиси static қилиб аниқланган.

Программа ишга тушганда клавиатурадан n ўзгарувчисининг қиймати сифатида 60 сони киритилса, экранга қуйидаги кўпайтма чоп этилади:

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

volatile синфи ўзгарувчилари. Агар программада ўзгарувчини бирорта ташқи қурилма ёки бошқа программа билан боғлаш учун ишлатиш зарур бўладиган бўлса, у volatile модификатори билан эълон қилинади. Компилятор бундай модификаторли ўзгарувчини регистрга жойлаштиришга ҳаракат қилмайди. Бундай ўзгарувчилар эълонига мисол қуйида келтирилган:

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

Мисолдан кўриниб турибдики, volatile модификаторли ўзгармас ҳам эълон қилиниши мумкин.

Номлар фазоси

Маълумки, программага қўшилган сарлавҳа файлларида эълон қилинган идентификатор ва ўзгармаслар компилятор томонидан ягона глобал номлар фазосига киритилади. Агар программа кўп миқдордаги сарлавҳа файлларни ишлатса ва ундаги идентификаторлар (функция номлари ва ўзгарувчилар номлари, синфлар номлари ва ҳакозалар) ва ўзгармаслар номлари турли программа тузувчилар томонидан мустақил равишда аниқланган бўлса, бир хил номларни ишлатиш билан боғлиқ муаммолар юзага келиш эҳтимоли катта бўлади. Номлар фазоси тушунчасини киритилиши мазкур муаммони маълум бир маънода ҳал қилишга ёрдам беради. Агар программада янги идентификаторни аниқлаш керак бўлса ва худди шу номни бошқа модул-

ларда ёки кутубхоналарда ишлатиши хавфи бўладиган бўлса, бу идентификаторлар учун ўзининг шахсий номлар фазосини аниқлаш мумкин. Бунга namespace калит сўзидан фойдаланилган ҳолда эришилади:

```
namespace <номлар фазосининг номи>
 {
 // эълонлар
 }
```

Номлар фазоси ичида эълон қилинган идентификаторлар фақат <номлар фазосининг номи> кўриниш соҳасида бўлади ва юзага келиши мумкин бўлган келишмовчиликларнинг олди олинади.

Мисол тариқасида қуйидаги номлар фазосини яратайлик:

```
namespace Shaxsiy_nomlar
{
 int x,y,z;
 void Mening_funksiyam(char belgi);
}
```

Компиляторга конкрет номлар фазосидаги номларни ишлатиш кераклигини кўрсатиш учун кўриниш соҳасига рухсат бериш амалидан фойдаланиш мумкин:

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

Агар программа матнида конкрет номлар фазосига нисбатан кўп мурожаат қилинадиган бўлса using namespace қурилмасини ишлатиш орқали ёзувни соддалаштириш мумкин:

```
using namespace <номлар фазоси номи>;
```

Масалан,

```
using namespace Shaxsiy_nomlar;
```

кўрсатмаси компиляторга, бундан кейин токи навбатдаги using учрамагунча Shaxsiy_nomlar фазосидаги номлар ишлатилиши кераклигини билдиради:

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

Программа ва унга қўшилган сарлавҳа файллари томонидан аниқланадиган номлар фазоси std деб номланади. Стандарт фазога ўтиш керак бўлса

```
using namespace std;
```

кўрсатмаси берилади.

Агар бирорта номлар фазосидаги алоҳида бир номга мурожаат қилиш зарур бўлса, using қурилмасини бошқа шаклида фойдаланилади. Мисол учун

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

кўрсатмаси x идентификаторини Shaxsiy_nomlar фазосидан ишлатиш кераклигини билдиради.

Шуни қайд этиш керакки, using namespace қурилмаси стандарт номлар фазоси кўриниш соҳасини беркитади ва ундаги номга мурожаат қилиш учун кўриниш соҳасига рухсат бериш амалидан (std::) фойдаланиш зарур бўлади.

Номлар фазоси функция ичида эълон қилиниши мумкин эмас, лекин улар бошқа номлар фазоси ичида эълон қилиниши мумкин. Ичма-ич жойлашган номлар фазосидаги идентификаторга мурожаат қилиш учун уни қамраб олган барча номлар фазоси номлар кетма-кет равишда кўрсатилиши керак. Мисол учун, қуйидаги кўринишда номлар фазоси эълон қилинган бўлсин:

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

Ichki_n ўзгарувчисига мурожаат қуйидаги кўринишда бўлади:

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

Номлар фазосида функцияни эълон қилишда номлар фазосида фақат функция прототипини эълон қилиш ва функция танасини бошқа жойда эълон қилиш маъқул вариант ҳисобланади. Бу ҳолатнинг кўринишига мисол:

```
namespace Nomlar_fazosi
{
 char c;
 int I;
 void Functsiya(char Bayroq);
}
…
void Nomlar_fazosi::Functsiya(char Bayroq)
{
 // функция танаси
}
```

Умуман олганда, ўз номига эга бўлмаган номлар фазосини эълон қилиш мумкин. Бу ҳолда namespace калит сўзидан кейин ҳеч нима ёзилмайди. Мисол учун

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

кўринишидаги номлар фазоси элементларига мурожаат ҳеч бир префикс ишлатмасдан амалга оширилади. Номсиз номлар фазоси фақат ўзи эълон қилинган файл чегарасида амал қилади.

C++ тили номлар фазосининг псевдонимларини аниқлаш имконини беради. Бу йўл орқали номлар фазосини бошқа ном билан ишлатиш мумкин бўлади. Масалан, номлар фазоси номи узун бўлганда унга қисқа ном билан мурожаат қилиш:

```
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;
```

Жойлаштириладиган (inline) функциялар

Компилятор ишлаши натижасида ҳар бир функция машина коди кўринишида бўлади. Агар программада функцияни чақириш кўрсатмаси бўлса, шу жойда функцияни адреси бўйича чақиришнинг машина коди шаклланади. Одатда функцияни чақириш процессор томонидан қўшимча вақт ва хотира ресурсларини талаб қилади. Шу сабабли, агар чақириладиган функция ҳажми унчалик катта бўлмаган ҳолларда, компиляторга функцияни чақириш коди ўрнига функция танасини ўзини жойлаштиришга кўрсатма бериш мумкин. Бу иш функция прототипини inline калит сўзи билан эълон қилиш орқали амалга оширилади. Натижада ҳажми ошган, лекин нисбатан тез бажариладиган программа коди юзага келади.

Функция коди жойлаштириладиган программага мисол.

```
#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;
}
```

Келтирилган программа кодини ҳосил қилишда Summa() функцияси чақирилган жойларга унинг танасидаги буйруқлар жойлаштирилади.

Рекурсив функциялар

Юқорида қайд қилингандек *рекурсия* деб функция танасида шу функциянинг ўзини чақиришига айтилади. Рекурсия икки хил бўлади:

1) *оддий* - агар функция ўз танасида ўзини чақирса;

2) *воситали* - агар биринчи функция иккинчи функцияни чақирса, иккинчиси эса ўз навбатида биринчи функцияни чақирса.

Одатда рекурсия математикада кенг қўлланилади. Чунки аксарият математик формулалар рекурсив аниқланади. Мисол тариқасида факториални ҳисоблаш формуласини

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

ва соннинг бутун даражасини ҳисоблашни кўришимиз мумкин:

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

Кўриниб турибдики, навбатдаги қийматни ҳисоблаш учун функциянинг «олдинги қиймати» маълум бўлиши керак. C++ тилида рекурсия математикадаги рекурсияга ўхшаш. Буни юқоридаги мисоллар учун тузилган функцияларда кўриш мумкин. Факториал учун:

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

Берилган ҳақиқий x сонинг n- даражасини ҳисоблаш функцияси:

```
double Butun_Daraja(double x, int n)
{
 if(!n) return 1;
```

```
 else return x*Butun_Daraja(x,n-1);
}
```

Агар факториал функциясига n>0 қиймат берилса, қуйидаги ҳолат рўй беради: шарт операторининг else шохидаги қиймати (n қиймати) стекда эслаб қолинади. Ҳозирча қиймати номаълум n-1 факториални ҳисоблаш учун шу функциянинг ўзи n-1 қиймати билан билан чақирилади. Ўз навбатида, бу қиймат ҳам эслаб қолинади (стекка жойланади) ва яна функция чақирилади ва ҳакоза. Функция n=0 қиймат билан чақирилганда if операторининг шарти (!n) рост бўлади ва «return 1;» амали бажарилиб, айни шу чақириш бўйича 1 қиймати қайтарилади. Шундан кейин «тескари» жараён бошланади - стекда сақланган қийматлар кетма-кет олинади ва кўпайтирилади: охирги қиймат - аниқлангандан кейин (1), у ундан олдинги сақланган қийматга 1 қийматига кўпайтириб F(1) қиймати ҳисобланади, бу қиймат 2 қийматига кўпайтириш билан F(2) топилади ва ҳакоза. Жараён F(n) қийматини ҳисоблашгача «кўтарилиб» боради. Бу жараённи, n=4 учун факториал ҳисоблаш схемасини 5.2-расмда кўриш мумкин:

↓	F(4)=4*F(3)	↓	F(4)=4*F(3)	↓	F(4)=4*F(3)	↓	F(4)=4*F(3)	↑	F(4)=4*6
↓	F(3)=3*F(2)	↓	F(3)=3*F(2)	↓	F(3)=3*F(2)	↑	F(3)=3*2		
↓	F(2)=2*F(1)	↓	F(2)=2*F(1)	↑	F(2)=2*1				
↓	F(1)=1*F(0)	↑	F(1)=1*1						
↑	F(0)=1								

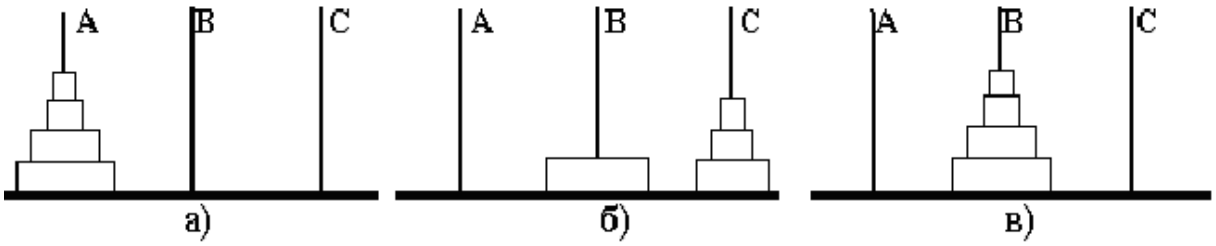
5.2-расм. 4! ҳисоблаш схемаси

Рекурсив функцияларни тўғри амал қилиши учун рекурсив чақиришларнинг тўхташ шарти бўлиши керак. Акс ҳолда рекурсия тўхтамаслиги ва ўз навбатида функция иши тугамаслиги мумкин. Факториал ҳисоблашида рекурсив тушишларнинг тўхташ шарти функция параметри n=0 бўлишидир (шарт операторининг рост шохи).

Ҳар бир рекурсив мурожаат қўшимча хотира талаб қилади - функцияларнинг локал объектлари (ўзгарувчилари) учун ҳар бир мурожаатда стекдан янгидан жой ажратилади. Масалан, рекурсив функцияга 100 марта мурожаат бўлса, жами 100 локал объектларнинг мажмуаси учун жой ажратилади. Айрим ҳолларда, яъни рекурсиялар сони етарлича катта бўлганда, стек ўлчами чекланганлиги сабабли (реал режимда 64Кб ўлчамгача) у тўлиб кетиши мумкин. Бу ҳолатда программа ўз ишини «Стек тўлиб кетди» хабари билан тўхтади.

Қуйида, рекурсия билан самарали ечиладиган «Ханой минораси» масаласини кўрайлик.

Масала. Учта А, В, С қозиқ ва n-та ҳар хил ўлчамли халқалар мавжуд. Халқаларни ўлчамлари ўсиш тартибида 1 дан n гача тартибланган. Бошда барча халқалар А қозиққа 5.3а -расмдагидек жойлаштирилган. А қозиқдаги барча халқаларни В қозиққа, ёрдамчи С қозиқдан фойдаланган ҳолда, қуйидаги қоидаларга амал қилган ҳолда ўтказиш талаб этилади: халқаларни биттадан кўчириш керак ва катта ўлчамли халқани кичик ўлчамли халқа устига қўйиш мумкин эмас.



5.3-расм. Ханой минораси масаласини ечиш жараёни

Амаллар кетма-кетлигини чоп этадиган («Халқа q дан r га ўтказилсин» кўринишида, бунда q ва r - 5.3-расмдаги А,В ёки С халқалар). Берилган n та халқа учун масала ечилсин.

Кўрсатма: халқаларни А дан В га тўғри ўтказишда 5.3б –расмлардаги ҳолат юзага келади, яъни n халқани А дан В ўтказиш масаласи n-1 халқани А дан С га ўтказиш, ҳамда битта халқани А дан В ўтказиш масаласига келади. Ундан кейин С қозиқдаги n-1 халқали А қозиқ ёрдамида В қозиққа ўтказиш масаласи юзага келади ва ҳакоза.

```
#include <iostream.h>
void Hanoy(int n,char a='A',char b='B',char c='C')
{
 if(n)
 {
  Hanoy(n-1,a,c,b);
  cout<<"Xalqa"<< a<<" dan "<<b<<" ga o'tkazilsin\n";
  Hanoy(n-1,c,b,a);
  }
}
int main()
{unsigned int Xalqalar_Soni;
 cout<<"Hanoy minorasi masalasi"<<endl;
 cout<<"Xalqalar sonini kiriting: ";
 cin>>Xalqalar_Soni;
 Hanoy(Xalqalar_Soni);
 return 0;
}
```

Халқалар сони 3 бўлганда (Xalqalar_Soni=3) программа экранга халқаларни кўчириш бўйича амаллар кетма-кетлигини чоп этади:

```
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
```

Рекурсия чиройли, ихчам кўрингани билан хотирани тежаш ва ҳисоблаш вақтини қисқартириш нуқтаи-назаридан уни имкон қадар итератив ҳисоблаш билан алмаштирилгани маъқул. Масалан, x ҳақиқий сонининг n-даражасини ҳисоблашнинг қуйидаги ечим варианти нисбатан кам ресурс талаб қилади (n- бутун ишорасиз сон):

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

Иккинчи томондан, шундай масалалар борки, уларни ечишда рекурсия жуда самарали, ҳаттоки ягона усулдир. Хусусан, грамматик таҳлил масалаларида рекурсия жуда ҳам ўнғай ҳисобланди.

Қайта юкланувчи функциялар

Айрим алгоритмлар берилганларнинг ҳар хил турдаги қийматлари учун қўлланиши мумкин. Масалан, иккита соннинг максимумини топиш алгоритмида бу сонлар бутун ёки ҳақиқий турда бўлиши мумкин. Бундай ҳолларда бу алгоритмлар амалга оширилган функциялар номлари бир хил бўлгани маъқул. Бир нечта функцияни бир хил номлаш, лекин ҳар хил турдаги параметрлар билан ишлатиш *функцияни қайта юклаш* дейилади.

Компилятор параметрлар турига ва сонига қараб мос функцияни чақиради. Бундай амални «*ҳал қилиш амали*» дейилади ва унинг мақсади параметрларга кўра айнан (нисбатан) тўғри келадиган функцияни чақиришдир. Агар бундай функция топилмаса компилятор хатолик ҳақида хабар беради. Функцияни аниқлашда функция қайтарувчи қиймат турининг аҳамияти йўқ. Мисол:

```
#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;);
}
```

Агар функция чақирилишида аргумент тури унинг прототипидаги худди шу ўриндаги параметр турига мос келмаса, компилятор уни параметр турига келтирилишга ҳаракат қилади - bool ва char турларини int турига, float турини double турига ва int турини double турига ўтказишга.

Қайта юкланувчи функциялардан фойдаланишда қуйидаги қоидаларга риоя қилиш керак:

-қайта юкланувчи функциялар битта кўриниш соҳасида бўлиши керак;

-қайта юкланувчи функцияларда келишув бўйича параметрлар ишлатилса, бундай параметрлар барча қайта юкланувчи функцияларда ҳам ишлатилиши ва улар бир хил қийматга эга бўлиш керак;

- агар функциялар параметрларининг тури фақат «const» ва ‘&’ белгилари билан фарқ қиладиган бўлса, бу функциялар қайта юкланмайди.

6-боб. Кўрсаткичлар ва адрес олувчи ўзгарувчилар

Кўрсаткичлар

Программа матнида ўзгарувчи эълон қилинганда, компилятор ўзгарувчига хотирадан жой ажратади. Бошқача айтганда, программа коди хотирага юкланганда берилганлар учун, улар жойлашадиган сегментнинг бошига нисбатан силжишини, яъни нисбий адресини аниқлайди ва объект код ҳосил қилишда ўзгарувчи учраган жойга унинг адресини жойлаштиради.

Умуман олганда, программадаги ўзгармаслар, ўзгарувчилар, функциялар ва синф объектлар адресларини хотиранинг алоҳида жойида сақлаш ва улар устидан амаллар бажариш мумкин. Қийматлари адрес бўлган ўзгарувчиларга *кўрсаткич ўзгарувчилар* дейилади.

Кўрсаткич уч хил турда бўлиши мумкин:

- бирорта объектга, хусусан ўзгарувчига кўрсаткич;
- функцияга кўрсаткич;
- void кўрсаткич.

Кўрсаткичнинг бу хусусиятлари унинг қабул қилиши мумкин бўлган қийматларида фарқланади.

Кўрсаткич албатта бирорта турга боғланган бўлиши керак, яъни у кўрсатган адресда қандайдир қиймат жойланиши мумкин ва бу қийматнинг хотирада қанча жой эгаллаши олдиндан маълум бўлиши шарт.

Функцияга кўрсаткич. Функцияга кўрсаткич программа жойлашган хотирадаги функция кодининг бошланғич адресини кўрсатади, яъни функция чақирилганда бошқарув айни шу адресга узатилади. Кўрсаткич орқали функцияни оддий ёки воситали чақириш амалга ошириш мумкин. Бунда функция унинг номи бўйича эмас, балки функцияга кўрсатувчи ўзгарувчи орқали чақирилади. Функцияни бошқа функцияга аргумент сифатида узатиш ҳам функция кўрсаткичи орқали бажарилади. Функцияга кўрсаткичнинг ёзилиш синтаксиси қуйидагича:

<тур> (* <ном>) (<параметрлар рўйхати>);

Бунда <тур>- функция қайтарувчи қиймат тури; *<ном> - кўрсаткич ўзгарувчининг номи; <параметрлар рўйхати> - функция параметрларининг ёки уларнинг турларининг рўйхати.

Масалан:

```
int (*fun)(float,float);
```

Бу ерда бутун сон турида қиймат қайтарадиган fun номидаги функцияга кўрсаткич эълон қилинган ва у иккита ҳақиқий турдаги параметрларга эга.

Масала. Берилган бутун n=100 ва a,b - ҳақиқий сонлар учун $f_1(x)=5\sin(3x)+x$, $f_2(x)=\cos(x)$ ва $f_3(x)=x^2+1$ функциялар учун $\int_a^b f(x)dx$ интегралини тўғри тўртбурчаклар формуласи билан тақрибан ҳисоблансин:

$$\int_a^b f(x)dx \approx h[f(x_1)+f(x_2)+...+f(x_n)],$$

бу ерда $h=\frac{b-a}{n}$, $x_i=a+ih-h/2, i=1..n$.

Программа бош функция, интеграл ҳисоблаш ва иккита математик функциялар - $f_1(x)$ ва $f_3(x)$ учун аниқланган функциялардан ташкил топади, $f_2(x)=\cos(x)$ функциянинг адреси «math.h» сарлавҳа файлидан олинади. Интеграл ҳисоблаш функциясига кўрсаткич орқали интеграли ҳисобланадиган функция адреси, a ва b - интеграл чегаралари қийматлари узатилади. Оралиқни бўлишлар сони - n глобал ўзгармас қилиб эълон қилинади.

```
#include <iostream.h>
#include <math.h>
const int n=100;
double f1(double x){return 5*sin(3*x)+x;}
double f3(double x){return x*x+1;}
double Integral(double(*f)(double),double a,double b)
{
 double x,s=0;
 double h=(b-a)/n;
 x=a-h/2;
 for(int i=1;i<=n; i++) s+=f(x+=h);
 s*=h;
 return s;
}
int main()
{
 double a,b;
 int menu;
 while(1)
 {
  cout<<"\nIsh regimini tanlang:\n";
  cout<<"1:f1(x)=5*sin(3*x)+x integralini\
  hisoblash\n";
  cout<<"2:f2(x)=cos(x) integralini hisoblash\n";
```

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

Программанинг иши чексиз такрорлаш оператори танасини бажаришдан иборат. Такрорлаш танасида фойдаланувчига иш режимини танлаш бўйича меню таклиф қилинади:

```
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->
```

Фойдаланувчи 0 ва 3 оралиғидаги бутун сонни киритиши керак. Агар киритилган сон (menu ўзгарувчи қиймати) 0 бўлса, break оператори ёрдамида такрорлашдан, кейин программадан чиқилади. Агар menu қиймати 1 ва 3 оралиғида бўлса, интегралнинг қуйи ва юқори чегараларини киритиш сўралади, ҳамда Integral() функцияси мос функция адреси билан чақирилади ва натижа чоп этилади. Шунга эътибор бериш керакки, интеграл чегараларининг қийматларини тўғри киритилишига фойдаланувчи жавобгар.

Объектга кўрсаткич. Бирор объектга кўрсаткич (шу жумладан ўзгарувчига). Бундай кўрсаткичда маълум турдаги (таянч ёки ҳосилавий турдаги) берилганларнинг хотирадаги адреси жойлашади. Объектга кўрсаткич қуйидагича эълон қилинади:

<тур> *<ном>;

Бу ерда <тур> - кўрсаткич аниқлайдиган адресдаги қийматнинг тури, <ном> - объект номи (идентификатор). Агар бир турда бир нечта кўрсаткичлар эълон қилинадиган бўлса, ҳар бир кўрсаткич учун ‘*’ белгиси қўйилиши шарт:

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

Келтирилган мисолда i ва k - бутун турдаги кўрсаткичлар ва j - бутун турдаги ўзгарувчи, иккинчи операторда x - ҳақиқий ўзгарувчи ва y,z - ҳақиқий турдаги кўрсаткичлар эълон қилинган.

void кўрсаткич. Бу кўрсаткич объект тури олдиндан номаълум бўлганда ишлатилади. void кўрсаткичининг муҳим афзалликларидан бири - унга ҳар қандай турдаги кўрсаткич қийматини юклаш мумкинлигидир. void кўрсаткич адресидаги қийматни ишлатишдан олдин, уни аниқ бир турга ошкор равишда келтириш керак бўлади. void кўрсаткични эълон қилиш куйидагича бўлади:

void *<ном>;

Кўрсаткичнинг ўзи ўзгармас ёки ўзгарувчан бўлиши ва ўзгармас ёки ўзгарувчилар адресига кўрсатиши мумкин, масалан:

```
int i; // бутун ўзгарувчи
const int ci=1; // бутун ўзгармас
int * pi; // бутун ўзгарувчига кўрсаткич
const int *pci; // бутун ўзгармасга кўрсаткич
int *const cp=&i;//бутун ўзгарувчига ўзгармас
                 //кўрсаткич
const int*const cpc=&ci; // бутун ўзгармасга ўзгармас
                         // кўрсаткич
```

Мисоллардан кўриниб турибдики, ‘*’ ва кўрсаткич номи орасида турган const модификатори фақат кўрсаткичнинг ўзига тегишли ҳисобланади ва уни ўзгартириш мумкин эмаслигини билдиради, ‘*’ белгисидан чапда турган const эса кўрсатилган адресдаги қиймат ўзгармас эканлигини билдиради.

Кўрсаткичга қийматни бериш учун ‘&’ - адресни олиш амали ишлатилади.

Кўрсаткич ўзгарувчиларининг амал қилиш соҳаси, яшаш даври ва кўриниш соҳаси умумий қоидаларга бўйсунади.

Кўрсаткичга бошланғич қиймат бериш

Кўрсаткичлар кўпинча динамик хотира (бошқача номи «уюм» ёки «heap») билан боғлиқ ҳолда ишлатилади. Хотиранинг динамик

дейилишига сабаб, бу соҳадаги бўш хотира программа ишлаш жараёнида, керакли пайтида ажратиб олинади ва зарурат қолмаганида қайтарилади (бўшатилади). Кейинчалик, бу хотира бўлаги программа томонидан бошқа мақсадда яна ишлатилиши мумкин. Динамик хотирага фақат кўрсаткичлар ёрдамида мурожаат қилиш мумкин. Бундай ўзгарувчилар *динамик ўзгарувчилар* дейилади ва уларни яшаш вақти яратилган нуқтадан бошлаб программа охиригача ёки ошкор равишда йўқотилган (боғланган хотира бўшатилган) жойгача бўлади.

Кўрсаткичларни эълон қилишда унга бошланғич қийматлар бериш мумкин. Бошланғич қиймат (инициализатор) кўрсаткич номидан сўнг ёки қавс ичида ёки ‘=’ белгидан кейин берилади. Бошланғич қийматлар қуйидаги усуллар билан берилиши мумкин:

I. Кўрсаткичга мавжуд бўлган объектнинг адресини бериш:

а) адресни олиш амал орқали:

```
int i=5,k=4; // бутун ўзгарувчилар
int *p=&i;   // p кўрсаткичга i ўзгарувчининг
             // адреси ёзилади
int *p1(&k); // p1 кўрсаткичга k ўзгарувчининг
             // адреси ёзилади
```

б) бошқа, инициализацияланган кўрсаткич қийматини бериш:

```
int * r=p; // p олдин эълон қилинган ва қийматга эга
           // бўлган кўрсаткич
```

в) массив ёки функция номини бериш:

```
int b[10];  // массивни эълон қилиш
int *t=b;  // массивнинг бошланғич адресини бериш
void f(int a){/* … */} // функцияни аниқлаш
void (*pf)(int);// функцияга кўрсаткични эълон қилиш
pf=f; // функция адресини кўрсаткичга бериш
```

II. Ошкор равишда хотиранинг абсолют адресини бериш:

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

Бунда 0xB8000000 - ўн олтилик ўзгармас сон ва (char*) - турга келтириш амали бўлиб, у vp ўзгарувчисини хотиранинг абсолют адресидаги байтларни char сифатида қайта ишловчи кўрсаткич турига айлантирилишини англатади.

III. Бўш қиймат бериш:

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

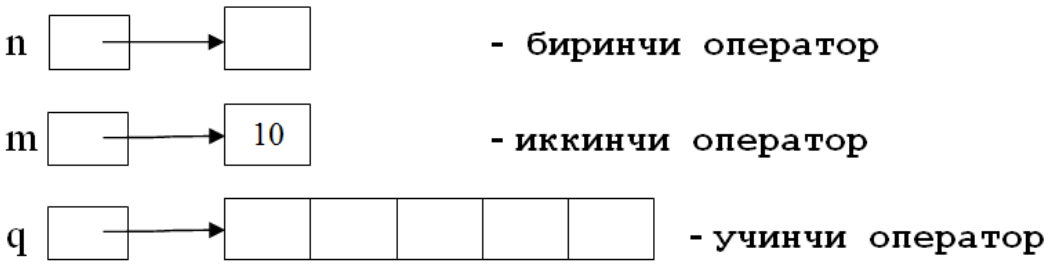
Биринчи сатрда махсус NULL ўзгармаси ишлатилган, иккинчи сатрда 0 қиймат ишлатилган. Иккала ҳолда ҳам кўрсаткич ҳеч қандай объектга мурожаат қилмайди. Бўш кўрсаткич асосан кўрсаткични

аниқ бир объектга кўрсатаётган ёки йўқлигини аниқлаш учун ишлатилади.

IV. Динамик хотирада new амали билан жой ажратиш ва уни адресини кўрсаткичга бериш:

```
int * n=new int;        // биринчи оператор
int * m=new int(10);    // иккинчи оператор
int * q=new int[5];     // учинчи оператор
```

Биринчи операторда new амали ёрдамида динамик хотирада int учун етарли жой ажратиб олиниб, унинг адреси n кўрсаткичга юкланади. Кўрсаткичнинг ўзи учун жой компиляция вақтида ажратилади.



6.1-расм. Динамик хотирадан жой ажратиш

Иккинчи операторда жой ажратишдан ташқари m адресига бошланғич қиймат - 10 сонини жойлаштиради.

Учинчи операторда int туридаги 5 элемент учун жой ажратилган ва унинг бошланғич адреси q кўрсаткичга берилаяпти.

Хотира new амали билан ажратилган бўлса, у delete амали билан бўшатилиши керак. Юқоридаги динамик ўзгарувчилар билан боғланган хотира қуйидагича бўшатилади:

```
delete n; delete m; delete[]q;
```

Агарда хотира new[] амали билан ажратилган бўлса, уни бўшатиш учун delete [] амалини ўлчови кўрсатилмаган ҳолда қўллаш керак.

Хотира бўшатилганлигига қарамасдан кўрсаткични ўзини кейинчалик қайта ишлатиш мумкин.

Кўрсаткич устида амаллар

Кўрсаткич устида қуйидаги амаллар бажарилиши мумкин:

1) объектга воситали мурожаат қилиш амали;
2) қиймат бериш амали;
3) кўрсаткичга ўзгармас қийматни қўшиш амали;
4) айириш амали;
5) инкремент ва декремент амаллари;

6) солиштириш амали;

7) турга келтириш амали.

Воситали мурожаат қилиш амали кўрсаткичдаги адрес бўйича жойлашган қийматни олиш ёки қиймат бериш учун ишлатилади:

```
char a;            // char туридаги ўзгарувчи эълони.
char *p=new char; // Кўрсаткични эълон қилиб,унга
                   // динамик хотирадан ажратилган
                   // хотиранинг адресини бериш
*p='b';  // p адресига қиймат жойлаштириш
a=*p; // a ўзгарувчисига p адресидаги қийматни бериш
```

Шуни қайд қилиб ўтиш керакки, хотиранинг аниқ бир жойидаги адресни бир пайтнинг ўзида бир нечта ва ҳар хил турдаги кўрсаткичларга бериш мумкин ва улар орқали мурожаат қилинганда берилганнинг ҳар хил турдаги қийматларини олиш мумкин:

```
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;
```

Экранга ҳар хил қийматлар чоп этилади:

```
cc77ffaa ffaa aa
```

Ўзгарувчилар битта адресда жойлашган ҳолда яхлит қийматнинг турли бўлакларини ўзлаштиради. Бунда, бир байтдан катта жой эгаллаган сон қийматининг хотирада «тескари» жойлашиши инобатга олиниши керак.

Агар ҳар хил турдаги кўрсаткичларга қийматлар берилса, албатта турга келтириш амалидан фойдаланиш керак:

```
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;
```

Кўрсаткич турини void турига келтириш амалда маънога эга эмас. Худди шундай, турлари бир хил бўлган кўрсаткичлар учун турни келтириш амалини бажаришга ҳожат йўқ.

Кўрсаткич устидан бажариладиган арифметик амалларда автоматик равишда турларнинг ўлчами ҳисобга олинади.

Арифметик амаллар фақат бир хил турдаги кўрсаткичлар устидан бажарилади ва улар асосан, массив тузилмаларига кўрсаткичлар устида бажарилади.

Инкремент амали кўрсаткични массивнинг кейинги элементига, декремент эса аксинча, битта олдинги элементининг адресига кўчиради. Бунда кўрсаткичнинг қиймати sizeof(<массив элементи-нинг тури>) қийматига ўзгаради. Агар кўрсаткич k ўзгармас қийматга оширилса ёки камайтирилса, унинг қиймати k*sizeof(<массив элементининг тури>) катталикка ўзгаради.

Масалан:

```
short int *p=new short[5];
long * q=new long [5];
p++;      // p қиймати 2 ошади
q++;      // q қиймати 4 га ошади
q+=3;     // q қиймати 3*4=12 ошади
```

Кўрсаткичларнинг айирмаси деб, улар айирмасининг тур ўлчамига бўлинишига айтилади. Кўрсаткичларни ўзаро қўшиш мумкин эмас.

Адресни олиш амали

Адресни олиш қуйидагича эълон қилинади:

<тур> & <ном>;

Бу ерда <тур> - адреси олинадиган қийматнинг тури, <ном>- адрес олувчи ўзгарувчи номи. Ўртадаги ‘&’ белгисига *адресни олиш амали* дейилади.

Бу кўринишда эълон қилинган ўзгарувчи шу турдаги ўзгарувчининг синоними деб қаралади. Адресни олиш амали орқали битта ўзгарувчига ҳар хил ном билан мурожаат қилиш мумкин бўлади.

Мисол:

```
int kol;
int & pal=kol;     // pal мурожаати, у kol
                   // ўзгарувчисининг альтернатив номи
const char & cr='\n'; // cr - ўзгармасга мурожаат
```

Адресни олиш амалини ишлатишда қуйидаги қоидаларга риоя қилиш керак: адрес олувчи ўзгарувчи функция параметри сифатида ишлатилган ёки extern билан тавсифланган ёки синф майдонига мурожаат қилингандан ҳолатлардан ташқари барча ҳолатларда бошланғич қийматга эга бўлиши керак.

Адресни олиш амали асосан функцияларда адрес орқали узатилувчи параметрлар сифатида ишлатилади.

Адрес олувчи ўзгарувчининг кўрсаткичдан фарқи шундаки, у алоҳида хотирани эгалламайди ва фақат ўз қиймати бўлган ўзгарувчининг бошқа номи сифатида ишлатилади.

Кўрсаткичлар ва адрес олувчи ўзгарувчилар функция параметри сифатида

Функция прототипида ёки аниқланиш сарлавҳасида кўрсатилган параметрлар *формал параметрлар* дейилади, функция чақиришида кўрсатилган аргументларга *фактик параметрлар* дейилади.

Функция чақирилишида фактик параметрнинг тури мос ўриндаги формал параметр турига тўғри келмаса ёки шу турга келтиришнинг иложи бўлмаса компиляция хатоси рўй беради.

Фактик параметрларни функцияга икки хил усул билан узатиш мумкин: *қиймати* ёки *адреси* билан.

Функция чақирилишида аргумент қиймат билан узатилганда, аргумент ёки унинг ўрнидаги келган ифода қиймати ва бошқа аргументларнинг нусхаси (қийматлари) стек хотирасига ёзилади. Функция фақат шу нусхалар билан амал қилади, керак бўлса бу нусхаларга ўзгартиришлар қилиниши мумкин, лекин бу ўзгаришлар аргументнинг ўзига таъсир қилмайди, чунки функция ўз ишини тугатиши билан нусхалар ўчирилади (стек тозаланади).

Агар параметр адрес билан узатилса, стекка адрес нусхаси ёзилади ва худди шу адрес бўйича қийматлар ўқилади (ёзилади). Функция ўз ишини тугатгандан кейин шу адрес бўйича қилинган ўзгаришлар сақланиб қолинади ва бу қийматларни бошқа функциялар ишлатиши мумкин.

Аргумент қиймат билан узатилиши учун мос формал параметр сифатида ўзгарувчини тури ва номи ёзилади. Функция чақирилишида мос аргумент сифатида ўзгарувчининг номи ёки ифода бўлиши мумкин.

Фактик параметр адрес билан узатилганда унга мос келувчи формал параметрни икки хил усул билан ёзиш мумкин: *кўрсаткич орқали* ёки *адресни олувчи параметрлар орқали*. Кўрсаткич орқали ёзилганда формал параметр туридан кейин ‘*’ белгиси ёзилади, мос аргументда эса ўзгарувчининг адреси (& амал орқали) ёки массив номи, ёки функция номи бўлиши мумкин. Адресни олиш амали орқали параметр узатишда формал параметрда туридан кейин ‘&’ белгиси ёзилади ва функция чақирилишида мос аргумент сифатида ўзгарувчи номи келади.

Мисол:

```
#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;
}
```

Программа ишлаши натижасида экранга қуйидаги қийматлар чоп қилинади:

```
1 2 3
1 6 8
```

Бу мисолда биринчи параметр і қиймат билан узатилади (“int i"). Унинг қиймати функция ичида ўзгаради, лекин ташқаридаги і ўзгарувчисининг қиймати ўзгармайди. Иккинчи параметрни кўрсаткич орқали адреси билан узатилиши талаб қилинади (“int *j”), адресни узатиш учун ‘&’- адресни олиш амали ишлатилган (“&j”). Функция танасида аргумент адресидан қиймат олиш учун ‘*’- қиймат олиш амали қўлланилган. Учинчи параметрда мурожаат орқали (“&k”) аргументнинг адреси узатиш кўзда тутилган. Бу ҳолда функция чақирилишида мос аргумент ўрнида ўзгарувчи номи туради, функция ичида эса қиймат олиш амалини ишлатишнинг ҳожати йўқ. Функция ишлаш натижасидаги қийматларни аргументлар рўйхати орқали олиш қулай ва тушунарли усул ҳисобланади.

Агар функция ичида адрес билан узатиладиган параметр қиймати ўзгармасдан қолиши зарур бўлса, бу параметр const модификатор билан ёзилиши керак:

```
fun(int n,const char*str);
```

Агарда функцияни чақиришда аргументлар фақат номлари билан берилган бўлса, келишув бўйича массивлар ва функциялар адреси билан, қолган турдаги параметрлар қийматлари билан узатилган деб ҳисобланади.

Мисол тариқасида дискриминантни ҳисоблаш усули ёрдамида $ax^2+bx+c=0$ кўринишидаги квадрат тенглама илдизларини функция параметрлари воситасида олиш масаласини кўрайлик.

```
#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;
}
```

Программадаги Kvadrat_Ildiz() функцияси квадрат тенглама илдизини ҳисоблайди. Унинг қайтарадиган қиймати тенгламанинг нечта илдизи борлигини англатади. Агар тенгламанинг ҳақиқий

илдизи мавжуд бўлмаса (D<0), функция 0 қийматини қайтаради. Агар D=0 бўлса, функция 1 қийматини қайтаради. Агар D>0 бўлса функция 2 қийматини қайтаради. Мавжуд илдизлар - x1 ва x2 адрес олувчи параметрларда қайтарилади.

Ўзгарувчан параметрли функциялар

C++ тилида параметрлар сони номаълум бўлган функцияларни ҳам ишлатиш мумкин. Бундан ташқари уларнинг турлари ҳам номаълум бўлиши мумкин. Параметрлар сони ва тури функцияни чақиришдаги аргументлар сони ва уларнинг турига қараб аниқланади. Бундай функциялар сарлавҳаси қуйидаги форматда ёзилади:

<функция тури> <функция номи> (<ошкор параметрлар рўйхати>, …)

Бу ерда <ошкор параметрлар рўйхати> - ошкор равишда ёзилган параметрлар номи ва тури. Бу параметрлар *мажбурий параметрлар* дейилади. Бундай параметрлардан камида биттаси бўлиши шарт. Қолган параметрлар сони ва тури номаълум ҳисобланади. Уларни аниқлаш ва ишлатиш тўла равишда программа тузувчи зиммасига юкланади.

Ўзгарувчан сондаги параметрларни ташкил қилиш усули умуман олганда иккита:

1-усул. Параметрлар рўйхати охирида яна бир махсус параметр ёзилади ва унинг қиймати параметрлар тугаганлигини билдиради. Компилятор томонидан функция танасида параметрлар бирма-бир аниқлаштирилади. Барча параметрлар тури охирги махсус параметр тури билан устма-уст тушади деб ҳисоб-ланади;

2-усул. Бирорта махсус параметр сифатида номаълум параметрлар сони киритилади ва унга қараб параметрлар сони аниқланади.

Иккала усулда ҳам параметрларга мурожаат қилиш учун кўрсаткичлар ишлатилади. Мисоллар келтирамиз.

1 - усул:

```
#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);
}
```

Натижа:

```
24
480
```

2 - усул:

```
#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;
}
```

Натижа:

```
Yigindi(2,6,4)=10
Yigindi(6,1,2,3,4,5,6)=21
```

Иккала мисолда ҳам номаълум параметрлар берилган махсус параметр турини қабул қилган. Ҳар хил турдаги параметрларни ишлатиш учун турни аниқлайдиган параметр киритиш керак:

```
#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;
    }
  }
}
```

Юқорида келтирилган мисолда номаълум параметрларни турини аниқлаш масаласи компилятор томонидан эмас, балки программа тузувчиси томонидан ҳал қилинган.

7-боб. Массивлар

Берилганлар массиви тушунчаси

Хотирада кетма-кет (регуляр) жойлашган бир хил турдаги қийматларга *массив* дейилади.

Одатда массивларга зарурат, катта ҳажмдаги, лекин чекланган миқдордаги ва тартибланган қийматларни қайта ишлаш билан боғлиқ масалаларни ечишда юзага келади. Фараз қилайлик, талабалар гуруҳининг рейтинг баллари билан ишлаш масаласи қўйилган. Унда гуруҳнинг ўртача рейтингини аниқлаш, рейтингларни камайиши бўйича тартиблаш, конкрет талабанинг рейтинги ҳақида маълумот бериш ва бошқа масала остиларини ечиш зарур бўлсин. Қайд этилган масалаларни ечиш учун берилганларнинг (рейтингларнинг) тартибланган кетма-кетлиги зарур бўлади. Бу ерда тартибланганлик маъноси шундаки, кетма-кетликнинг ҳар бир қиймати ўз ўрнига эга бўлади (биринчи талабанинг рейтинги массивда биринчи ўринда, иккинчи талабаники - иккинчи ўринда ва ҳакоза). Берилганлар кетма-кетлигини икки хил усулда ҳосил қилиш мумкин. Биринчи йўл - ҳар бир рейтинг учун алоҳида ўзгарувчи аниқлаш: Reyting1,…,ReytingN. Лекин, гуруҳдаги талабалар сони етарлича катта бўлганда, бу ўзгарувчилар қатнашган программани тузиш катта қийинчиликларни юзага келтиради. Иккинчи йўл - берилганлар кетма-кетлигини ягона ном билан аниқлаб, унинг қийматларига мурожаатни, шу қийматларнинг кетма-кетликда жойлашган ўрнининг номери (индекси) орқали амалга оширишдир. Рейтинглар кетма-кетлигини Reyting деб номлаб, ундаги қийматларига $Reyting_1$,…,$Reyting_N$ кўринишида мурожаат қилиш мумкин. Одатда берилганларнинг бундай кўринишига массивлар дейилади. Массивларни математикадаги сонлар векторига ўхшатиш мумкин, чунки вектор ҳам ўзининг индивидуал номига эга ва у фиксирланган миқдордаги бир турдаги қийматлардан - сонлардан иборатдир.

Демак, массив - бу фиксирланган миқдордаги айрим қийматларнинг (массив элементларининг) тартибланган мажмуасидир. Барча элементлар бир хил турда бўлиши керак ва бу тур *элемент тури* ёки массив учун *таянч тур* деб номланади. Юқоридаги келтирилган мисолда Reyting - ҳақиқий турдаги *вектор* деб номланади.

Программада ишлатиладиган ҳар бир конкрет массив ўзининг индивидуал номига эга бўлиши керак. Бу номни *тўлиқ ўзгарувчи* дейилади, чунки унинг қиймати массивнинг ўзи бўлади. Массивнинг ҳар бир элементи массив номи, ҳамда квадрат қавсга олинган ва

элемент селектори деб номланувчи индексни кўрсатиш орқали ошкор равишда белгиланади. Мурожаат синтаксиси:

<массив номи >[<индекс>]

Бу кўринишга *хусусий ўзгарувчи* дейилади, чунки унинг қиймати массивнинг алоҳида элементидир. Бизнинг мисолда Reyting массивининг алоҳида компоненталарига Reyting[1],…,Reyting[N] хусусий ўзгарувчилар орқали мурожаат қилиш мумкин. Бошқача бу ўзгарувчилар *индексли ўзгарувчилар* дейилади.

Массив индекси сифатида бутун сон қўлланилади. Умуман олганда индекс сифатида бутун сон қийматини қабул қиладиган ихтиёрий ифода ишлатилиши мумкин ва унинг қиймати массив элементи номерини аниқлайди. Ифода сифатида ўзгарувчи ҳам олиниши мумкинки, ўзгарувчининг қиймати ўзгариши билан мурожаат қилинаётган массив элементини аниқловчи индекс ҳам ўзгаради. Шундай қилиб, программадаги битта индексли ўзгарувчи орқали массивнинг барча элементларини белгилаш (аниқлаш) мумкин бўлади. Масалан, Reyting[I] ўзгарувчиси орқали I ўзгарувчининг қийматига боғлиқ равишда Reyting массивининг ихтиёрий элементига мурожаат қилиш мавжуд.

Ҳақиқий турдаги (float, double) қийматлар тўплами чексиз бўлганлиги сабабли улар индекс сифатида ишлатилмайди.

C++ тилида индекс доимо 0 дан бошланади ва унинг энг катта қиймати массив эълонидаги узунликдан биттага кам бўлади.

Массив эълони қуйидагича бўлади:

<тур> <ном> [<узунлик>]={бошланғич қийматлар}.

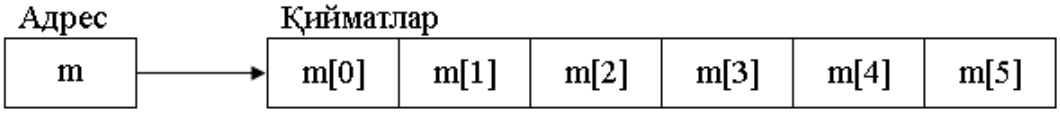
Бу ерда <узунлик> - ўзгармас ифода. Мисоллар:

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

Массив статик ва динамик бўлиши мумкин. Статик массивнинг узунлиги олдиндан маълум бўлиб, у хотирада маълум адресдан бошлаб кетма-кет жойлашади. Динамик массивни узунлиги программа бажарилиш жараёнида аниқланиб, у динамик хотирадаги айни пайтда бўш бўлган адресларга жойлашади. Масалан,

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

кўринишида эълон қилинган бир ўлчамли массив элементлари хотирада қуйидагича жойлашади:



7.1-расм. Бир ўлчамли массивнинг хотирадаги жойлашуви

Массивнинг i- элементига m[i] ёки *(m+i) - воситали мурожаат қилиш мумкин. Массив узунлигини sizeof(m) амали орқали аниқлади.

Массив эълонида унинг элементларига бошланғич қийматлар бериш мумкин ва унинг бир нечта вариантлари мавжуд.

1) ўлчами кўрсатилган массив элементларини тўлиқ инициализациялаш:

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

Бунда 5 та элементдан иборат бўлган t номли бутун турдаги бир ўлчамли массив эълон қилинган ва унинг барча элементларига бошланғич қийматлар берилган. Бу эълон қуйидаги эълон билан эквивалент:

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

2) ўлчами кўрсатилган массив элементларини тўлиқмас инициализациялаш:

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

Бу ерда фақат массив бошидаги учта элементга бошланғич қийматлар берилган. Шуни айтиб ўтиш керакки, массивнинг бошидаги ёки ўртасидаги элементларига қийматлар бермасдан, унинг охиридаги элементларга бошланғич қиймат бериш мумкин эмас. Агарда массив элементларига бошланғич қиймат берилмаса, унда келишув бўйича static ва extern модификатори билан эълон қилинган массив учун элементларининг қиймати 0 сонига тенг деб, automatic массивлар элементларининг бошланғич қийматлари номаълум ҳисобланади.

3) ўлчами кўрсатилмаган массив элементларини тўлиқ инициализациялаш:

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

Бу мисолда массивни барча элементларига қийматлар берилган ҳисобланади, массив узунлиги компилятор томонидан бошланғич қийматлар сонига қараб аниқланади. Агарда массив узунлиги берилмаса, бошланғич қиймати берилиши шарт.

Массивни эълон қилишга мисоллар:

```
char ch[4]={'a','b','c','d'}; //белгилар массиви
int in[6] ={10,20,30,40};     // бутун сонлар массиви
```

```
char str[]=”abcd”;
//сатр узунлиги 5 га тенг, чунки унинг охирига
// ’\0’ белгиси қўшилади
char str[]={’a’,’b’,’c’,’d’};
// юқоридаги сатрнинг бошқача ёзилиши
```

Масала. Бир ой ичидаги кундалик ҳароратлар берилган. Ой учун ўртача ҳароратни ҳисоблаш программаси тузилсин.

Программа матни:

```
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;
}
```

Кўп ўлчамли статик массивлар

C++ тилида массивлар элементининг турига чекловлар қўйилмайди, лекин бу турлар чекли ўлчамдаги объектларнинг тури бўлиши керак. Чунки компилятор массивнинг хотирадан қанча жой (байт) эгаллашини ҳисоблай олиши керак. Хусусан, массив компонентаси массив бўлиши мумкин («векторлар-вектори»), натижада *матрица* деб номланувчи икки ўлчамли массив ҳосил бўлади.

Агар матрицанинг элементи ҳам вектор бўлса, уч ўлчамли массивлар - *куб* ҳосил бўлади. Шу йўл билан ечилаётган масалага боғлиқ равишда ихтиёрий ўлчамдаги массивларни яратиш мумкин.

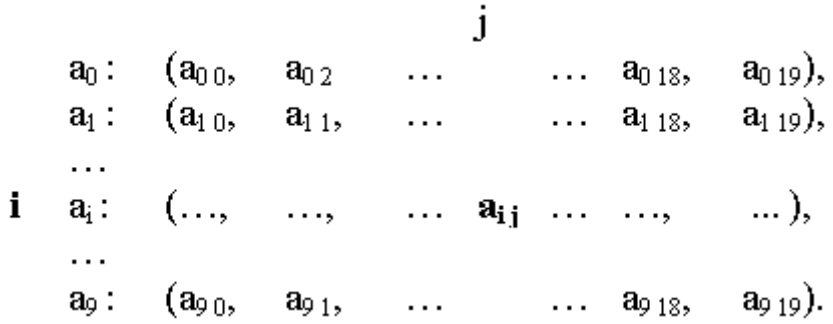
Икки ўлчамли массивнинг синтаксиси қуйидаги кўринишда бўлади:

<тур> <ном> [<узунлик >] [<узунлик>]

Масалан, 10×20 ўлчамли ҳақиқий сонлар массивининг эълони:

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

Эълон қилинган A матрицани кўриниши 7.2-расмда келтирилган.



7.2-расм. Икки ўлчамли массивнинг хотирадаги жойлашуви

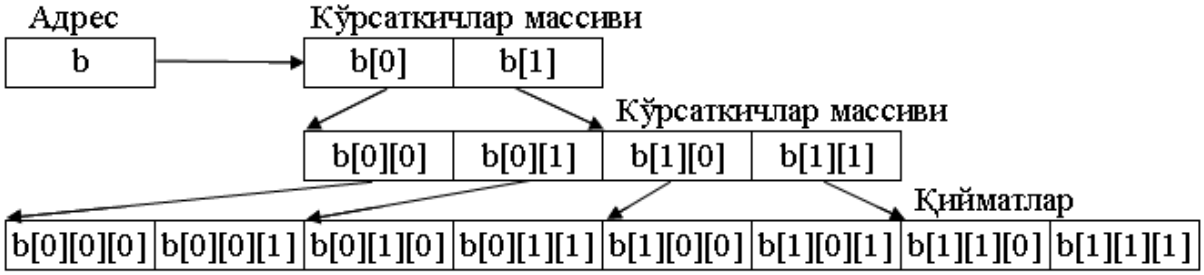
Энди адрес нуқтаи - назаридан кўп ўлчамли массив элементларига мурожаат қилишни кўрайлик. Қуйидаги эълонлар берилган бўлсин:

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

Биринчи эълонда икки ўлчамли массив, яъни 2 сатр ва 3 устундан иборат матрица эълон қилинган, иккинчисида уч ўлчамли - 3 та 2x2 матрицадан иборат бўлган массив эълон қилинган. Унинг элементларига мурожаат схемаси:

7.3-расм. Икки ўлчамли массив элементларига мурожаат

Бу ерда a[i] кўрсаткичда i-чи сатрнинг бошланғич адреси жойлашади, массив элементига a[i][j] кўринишидаги асосий мурожаатдан ташқари воситали мурожаат қилиш мумкин: *(*(a+i)+j) ёки *(a[i]+j).



7.3-расм. Уч ўлчамли массивнинг хотирада ташкил бўлиши

Массив элементларига мурожаат қилиш учун номдан кейин квадрат қавсда ҳар бир ўлчам учун индекс ёзилиши керак, масалан b[i][j][k]. Бу элементга воситали мурожаат ҳам қилиш мумкин ва унинг вариантлари:

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

Кўп ўлчамли массивларни инициализациялаш

Массивларни инициализациялаш қуйидаги мисолларда кўрсатилган:

```
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}};
```

Биринчи операторда бошланғич қийматлар кетма-кет ёзилган, иккинчи операторда қийматлар гуруҳлашган, учинчи операторда ҳам гуруҳлашган, лекин баъзи гуруҳларда охирги қийматлар берилмаган.

Мисол учун, матрицалар ва вектор кўпайтмасини - $C = A \times b$ ҳисоблаш масаласини кўрайлик. Бу ерда $A = \{a_{ij}\}$, $b = \{b_j\}$, $c = \{c_i\}$, $0 \le i < m, 0 \le j < n$. Ҳисоблаш формуласи - $c_i = \sum_{j=0}^{n-1} a_{ij} b_j$.

Мос програма матни:

```
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;
}
```

Динамик массивлар билан ишлаш

Статик массивларнинг камчиликлари шундаки, уларнинг ўлчамлари олдиндан маълум бўлиши керак, бундан ташқари бу ўлчамлар берилганларга ажратилган хотира сегментининг ўлчами билан чегараланган. Иккинчи томондан, етарлича катта ўлчамдаги массив эълон қилиб, конкрет масала ечилишида ажратилган хотира тўлиқ ишлатилмаслиги мумкин. Бу камчиликлар динамик массивлардан фойдаланиш орқали бартараф этилади, чунки улар программа

ишлаши жараёнида керак бўлган ўлчамдаги массивларни яратиш ва зарурат қолмаганда йўқотиш имкониятини беради.

Динамик массивларга хотира ажратиш учун malloc(), calloc() функцияларидан ёки new операторидан фойдаланиш мумкин. Динамик объектга ажратилган хотирани бўшатиш учун free() функцияси ёки delete оператори ишлатилади.

Юқорида қайд қилинган функциялар «alloc.h» кутубхонасида жойлашган.

malloc() функциясининг синтаксиси

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

кўринишида бўлиб, у хотиранинг уюм қисмидан size байт ўлчамидаги узлуксиз соҳани ажратади. Агар хотира ажратиш муваффақиятли бўлса, malloc() функцияси ажратилган соҳанинг бошланиш адресини қайтаради. Талаб қилинган хотирани ажратиш муваффақиятсиз бўлса, функция NULL қийматини қайтаради.

Синтаксисдан кўриниб турибдики, функция void туридаги қиймат қайтаради. Амалда эса конкрет турдаги объект учун хотира ажратиш зарур бўлади. Бунинг учун void турини конкрет турга келтириш технологиясидан фойдаланилади. Масалан, бутун турдаги узунлиги 3 га тенг массивга жой ажратишни қуйидагича амалга ошириш мумкин:

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

calloc() функцияси malloc() функциясидан фарқли равишда массив учун жой ажратишдан ташқари массив элементларини 0 қиймати билан инициализация қилади. Бу функция синтаксиси

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

кўринишда бўлиб, num параметри ажратилган соҳада нечта элемент борлигини, size ҳар бир элемент ўлчамини билдиради.

free() хотирани бўшатиш функцияси ўчириладиган хотира бўлагига кўрсаткич бўлган ягона параметрга эга бўлади:

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

free() функцияси параметрининг void турида бўлиши ихтиёрий турдаги хотира бўлагини ўчириш имконини беради.

Қуйидаги программада 10 та бутун сондан иборат динамик массив яратиш, унга қиймат бериш ва ўчириш амаллари бажарилган.

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

```
 int * pVector;
 if ((pVector=(int*)malloc(10*sizeof(int)))==NULL)
 {
  cout<<"Xotira etarli emas!!!";
  return 1;
 }
 // ажратилган хотира соҳасини тўлдириш
 for(int i=0;i<10;i++) *(pVector+i)=i;
 // вектор элементларини чоп этиш
 for(int i=0; i<10; i++) cout<<*(pVector+i)<<endl;
 // ажратилган хотира бўлагини қайтариш (ўчириш)
 free(pVector);
 return 0;
}
```

Кейинги программада $n \times n$ ўлчамли ҳақиқий сонлар массивининг бош диагоналидан юқорида жойлашган элементлар йиғиндисини ҳисоблаш масаласи ечилган.

```
#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 оператори ёрдамида, массивга хотира ажратишда объект туридан кейин квадрат қавс ичида объектлар сони кўрсатилади. Масалан, бутун турдаги 10 та сондан иборат массивга жой ажратиш учун

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

ифодаси ёзилиши керак. Бунга қарама-қарши равишда, бу усулда ажратилган хотирани бўшатиш учун

```
delete [] pVector;
```

кўрсатмасини бериш керак бўлади.

Икки ўлчамли динамик массивни ташкил қилиш учун

```
int **a;
```

кўринишидаги «кўрсаткичга кўрсаткич» ишлатилади.

Бошда массив сатрлари сонига қараб кўрсаткичлар массивига динамик хотирадан жой ажратиш керак:

```
a=new int *[m] // бу ерда m массив сатрлари сони
```

Кейин, ҳар бир сатр учун такрорлаш оператори ёрдамида хотира ажратиш ва уларнинг бошланғич адресларини а массив элементларига жойлаштириш зарур бўлади:

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

Шуни қайд этиш керакки, динамик массивнинг ҳар бир сатри хотиранинг турли жойларида жойлашиши мумкин (7.1 ва 7.3-расмлар).

Икки ўлчамли массивни ўчиришда олдин массивнинг ҳар бир элементи (сатри), сўнгра массивнинг ўзи йўқотилади:

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

Матрицани векторга кўпайтириш масаласи учун динамик массивлардан фойдаланишга мисол:

```
void main ()
{
 int n,m;
 int i,j; float s;
 cout<<"\n n="; cin>>n; // матрица сатрлари сони
 cout<<"\n m="; cin>>m; // матрица устунлари сони
 float *b=new float[m];
 float *c=new float[n];
 // кўрсаткичлар массивига хотира ажратиш
 float **a=new float *[n] ;
 for(i=0;i<n;i++)      // ҳар бир сатр учун
 a[i]=new float[m];    //динамик хотира ажратиш
 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;
}
```

Функция ва массивлар

Функциялар массивни параметр сифатида ишлатиши ва уни функциянинг натижаси сифатида қайтариши мумкин.

Агар массив параметр орқали функцияга узатилса, элементлар сонини аниқлаш муаммоси туғилади, чунки массив номидан унинг узунлигини аниқлашнинг иложи йўқ. Айрим ҳолларда, масалан, белгилар массиви сифатида аниқланган сатр (ASCIIZ сатрлар) билан ишлаганда массив узунлигини аниқлаш мумкин, чунки сатрлар ‘\0’ белгиси билан тугайди.

Мисол учун:

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

Функция параметри сатр бўлмаган ҳолларда фиксирланган узунликдаги массивлар ишлатилади. Агар турли узунликдаги массивларни узатиш зарур бўлса, массив ўлчамларини параметр сифатида узатиш мумкин ёки бу мақсадда глобал ўзгарувчидан фойдаланишга тўғри келади.

Мисол:

```
#include <iostream.h>
float sum(int n,float *x) //бу иккинчи усул
{
 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);
}
```

Массив номи кўрсаткич бўлганлиги сабабли массив элементларини функцияда ўзгартириш мумкин ва бу ўзгартиришлар функциядан чиққандан кейин ҳам сақланиб қолади.

```
#include <iostream.h>
void vector_01(int n,int*x,int * y) //бу иккинчи усул
{
 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];
}
```

Масала. Бутун турдаги ва элементлари камаймайдиган ҳолда тартибланган бир ўлчамли иккита массивларни ягона массивга, тартибланиш сақланган ҳолда бирлаштириш амалга оширилсин.

Программа матни:

```
#include <iostream.h>
\\бутун турдаги массивга кўрсаткич қайтарадиган
\\функция
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;
}
```

Кўп ўлчамли массивлар билан ишлаш маълум бир мураккабликка эга, чунки массивлар хотирада жойлаш тартиби турли вариантда бўлиши мумкин. Масалан, функция параметрлар рўйхатида $n \times n$ ўлчамдаги ҳақиқий турдаги x[n][n] массивга мос келувчи параметрни

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

кўринишда ёзиб бўлмайди. Муаммо ечими - бу массив ўлчамини параметр сифатида узатиш ва функция сарлавҳасини қуйидагича ёзиш керак:

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

Кўп ўлчамли массивларни параметр сифатида ишлатишда бир нечта усуллардан фойдаланиш мумкин.

1-усул. Массивнинг иккинчи ўлчамини ўзгармас ифода (сон) билан кўрсатиш:

```
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-усул. Икки ўлчамли массив кўрсаткичлар массиви кўринишида аниқланган ҳолатлар учун кўрсаткичлар массивини (матрица сатрлар адресларини) бериш орқали:

```
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]" эмас,чунки массивга мурожат
 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-усул. Кўрсаткичларга кўрсаткич кўринишида аниқланган динамик массивларни ишлатиш билан:

```
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;
}
```

Навбатдаги программада функция томонидан натижа сифатида икки ўлчамли массивни қайтаришига мисол келтирилган. Массив элементларнинг қийматлари тасодифий сонлардан ташкил топади. Тасоддифий сонлар «math.h» кутубхонасидаги random() функция ёрдамида ҳосил қилинади:

```
#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;
}
```

8-боб. ASCIIZ сатрлар ва улар устида амаллар

Белги ва сатрлар

Стандарт C++ тили икки хилдаги белгилар мажмуасини қўллаб-қувватлайди. Биринчи тоифага, анъанавий, «тор» белгилар деб номланувчи 8-битли белгилар мажмуаси киради, иккинчисига 16-битли «кенг» белгилар киради. Тил кутубхонасида ҳар бир гуруҳ белгилари учун махсус функциялар тўплами аниқланган.

C++ тилида сатр учун махсус тур аниқланмаган. Сатр char туридаги белгилар массиви сифатида қаралади ва бу белгилар кетмакетлиги *сатр терминатори* деб номланувчи 0 кодли белги билан тугайди (‘\0’). Одатда, нол-терминатор билан тугайдиган сатрларни *ASCIIZ-сатрлар* дейилади.

Қуйидаги жадвалда C++ тилида белги сифатида ишлатилиши мумкин бўлган ўзгармаслар тўплами келтирилган.

8.1-жадвал. C++ тилидаги белги ўзгармаслар

Белгилар синфлари	Белги ўзгармаслар
Катта ҳарфлар	‘A’ …’Z’, ‘A’…’Я’
Кичик ҳарфлар	‘a’ …’z’, ‘a’…’я’
Рақамлар	‘0’ …’9’
Бўш жой	горизонтал табуляция (ASCII коди 9), сатрни ўтказиш (ASCII коди 10), вертикал табуляция (ASCII коди 11), формани ўтказиш (ASCII коди 12), кареткани қайтариш (ASCII коди 13)
Пунктуация белгилари (ажратувчилар)	! ” # $ & ‘ ( ) * + - , . / : ; < = > ? @ [\ ] ^ _ { | } ~
Бошқарув белгилари	ASCII коди 0...1Fh оралиғида ва 7Fh бўлган белгилар
Пробел	ASCII коди 32 бўлган белги
Ўн олтилик рақамлар	‘0’…’9’, ‘A’…’F’, ‘a’…’f’

Сатр массиви эълон қилинишида, сатр охирига терминатор қўйилиши ва натижада сатрга қўшимча битта байт бўлишини инобатга олиниши керак:

```
char satr[10];
```

Ушбу эълонда satr сатри учун жами 10 байт ажратилади - 9 сатр ҳосил қилувчи белгилар учун ва 1 байт терминатор учун.

Сатр ўзгарувчилар эълон қилинишида бошланғич қийматларни қабул қилиши мумкин. Бу ҳолда компилятор автоматик равишда сатр узунлиги ҳисоблайди ва сатр охирига терминаторни қўшиб қўяди:

```
char Hafta_kuni[]=”Juma”;
```

Ушбу эълон қуйидаги эълон билан эквивалент:

```
char Hafta_kuni[]={'J','u','m','a','\0'};
```

Сатр қийматини ўқишда оқимли ўқиш оператори “>>” ўрнига getline() функциясини ишлатган маъқул ҳисобланади, чунки оқимли ўқишда пробеллар инкор қилинади (гарчи улар сатр белгиси ҳисобланса ҳам) ва ўқилаётган белгилар кетма-кетлиги сатрдан «ошиб» кетганда ҳам белгиларни киритиш давом этиши мумкин. Натижада сатр ўзига ажратилган ўлчамдан ортиқ белгиларни «қабул» қилади. Шу сабабли, getline() функцияси иккита параметрга эга бўлиб, биринчи параметр ўқиш амалга оширилаётган сатрга кўрсаткич, иккинчи параметрда эса ўқилиши керак бўлган белгилар сони кўрсатилади. Сатрни getline() функцияси орқали ўқишга мисол кўрайлик:

```
#include <iostream.h>
int main()
{
 char satr[6];
 cout<<"Satrni kiriting: "<<'\n';
 cin.getline(satr,6);
 cout<<"Siz kiritgan satr: "<<satr;
 return 0;
}
```

Программада ишлатилган satr сатри 5 та белгини қабул қилиши мумкин, ортиқчалари ташлаб юборилади. getline() функциясига мурожаатда иккинчи параметр қиймати ўқилаётган сатр узунлигидан катта бўлмаслиги керак.

Сатр билан ишлайдиган функцияларнинг аксарияти «string.h» кутубхонасида жамланган. Нисбатан кўп ишлатиладиган функцияларнинг тавсифини келтирамиз.

Сатр узунлигини аниқлаш функциялари

Сатрлар билан ишлашда, аксарият ҳолларда сатр узунлигини билиш зарур бўлади. Бунинг учун «string.h» кутубхонасида strlen() функцияси аниқланган бўлиб, унинг синтаксиси қуйидагича бўлади:

```
size_t strlen(const char* string)
```

Бу функция узунлиги ҳисобланиши керак бўлган сатр бошига кўрсаткич бўлган ягона параметрга эга ва у натижа сифатида ишорасиз бутун сонни қайтаради. strlen() функцияси сатрнинг реал узунлигидан битта кам қиймат қайтаради, яъни нол-терминатор ўрни ҳисобга олинмайди.

Худди шу мақсадда sizeof() функциясидан ҳам фойдаланиш мумкин ва у strlen() функциясидан фарқли равишда сатрнинг реал узунлигини қайтаради. Қуйида келтирилган мисолда сатр узунлигини ҳисоблашнинг ҳар иккита варианти келтирилган:

```
#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;
}
```

Программа ишлаши натижасида экранга

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

хабарлари чиқади.

Одатда sizeof() функциясидан getline() функциясининг иккинчи аргументи сифати ишлатилади ва сатр узунлигини яққол кўрсатмаслик имконини беради:

```
cin.getline(Satr, sizeof(Satr));
```

Масала. Фақат лотин ҳарфларидан ташкил топган сатр берилган. Ундаги ҳар хил ҳарфлар миқдори аниқлансин.

```
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;
}
```

Программада сатр учун 80 узунлигидаги Satr белгилар массиви эълон қилинган ва унинг қиймати клавиатурадан киритилади. Масала қуйидагича ечилади. Ичма-ич жойлашган такрорлаш оператори ёрдамида Satr массивининг ҳар бир элементи - Satr[i] массивнинг барча элементлари - Satr[j] билан устма-уст тушиши ёки улар бир-биридан 32 сонига фарқ қилиши (катта ва кичик лотин ҳарфларининг кодлари ўртасидаги фарқ) ҳолатлари k ўзгарувчисида саналади ва s умумий йиғиндига 1/k қиймати билан қўшилади. Программа охирида s қиймати бутун турга айлантирилган ҳолда чоп этилади. Сатрдаги сўзларни бир-биридан ажратувчи пробел белгиси чеклаб ўтилади.

Программага

```
Satrdagi turli harflar miqdori
```

сатри киритилса, экранга жавоб тариқасида

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

сатри чоп этилади.

Сатрларни нусхалаш

Сатр қийматини биридан иккинчисига нусхалаш мумкин. Бу мақсадда бир қатор стандарт функциялар аниқланган бўлиб, уларнинг айримларининг тавсифларини келтирамиз.

strcpy() функцияси прототипи

```
char* strcpy(char* str1, const char* str2)
```

кўринишга эга ва бу функция str2 сатрдаги белгиларни str1 сатрга байтма-байт нусхалайди. Нусхалаш str2 кўрсатиб турган сатрдаги нол-терминал учрагунча давом этади. Шу сабабли, str2 сатр узунлиги str1 сатр узунлигидан катта эмаслигига ишонч ҳосил қилиш керак, акс ҳолда берилган соҳасида (сегментда) str1 сатрдан кейин жойлашган берилганлар «устига» str2 сатрнинг «ортиб қолган» қисми ёзилиши мумкин.

Навбатдаги программа қисми “Satrni nusxalash!” сатрини Str сатрга нусхалайди:

```
char Str[20];
strcpy(Str, ”Satrni nusxalash!”);
```

Зарур бўлганда сатрнинг қайсидир жойидан бошлаб, охиригача нусхалаш мумкин. Масалан, “Satrni nusxalash!” сатрини 8-белгисидан бошлаб нусха олиш зарур бўлса, уни қуйидагича ечиш мумкин:

```
#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() функциясининг strcpy() функциясидан фарқли жойи шундаки, унда бир сатрдан иккинчисига нусхаланадиган белгилар сони кўрсатилади. Унинг прототипи қуйидаги кўринишга эга:

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

Агар str1 сатр узунлиги str2 сатр узунлигидан кичик бўлса, ортиқча белгилар «кесиб» ташланади. strncpy() функцияси ишлатилишига мисол кўрайлик:

```
#include <iostream.h>
#include <string.h>
int main()
{
 char Uzun_str[]="012345678901234567890";
 char Qisqa_str[]="ABCDEF";
 strncpy(Qisqa_str,Uzun_str,4);
 cout <<"Uzun_str= "<<Uzun_str<<endl;
 cout<<"Qisqa_str="<<Qisqa_str<<endl;
 return 0;
}
```

Программада Uzun_str сатри бошидан 4 белги Qisqa_str сатрига, унинг олдинги қийматлари устига жойланади ва натижада экранга

```
01234567890123456789
0123EF
```

сатрлар чоп этилади.

strdup() функциясига ягона параметр сифатида сатр-манбага кўрсаткич узатилади. Функция, сатрга мос хотирадан жой ажратади, унга сатрни нусхалайди ва юзага келган сатр-нусха адресини жавоб сифатида қайтаради. strdup() функция синтаксиси:

```
char* strdup(const char* sourse)
```

Қуйидаги программа бўлагида satr1 сатрининг нусхаси хотирани satr2 кўрсатган жойида пайдо бўлади:

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

Сатрларни улаш

Сатрларни улаш (конкатенация) амали янги сатрларни ҳосил қилишда кенг қўлланилади. Бу мақсадда «string.h» кутубхонасида strcat() ва strncat() функциялари аниқланган.

strcat() функцияси синтаксиси қуйидаги кўринишга эга:

```
char* strcat(char* str1, const char* str2)
```

Функция ишлаши натижасида str2 сатр, функция қайтарувчи сатр - str1 сатр охирига уланади. Функцияни чақиришдан олдин str1 сатр узунлиги, унга str2 сатри уланиши учун етарли бўлиши ҳисобга олинган бўлиши керак.

Қуйида келтирилган амаллар кетма-кетлигининг бажарилиши натижасида satr сатрига қўшимча сатр уланиши кўрсатилган:

```
char satr[80];
strcpy(satr,"Bu satrga ");
strcat(satr,"satr osti ulandi.");
```

Амаллар кетма-кетлигини бажарилиши натижасида satr кўрсатаётган жойда "Bu satrga satr osti ulandi." сатри пайдо бўлади.

strncat() функцияси strcat() функциядан фарқли равишда str1 сатрга str2 сатрнинг кўрсатилган узунликдаги сатр қисмини улайди. Уланадиган сатр қисми узунлиги функциянинг учинчи параметри сифатида берилади.

Функция синтаксиси

```
char* strncat(char* str1,const char* str2,size_t num)
```

Пастда келтирилган программа бўлагида str1 сатрга str2 сатрнинг бошланғич 10 та белгидан иборат сатр қисмини улайди:

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

Амаллар бажарилиши натижасида экранга

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

сатри чоп этилади.

Масала. Нол-терминатор билан тугайдиган S,S1 ва S2 сатрлар берилган. S сатрдаги S1 сатр остилари S2 сатр ости билан алмаштирилсин. Масалани ечиш учун қуйидаги масала остиларини ечиш зарур бўлади:

1) S сатрида S1 сатр остини кириш ўрнини аниқлаш;
2) S сатридан S1 сатр остини ўчириш;
3) S сатрида S1 сатр ости ўрнига S2 сатр остини ўрнатиш.

Гарчи бу масала остиларининг ечимлари C++ тилнинг стандарт кутубхоналарида функциялар кўринишида мавжуд бўлса ҳам, улар кодини қайта ёзиш фойдаланувчига бу амалларнинг ички моҳиятини тушунишга имкон беради. Қуйида масала ечимининг программа матни келтирилган:

```
#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);
}
```

Программада ҳар бир масала остига мос функциялар тузилган:

1) int Izlash(char satr[],char satr_osti[]) - функцияси satr сатрига satr_osti сатрининг чап томондан биринчи киришининг ўрнини қайтаради. Агар satr сатрида satr_osti учрамаса -1 қийматини қайтаради.

2) void Qirqish(char satr[],int joy,int qirqish_soni) - функцияси satr сатрининг joy ўрнидан бошлаб qirqish_soni сондаги белгиларни қирқиб ташлайди. Функция натижаси satr сатрида ҳосил бўлади;

3) void Joylash(char satr[],char satr_osti[],int joy) - функцияси satr сатрига, унинг joy ўрнидан бошлаб satr_osti сатрини жойлаштиради.

Бош функцияда сатр (S), унда алмаштириладиган сатр (S1) ва S1 ўрнига жойлаштириладиган сатр (S2) оқимдан ўқилади. Такрорлаш оператори бажарилишининг ҳар бир қадамида S сатрининг чап томонидан бошлаб S1 сатри изланади. Агар S сатрида S1 мавжуд бўлса, у қирқилади ва шу ўринга S2 сатри жойлаштирилади. Такрорлаш жараёни Izlash() функцияси -1 қийматини қайтаргунча давом этади.

Сатрларни солиштириш

Сатрларни солиштириш улардаги мос ўринда жойлашган белгилар кодларини ўзаро солиштириш билан аниқланади. Бунинг учун «string.h» кутубхонасида стандарт функциялар мавжуд.

strcmp() функцияси синтаксиси

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

кўринишига эга бўлиб, функция str1 ва str2 солиштириш натижаси сифатида сон қийматни қайтаради (масалан, бутун і ўзгарувчисида) ва улар қуйидагича изоҳланади:

a) i<0 - агар str1 сатри str2 сатридан кичик бўлса;
b) i=0 - агар str1 сатри str2 сатрига тенг бўлса;
c) i>0 - агар str1 сатри str2 сатридан катта бўлса.

Функция ҳарфларнинг регистрини фарқлайди. Буни мисолда кўришимиз мумкин:

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

Натижада і ўзгарувчиси мусбат қиймат қабул қилади, чунки солиштирилаётган сатрлардаги «pascal» ва «Pascal» сатр қисмларида биринчи ҳарфлар фарқ қилади. Келтирилган мисолда і қиймати 32 бўлади. Бу фарқланувчи ҳарфлар кодларининг айирмаси. Агар функцияга

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

кўринишида мурожаат қилинса і қиймати манфий сон -32 бўлади.

Агар сатрлардаги бош ёки кичик ҳарфларни фарқламасдан солиштириш амалини бажариш зарур бўлса, бунинг учун strcmpi() функциясидан фойдаланиш мумкин. Юқорида келтирилган мисолдаги сатрлар учун

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

амали бажарилганда і қиймати 0 бўлади.

strncmp( ) функцияси синтаксиси

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

кўринишида бўлиб, str1 ва str2 сатрларни бошланғич num сонидаги белгиларини солиштиради. Функция ҳарфлар регистрини инобатга олади. Юқорида мисолда аниқланган satr1 ва satr2 сатрлар учун

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

амали бажарилишида і қиймати 0 бўлади, чунки сатрлар бошидаги 31 белгилар бир хил.

strncmpi() функцияси strncmp() функциясидек амал қилади, фарқли томони шундаки, солиштиришда ҳарфларнинг регистрини ҳисобга олинмайди. Худди шу сатрлар учун

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

амали бажарилиши натижасида і ўзгарувчи қиймати 0 бўлади.

Сатрдаги ҳарфлар регистрини алмаштириш

Берилган сатрдаги кичик ҳарфларни бош ҳарфларга ёки тескарисига алмаштиришга мос равишда _strupr() ва _strlwr() функциялар ёрдамида амалга ошириш мумкин. Компиляторларнинг айрим вариантларида функциялар номидаги тагчизиқ (‘_’) бўлмаслиги мумкин.

_strlwr() функцияси синтаксиси

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

кўринишида бўлиб, аргумент сифатида берилган сатрдаги бош ҳарфларни кичик ҳарфларга алмаштиради ва ҳосил бўлган сатр адресини функция натижаси сифатида қайтаради. Қуйидаги программа бўлаги _strlwr() функциясидан фойдалнишга мисол бўлади.

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

Натижада экранга

```
10 ta katta harflar
```

сатри чоп этилади.

_strupr() функцияси худди _strlwr() функциясидек амал қилади, лекин сатрдаги кичик ҳарфларни бош ҳарфларга алмаштиради:

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

Натижада экранга

```
10 TA KATTA HARFLAR
```

сатри чоп этилади.

Программалаш амалиётида белгиларни қайсидир оралиққа тегишли эканлигини билиш зарур бўлади. Буни «ctype.h» сарлавҳа файлида эълон қилинган функциялар ёрдамида аниқлаш мумкин. Қуйида уларнинг бир қисмининг тавсифи келтирилган:

isalnum() - белги рақам ёки ҳарф (true) ёки йўқлигини (false) аниқлайди;

isalpha() - белгини ҳарф (true) ёки йўқлигини (false) аниқлайди;

isascii() - белгини коди 0..127 оралиғида (true) ёки йўқлигини (false) аниқлайди;

isdigit() - белгини рақамлар диапазонига тегишли (true) ёки йўқлигини (false) аниқлайди.

Бу функциялардан фойдаланишга мисол келтирамиз.

```
#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);
}
```

Программада фойдаланувчига туғилган йилини киритиш таклиф этилади. Киритилган сана satr ўзгарувчисига ўқилади ва агар сатрнинг

ҳар бир белгиси (satr[i]) ҳарф ёки бошқарув белгиси ёки пунктуация белгиси бўлса, шу ҳақда хабар берилади ва туғилган йилни қайта киритиш таклиф этилади. Программа туғилган йил (тўртта рақам) тўғри киритилганда “Sizni tug'ilgan yilingiz: XXXX” сатрини чоп қилиш билан ўз ишини тугатади.

Сатрни тескари тартиблаш

Сатрни тескари тартиблашни учун strrev() функциясидан фойдаланиш мумкин. Бу функция қуйидагича прототипга эга:

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

Сатр реверсини ҳосил этишга мисол:

```
char str[]=”telefon”;
cout<<strrev(str);
```

амаллар бажарилиши натижасида экранга

```
nofelet
```

сатри чоп этилади.

Сатрда белгини излаш функциялари

Сатрлар билан ишлашда ундаги бирорта белгини излаш учун «string.h» кутубхонасида бир қатор стандарт функциялар мавжуд.

Бирорта белгини берилган сатрда бор ёки йўқлигини аниқлаб берувчи strchr() функциясининг прототипи

```
char* strchr(const char* string, int c);
```

кўринишида бўлиб, у c белгинининг string сатрида излайди. Агар излаш мувофаққиятли бўлса, функция шу белгининг сатрдаги ўрнини (адресини) функция натижаси сифатида қайтаради, акс ҳолда, яъни белги сатрда учрамаса функция NULL қийматини қайтаради. Белгини излаш сатр бошидан бошланади.

Қуйида келтирилган программа бўлаги белгини сатрдан излаш билан боғлиқ.

```
char satr[]=”0123456789”;
char* pSatr;
pSatr=strchr(satr,’6’);
```

Программа ишлаши натижасида pSatr кўрсаткичи satr сатрининг ‘6’ белгиси жойлашган ўрни адресини кўрсатади.

strrchr() функцияси берилган белгини берилган сатр охиридан бошлаб излайди. Агар излаш муваффақиятли бўлса, белгини сатрга охирги киришининг ўрнини қайтаради, акс ҳолда NULL.

Мисол учун

```
char satr[]=”0123456789101112”;
char* pSatr;
pSatr=strrchr(satr,’0’);
```

амалларини бажарилишида pSatr кўрсаткичи satr сатрининг “01112” сатр қисмининг бошланишига кўрсатади.

strspn() функцияси иккита сатрни белгиларни солиштиради. Функция қуйидаги

```
size_t strspn(const char* str1, const char* str2);
```

кўринишга эга бўлиб, у str1 сатрдаги str2 сатрга кирувчи бирорта белгини излайди ва агар бундай элемент топилса, унинг индекси функция қиймати сифатида қайтарилади, акс ҳолда функция сатр узунлигидан битта ортиқ қийматни қайтаради.

Мисол:

```
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.";
```

амаллар бажарилиши натижасида экранга

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

сатри чоп этилади.

strcspn() функциясининг прототипи

```
size_t strcspn(const char* str1, const char* str2);
```

кўринишида бўлиб, у str1 ва str2 сатрларни солиштиради ва str1 сатрининг str2 сатрига кирган биринчи белгини индексини қайтаради. Масалан,

```
char satr[]=”Birinchi satr”;
int index;
index=strcspn(satr,”sanoq tizimi”);
```

амаллари бажарилгандан кейин index ўзгарувчиси 1 қийматини қабул қилади, чунки биринчи сатрнинг биринчи ўриндаги белгиси иккинчи сатрда учрайди.

strpbrk() функциясининг прототипи

```
char* strpbrk(const char* str1, const char* str2);
```

кўринишга эга бўлиб, у str1 сатрдаги str2 сатрга кирувчи бирорта белгини излайди ва агар бундай элемент топилса, унинг адреси функция қиймати сифатида қайтарилади, акс ҳолда функция NULL

қиймати қайтаради. Қуйидаги мисол функцияни қандай ишлашини кўрсатади.

```
char satr1[]="0123456789ABCDEF";
char satr2[]="ZXYabcdefABC";
char* element;
element = strpbrk(satr1,satr2);
cout<<element<<'\n';
```

Программа ишлаши натижасида экранга str1 сатрининг

```
ABCDEF
```

сатр остиси чоп этилади.

Сатр қисмларини излаш функциялари

Сатрлар билан ишлашда бир сатрда иккинчи бир сатрнинг (ёки унинг бирор қисмини) тўлиқ киришини аниқлаш билан боғлиқ масалалар нисбатан кўп учрайди. Масалан, матн таҳрирларидаги сатрдаги бирорта сатр қисмини иккинчи сатр қисми билан алмаштириш масаласини мисол келтириш мумкин (юқорида худди шундай масала учун программа келтирилган). Стандарт «string.h» кутубхонаси бу тоифадаги масалалар учун бир нечта функцияларни таклиф этади.

strstr() функцияси қуйидагича эълон қилинади:

```
char* strstr(const char* str, const char* substr);
```

Бу функция str сатрига substr сатр қисми кириши текширади, агар substr сатр қисми str сатрига тўлиқ кириши мавжуд бўлса, сатрнинг чап томонидан биринчи киришдаги биринчи белгининг адреси жавоб тариқасида қайтарилади, акс ҳолда функция NULL қийматини қайтаради.

Қуйидаги мисол strstr() функциясини ишлатишни кўрсатади.

```
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';
```

Программа буйруқлари бажарилиши натижасида экранга

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

сатри чоп этилади.

Кейинги программа бўлагида сатрда бошқа бир сатр қисми мавжуд ёки йўқлигини назорат қилиш ҳолати кўрсатилган:

```
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';
```

Программада фойдаланувчидан сатр қисми сифатида бирорта номни киритиш талаб қилинади ва бу қиймат Ism сатрига ўқилади. Киритилган исм программада аниқланган рўйхатда (Ismlar сатрида) бор ёки йўқлиги аниқланади ва хабар берилади.

strtok() функциясининг синтаксиси

```
char* strtok(char* str, const char* delim);
```

кўринишда бўлиб, у str сатрида delim сатр-рўйхатида берилган ажратувчилар оралиғига олинган сатр қисмларни ажратиб олиш имконини беради. Функция биринчи сатрда иккинчи сатр-рўйхатдаги ажратувчини учратса, ундан кейин нол-терминаторни қўйиш орқали str сатрни иккига ажратади. Сатрнинг иккинчи бўлагидан ажратувчилар билан «ўраб олинган» сатр қисмлари топиш учун функцияни кейинги чақирилишида биринчи параметр ўрнига NULL қийматини қўйиш керак бўлади. Қуйидаги мисолда сатрни бўлакларга ажратиш масаласи қаралган:

```
#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;
}
```

Программа ишлаши натижасида экранга Ismlar сатридаги ‘␣’ (пробел), ‘,’ (вергул), ‘?’ (сўроқ белгиси) ва ‘0’ (рақам) билан ажратилган сатр қисмлари - исмлар чоп қилинади:

```
Alisher
Farxod
Munisa
Erkin
Akmal
Nodira
```

Турларни ўзгартириш функциялари

Сатрлар билан ишлашда сатр кўринишида берилган сонларни, сон турларидаги қийматларга айлантириш ёки тескари амални бажаришга тўғри келади. C++ тилининг «strlib.h» кутубхонасида бу амалларни бажарувчи функциялар тўплами мавжуд. Қуйида нисбатан кўп ишлатиладиган функциялар тавсифи келтирилган.

atoi( ) функциясининг синтаксиси

```
int atoi(const char* ptr);
```

кўринишга эга бўлиб, ptr кўрсатувчи ASCIIZ-сатрни int туридаги сонга ўтказишни амалга оширади. Функция сатр бошидан белгиларни сонга айлантира бошлайди ва сатр охиригача ёки биринчи рақам бўлмаган белгигача ишлайди. Агар сатр бошида сонга айлантириш мумкин бўлмаган белги бўлса, функция 0 қийматини қайтаради. Лекин, шунга эътибор бериш керакки, “0” сатри учун ҳам функция 0 қайтаради. Агар сатрни сонга айлантиришдаги ҳосил бўлган сон int чегарасидан чиқиб кетса, соннинг кичик икки байти натижа сифатида қайтарилади. Мисол учун

```
#include <stdlib.h>
#include <iostream.h>
int main()
 {
 char str[]="32secund";
 int i=atoi(str);
 cout<<i<<endl;
 return 0;
 }
```

программасининг натижаси сифатида экранга 32 сонини чоп этади. Агар str қиймати ”100000” бўлса, экранга -31072 қиймати чоп этилади, чунки 100000 сонинг ички кўриниши 0x186A0 ва унинг охирги икки байтидаги 0x86A0 қиймати 31072 сонининг қўшимча коддаги кўринишидир.

atol() функцияси худди atoi() функциясидек амал қилади, фақат функция натижаси long турида бўлади. Агар ҳосил бўлган сон қиймати long чегарасига сиғмаса, функция кутилмаган қийматни қайтаради.

atof() функцияси эълони

```
double atof (const char* ptr);
```

кўринишида бўлиб, ptr кўрсатувчи ASCIIZ-сатрни double туридаги сузувчи нуқтали сонга ўтказишни амалга оширади. Сатр сузувчи нуқтали сон форматида бўлиши керак.

Сонга айлантириш биринчи форматга мос келмайдиган белги учрагунча ёки сатр охиригача давом этади.

strtod() функцияси atof() функциясидан фарқли равишда сатрни double туридаги сонга ўтказишда конвертация жараёни узилган пайтда айлантириш мумкин бўлмаган биринчи белги адресини ҳам қайтаради. Бу ўз навбатида сатрни хато қисмини қайта ишлаш имконини беради.

strtod() функциясининг синтаксиси

```
double strtod(const char *s, char **endptr);
```

кўринишга эга ва endptr кўрсаткичи конвертация қилиниши мумкин бўлмаган биринчи белги адреси. Конвертация қилинувчи сатрда хато бўлган ҳолатни кўрсатувчи мисол:

```
#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;
}
```

Программа бажарилишида x ўзгарувчи 3.14 сонини қабул қилади, kursatkich ўзгарувчиси сатрдаги ‘D’ белгисининг адресини кўрсатади. Экранга қуйидаги сатрлар кетма-кетлиги чоп этилади:

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

Бир қатор функциялар тескари амални, яъни берилган сонни сатрга айлантириш амалларини бажаради.

itoa() ва ltoa() функциялари мос равишда int ва long туридаги сонларни сатрга кўринишга ўтказади. Бу функциялар мос равишда қуйидаги синтаксисга эга:

```
char* itoa(int num, char *str, int radix);
```

ва

```
char* ltoa(long num, char *str, int radix);
```

Бу функциялар num сонини radix аргументда кўрсатилган саноқ системасидаги кўринишини str сатрда ҳосил қилади. Мисол учун 12345 сонини турли саноқ системадаги сатр кўринишини ҳосил қилиш масаласини кўрайлик:

```
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;
}
```

Программа экранга қуйидаги сатрларни чиқаради:

```
Son ko'rinishlari
2 sanoq sistemasida : 11000000111001
8 sanoq sistemasida : 30071
10 sanoq sistemasida: 12345
16 sanoq sistemasida: 3039
```

gcvt( ) функцияси

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

кўринишдаги прототипга эга бўлиб, double туридаги val сонини buf кўрсатувчи ASCIIZ сатрга айлантиради. Иккинчи аргумент сифатида бериладиган ndec қиймати сон кўринишида рақамлар миқдорини кўрсатади. Агар рақамлар сони ndec қийматидан кўп бўлса, имкон бўлса соннинг каср қисмидан ортиқча рақамлар қирқиб ташланади (яхлитланган ҳолда), акс ҳолда сон экспоненциал кўринишда ҳосил қилинади. Қуйидаги келтирилган программада gcvt() функциясидан фойдаланишнинг турли вариантлари кўрсатилган.

```
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;
}
```

Программа экранга кетма-кет равишда сон кўринишларини чоп этади:

```
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-боб. string туридаги сатрлар

C++ тилида стандарт сатр турига қўшимча сифатида string тури киритилган ва у string синфи кўринишида амалга оширилган. Бу турдаги сатр учун ‘\0’ белгиси тугаш белгиси ҳисобланмайди ва у оддийгина белгилар массиви сифатида қаралади. string турида сатрлар узунлигининг бажарила-диган амаллар натижасида динамик равишда ўзгариб туриши, унинг таркибида бир қатор функциялар аниқланганлиги бу тур билан ишлашда маълум бир қулайликлар яратади.

string туридаги ўзгарувчилар қуйидагича эълон қилиниши мумкин:

```
string s1,s2,s3;
```

Бу турдаги сатрлар учун махсус амаллар ва функциялар аниқ-ланган.

string сатрга бошланғич қийматлар ҳар хил усуллар орқали бериш мумкин:

```
string s1=”birinchi usul”;
string s2(”ikkinchi usul”);
string s3(s2);
string s4=s2;
```

Худди шундай, string туридаги ўзгарувчилар устида қиймат бериш амаллари ҳам ҳар хил:

```
string s1,s2,s3; char *str=”misol”;
//сатрли ўзгармас қиймати бериш
s1=”Qiymat berish 1-usul”;
s2=str;       // char туридаги сатр юкланмоқда
s3=’A’;       // битта белги қиймат сифатида бериш
s3=s3+s1+s2+”0123abc”; //қиймат сифатида сатр ифода
```

8.2-жадвалида string туридаги сатрлар устидан амаллар келти-рилган.

Сатр элементига индекс воситасидан ташқари at() функцияси орқали мурожаат қилиш мумкин:

```
string s1=”satr misoli“;
cout<<s.at(3) // натижада ‘r’ белгиси экранга чиқади
```

Шуни айтиб ўтиш керакки, string синфда шу турдаги ўзгарувчилар билан ишлайдиган функциялар аниқланган. Бошқача айтганда, string турида эълон қилинган ўзгарувчилар (объектлар) ўз функцияларига эга ҳисобланади ва уларни чақириш учун олдин

ўзгарувчи номи, кейин ‘.’ (нуқта) ва зарур функция номи (аргументлари билан) ёзилади.

8.2-жадвал. string туридаги сатрлар устидан амаллар

Амал	Мазмуни	Мисол
=, +=	Қиймат бериш амали	s=”satr01234” s+=”2satr000”
+	Сатрлар улаш амали (конкантенация)	s1+s2
==, !=, <, <=, >, >=	Сатрларни солиштириш амаллари	s1==s2 || s1>s2 && s1!=s2
[]	Индекс бериш	s[4]
<<	Оқимга чиқариш	cout << s
>>	Оқимдан ўқиш	cin >> s (пробелгача)

Сатр қисмини бошқа сатрга нусхалаш функцияси

Бир сатр қисмини бошқа сатрга юклаш учун куйидаги функция-ларни ишлатиш мумкин, уларни прототипи куйидагича:

```
assign(const string &str);
assign(const string &str,unsigned int pos,
       unsigned int n);
assign(const char *str, int n);
```

Биринчи функция қиймат бериш амал билан эквивалентдир: string туридаги str сатр ўзгарувчи ёки сатр ўзгармасни амални чақирувчи сатрга беради:

```
string s1,s2;
s1=”birinchi satr”;
s2.assign(s1); // s2=s1 амалга эквивалент
```

Иккинчи функция чақирувчи сатрга аргументдаги str сатрнинг pos ўрнидан n та белгидан иборат бўлган сатр қисмини нусхалайди. Агарда pos қиймати str сатр узунлигидан катта бўлса, хатолик ҳақида огоҳлантирилади, агар pos + n ифода қиймати str сатр узунлигидан катта бўлса, str сатрининг pos ўрнидан бошлаб сатр охиригача бўлган белгилар нусхаланади. Бу қоида барча функциялар учун тегишлидир.

Мисол:

```
string s1,s2,s3;
s1=”0123456789”;
s2.assign(s1,4,5);      // s2=”45678”
s3.assign(s1,2,20);     // s3=”23456789”
```

Учинчи функция аргументдаги char туридаги str сатрни string турига айлантириб, функцияни чақирувчи сатрга ўзлаштиради:

```
char * strold;
cin.getline(strold,100);//"0123456789" киритилади
string s1,s2;
s2.assign(strold,6);   // s2="012345"
s3.assign(strold,20);  // s3="0123456789"
```

Сатр қисмини бошқа сатрга қўшиш функцияси

Сатр қисмини бошқа сатрга қўшиш функциялари қуйидагича:

```
append(const string &str);
append(const string & str,unsigned int pos,
       unsigned int n);
append(const char *str, int n);
```

Бу функцияларни юқорида келтирилган мос assign функция-лардан фарқи - функцияни чақирувчи сатр охирига str сатрни ўзини ёки унинг қисмини қўшади.

```
char * sc;
cin.getline(sc,100);   //"0123456789" киритилади
string s1,s,s2;
s2=sc; s1="misol";
s="aaa";              //s2="0123456789"
s2.append("abcdef"); //s2+="abcdef" амали
                      //ва s2="0123456789abcdef"
s1.append(s2,4,5);    //s1="misol45678"
s.append(ss,5);       // s="aaa012345"
```

Сатр қисмини бошқа сатр ичига жойлаштириш функцияси

Бир сатрга иккинчи сатр қисмини жойлаштириш учун қуйидаги функциялар ишлатилади:

```
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);
```

Бу фукциялар append каби ишлайди, фарқи шундаки, str сатрини ёки унинг қисмини функцияни чақирувчи сатрнинг кўрсатилган pos1 ўрнидан бошлаб жойлаштиради. Бунда амал чақирувчи сатрнинг pos1 ўриндан кейин жойлашган белгилар ўнга сурилади.

Мисол:

```
char * sc;
cin.getline (sc,100); //"0123456789" сатри киритилади
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”
```

Сатр қисмини ўчириш функцияси

Сатр қисмини ўчириш учун қуйидаги функцияни ишлатиш мумкин:

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

Бу функция, уни чақирувчи сатрнинг pos ўрнидан бошлаб n та белгини ўчиради. Агарда pos кўрсатилмаса, сатр бошидан бошлаб ўчирилади. Агар n кўрсатилмаса, сатрни охиригача бўлган белгилар ўчирилади:

```
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() функцияси, уни чақирувчи сатрни тўлиқ тозалайди.

Масалан:

```
s1.clear(); //сатр бўш ҳисобланади (s1=””)
```

Сатр қисмини алмаштириш функцияси

Бир сатр қисмининг ўрнига бошқа сатр қисмини қўйиш учун қуйидаги функциялардан фойдаланиш мумкин:

```
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);
```

Бу функциялар insert каби ишлайди, ундан фарқли равишда амал чақирувчи сатрнинг кўрсатилган ўрнидан (pos1) n1 белгилар ўрнига str сатрини ёки унинг pos2 ўриндан бошланган n2 белгидан иборат қисмини қўяди (алмаштиради).

Мисол:

```
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) функцияси иккита сатрларни ўзаро алмаштириш учун ишлатилади. Масалан:

```
string s1,s2;
s1="01234";
s2="98765432";
s1.swap(s2); // s2="01234" ва s1="98765432" бўлади.
```

Сатр қисмини ажратиб олиш функцияси

Функция прототипи куйидагича:

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

Бу функция, уни чақирувчи сатрнинг pos ўрнидан бошлаб n белгини натижа сифатида қайтаради. Агарда pos кўрсатилмаса, сатр бошидан бошлаб ажратиб олинади, агар n кўрсатилмаса, сатр охиригача бўлган белгилар натижа сифатида қайтарилади:

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

string туридаги сатрни char турига ўтказиш

string туридаги сатрни char турига ўтказиш учун

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

функцияни ишлатиш керак. Бу функция char турдаги '\0' белгиси билан тугайдиган сатрга ўзгармас кўрсаткични қайтаради:

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

Худди шу мақсадда

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

функциясидан ҳам фойдаланиш мумкин. Лекин бу функция сатр охирига '\0' белгисини қўшмайди.

Сатр қисмини излаш функциялари

string синфида сатр қисмини излаш учун ҳар хил вариантдаги функциялар аниқланган. Қуйида улардан асосийларининг тавсифини келтирамиз.

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

Функция, уни чақирган сатрнинг кўрсатилган жойдан (pos) бошлаб str сатрни қидиради ва биринчи мос келувчи сатр қисмининг бошланиш индексини жавоб сифатида қайтаради, акс ҳолда максимал мусбат бутун npos сонни қайтаради (npos=4294967295), агар излаш ўрни (pos) берилмаса, сатр бошидан бошлаб изланади.

```
unsigned int find(char c.unsigned int pos=0)const;
```

Бу функция олдингидан фарқи равишда сатрдан с белгисини излайди.

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

Функция, уни чақирган сатрнинг кўрсатилган pos ўрнигача str сатрнинг биринчи учраган жойини индексини қайтаради, акс ҳолда npos қийматини қайтаради, агар pos кўрсатилмаса сатр охиригача излайди.

```
unsigned int rfind(char c.unsigned int pos=npos)
const;
```

Бу функциянинг олдингидан фарқи - сатрдан с белгиси изланади.

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

Функция, уни чақирган сатрнинг кўрсатилган (pos) жойидан бошлаб str сатрининг ихтиёрий бирорта белгисини қидиради ва биринчи учраганининг индексини, акс ҳолда npos сонини қайтаради.

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

Бу функциянинг олдингидан фарқи - сатрдан с белгисини излайди;

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

Функция, уни чақирган сатрнинг кўрсатилган (pos) жойдан бошлаб str сатрни ихтиёрий бирорта белгисини қидиради ва ўнг томондан биринчи учраганининг индексини, акс ҳолда npos сонини қайтаради.

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

Бу функция олдингидан фарқи - сатрдан с белгисини излайди;

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

Функция, уни чақирган сатрнинг кўрсатилган (pos) жойдан бошлаб str сатрнинг бирорта ҳам белгиси кирмайдиган сатр қисмини қидиради

ва чап томондан биринчи учраганининг индексини, акс ҳолда npos сонини қайтарилади.

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

Бу функциянинг олдингидан фарқи - сатрдан с белгисидан фарқли биринчи белгини излайди;

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

Функция, уни чақирувчи сатрнинг кўрсатилган жойдан бошлаб str сатрини ташкил этувчи белгилар тўпламига кирмаган белгини қидиради ва энг ўнг томондан биринчи топилган белгининг индексини, акс ҳолда npos сонини қайтаради.

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

Бу функциянинг олдингидан фарқи - сатр охиридан бошлаб с белгисига ўхшамаган белгини излайди.

Излаш функцияларини қўллашга мисол:

```
#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; // натижа 3
 cout<<s1.rfind('3') <<endl;// натижа 10
 cout<<s1.find_first_of(s3)<<endl; // натижа 2
 cout<<s1.find_last_of(s3)<<endl;   // натижа 16
 cout<<s1.find_first_not_of(s2)<<endl; // натижа 14
 cout<<s1.find_last_not_of(s2)<<endl; // натижа 20
}
```

Сатрларни солиштириш

Сатрлар қисмларини солиштириш учун compare функцияси ишлатилади:

```
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;
```

Функциянинг биринчи шаклида иккита сатрлар тўла солиштирилади: функция манфий сон қайтаради, агар функцияни чақирувчи сатр str сатрдан кичик бўлса, 0 қайтаради агар улар тенг бўлса ва мусбат сон қайтаради, агар функция чақирувчи сатр str сатрдан катта бўлса.

Иккинчи шаклда худди биринчидек амаллар бажарилади, фақат функция чақирувчи сатрнинг pos1 ўрнидан бошлаб n1 та белгили сатр ости str сатр билан солиштирилади.

Учинчи кўринишда функция чақирувчи сатрнинг pos1 ўрнидан бошлаб n1 та белгили сатр қисми ва str сатрдан pos2 ўрнидан бошлаб n2 та белгили сатр қисмлари ўзаро солиштирилади.

Мисол:

```
#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;
 }
```

Масала. Фамилия, исми ва шарифлари билан талабалар рўйхати берилган. Рўйхат алфавит бўйича тартиблансин.

Программа матни:

```
#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;
}
```

Программада талабалар рўйхати string туридаги Talaba динамик массив кўринишида эълон қилинган ва унинг ўлчами фойдаланувчи томонидан киритилган talabar_soni билан аниқланади. Талабалар сонини киритишда назорат қилинади: клавиатурадан сатр ўқилади ва у atoi() функцияси ёрдамида сонга айлантирилади. Агар ҳосил бўлган сон нолдан катта сон бўлмаса, сонни киритиш жараёни такрорланади. Талабалар сони аниқ бўлгандан кейин ҳар бир талабанинг фамилия, исми ва шарифи битта сатр сифатида оқимдан ўқилади. Кейин, string турида аниқланган compare() функцияси ёрдамида массивдаги сатрлар ўзаро солиштирилади ва мос ўриндаги белгилар кодларини ўсиши бўйича «пуфакчали саралаш» орқали тартибланади. Программа охирида ҳосил бўлган массив чоп этилади, ҳамда динамик массивлар йўқотилади.

Сатр хоссаларини аниқлаш функциялари

string синфида сатр узунлиги, унинг бўшлигини ёки эгаллаган хотира ҳажмини аниқлайдиган функциялар бор:

```
unsigned int size()const; // сатр ўлчами
unsigned int length()const; // сатр элементлар сони
unsigned int max_size()const; // сатрнинг максимал
                              // узунлиги(4294967295)
unsigned int capacity()const;// сатр эгаллаган хотира
                              // ҳажми
bool empty()const; // true, агар сатр бўш бўлса
```

10-боб. Структуралар ва бирлашмалар

Структуралар

Маълумки, бирор предмет соҳасидаги масалани ечишда ундаги объектлар бир нечта, ҳар хил турдаги параметрлар билан аниқланиши мумкин. Масалан, текисликдаги нуқта ҳақиқий турдаги x- абцисса ва y- ордината жуфтлиги - (x,y) кўринишида берилади. Талаба ҳақидаги маълумотлар: сатр туридаги талаба фамилия, исми ва шарифи, мутахассислик йўналиш, талаба яшаш адреси, бутун турдаги туғилган йили, ўқув босқичи, ҳақиқий турдаги рейтинг бали, мантиқий турдаги талаба жинси ҳақидаги маълумот ва бошқалардан шаклланади.

Программада ҳолат ёки тушунчани тавсифловчи ҳар бир берилганлар учун алоҳида ўзгарувчи аниқлаб масалани ечиш мумкин. Лекин бу ҳолда объект ҳақидаги маълумотлар «тарқоқ» бўлади, уларни қайта ишлаш мураккаблашади, объект ҳақидаги берилганларни яхлит ҳолда кўриш қийинлашади.

C++ тилида бир ёки ҳар хил турдаги берилганларни жамланмаси *структура* деб номланади. Структура фойдаланувчи томонидан аниқланган берилганларнинг янги тури ҳисобланади. Структура қуйидагича аниқланади:

struct <структура номи>
{
  <тур$_1$ > <ном$_1$>;
  <тур$_2$ > <ном$_2$>;
  …
  <тур$_n$> <ном$_n$>;
};

Бу ерда <структура номи> - структура кўринишида яратилаётган янги турнинг номи, “<тур$_i$> <ном$_i$>;” - структуранинг i-майдонининг (ном$_i$) эълони.

Бошқача айтганда, структура эълон қилинган ўзгарувчилардан (майдонлардан) ташкил топади. Унга ҳар хил турдаги берилганларни ўз ичига олувчи *қобиқ* деб қараш мумкин. Қобиқдаги берилганларни яхлит ҳолда кўчириш, ташқи қурилмалар (бинар файлларга) ёзиш, ўқиш мумкин бўлади.

Талаба ҳақидаги берилганларни ўз ичига олувчи структура турининг эълон қилинишини кўрайлик.

```
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;
};
```

Программада структуралардан фойдаланиш, шу турдаги ўзгарувчилар эълон қилиш ва уларни қайта ишлаш орқали амалга оширилади:

```
Talaba talaba;
```

Структура турини эълонида турнинг номи бўлмаслиги мумкин, лекин бу ҳолда структура аниқланишидан кейин албатта ўзгарувчилар номлари ёзилиши керак:

```
struct
{
 unsigned int x,y;
 unsigned char Rang;
} Nuqta1, Nuqta2;
```

Келтирилган мисолда структура туридаги Nuqta1, Nuqta2 ўзгарувчилари эълон қилинган.

Структура туридаги ўзгарувчилар билан ишлаш, унинг майдонлари билан ишлашни англатади. Структура майдонига мурожаат қилиш ‘.’ (нуқта) орқали амалга оширилади. Бунда структура туридаги ўзгарувчи номи, ундан кейин нуқта қўйилади ва майдон ўзгарувчисининг номи ёзилади. Масалан, талаба ҳақидаги структура майдонларига мурожаат қуйидагича бўлади:

```
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;
```

Келтирилган мисолда talaba структурасининг сон туридаги майдонларига оддий кўринишда қийматлар берилган, сатр туридаги майдонлар учун strcpy функцияси орқали қиймат бериш амалга оширилган.

Структура туридаги объектнинг хотирадан қанча жой эгаллаганлигини sizeof функцияси (оператори) орқали аниқлаш мумкин:

```
int i=sizeof(Talaba);
```

Айрим ҳолларда структура майдонлари ўлчамини битларда аниқлаш орқали эгалланадиган хотирани камайтириш мумкин. Бунинг учун структура майдони қуйидагича эълон қилинади:

<майдон номи> : <ўзгармас ифода>

Бу ерда <майдон номи>- майдон тури ва номи, <ўзгармас ифода>- майдоннинг битлардаги узунлиги. Майдон тури бутун турлар бўлиши керак (int, long, unsigned, char).

Агар фойдаланувчи структуранинг майдони фақат 0 ва 1 қийматини қабул қилишини билса, бу майдон учун бир бит жой ажратиши мумкин (бир байт ёки икки байт ўрнига). Хотирани тежаш эвазига майдон устида амал бажаришда разрядли арифметикани қўллаш зарур бўлади.

Мисол учун сана-вақт билан боғлиқ структурани яратишнинг иккита вариантини кўрайлик. Структура йил, ой, кун, соат, минут ва секунд майдонларидан иборат бўлсин ва уни қуйидагича аниқлаш мумкин:

```
struct Sana_vaqt
{
 unsigned short Yil;
 unsigned short Oy;
 unsigned short Kun;
 unsigned short Soat;
 unsigned short Minut;
 unsigned short Sekund;
};
```

Бундай аниқлашда Sana_vaqt структураси хотирада 6 майдон*2 байт=12 байт жой эгаллайди. Агар эътибор берилса структурада ортиқча жой эгалланган ҳолатлар мавжуд. Масалан, йил учун қиймати 0 сонидан 99 сонигача қиймат билан аниқланиши етарли (масалан, 2008 йилни 8 қиймати билан ифодалаш мумкин). Шунинг учун унга 2 байт эмас, балки 7 бит ажратиш етарли. Худди шундай ой учун 1..12 қийматларини ифодалашга 4 бит жой етарли ва ҳакоза.

Юқорида келтирилган чекловлардан кейин сана-вақт структурасини тежамли вариантини аниқлаш мумкин:

```
struct Sana_vaqt2
{
 unsigned Yil:7;
 unsigned Oy:4;
```

```
 unsigned Kun:5;
 unsigned Soat:6;
 unsigned Minut:6;
 unsigned Sekund:6;
};
```

Бу структура хотирадан 5 байт жой эгаллайди.

Структура функция аргументи сифатида

Структуралар функция аргументи сифатида ишлатилиши мумкин. Бунинг учун функция прототипида структура тури кўрсатилиши керак бўлади. Масалан, талаба ҳақидаги берилганларни ўз ичига олувчи Talaba структураси туридаги берилганларни Talaba_Manzili() функциясига параметр сифатида бериш учун функция прототипи қуйидаги кўринишда бўлиши керак:

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

Функцияга структурани аргумент сифатида узатишга мисол сифатидаги программанинг матни:

```
#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;
}
```

Программа бош функциясида talaba структураси аниқланиб, унинг майдонларига қийматлар берилади. Кейин talaba структураси Talaba_Manzili() функциясига аргумент сифатида узатилади. Программа ишлаши натижасида экранга қуйидаги маълумотлар чоп этилади.

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

Структуралар массиви

Ўз-ўзидан маълумки, структура туридаги ягона берилган билан ечиш мумкин бўлган масалалар доираси жуда тор ва аксарият ҳолатларда, қўйилган масала структуралар мажмуасини ишлатишни талаб қилади. Бу турдаги масалаларга берилганлар базасини қайта ишлаш масалалари деб қараш мумкин.

Структуралар массивини эълон қилиш худди стандарт массивларни эълон қилишдек, фарқи массив тури ўрнида фойдаланувчи томонидан аниқланган структура турининг номи ёзилади. Масалан, талабалар ҳақидаги берилганларни ўз ичига олган массив яратиш эълони қуйидагича бўлади:

```
const int n=25;
Talaba talabalar[n];
```

Структуралар массивининг элементларига мурожаат одатдаги массив элементларига мурожаат усуллари орқали, ҳар бир элементнинг майдонларига мурожаат эса ‘.’ орқали амалга оширилади.

Қуйидаги программада гуруҳидаги ҳар бир талаба ҳақидаги берилганларни клавиатурадан киритиш ва гуруҳ талабаларини фамилия, исми ва шарифини чоп қилинади.

```
#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);
 }
}
```

Структураларга кўрсаткич

Структура элементларига кўрсаткичлар орқали мурожаат қилиш мумкин. Бунинг учун структурага кўрсаткич ўзгарувчиси эълон қилиниши керак. Масалан, юқорида келтирилган мисолда Talaba структурасига кўрсаткич қуйидагича эълон қилинади:

```
Talaba * k_talaba;
```

Кўрсаткич орқали аниқланган структура элементларига мурожаат «.» билан эмас, балки «->» воситасида амалга оширилади:

```
cout<<k_talaba ->FISh;
```

Структураларни кўрсаткич ва адресни олиш (&) воситасида функция аргументи сифатида узатиш мумкин. Қуйида келтирилган программа бўлагида структурани Talaba_Kiritish() функциясига кўрсаткич орқали, Talabalar_FISh() функциясига эса адресни олиш воситасида узатишга мисол келтирилган.

```
...
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;
  ...
 }
}
```

Шунга эътибор бериш керакки, динамик равишда ҳосил қилинган структуралар массиви элементи бўлган структуранинг майдонига мурожаатда «*» белгиси қўлланилмайди.

Масала. Футбол жамоалари ҳақидаги маълумотлар - жамоа номи, айни пайтдаги ютуқлар, дуранг ва мағлубиятлар сонлари, ҳамда рақиб дарвозасига киритилган ва ўз дарвозасидан ўтказиб юборилган тўплар сонлари билан берилган. Футбол жамоаларининг турнир

жадвали чоп қилинсин. Жамоаларни жадвалда тартиблашда қуйидаги қоидаларга амал қилинсин:

1) жамоалар тўплаган очколарини камайиши бўйича тартибланиши керак;

2) агар жамоалар тўплаган очколари тенг бўлса, улардан нисбатан кўп ғалабага эришган жамоа жадвалда юқори ўринни эгаллайди;

3) агар иккита жамоанинг тўплаган очколари ва ғалабалар сони тенг бўлса, улардан нисбатан кўп тўп киритган жамоа жадвалда юқори ўринни эгаллайди.

Жамоа ҳақидаги берилганлар структура кўринишида, жадвал эса структура массиви сифати аниқланади:

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

Бу ерда Uyin майдони Yutuq, Durang ва Maglub майдонлар йиғиндиси, жамоа тўплаган очколар - Ochko=3*Yutuq+1*Durang кўринишида аниқланади. Жамоалар массиви Ochko, Yutuq ва Urgan_tup майдонлари бўйича тартибланади.

Программа матни:

```
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-жамоанинг очкоси (j+1)- жамоа очкосидан катта
   // бўлса, такрорлашнинг кейинги қадамига ўтилсин.
   if(jm[j].Ochko>jm[j+1].Ochko) continue;
   //j ва (j+1)-жамоаларнинг очколари тенг ва j-жамоа
   // ютуқлари (j+1)- жамоа ютуқларидан кўп бўлса,
   // такрорлашнинг кейинги қадамига ўтилсин.
   if(jm[j].Ochko==jm[j+1].Ochko &&
      jm[j].Yutuq>jm[j+1].Yutuq) continue;
   //j ва (j+1)-жамоаларнинг очколари ва ютуқлар сони
   // тенг ва j-жамоа урган тўплар сони (j+1)- жамоа
```

```
      //урган тўплардан кўп бўлса, такрорлашнинг кейинги
      // қадамига ўтилсин.
      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;
      //юқоридаги шартларнинг бирортаси ҳам бажарилмаса,
      //j ва (j+1)-жамоалар ўринлари алмаштирилсин.
      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;
  }
```

Программа бош функция ва қуйидаги вазифаларни бажарувчи тўртта функциядан ташкил топган:

1) Jamoa * Jamoalar_Jadvali()- жамоалар ҳақидаги берилганларни сақлайдиган Jamoa структураларидан ташкил топган динамик массив яратади ва унга оқимдан ҳар бир жамоа берилганларни ўқиб жойлаштиради. Ҳосил бўлган массивга кўрсаткични функция натижаси сифатида қайтаради;

2) Jamoa*Jadvalni_Tartiblash(Jamoa*jm) - аргумент орқали кўрсатилган массивни масала шарти бўйича тартиблайди ва шу массивга кўрсаткични қайтаради;

3) void Utkazish(Jamoa & jamoa1, Jamoa & jamoa2) - jamoa2 структурасидаги майдонларни jamoa1 структурасига ўтказади. Бу функция Jadvalni_Tartiblash() функциясидан массивдаги иккита структурани ўзаро ўринларини алмаштириш учун чақирилади;

4) void Jadavlni_Chop_Qilish(const Jamoa *jm) - аргументда берилган массивни турнир жадвали қолипида чоп қилади.

Учта жамоа ҳақида маълумот берилганда программа ишлашининг натижаси қуйидагича бўлиши мумкин:

```
   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  |
-----------------------------------------
```

Динамик структуралар

Берилганлар устида ишлашда уларнинг миқдори қанча бўлиши ва уларга хотирадан қанча жой ажратиш кераклиги олдиндан номаълум бўлиши мумкин. Программа ишлаш пайтида берилганлар учун зарурат бўйича хотирадан жой ажратиш ва уларни кўрсаткичлар билан боғлаш орқали ягона структура ҳосил қилиш жараёни *хотиранинг динамик тақсимоти* дейилади. Бу усулда ҳосил бўлган берилганлар мажмуасига *берилганларнинг динамик структураси* дейилади, чунки уларнинг ўлчами программа бажарилишида ўзгариб туради. Программалашда динамик структуралардан чизиқли рўйхатлар (занжирлар), стеклар, навбатлар ва бинар дарахтлар нисбатан кўп ишлатилади. Улар бир - биридан элементлар ўртасидаги боғланишлари ва улар устида бажариладиган амаллари билан фарқланади.

Программа ишлашида структурага янги элементлар қўшилиши ёки ўчирилиши мумкин.

Ҳар қандай берилганларнинг динамик структураси майдонлардан ташкил топади ва уларнинг айримлари қўшни элементлар билан боғланиш учун хизмат қилади.

Масала. Нолдан фарқли бутун сонлардан иборат чизиқли рўйхат яратилсин ва ундан кўрсатилган сонга тенг элемент ўчирилсин.

Бутун сонларнинг чизиқли рўйхат кўринишидаги динамик структураси қуйидаги майдонлардан ташкил топади:

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

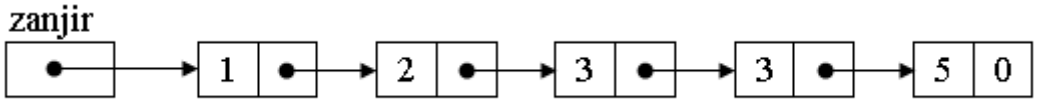
Программа матни:

```
#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)             // рўйхат бўш эмас
 {
  Zanjir * temp=z;
  while(temp->keyingi)
   temp=temp->keyingi;// рўйхатнинг охирги элементини
                      // топиш
  temp->keyingi=yangi;// янги элементни рўйхат
                      // охирига қўшиш
 }
 else z=yangi;     // рўйхат бўш
 return z;         // рўйхат боши адресини қайтариш
}
Zanjir * Element_Uchirish(Zanjir * z, int del_elem)
{
 if(z)
 {
  Zanjir * temp=z;
  Zanjir * oldingi=0;  // жорий элементдан олдинги
                       // элементга кўрсаткич
  while (temp)
  {
   if (temp->element==del_elem)
```

```
   {
    if(oldingi)    //ўчириладиган элемент биринчи эмас
    {
      // ўчириладиган элементдан олдинги элементни
      // кейинги элементга улаш
     oldingi->keyingi = temp->keyingi;
      delete temp;      // элементни ўчириш
      temp=oldingi->keyingi;
    }
    else
    {
      // ўчириладиган элемент рўйхат бошида
     z=z->keyingi;
     delete temp;
     temp=z;
    }
   }
   else // элемент ўчириладиган сонга тенг эмас
   {
    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)
  {
   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;
}
```

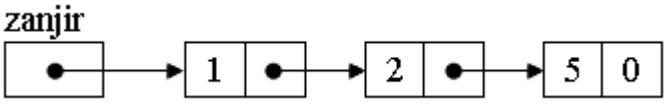
Программанинг бош функциясида чизиқли рўйхат ҳосил қилиш учун Zanjir туридаги zanjir ўзгарувчиси аниқланган бўлиб, унга бўш кўрсаткич қиймати 0 берилган (унинг эквиваленти - NULL). Кейин такрорлаш оператори танасида клавиатурадан бутун сон ўқилади ва Element_Joylash() функциясини чақириш орқали бу сон рўйхатга охирига қўшилади. Функция янги ҳосил бўлган рўйхат бошининг адресини яна zanjir ўзгарувчисига қайтаради. Агар клавиатурадан 0 сони киритилса рўйхатни ҳосил қилиш жараёни тугайди. Фараз қилайлик қуйидаги сонлар кетма-кетлиги киритилган бўлсин: 1,2,3,3,5,0. У ҳолда ҳосил бўлган рўйхат қуйидаги кўринишда бўлади (10.1-расм):



10.1-расм. Бешта сондан ташкил топган чизиқли рўйхат

Ҳосил бўлган рўйхатни кўриш учун Zanjir_Ekranga() функцияси чақирилади ва экранда рўйхат элементлари чоп этилади. Рўйхат устида амал сифатида берилган сон билан устма-уст тушадиган элементларни ўчириш масаласи қаралган. Бунинг учун ўчириладиган сон del_element ўзгарувчига ўқилади ва у Element_Uchirish() функцияси чақирилишида аргумент сифатида узатилади. Функция бу сон билан устма-уст тушадиган рўйхат элементларини ўчиради (агар бундай элемент мавжуд бўлса) ва ўзгарган рўйхат бошининг адресини zanjir ўзгарувчисига қайтариб беради. Масалан, рўйхатдан 3 сони

билан устма-уст тушадиган элементлар ўчирилгандан кейин у қуйидаги кўринишга эга бўлади (10.2-расм):



10.2-расм. Рўйхатдан 3 сонини ўчирилгандан кейинги кўриниш

Ўзгарган рўйхат элементлари экранга чоп этилади. Программа охирида, Zanjirni_Uchirish() функциясини чақириш орқали рўйхат учун динамик равишда ажратилган хотира бўшатилади (гарчи бу ишнинг программа тугаши пайтида бажарилишининг маъноси йўқ).

Динамик структураларда ўзгартиришлар (рўйхатга элемент қўшиш ёки ўчириш) нисбатан кам амалларда бажарилиши, улар воситасида масалаларни самарали ечишнинг асосларидан бири ҳисобланади.

Бирлашмалар ва улар устида амаллар

Бирлашмалар хотиранинг битта соҳасида (битта адрес бўйича) ҳар хил турдаги бир нечта берилганларни сақлаш имконини беради.

Бирлашма эълони union калит сўзи, ундан кейин идентификатор ва блок ичида ҳар хил турдаги элементлар эълонидан иборат бўлади, масалан:

```
union Birlashma
 {
 int n;
 unsigned long N;
 char Satr[10];
 };
```

Бирлашманинг бу эълонида компилятор томонидан Birlashma учун унинг ичидаги энг кўп жой эгалловчи элементнинг - Satr сатрининг ўлчамида, яъни 10 байт жой ажратилади. Вақтнинг ҳар бир моментида бирлашмада, эълон қилинган майдонларнинг фақат биттасининг туридаги берилган мавжуд деб ҳисобланади. Юқоридаги мисолда Birlashma устида амал бажарилишида унинг учун ажратилган хотирада ёки int туридаги n ёки unsigned long туридаги N ёки Satr сатр қиймати жойлашган деб ҳисобланади.

Бирлашма майдонларига худди структура майдонларига мурожаат қилгандек ‘.’ орқали мурожаат қилинади.

Структуралардан фарқли равишда бирлашма эълонида фақат унинг биринчи элементига бошланғич қиймат бериш мумкин:

```
union Birlashma
{
 int n;
 unsigned long N;
 char Satr[10];
}
 birlashma={25};
```

Бу мисолда birlashma бирлашмасининг n майдони бошланғич қиймат олган ҳисобланади.

Бирлашма элементи сифатида структуралар келиши мумкин ва улар одатда берилганни «бўлакларга» ажратиш ёки «бўлаклардан» яхлит берилганни ҳосил қилиш учун хизмат қилади. Мисол учун сўзни байтларга, байтларни тетрадаларга (4 битга) ажратиш ва қайтадан бирлаштириш мумкин.

Қуйида байтни катта ва кичик ярим байтларга ажратишда бирлашма ва структурадан фойдаланилган программани матни келтирилган.

```
#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;
}
```

Программа бош функциясида bcd бирлашмасининг байт ўлчамида bayt майдонига 127 қиймати берилади ва унинг катта ва кичик ярим байтлари чоп этилади.

Программа ишлаши натижасида экранга қуйидаги натижалар чиқади:

```
Katta yarim bayt : 7
Kichik yarim bayt: 15
```

Масала. Ҳақиқий турдаги соннинг компьютер хотирасидаги ички кўринишини чоп қилиш. Ҳақиқий сон float турида деб ҳисобланади ва у хотирада 4 байт жой эгаллайди (1-иловага қаранг).

Қўйилган масалани ечиш учун бирлашма хусусиятдан фойдаланилади, яъни хотиранинг битта адресига ҳақиқий сон ва белгилар массиви жойлаштирилади. Ҳақиқий сон хотирага ўқилиб, белгилар массивининг ҳар бир элементининг (байтининг) иккилик кўриниши чоп этилади.

Программа матни:

```
#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 l0000000=128;
 for(int i=0;i<=bitlar_soni;i++)
 {
  if(blg&l0000000)cout<<'1';
  else cout<<'0';
  blg=blg<<1;
 }
 cout<<' ';
}
```

Программада Son_va_Belgi бирлашмасини эълон қилиш орқали float туридаги x ўзгарувчисини ва float тури форматининг байтлардаги узунлигидаги белгилардан иборат belgi массивини хотиранинг битта жойига жойлашувига эришилади. Бош функцияда бирлашма туридаги son_va_belgi ўзгарувчиси эълон қилинади ва унинг x майдонига клавиатурадан ҳақиқий сон ўқилади. Кейин белгилар массивидаги ҳар бир элементнинг иккилик коди чоп этилади. Иккилик кодни чоп этиш 8 марта байтни 7-разрядидаги сонни чоп этиш ва байт разрядларини биттага чапга суриш орқали амалга оширилади. Шунга эътибор бериш керакки, белгилар массивидаги элементларнинг иккилик

кодларини чоп қилиш ўнгдан чап томонга бажарилган. Бунга сабаб, сон ички форматидаги байтларнинг хотирада «кичик байт - кичик адресда» қоидасига кўра жойлашувидир.

Программага -8.5 сони киритилса, экранда

```
11000001 00001000 00000000 00000000
```

кўринишидаги иккилик сонлари кетма-кетлиги пайдо бўлади.

Фойдаланувчи томонидан аниқланган берилганлар тури

C++ тилида фойдаланувчи томонидан нафақат структура ёки бирлашма турлари, балки айни пайтда мавжуд (аниқланган) турлар асосида янги турларни яратиши мумкин.

Фойдаланувчи томонидан аниқланадиган тур typedef калит сўзи билан бошланади, ундан кейин мавжуд тур кўрсатилади ва идентификатор ёзилади. Охирида ёзилган идентификатор - янги яратилган турнинг номи ҳисобланади. Масалан,

```
typedef unsigned char byte;
```

ифодаси byte деб номланувчи янги турни яратади ва ўз мазмунига кўра unsigned char тури билан эквивалент бўлади. Кейинчалик, программада хотирадан бир байт жой эгаллайдиган ва [0..255] оралиқдаги қийматларни қабул қиладиган byte туридаги ўзгарувчи (ўзгармасларни) эълон қилиш мумкин:

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

Массив кўринишидаги фойдаланувчи томонидан аниқланувчи тур эълони қуйидагича бўлади:

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

Ism туридаги ism ўзгарувчиси эълони - бу 30 белгидан иборат массив (сатр) эълонидир.

Одатда ечилаётган масаланинг предмет соҳаси терминларида ишлаш учун структуралар қайта номланади. Натижада мураккаб тузилишга эга бўлган ва зарур хусусиятларни ўзига жамлаган янги турларни яратишга мувофиқ бўлинади.

Масалан, комплекс сон ҳақидаги маълумотларни ўз ичига олувчи Complex тури қуйидагича аниқланади:

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

Энди комплекс сон эълонини

```
Complex KSon;
```

ёзиш мумкин ва унинг майдонларига мурожаат қилиш мумкин:

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

11-боб. Макрослар

Макросларни аниқлаш ва жойлаштириш

Макрос - бу программа (код) бўлаги бўлиб, кўриниши ва ишлаши худди функциядек. Бироқ у функция эмас. Функциялар ва макрослар ўртасида бир нечта фарқлар мавжуд:

– программа матнида учраган макрос ифодаси ўз аниқланиши (танаси билан) билан препроцессор ишлаш пайтида, яъни программа компиляциясидан олдин алмаштирилади. Шу сабабли макрос функцияни чақириш билан боғлиқ қўшимча вақт сарфини талаб қилмайди;

– макрослардан фойдаланиш программанинг бошланғич коди (матнини) катталашувига олиб келади. Бунга қарама-қарши ҳолда функция коди ягона нусхада бўлади ва у программа кодини қисқаришига олиб келади. Лекин функцияни чақириш учун қўшимча ресурслар сарфланади;

– компилятор макросдаги турлар мослигини текширмайди. Шу сабабли, макросга аргумент жўнатишда турларнинг мослиги ёки аргументлар сонининг тўғри келиши ёки келмаслиги ҳақидаги хатолик хабарлари берилмайди;

– макрос бошланғич кодга программа бўлагини қўйиш воситаси бўлганлиги ва бундай бўлаклар матннинг турли жой-ларига қўйиш мумкинлиги сабабли макрослар билан боғлиқ фиксирланган, ягона адреслар бўлмайди. Шу сабабли макросларда кўрсаткичлар эълон қилиш ёки макрос адресларини ишлатиш имконияти йўқ.

Макросларни аниқлаш учун #define директивасидан фойдаланилади. Функцияга ўхшаб макрослар ҳам параметрларга эга бўлиши мумкин. Мисол учун иккита сонни кўпайтмасини ҳисобловчи макрос қуйидагича аниқланади:

```
#include <iostream.h>
#define KUPAYTMA(x,y)((x)+(y))
int main()
{
 int a=2, b=3;
 c=KUPAYTMA(a,b);
 cout<<c;
 return 0;
```

```
}
```

Мисолдан кўриниб турибдики, ташқи кўриниши бўйича макрослардан фойдаланиш функциялардан фойдаланишга ўхшаш. Шунинг учун уларни айрим ҳолларда уларга *псевдофункциялар* деб аташади. Макрослар аниқланишининг яна бир ўзига хос томони шундаки, C++ тилида уларнинг номларини катта ҳарфлар билан ёзишга келишилган.

Юқоридаги мисолнинг ўзига хос кўринишидан бири бу макрос параметрларини қавс ичида ёзилишидир. Акс ҳолда макрос аниқланишини (танасини) матнга қўйишда мазмунан хатолик юзага келиши мумкин. Масалан,

```
#define KVADRAT(x) x*x
```

Программа матнида ушбу макрос ишлатилган сатр мавжуд бўлсин:

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

у ҳолда, макрос аниқланишини матнга қўйиш натижасида программа матнида юқоридаги сатр қуйидаги кўринишга келади:

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

Лекин, программада макросни ишлатиш

```
int y=KVADRAT(x+1);
```

кўринишида бўлса, макрос аниқланишини матнга қўйиш натижасида ушбу сатр

```
int y=x+1*x+1;
```

кўрсатмаси билан алмаштириладики, бу албатта кутилган алмаштириш эмас. Шу сабабли, макрос аниқланишида умумий қоида сифатида параметрларни қавсга олиш тавсия этилади:

```
#define KVADRAT(x)(x)*(x)
```

Агар макрос чақирилишида тургa келтириш операторидан фойдаланган ҳолат бўлса, макрос танасини тўлиқлигича қавсга олиш талаб қилинади. Мисол учун программа матнида макросга мурожаат қуйидагича бўлсин:

```
double x=(double)KVADRAT(x+1);
```

Бу ҳолда макрос аниқланиши

```
#define KVADRAT(x)((x)*(x))
```

кўриниши тўғри ҳисобланади.

Макрос аниқланишида охирги эслатма сифатида шуни қайд этиш керакки, ортиқча пробеллар макросдан фойдаланишда хатоликларга олиб келиши мумкин. Масалан

```
#define CHOP_QILISH (x)cout<<x
```

макрос аниқланишида макрос номи CHOP_QILISH ва параметрлар рўйхати (x) ўртасида ортиқча пробел қўйилган. Препроцессор бу макросни параметрсиз макрос деб қабул қилади, ҳамда “(x)cout<<x” сатр остини макрос танаси деб ҳисоблайди ва макрос алмаштиришларда шу сатрни программа матнига қўйилада. Натижада компиляция хатоси рўй беради. Хатони тузатиш учун макрос номи ва параметрлар рўйхати ўртасидаги пробелни олиб ташлаш етарли:

```
#define CHOP_QILISH(x)cout<<x
```

Агар макрос аниқланиши битта сатрга сиғмаса, шу сатр охирига ‘\’ белгисини қўйиш орқали кейинги сатрда давом эттириш мумкин:

```
#define BURCHAK3(a,b,c)(unsigned int)a+(unsigned int)b\
  >(unsigned int)c &&(unsigned int)a+(unsigned int)c>\
  (unsigned int)b &&(unsigned int)b+(unsigned int)c>\
  (unsigned int)a
```

Макрос аниқланишида бошқа макрослар иштирок этиши мумкин. Қуйидаги мисолда ичма-ич жойлашган макрос аниқланиши кўрсатилган.

```
#define PI 3.14159
#define KVADRAT(x) ((x)*(x))
#define AYLANA_YUZI(r)(PI* KVADRAT(r))
```

Фойдаланишга зарурати қолмаган макросни программа матнининг ихтиёрий жойида #undef директиваси билан бекор қилиш мумкин, яъни шу сатрдан кейин макрос препроцессор учун ноаниқ ҳисобланади. Қуйида айлана юзасини ҳисоблайдиган программа матни келтирилган.

```
#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;
}
```

Программа компиляциясида “c1=AYLANA_YUZI(r1);” сатр нормал қайти ишланган ҳолда “c2=AYLANA_YUZI(r2);” сатри учун AYLANA_YUZI функцияси аниқланмаганлиги ҳақида хатолик хабари чоп этилади.

Макросларда ишлатиладиган амаллар

Макрослар ишлатилиши мумкин бўлган иккита амал мавжуд: ‘#’- сатрни жойлаштириш ва ” ##” - сатрни улаш амаллари.

Агар макрос параметри олдида ‘#’- сатрни жойлаштириш амали қўйилган бўлса, макрос аниқланишини матнга қўйиш пайтида шу ўринга мос аргументнинг (ўзгарувчининг) номи қўйилади. Буни қуйидаги мисолда кўриш мумкин:

```
#include <iostream.h>
#define UZG_NOMI(uzg) cout<<#uzg<<'='<<uzg;
int main()
{
 int x=10;
 UZG_NOMI(x);
 return 0;
}
```

Программа ишлаши натижасида экранда

```
x=10
```

сатри пайдо бўлади.

Сатр улаш амали иккита сатрни биттага бирлаштириш учун хизмат қилади. Сатрларни бирлаштиришдан олдин уларни ажратиб турган пробеллар ўчирилади. Агар ҳосил бўлган сатр номидаги макрос мавжуд бўлса, препроцессор шу макрос танасини чақирув бўлган жойга жойлаштиради.

Мисол учун,

```
#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;
```

```
}
```

программмаси препроцессор томонидан қайта ишлангандан кейин унинг оралиқ матни куйидаги кўринишда бўлади:

```
int main(int argc, char* argv[])
{
 int x=10;
 cout<<"MACRO_1";
 cin>>x;
 return 0;
}
```

Сатрларни улаш амалидан янги ўзгарувчиларни ҳосил қилиш учун фойдаланиш мумкин.

```
#define UZG_ELONI(i) int var ## i
...
UZG_ELONI(1);
...
```

Юқоридаги мисолда макрос ўз аниқланиши билан алмаштириш натижасида “UZG_ELONI(1);” сатри ўрнида

```
int var1;
```

кўрсатмаси пайдо бўлади.

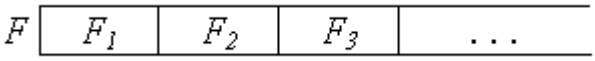
12-боб. Ўқиш - ёзиш функциялари

Файл тушунчаси

С++ тилидаги стандарт ва фойдаланувчи томонидан аниқланган турларнинг муҳим хусусияти шундан иборатки, уларнинг олдиндан аниқланган миқдордаги чекли элементлардан иборатлигидир. Ҳатто берилганлар динамик аниқланганда ҳам, оператив хотиранинг (уюмнинг) амалда чекланганлиги сабабли, бу берилганлар миқдори юқоридан чегараланган элементлардан иборат бўлади. Айрим бир тадбиқий масалалар учун олдиндан берилганнинг компонентаси сонини аниқлаш имкони йўқ. Улар масалани ечиш жараёнида аниқланади ва етарлича катта ҳажмда бўлиши мумкин. Иккинчи томондан, программада эълон қилинган ўзгарувчиларнинг қийматлари сифатида аниқланган берилганлар фақат программа ишлаш пайтидагина мавжуд бўлади ва программа ўз ишини тугатгандан кейин йўқолиб кетади. Агар программа янгидан ишга туширилса, бу берилганларни янгидан ҳосил қилиш зарур бўлади. Аксарият тадбиқий масалалар эса берилганларни доимий равишда сақлаб туришни талаб қилади. Масалан, корхона ходимлариннинг ойлик маошини ҳисобловчи программада ходимлар рўйхатини, штат ставкалари ва ходимлар томонидан олинган маошлар ҳақидаги маълумотларни доимий равишда сақлаб туриш зарур. Бу талабларга файл туридаги объектлар (ўзгарувчилар) жавоб беради.

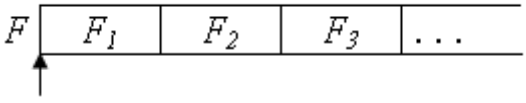
Файл - бу бир хил турдаги қийматлар жойлашган ташқи хотирадаги номланган соҳадир.

Файлни, бошида кетма-кет равишда жойлашган ёзувлар (масалан, мусиқа) билан тўлдирилган ва охири бўш бўлган етарлича узун магнит тасмасига ўхшатиш мумкин.



12.1-расм. Файл тасвири

12.1-расмда F- файл номи, F_1,F_2, F_3 - файл элементлари (компоненталари). Худди янги мусиқани тасма охирига қўшиш мумкин бўлгандек, янги ёзувлар файл охирига кўшилиши мумкин.



12.2-расм. Файл кўрсаткичи

Яна бир муҳим тушунчалардан бири файл кўрсаткичи тушунчасидир. *Файл кўрсаткичи* - айни пайтда файлдан ўқилаётган ёки унга ёзилаётган жой (ёзув ўрнини) кўрсатиб туради, яъни файл кўрсаткичи кўрсатиб турган жойдан битта ёзувни ўқиш ёки шу жойга янги ёзувни жойлаштириш мумкин. 12.2-расмда файл кўрсаткичи файл бошини кўрсатмоқда.

Файл ёзувларига мурожаат кетма-кет равишда амалга оширилади: n- ёзувга мурожаат килиш учун n-1 ёзувни ўқиш зарур бўлади. Шуни таъкидлаб ўтиш зарурки, файлдан ёзувларни ўқиш жараёни қисман «автоматлашган», унда i - ёзувни ўқилгандан кейин, кўрсаткич навбатдаги i+1 ёзув бошига кўрсатиб туради ва шу тарзда ўқишни давом эттириш мумкин (массивлардагидек индексни ошириш шарт эмас). Файл - бу берилганларни сақлаш жойидир ва шу сабабли унинг ёзувлари устида тўғридан-тўғри амал бажариб бўлмайди. Файл ёзуви устида амал бажариш учун ёзув қиймати оператив хотирага мос турдаги ўзгарувчига ўқилиши керак. Кейинчалик, зарур амаллар шу ўзгарувчи устида бажарилади ва керак бўлса натижалар яна файлга ёзилиши мумкин.

Операцион система нуқтаи-назаридан файл ҳисобланган ҳар қандай файл C++ тили учун *моддий файл* ҳисобланади. MS DOS учун моддий файллар <файл номи>.<файл кенгайтмаси> кўринишидаги «8.3» форматидаги сатр (ном) орқали берилади. Файл номлари сатр ўзгармаслар ёки сатр ўзгарувчиларида берилиши мумкин. MS DOS қоидаларига кўра файл номи тўлиқ бўлиши, яъни файл номининг бошида адрес қисми бўлиши мумкин: ”C:\\USER\\Misol.cpp”, “A:\\matn.txt”.

C++ тилида *мантиқий файл* тушунчаси бўлиб, у файл туридаги ўзгарувчини англатади. Файл туридаги ўзгарувчиларга бошқа турдаги ўзгарувчилар каби қиймат бериш оператори орқали қиймат бериб бўлмайди. Бошқача айтганда файл туридаги ўзгарувчилар устида ҳеч қандай амал аниқланмаган. Улар устида бажариладиган барча амаллар функциялар воситасида бажарилади.

Файллар билан ишлаш қуйидаги босқичларни ўз ичига олади:

– файл ўзгарувчиси албатта дискдаги файл билан боғланади;

– файл очилади;

– файл устида ёзиш ёки ўқиш амаллари бажарилади;

– файл ёпилади;

– файл номини ўзгартириш ёки файлни дискдан ўчириш амалларини бажарилиши мумкин.

Матн ва бинар файллар

C++ тили C тилидан ўқиш-ёзиш амалини бажарувчи стандарт функциялар кутубхонасини ворислик бўйича олган. Бу функциялар <stdio.h> сарлавҳа файлида эълон қилинган. Ўқиш-ёзиш амаллари файллар билан бажарилади. Файл матн ёки бинар (иккилик) бўлиши мумкин.

Матн файл - ASCII кодидаги белгилар билан берилганлар мажмуаси. Белгилар кетма-кетлиги сатрларга бўлинган бўлади ва сатрнинг тугаш аломати сифатида CR (кареткани қайтариш ёки ‘\r’) LF (сатрни ўтказиш ёки ‘\n’) белгилар жуфтлиги ҳисобланади. Матн файлдан берилганларни ўқишда бу белгилар жуфтлиги битта CR белгиси билан алмаштирилади ва аксинча, ёзишда CR белгиси иккита CR ва LF белгиларига алмаштирилади. Файл охири #26 (^Z) белгиси билан белгиланади.

Матн файлга бошқача таъриф бериш ҳам мумкин. Агар файлни матн таҳририда экранга чиқариш ва ўқиш мумкин бўлса, бу матн файл. Клавиатура ҳам компьютерга фақат матнларни жўнатади. Бошқача айтганда программа томонидан экранга чиқариладиган барча маълумотларни stdout номидаги матн файлига чиқарилмоқда деб қараш мумкин. Худди шундай клавиатурадан ўқилаётган ҳар қандай берилганларни матн файлидан ўқилмоқда деб ҳисобланади.

Матн файлларининг компоненталари *сатрлар* деб номланади. Сатрлар узлуксиз жойлашиб, турли узунликда ва бўш бўлиши мумкин. Фараз қилайлик, T матн файли 4 сатрдан иборат бўлсин:

1- satr#13#10	2- satr uzunroq #13#10	#13#10	4-satr#13#10#26

12.3-расм. Тўртта сатрдан ташкил топган матн файли

Матнни экранга чиқаришда сатр охиридаги #13#10 бошқарув белгилари жуфтлиги курсорни кейинги қаторга туширади ва уни сатр бошига олиб келади. Бу матн файл экранга чоп этилса, унинг кўриниши қуйидагича бўлади:

```
1- satr[13][10]
2- satr uzunroq[13][10]
[13][10]
4- satr[13][10]
[26]
```

Матндаги [n] - n- кодли бошқарув белгисини билдиради. Одатда матн таҳрирлари бу белгиларни кўрсатмайди.

Бинар файллар - бу оддийгина байтлар кетма-кетлиги. Одатда бинар файллардан берилганларни фойдаланувчи томонидан бевосита «кўриш» зарур бўлмаган ҳолларда ишлатилади. Бинар файллардан

ўқиш-ёзишда байтлар устида ҳеч қандай конвертация амаллари бажарилмайди.

Ўқиш-ёзиш оқимлари. Стандарт оқимлар

Оқим тушунчаси берилганларни файлга ўқиш-ёзишда уларни белгилар кетма-кетлиги ёки оқими кўринишида тасаввур қилишдан келиб чиққан. Оқим устида қуйидаги амалларни бажариш мумкин:

– оқимдан берилганлар блокини оператив хотирага ўқиш;

– оператив хотирадаги берилганлар блокини оқимга чиқариш;

– оқимдаги берилганлар блокини янгилаш;

– оқимдан ёзувни ўқиш;

– оқимга ёзувни чиқариш.

Оқим билан ишлайдиган барча функциялар буферли, форматлашган ёки форматлашмаган ўқиш-ёзишни таъминлайди.

Программа ишга тушганда ўқиш-ёзишнинг қуйидаги стандарт оқимлар очилади:

stdin - ўқишнинг стандарт воситаси;

stdout - ёзишнинг стандарт воситаси;

stderr - хатолик ҳақида хабар беришнинг стандарт воситаси;

stdprn - қоғозга чоп қилишнинг стандарт воситаси;

stdaux - стандарт ёрдамчи қурилма.

Келишув бўйича stdin - фойдаланувчи клавиатураси, stdout ва stderr - терминал (экран), stdprn - принтер билан, ҳамда stdaux - компьютер ёрдамчи портларига боғланган ҳисобланади. Берилганларни ўқиш-ёзишда stderr ва stdaux оқимидан бошқа оқимлар буферланади, яъни белгилар кетма-кетлиги оператив хотиранинг буфер деб номланувчи соҳасида вақтинча жамланади. Масалан, белгиларни ташқи қурилмага чиқаришда белгилар кетма-кетлиги буферда жамланади ва буфер тўлгандан кейингина ташқи қурилмага чиқарилади.

Ҳозирдаги операцион системаларда клавиатура ва дисплейлар матн файллари сифатида қаралади. Ҳақиқатдан ҳам берилганларни клавиатурадан программага киритиш (ўқиш) мумкин, экранга эса чиқариш (ёзиш) мумкин. Программа ишга тушганда стандарт ўқиш ва ёзиш оқимлари ўрнига матн файлларни тайинлаш орқали бу оқимларни қайта аниқлаш мумкин. Бу ҳолатни *ўқишни (ёзишни) қайта адреслаш рўй берди* дейилади. Ўқиш учун қайта адреслашда ‘<’ белгисидан, ёзиш учун эса ‘>’ белгисидан фойдаланилади. Мисол учун gauss.exe бажарилувчи программа берилганларни ўқишни клавиатурадан эмас, балки massiv.txt файлидан амалга ошириш зарур

бўлса, у буйруқ сатрида қуйидаги кўринишда юкланиши зарур бўлади:

```
gauss.exe < massiv.txt
```

Агар программа натижасини natija.txt файлига чиқариш зарур бўлса

```
gauss.exe > natija.txt
```

сатри ёзилади.

Ва ниҳоят, агар берилганларни massiv.txt файлидан ўқиш ва натижани natija.txt файлига ёзиш учун

```
gauss.exe < massiv.txt > natija.txt
```

буйруқ сатри терилади.

Умуман олганда, бир программанинг чиқиш оқимини иккинчи программанинг кириш оқими билан боғлаш мумкин. Буни *конвейрли жўнатиш* дейилади. Агар иккита junat.exe программаси qabul.exe программасига берилганларни жўнатиши керак бўлса, у ҳолда улар ўртасига ‘|’ белги қўйиб ёзилади:

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

Бу кўринишдаги программалар ўртасидаги конвейрли жўнатишни операцион системанинг ўзи таъминлайди.

Белгиларни ўқиш-ёзиш функциялари

Белгиларни ўқиш-ёзиш функциялари макрос кўринишида амалга оширилган.

getc() макроси тайинланган оқимдан навбатдаги белгини қайтаради ва кириш оқими кўрсаткичини кейинги белгини ўқишга мослаган ҳолда оширади. Агар ўқиш муваффақиятли бўлса getc() функцияси ишорасиз int кўринишидаги қийматни, акс ҳолда EOF қайтаради. Ушбу функция прототипи қуйидагича:

```
int getc(FILE * stream)
```

EOF идентификатор макроси

```
#define EOF(-1)
```

кўринишида аниқланган ва ўқиш-ёзиш амалларида файл охирини белгилаш учун хизмат қилади. EOF қиймати ишорали char турида деб ҳисобланади. Шу сабабли ўқиш-ёзиш жараёнида unsigned char туридаги белгилар ишлатилса, EOF макросини ишлатиб бўлмайди.

Навбатдаги мисол getc() макросини ишлатишни намоён қилади.

```
#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() макроси аксарият ҳолатларда stdin оқими билан ишлатилганлиги сабабли, унинг getc(stdin) кўринишига эквивалент бўлган int getchar() макроси аниқланган. Юқоридаги мисолда «ch=getc(stdin);» қаторини «ch=getchar();» қатори билан алмаштириш мумкин.

Белгини оқимга чиқариш учун putc() макроси аниқланган ва унинг прототипи

```
int putc(int c, FILE*stream)
```

кўринишида аниқланган. putc() функцияси stream номи билан берилган оқимга c белгини чиқаради. Функция қайтарувчи қиймати сифатида int турига айлантирилган c белги бўлади. Агар белгини чиқаришда хатолик рўй берса EOF қайтарилади.

putc() функциясини стандарт stdout оқими билан боғланган ҳолати - putc(c,strout) учун putchar(c) макроси аниқланган.

Сатрларни ўқиш - ёзиш функциялари

Оқимдан сатрни ўқишга мўлжалланган gets() функциясининг прототипи

```
char * gets(char *s);
```

кўринишида аниқланган. gets() функцияси стандарт оқимдан сатрни ўқийди ва уни s ўзгарувчисига жойлаштиради. Жойлаштириш пайтида оқимдаги ‘\n’ белгиси ‘\0’ белгиси билан алмаштирилади. Бу функцияни ишлатишда ўқилаётган сатрнинг узунлиги s сатр учун ажратилган жой узунлигидан ошиб кетмаслигини назорат қилиш керак бўлади.

puts() функцияси

```
int puts(const char *s);
```

кўринишида бўлиб, у стандрат оқимга аргументда кўрсатилган сатрни чиқаради. Бунда сатр охирига янги сатрга ўтиш белгиси ‘\n’ қўшилади. Агар сатрни оқимга чиқариш муваффақиятли бўлса puts() функцияси манфий бўлмаган сонни, акс ҳолда EOF қайтаради.

Сатрни ўқиш-ёзиш функцияларини ишлатишга мисол тариқасида қўйидаги программани келтириш мумкин.

```
#include <stdio.h>
int main()
{
 char *s;
 puts(”Satrni kiriting: “);
 gets(s);
 puts(”Kiritilgan satr: ”);
 puts(s);
 return 0;
}
```

Форматли ўқиш ва ёзиш функциялари

Форматли ўқиш ва ёзиш функциялари - scanf() ва printf() функциялари С тилидан ворислик билан олинган. Бу функцияларни ишлатиш учун «stdio.h» сарлавҳа файлини программага қўшиш керак бўлади.

Форматли ўқиш функцияси scanf() қуйидаги прототипга эга:

int scanf(const char * <формат>[<адрес>,...])

Бу функция стандарт оқимдан берилганларни форматли ўқишни амалга оширади. Функция, кириш оқимидаги майдонлар кетма-кетлиги кўринишидаги белгиларни бирма-бир ўқийди ва ҳар бир майдонни <формат> сатрида келтирилган формат аниқлаштирувчисига мос равишда форматлайди. Оқимдаги ҳар бир майдонга формат аниқлаштирувчиси ва натижа жойлашадиган ўзгарувчининг адреси бўлиши шарт. Бошқача айтганда, оқимдаги майдон (ажратилган белгилар кетма-кетлиги) кўрсатилган форматдаги қийматга аксланти-рилади ва ўзгарувчи билан номланган хотира бўлагига жойлашти-рилади (сақланади). Функция оқимдан берилганларни ўқиш жараё-нини «тўлдирувчи белгини» учратганда ёки оқим тугаши натижасида тўхтатиши мумкин. Оқимдан берилганларни ўқиш муваффақиятли бўлса, функция муваффақиятли айлантирилган ва хотирага сақланган майдонлар сонини қайтаради. Агар ҳеч бир майдонни сақлаш имкони бўлмаган бўлса, функция 0 қийматини қайтаради. Оқим охирига келиб қолганда (файл ёки сатр охирига) ўқишга ҳаракат бўлса, функция EOF қийматини қайтаради.

Форматлаш сатри - <формат> белгилар сатри бўлиб, у учта тоифага бўлинади:

– тўлдирувчи белгилар;
– тўлдирувчи белгилардан фарқли белгилар;
– формат аниқлаштирувчилари.

Тўлдирувчи-белгилар – бу пробел, ‘\t’, ‘\n’ белгилари. Бу белги-лар форматлаш сатридан ўқилади, лекин сақланмайди.

Тўлдирувчи белгилардан фарқли белгилар – бу қолган барча ASCII белгилари, ‘%’ белгисилан ташқари. Бу белгилар форматлаш сатридан ўқилади, лекин сақланмайди.

12.1–жадвал. Формат аниқлаштирувчилари ва уларнинг вазифаси

Компонента	**Бўлиши шарт ёки йўқ**	**Вазифаси**
[*]	Йўқ	Навбатдаги кўриб чиқилаётган майдон қийматини ўзгарувчига ўзлаштирмаслик белгиси. Кириш оқимидаги майдон кўриб чиқилади, лекин ўзгарувчида сақланмайди.
[<кенглик>]	Йўқ	Майдон кенглигини аниқлаштирувчиси. Ўқиладиган белгиларнинг максимал сонини аниқлайди. Агар оқимда тўлдирув–чи белги ёки алмаштирилмайдиган белги учраси функция нисбатан кам сондаги белгиларни ўқиши мумкин.
[F|N]	Йўқ	Ўзгарувчи кўрсаткичининг (адресининг) модификатори: F– far pointer; N- near pointer
[h|l|L]	Йўқ	Аргумент турининг модификатори. <тур белгиси> билан аниқланган ўзгарувчининг қисқа (short - h) ёки узун (long –l,L) кўри–нишини аниқлайди.
<тур белгиси>	Ҳа	Оқимдаги белгиларни алмаштириладиган тур белгиси

Формат аниқлаштирувчилари – оқим майдонидаги белгиларни кўриб чиқиш, ўқиш ва адреси билан берилган ўзгарувчилар турига мос равишда алмаштириш жараёнини бошқаради. Ҳар бир формат аниқлаштирувчисига битта ўзгарувчи адреси мос келиши керак. Агар формат аниқлаштирувчилари сони ўзгарувчилардан кўп бўлса, натижа нима бўлишини олдиндан айтиб бўлмайди. Акс ҳолда, яъни ўзгарувчилар сони кўп бўлса, ортиқча ўзгарувчилар инобатга олинмайди.

Формат аниқлаштирувчиси қуйидаги кўринишга эга:

% [*][<кенглик>] [F|N] [h|l|L] <тур белгиси>

Формат аниқлаштирувчиси ‘%’ белгисидан бошланади ва ундан кейин 12.1–жадвалда келтирилган шарт ёки шарт бўлмаган компонен-талар келади.

12.2–жадвал. Алмаштириладиган тур аломати белгилари

Тур аломати	Кутилаётган қиймат	Аргумент тури
Сон туридаги аргумент		
d, D	Ўнлик бутун	int * arg ёки long * arg
E,e	Сузувчи нуқтали сон	float * arg
F	Сузувчи нуқтали сон	float * arg
G,g	Сузувчи нуқтали сон	float * arg
O	Саккизлик сон	int * arg
O	Саккизлик сон	long * arg
I	Ўнлик, саккизлик ва ўн олтилик бутун сон	int * arg
I	Ўнлик, саккизлик ва ўн олтилик бутун сон	long * arg
U	Ишорасиз ўнлик сон	Unsigned int * arg
U	Ишорасиз ўнлик сон	Unsigned long * arg
X	Ўн олтилик сон	int * arg
X	Ўн олтилик сон	int * arg
Белгилар		
S	Сатр	char * arg (белгилар массиви)
C	Белги	char * arg (белги учун майдон кенглиги берилиши мумкин (масалан, %4c). N белгидан ташкил топган белгилар массивига кўрсаткич: char arg[N])
%	‘%’ белгиси	Ҳеч қандай алмаштиришлар бажа–рилмайди, ‘%’ белгиси сақланади.
Кўрсаткичлар		
N	int * arg	%n аргументигача муваффақиятли ўқилган белгилар сони, айнан шу int кўрсаткичи бўйича адресда сақланади.
P	YYYY:ZZZZ ёки ZZZZ кўринишидаги ўн олтилик	Объектга кўрсаткич (far* ёки near*).

Оқимдаги белгилар бўлагини алмаштириладиган тур аломатининг қабул қилиши мумкин бўлган белгилар 12.2-жадвалда келтирилган.

12.3–жадвал. Формат аниқлаштирувчилари ва уларнинг вазифаси

Компонента	Бўлиши шарт ёки йўқ	Вазифаси
[байроқ]	Йўқ	Байроқ белгилари. Чиқарилаётган қийматни чапга ёки ўнга текислашни,

		соннинг ишорасини, ўнлик каср нуқтасини, охирдаги нолларни, саккизлик ва ўн олтилик сонларнинг аломатларни чоп этишни бошқаради. Масалан, ‘-‘ байроғи қийматни ажратилган ўринга нисбатан чапдан бошлаб чиқаришни ва керак бўлса ўнгдан пробел билан тўлдиришни билдиради, акс ҳолда чап томондан пробеллар билан тўлдиради ва давомига қиймат чиқарилади.
[<кенглик>]	Йўқ	Майдон кенглигини аниқлаштирувчиси. Чиқариладиган белгиларнинг минимал сонини аниқлайди. Зарур бўлса қиймат ёзилишидан ортган жойлар пробел билан тўлдирилади.
[.<хона>]	Йўқ	Аниқлик. Чиқариладиган белгиларнинг максимал сонини кўрсатади. Сондаги рақамларнинг минимал сонини.
[F|N|h|l|L]	Йўқ	Ўлчам модификатори. Аргументнинг қисқа (short - h) ёки узун (long –l,L) кўринишини, адрес турини аниқлайди.
<тур белгиси>	Ҳа	Аргумент қиймати алмаштириладиган тур аломати белгиси

Форматли ёзиш функцияси printf() қуйидаги прототипга эга:

int printf(const char * <формат>[,<аргумент>,...])

Бу функция стандарт оқимга форматлашган чиқаришни амалга оширади. Функция аргументлар кетма-кетлигидаги ҳар бир аргумент қийматини қабул қилади ва унга <формат> сатридаги мос формат аниқлаштирувчисини қўллайди ва оқимга чиқаради.

12.4–жадвал. printf() функциясининг алмаштириладиган тур белгилари

Тур аломати	**Кутилаётган қиймат**	**Чиқиш формати**
Сон қийматлари		
D	Бутун сон	Ишорали ўнлик бутун сон
I	Бутун сон	Ишорали ўнлик бутун сон
O	Бутун сон	Ишорасиз саккизлик бутун сон
U	Бутун сон	Ишорасиз ўнлик бутун сон
X	Бутун сон	Ишорасиз ўн олтилик бутун сон (a,b,c,d,e,f белгилари ишлатилади)
X	Бутун сон	Ишорасиз ўн олтилик бутун сон (A,B,C,D,E,F белгилари ишлатилади)

F	Сузувчи нуқтали сон	[-]dddd.dddd кўринишидаги сузувчи нуқтали сон
E	Сузувчи нуқтали сон	[-]d.dddd ёки e[+/-]ddd кўрини–шидаги сузувчи нуқтали сон
G	Сузувчи нуқтали сон	Кўрсатилган аниқликка мос e ёки f шаклидаги сузувчи нуқтали сон
E, G	Сузувчи нуқтали сон	Кўрсатилган аниқликка мос e ёки f шаклидаги сузувчи нуқтали сон. e формат учун ‘E’ чоп этилади.
Белгилар		
S	Сатрга кўрсаткич	0-белгиси учрамагунча ёки кўрсатилган аниқликка эришилмагунча белгилар оқимга чиқарилади.
C	Белги	Битта белги чиқарилади
%	Ҳеч нима	‘%’ белгиси оқимга чиқарилади.
Кўрсаткичлар		
N	int кўрсаткич (int* arg)	%n аргументигача муваффақиятли чиқарилган белгилар сони, айнан шу int кўрсаткичи бўйича адресда сақланади.
P	Кўрсаткич	Аргументни YYYY:ZZZZ ёки ZZZZ кўринишидаги ўн олтилик сонга айлантириб оқимга чиқаради.

Ҳар бир формат аниқлаштирувчисига битта ўзгарувчи адреси мос келиши керак. Агар формат аниқлаштирувчилари сони ўзгарувчилардан кўп бўлса, натижада нима бўлишини олдиндан айтиб бўлмайди. Акс ҳолда, яъни ўзгарувчилар сони кўп бўлса, ортиқча ўзгарувчилар инобатга олинмайди. Агар оқимга чиқариш муваффақиятли бўлса, функция чиқарилган байтлар сонини қайтаради, акс ҳолда EOF.

printf() функциясининг <формат> сатри аргументларни алмаштириш, форматлаш ва берилганларни оқимга чиқариш жараёнини бошқаради ва у икки турдаги объектлардан ташкил топади:

– оқимга ўзгаришсиз чиқариладиган оддий белгилар;

– аргументлар рўйхатидаги танланадиган аргументга қўлланиладиган формат аниқлаштирувчилари.

Формат аниқлаштирувчиси қуйидаги кўринишга эга:

% [<байроқ>][<.кенглик>] [.<хона>][F|N|h|l|L] <тур белгиси>

Формат аниқлаштирувчиси ‘%’ белгисидан бошланади ва ундан кейин 12.3–жадвалда келтирилган шарт ёки шарт бўлмаган компоненталар келади.

Алмаштириладиган тур белгисининг қабул қилиши мумкин бўлган белгилар 12.4- жадвалда келтирилган.

Берилганлар қийматларини оқимдан ўқиш ва оқимга чиқаришда scanf() ва prinf() функцияларидан фойдаланишга мисол:

```
#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;
}
```

Программа фойдаланувчидан бутун ва сузувчи нуқтали сонларни, белги ва сатрни киритишни сўрайди. Бунга жавобан фойдаланувчи томонидан

```
10 12.35 A Satr
```

қийматлари киритилса, экранга

```
Oqimdan 4 ta qiymat o'qildi va ular quyidagilar:
10 12.35 A Satr
```

сатрлари чоп этилади.

Файлдан ўқиш-ёзиш функциялари

Файл оқими билан ўқиш-ёзиш амалини бажариш учун файл оқимини очиш зарур. Бу ишни, прототипи

```
FILE * fopen(const char* filename, const char *mode);
```

кўринишида аниқланган fopen() функцияси орқали амалга оширилади. Функция filename номи билан файлни очади, у билан оқимни боғлайди ва оқимни идентификация қилувчи кўрсаткични жавоб тариқасида қайтаради. Файлни очиш муваффақиятсиз бўлганлигини fopen() функциясининг NULL қийматли жавоби билдиради.

Параметрлар рўйхатидаги иккинчи - mode параметри файлни очиш режимини аниқлайди. У қабул қилиши мумкин бўлган қийматлар 12.5- жадвалда келтирилган.

12.5-жадвал. Файл очиш режимлари

Mode	Файл очилиш ҳолати тавсифи

қиймати	
R	Файл фақат ўқиш учун очилади
W	Файл ёзиш учун очилади. Агар бундай файл мавжуд бўлса, у қайтадан ёзилади (янгиланади).
A	Файлга ёзувни қўшиш режими. Агар файл мавжуд бўлса, файл унинг охирига ёзувни ёзиш учун очилади, акс ҳолда янги файл яратилади ва ёзиш режимида очилади.
r+	Мавжуд файл ўзгартириш (ўқиш ва ёзиш) учун очилади.
w+	Янги файл яратилиб, ўзгартириш (ўқиш ва ёзиш) учун очилади. Агар файл мавжуд бўлса, ундаги олдинги ёзувлар ўчирилади ва у қайта ёзишга тайёрланади.
a+	Файлга ёзувни қўшиш режими. Агар файл мавжуд бўлса, унинг охирига (EOF аломатидан кейин) ёзувни ёзиш (ўқиш) учун очилади, акс ҳолда янги файл яратилади ва ёзиш режимида очилади.

Матн файли очилаётганлигини билдириш учун файл очилиш режими сатрига ‘t’ белгисини қўшиб ёзиш зарур бўлади. Масалан, матн файл ўзгартириш (ўқиш ва ёзиш) учун очилаётганлигини билдириш учун “rt+” сатри ёзиш керак бўлади. Худди шундай бинар файллар устида ишлаш учун ‘b’ белгисини ишлатиш керак. Мисол учун файл очилишининг “wb+” режими бинар файл янгиланишини билдиради.

Файл ўзгартириш (ўқиш-ёзиш) учун очилганда, берилганларни оқимдан ўқиш, ҳамда оқимга ёзиш мумкин. Бироқ ёзиш амалидан кейин дарҳол ўқиб бўлмайди, бунинг учун ўқиш амалидан олдин fseek() ёки rewind() функциялари чақирилиши шарт.

Фараз қилайлик «C:\\USER\\TALABA\\iat1kuz.txt» номли матн файлни ўқиш учун очиш зарур бўлсин. Бу талаб

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

ифодасини ёзиш орқали амалга оширалади. Натижада дискда мавжуд бўлган файл программада f ўзгарувчиси номи билан айнан бир нарса деб тушунилади. Бошқача айтганда, программада кейинчалик f устида бажарилган барча амаллар, дискдаги «iat1kuz.txt» файли устида рўй беради.

Файл оқими билан ишлаш тугагандан кейин у албатта ёпилиши керак. Бунинг учун fclose() функциясидан фойдаланилади. Функция прототипи қуйидаги кўринишга эга:

```
int fclose(FILE * stream);
```

fclose() функцияси оқим билан боғлиқ буферларни тозалайди (масалан, файлга ёзиш кўрсатмалари берилиши натижасида буферда

йиғилган берилганларни дискдаги файлга кўчиради) ва файлни ёпади. Агар файлни ёпиш хатоликка олиб келса, функция EOF қийматини, нормал ҳолатда 0 қийматини қайтаради.

fgetc() функцияси прототипи

```
int fgetc(FILE *stream);
```

кўринишида аниқланган бўлиб, файл оқимидан белгини ўқишни амалга оширади. Агар ўқиш муваффақиятли бўлса, функция ўқилган белгини int туридаги ишорасиз бутун сонга айлантиради. Агар файл охирини ўқишга ҳаракат қилинса ёки хатолик рўй берса, функция EOF қийматини қайтаради.

Кўриниб турибдики, getc() ва fgetc() функциялари деярли бир хил ишни бажаради, фарқи шундаки, getc() функцияси белгини стандарт оқимдан ўқийди. Бошқача айтганда, getc() функцияси, файл оқими стандарт қурилма бўлган fgetc() функцияси билан аниқланган макросдир.

fputc() функцияси

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

прототипи билан аниқланган. fputc() функцияси файл оқимига аргументда кўрсатилган белгини ёзади (чиқаради) ва у амал қилишида putc() функцияси билан бир хил.

Файл оқимидан сатр ўқиш учун

```
char * fgets(char * s, int n, FILE *stream)
```

прототипи билан fgets() аниқланган. fgets() функцияси файл оқимидан белгилар кетма-кетлигини s сатрига ўқийди. Функция ўқиш жараёнини оқимдан n-1 белги ўқилгандан кейин ёки кейинги сатрга ўтиш белгиси (‘\n’) учраганда тўхтатади. Охирги ҳолда ‘\n’ белгиси ҳам s сатрга қўшилади. Белгиларни ўқиш тугагандан кейин s сатр охирига, сатр тугаш аломати ‘\0’ белгиси қўшилади. Агар сатрни ўқиш муваффақиятли бўлса, функция s аргумент кўрсатадиган сатрни қайтаради, акс ҳолда NULL.

Файл оқимига сатрни fputs() функцияси ёрдамида чиқариш мумкин. Бу функция прототипи

```
int fputs (const char *s, FILE *stream);
```

кўринишида аниқланган. Сатр охиридаги янги сатрга ўтиш белгиси ва терминаторлар оқимга чиқарилмайди. Оқимга чиқариш муваффақиятли бўлса, функция номанфий сон қайтаради, акс ҳолда EOF.

foef() функцияси аслида макрос бўлиб, файл устида ўқиш-ёзиш амаллари бажарилаётганда файл охири белгиси учраган ёки йўқлигини билдиради. Функция

```
int feof(FILE *stream);
```

прототипига эга бўлиб у файл охири белгиси учраса, нолдан фарқли сонни қайтаради, бошқа ҳолатларда 0 қийматини қайтаради.

Қуйида келтирилган мисолда файлга ёзиш ва ўқишга амаллари кўрсатилган.

```
#include <iostream.h>
#include <stdio.h>
int main()
{
 char c;
 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;
}
```

Программада «talaba.txt» файли матн файли сифатида ўқиш учун очилган ва у in ўзгарувчиси билан боғланган. Худди шундай, «talaba.dbl» матн файли ёзиш учун очилган ва out билан боғланган. Агар файлларни очиш муваффақиятсиз бўлса, мос хабар берилади ва программа ўз ишини тугатади. Кейинчалик, токи in файли охирига етмагунча, ундан белгилар ўқилади ва экранга, ҳамда out файлига чиқарилади. Программа охирида иккита файл ҳам ёпилади.

Масала. Ғалвирли тартиблаш усули.

Берилган x векторини пуфакча усулида камаймайдиган қилиб тартиблаш қуйидагича амалга оширилади: массивнинг қўшни элементлари x_k ва x_{k+1} (k=1,..,n-1) солиштирилади. Агар $x_k>x_{k+1}$ бўлса, у ҳолда бу элементлар ўзаро ўрин алмашади. Шу йўл билан биринчи ўтишда энг катта элемент векторнинг охирига жойлашади. Кейинги

қадамда вектор бошидан n-1 ўриндаги элементгача юқорида қайд қилинган йўл билан қолган элементларнинг энг каттаси n-1 ўринга жойлаштирилади ва ҳ.к.

Ғалвирли тартиблаш усули пуфакчали тартиблаш усулига ўхшаш, лекин x_k ва x_{k+1} (k=1,2,3,..,n-1) элементлар ўрин алмашгандан кейин «ғалвирдан» ўтказиш амали қўлланилади: чап томондаги кичик элемент имкон қадар чап томонга тартиблаш сақланган ҳолда кўчирилади. Бу усул оддий пуфакчали тартиблаш усулига нисбатан тез ишлайди.

Программа матни:

```
#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;
}
```

Программада берилганларни оқимдан ўқиш ёки оқимга чиқаришда файлдан форматли ўқиш - fscanf() ва ёзиш - fprintf() функцияларидан фойдаланилган. Бу функцияларнинг мос равишда scanf() ва printf() функцияларидан фарқи - улар берилганларни биринчи аргумент сифатида бериладиган матн файлдан ўқийди ва ёзади.

Номи фойдаланувчи томонидан киритиладиган f1 файлдан бутун сонлар массивининг узунлиги ва қийматлари ўқилади ва тартибланган массив f2 файлга ёзилади.

Векторни тартиблаш Pufakchali_Tartiblash() функцияси томонидан амалга оширилади. Унга вектор ва унинг узунлиги кирувчи параметр бўлади ва тартибланган вектор функция натижаси сифатида қайтарилади.

Навбатдаги иккита функция файл оқимидан форматлашмаган ўқиш-ёзишни амалга оширишга мўлжалланган.

fread() функцияси қуйидаги прототипга эга:

```
size_t fread(void * ptr, size_t size, size_t n,
             FILE *stream);
```

Бу функция оқимдан ptr кўрсатиб турган буферга, ҳар бири size байт бўлган n та берилганлар блокини ўқийди. Ўқиш муваффақиятли бўлса, функция ўқилган блоклар сонини қайтаради. Агар ўқиш жараёнида файл охири учраб қолса ёки хатолик рўй берса, функция тўлиқ ўқилган блоклар сонини ёки 0 қайтаради.

fwrite() функцияси прототипи

```
size_t fwrite(const void*ptr,size_t size,
              size_t n,FILE *stream);
```

кўриниши аниқланган. Бу функция ptr кўрсатиб турган буфердан, ҳар бири size байт бўлган n та берилганлар блокини оқимга чиқаради. Ёзиш муваффақиятли бўлса, функция ёзилган блоклар сонини қайтаради. Агар ёзиш жараёнида хатолик рўй берса, функция тўлиқ ёзилган блоклар сонини ёки 0 қайтаради.

Файл кўрсаткичини бошқариш функциялари

Файл очилганда, у билан «stdio.h» сарлавҳа файлида аниқланган FILE структураси боғланади. Бу структура ҳар бир очилган файл учун жорий ёзув ўрнини кўрсатувчи ҳисоблагични - файл кўрсаткичини мос қўяди. Одатда файл очилганда кўрсаткич қиймати 0 бўлади. Файл устида бажарилган ҳар бир амалдан кейин кўрсаткич қиймати ўқилган ёки ёзилган байтлар сонига ошади. Файл кўрсаткичини бошқариш функциялари - fseek(), ftell() ва rewind() функциялари файл кўрсаткичини ўзгартириш, қийматини олиш имконини беради.

ftell() функциясининг прототипи

```
long int ftell(FILE *stream);
```

кўринишида аниқланган бўлиб, аргументда кўрсатилган файл билан боғланган файл кўрсаткичи қийматини қайтаради. Агар хатолик рўй берса функция -1L қийматини қайтаради.

```
int fseek(FILE *stream, long offset, int from);
```

прототипига эга бўлган fseek() функцияси stream файли кўрсаткичини from жойига нисбатан offset байт масофага суришни амалга оширади. Матн режимидаги оқимлар учун offset қиймати 0 ёки ftell() функцияси қайтарган қиймат бўлиши керак. from параметри қуйидаги қийматларни қабул қилиши мумкин:

SEEK_SET (=0) - файл боши;

SEEK_CUR (=1) - файл кўрсаткичининг айни пайтдаги қиймати;

SEEK_END (=2) - файл охири.

Функция файл кўрсаткичи қийматини ўзгартириш муваффақиятли бўлса, 0 қийматини, акс ҳолда нолдан фарқли қиймат қайтаради.

rewind() функцияси

```
void rewind(FILE *stream);
```

прототипи билан аниқланган бўлиб, файл кўрсаткичини файл бошланишига олиб келади.

Қуйида келтирилган программада бинар файл билан ишлаш кўрсатилган.

```
#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;
}
```

Юқорида келтирилган программада, олдин «Talaba.dat» файли бинар файл сифатида ёзиш учун очилади ва у oqim1 ўзгарувчиси билан боғланади. Шахс ҳақидаги маълумотни сақловчи n ўлчамли динамик shaxs1 структуралар массиви oqim1 файлига ёзилади, файл узунлиги чоп қилиниб файл ёпилади. Кейин, худди шу файл oqim2 номи билан ўқиш учун очилади ва ундаги берилганлар shaxs2 структуралар массивига ўқилади ва экранга чоп қилинади. Программада файлдаги ёзувни ўзгартириш (қайта ёзиш) амалга оширилган. Ўзгартириш қилиниши керак бўлган ёзув тартиб номери фойдаланувчи томонидан киритилади (k ўзгарувчиси) ва shaxs2 структуралар массивидаги мос ўриндаги структуранинг Familiya майдони клавиатурадан киритилган янги сатр билан ўзгартирилади. oqim2 файл

кўрсаткичи файл бошидан k*sizeof(Shaxs) байтга сурилади ва shaxs2 массивнинг k - структураси (shaxs2+k) шу ўриндан бошлаб файлга ёзилади. Кейин oqim2 файли кўрсаткичи ўзгартириш киритилган ёзув бошига қайтарилади ва бу ёзув shaxsk структурасига ўқилади ҳамда экранга чоп этилади.

Масала. Ҳақиқий сонлар ёзилган f файли берилган. f файлдаги элементларнинг ўрта арифметигидан кичик бўлган элементлар миқдорини аниқлансин.

Масалани ечиш учун f файлини яратиш ва қайтадан уни ўқиш учун очиш зарур бўлади. Яратилган файлнинг барча элементларининг йиғиндиси s ўзгарувчисида ҳосил қилинади ва у файл элементлари сонига бўлинади. Кейин f файл кўрсаткичи файл бошига олиб келинади ва элементлар қайта ўқилади ва s қийматидан кичик элементлар сони - k санаб борилади.

Файлни яратиш ва ундаги ўрта арифметикдан кичик сонлар миқдорини аниқлашни алоҳида функция кўринишида аниқлаш мумкин.

Программа матни:

```
#include <iostream.h>
#include <stdio.h>
#include <string.h>
int Fayl_Yaratish()
{
 FILE * f;
 double x;
 // f файли янгидан ҳосил қилиш учун очилади
 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 бўш бўлмаса,такрорлаш
 fclose(f);
 return 1;
}
int OAdan_Kichiklar_Soni()
{
 FILE * f;
 double x;
 f=fopen("Sonlar.dbl", "rb+");
```

```
 double s=0; // s - f файл элементлари йиғиндиси
 while (!feof(f))
 {
  if (fread(&x,sizeof(double),1,f)) s+=x;
 }
 long sonlar_miqdori=ftell(f)/sizeof(double);
 s/=sonlar_miqdori; // s- ўрта арифметик
 cout<<"Fayldagi sonlar o'rta arifmetiki="<<s<<endl;
 fseek(f,SEEK_SET,0); // файл бошига келинсин
 int k=0;
 while (fread(&x,sizeof(x),1,f))
 {
  k+=(x<s); //ўрта арифметикдан кичик элементлар сони
 }
 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 файлини яратиш муваффақиятсиз бўлди.
  cout<<"Faylini ochish imkoni bo'lmadi!!!";
  return 0;
}
```

Программада бош функциядан ташқари иккита функция аниқланган:

int Fayl_Yaratish() - дискда «Sonlar.dbl» номли файлни яратади. Агар файлни яратиш муваффақиятли бўлса, функция 1 қийматини, акс ҳолда 0 қийматини қайтаради. Файлни яратишда клавиатурадан сонларнинг сатр кўриниши ўқилади ва сонга айлантирилиб, файлга ёзилади. Агар бўш сатр киритилса, сонларни киритиш жараёни тўхтатилади ва файл ёпилади;

int OAdan_Kichiklar_Soni() - дискдаги «Sonlar.dbl» номли файли ўқиш учун очилади ва файл элементларининг s ўрта арифметигидан кичик элементлари сони k топилади ва функция натижаси сифатида қайтарилади.

Бош функцияда файлни яратиш муваффақиятли кечганлиги текширилади ва шунга мос хабар берилади.

Адабиётлар

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. Абдуқодиров, У.М.Мирзаев СИ тилида программалаш асослари. Ўқув 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.

Иловалар

1-илова

Берилганларнинг компьютер хотирасидаги ички кўриниши

Компьютернинг жорий (оператив) хотираси катта сондаги, иккита ҳолатларни эслаб қолиш элементларидан ва уларни бошқариш схемаларидан иборат бўлган электрон қурилмадир. Хотирадаги мурожаат қилиш мумкин бўлган энг кичик маълумот бирлиги байт (8 иккилик разряд, ёки битлар).

Айрим берилганларни хотирада сақлаш учун бир байт етарлидир, масалан белгилар кодларини, бошқалари учун 2, 4, 8 байтлар талаб қилиниши мумкин. Шу сабабли берилганларни хотирада сақлаш учун *сўз* (2 байт), *иккиланган сўз* (4 байт) тушунчалари киритилган.

Кўп байтли берилганларни қайта ишлашда уларнинг ички байтларига мурожаат қилишга тўғри келади: бу байтлар шартли равишда нолдан бошлаб номерланади ва ўнгдан чапга жойлашади (уларнинг қоғоздаги кўринишида). Ўнгдаги (нолинчи) байт - *кичик байт*, чапдаги охирги байт - *катта байт* деб номланади (1и-расм).

1и- расм. Байт, сўз ва иккиланган сўз катталиклари

Умуман олганда хотирада фақат бутун иккилик сонларни сақлаш мумкин. Бошқа турдаги берилганлар учун, масалан белги ва каср сонлар учун кодлаш қоидаси кўзда тутилган.

Шуни таъкидлаб ўтиш керакки, берилганлар байтлари хотирада жойлашиши қуйидагича: ҳар бир сўз ёки иккиланган сўз хотирада кичик байтдан бошланади ва катта байт билан тугайди (2и-расм).

Хотирадаги байтлар кетма-кетликларининг номерлари

2и- расм. Кўпбайтли берилганларнинг байтларининг номерлари

Компьютернинг рақамли электрон қурилмалари амал қиладиган иккилик саноқ системаси билан ишлаш фойдаланувчи учун ноқулай. Хотирадаги, регистрлардаги берилганларини ифодалаш учун айрим ҳолларда 8 саноқ системаси, асосан 16 саноқ системаси ишлатилади. Бунда байт қиймат иккита 16 саноқ системасидаги рақам бўлган ифодаланади: 00h сонидан FFh сонигача, бу ерда h- соннинг 16 саноқ системасида тасвирланганини билдиради. Сўз тўртта 16 саноқ системасидаги рақам билан ифодаланади (0000h...FFFFh оралиғидаги сонлар, 10 саноқ системасида 0...65535).

Ишорасиз бутун сонлар хотирада иккилик саноқ системасида ёзилиб, байт, сўз, иккилик сўз, тўртлик сўз кўринишида ёзилиши мумкин. Масалан, $98_{10}=62_{16}=01100010_2$. Бу ерда индекс саноқ системасининг асоси. Ушбу сон битта байтдаги кўринишида қуйидагича:

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

3и- расм. 98 сонининг байтдаги иккилик кўриниши

Одатда битта байтдаги сон иккита ўн олтилик рақам билан кўрсатилади (62_{16}). Агар, 1100010_2 сонини икки байтда (сўзда) тасвирлаш зарур бўлса, унинг катта разрядлари 0 билан тўлдирилади.

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

4и- расм. 98 сонининг сўздаги иккилик кўриниши

Қуйидаги жадвалда эгаллаган байт ўлчамига мос бутун ишорасиз сонларнинг қиймат чегаралари кўрсатилган (1и-жадвал).

1и-жадвал. Ишорасиз бутун сон турлари

Битлар сони	**Ўлчами**	**Тур**	**Қиймат чегараси**

8	Байт	unsigned char	0 .. 255
16	Сўз	unsigned int	0 .. 65535
32	Иккилик сўз	unsigned long	0 .. 4294967295
64	Тўртлик сўз	unsigned __int64	0..18446744973709551615

Ўлчами байтдан катта турларда ишорасиз сон тескари кўринишда сақланади, яъни, олдин кичик байтлар кейин катта байтлар жойлашади. Масалан, сўз кўринишидаги 0062_{16} сонининг компьютер хотирасидаги жойлашуви қуйидагича бўлади:

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 - адресли байт								A+1 адресли байт							

5и- расм. Сўздаги 98 сонининг хотирада жойлашуви

Ишорали бутун сонлар компьютер хотирасида қўшимча код кўринишида сақланади. Мусбат бутун сонлар ишорасиз сонлар каби ёзилади. Манфий x сони эса 2^k -|x| ишорасиз сон кўринишида ёзилади, бу ерда k - ажратилган ўлчамдаги битлар сони.

Масалан, бир байтда жойлашган -98_{10} (-62_{16}) сонини қўшимча кодининг кўриниши:

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

Ишорали бутун сон ёзилган байтнинг катта разряди (7 разряди) сон ишорасининг аломати ҳисобланади. Агар 7-разрядда 1 бўлса байтда қўшимча коддаги манфий сон сақланаяпти, акс ҳолда байтда мусбат сон жойлашган деб ҳисобланади.

Агар -62_{16} сони сўз катталигида бўлса, у (10000_{16}-62_{16})=$FF9E_{16}$ сонига тенг бўлади (6и.а-расм) ва хотирада тескари кўринишда сақланади (6и.б- расм).

а)

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

б)

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 - адресли байт								A+1 адресли байт							

6и- расм. Сўздаги 98 сонининг хотирада жойлашуви

Қўшимча кодни топиш бошқа усули ҳам мавжуд: олдин манфий сонни ишорасиз кўриниши 2 саноқ системасида ёзилади, кейин ҳар бир разряддаги 0 рақам 1 рақамига, 1 рақами эса 0 алмаштирилади.

Ҳосил бўлган сонга 1 қўшилади. Мисол учун шу усулда 98_{10} сонини қўшимча коди қуйидагича топилади:

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

Қуйидаги жадвалда байт ўлчамидаги сонларнинг компьютер хотирасидаги ички кўринишига мисоллар келтирилган (2и-жадвал).

2и-жадвал. Байт ўлчамидаги сонларнинг ички кўриниши

10 с/с сон	**2 с/с код**	**10 с/с сон**	**2 с/с қўшимча код**
0	00000000	-1	11111111
1	00000001	-2	11111110
2	00000010	-3	11111101
3	00000011	-126	10000010
126	01111110	-127	10000001
127	01111111	-128	10000000

Ишорали бутун сон турига мос равишда манфий ва мусбат қийматлар чегараси мавжуд (3и-жадвал).

3и-жадвал. Ишорали сон қийматлар чегараси

Битлар сони	**Ўлчами**	**Тури**	**Чегараси**
8	Байт	char	-128 ... +127
16	Сўз	int	-32768 ... +32767
32	иккилик сўз	long int	-2147483648... +2147483647
64	тўртлик сўз	__int64	-4294967296... +4294967295

Ҳақиқий сонлар хотирада иккилик саноқ системасида нормаллашган экспоненциал шаклда сақланади.

Иккилик саноқ системасидаги нормаллашган сон деб, бутун қисми доимо 1 тенг, каср қисми - мантисса (M) ва экспонента деб номланувчи даражаси (тартиби p) билан тасвирланган сонга айтилади. Масалан 111.01_2 сонинг нормал кўриниши $1.1101*10^{10}_2$ тенг. Бу ерда $M=0.1101_2$ $p=10_2$ қийматига тенг.

Intel процессорлари учун нормаллашган сон

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

кўринишда бўлади.

Бу ерда:

s - сон ишораси аниқловчи разряд қиймати. Агар s=0 бўлса, сон мусбат, s=1 ҳолда сон манфий эканлигини билдиради;

M мантисса ва у $0 \le M < 1$ шартни қаноатлантиради;

N - саноқ система асоси (N=2);

q - характеристика.

Характеристика сон тартиби p билан қуйи-даги муносабатда бўлади: q=p+*фиксирланган силжиш*. Юқорида қайд қилинган учта форматнинг ҳар бири учун *фиксирланган силжиш* тур-лича бўлади. Одатда у 2^{k-1}-1 қийматига тенг бўлади. Бу ерда k - характеристика учун ажратилган разрядлар сони. Нормаллашган соннинг бутун қисмидаги рақам доимий равишда 1 бўлгани учун у катакка ёзил-майди ва бу ҳолат сон устида амал бажаришда аппарат даражасида инобатга олинади.

IEEE 754 стандарти бўйича ҳақиқий сонлар иккилик саноқ системасида учта форматда сақланади: қисқа формат; узун формат () ва кенгайтирилган формат (80 битлик):

- қисқа формат, 32 битлик, силжиш - $127_{10}=7F_{16}$ (7и.а-расм);
- узун формат, 64 битлик, силжиш - $1023_{10}=3FF_{16}$(7и.б-расм);
- кенгайтирилган формат, 80 битлик, силжиш -$16383_{10}=3FFF_{16}$ (7и.в-расм).

1 бит	8 бит	23 бит
Ишора (s)	Характеристика (q)	Мантисса (M)
31	30 23	22 0

а)

1 бит	11 бит	52 бит
Ишора (s)	Характеристика (q)	Мантисса (M)
63	62 52	51 0

б)

1 бит	15 бит	64 бит
Ишора (s)	Характеристика (q)	Мантисса (M)
79	78 64	63 0

в)

7и- расм. Ҳақиқий соннинг ички форматлари

Қуйидаги жадвалда ҳақиқий сон форматларининг чегаралари берилган (4и-жадвал).

4и-жадвал. Ҳақиқий сон форматларининг чегаралари

Берилган формати	**Қиймат чегараси**	**Аниқлиги (ўнлик рақамда)**
Қисқа формат	3.4×10^{-38} ... $3.4\times10^{+38}$	7
Иккиланган аниқлик	1.7×10^{-308} ... $1.7\times10^{+308}$	16
Кенгайтирилган формат	3.4×10^{-4932} ... $3.4\times10^{+4932}$	19

Мисол сифатида 0.5_{10} ва -8.5_{10} сонларининг хотирадаги ички кўриниши аниқлайлик:

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

Ушбу соннинг қисқа форматдаги кўриниши қуйидагича бўлади:

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 0
31	30 23	22 0

8и- расм. 0.5 сонинг қисқа форматдаги ички кўриниши

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

Ушбу соннинг қисқа форматдаги кўриниши қуйидагича бўлади:

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

9и- расм. -8.5 сонинг қисқа форматдаги ички кўриниши

Белгилар хотирада бир байт жой эгаллайди ва ҳар бир белги ўзини иккилик ASCII (2-илова) коди билан ёзилади. Унда максимал равишда 256 белги аниқланиши мумкин. Windows тизимида икки байт ажратилган Unicode код тизими киритилган. Бу тизимда ҳар миллий алфавитлар учун 256 белгидан иборат бўлган ва махсус номерланган кодлашлар киритилган.

Сатр бу - белгилар кетма-кетлиги ва у хотирада ҳам худди шундай кетма-кетликдаги байтларда жойлашади. Масалан, «Bu satr» сатри хотирада қуйидагича ёзилади:

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

9и- расм. ASCIIZ сатрнинг хотирадаги ички кўриниши

Бу ерда A - сатр бошланишининг (унинг кичик байтининг) адреси.

2-илова

ASCII кодлар жадваллари

5и-жадвал. Бошқарув белгилар кодлари (0-31)

Мнемоник номи	10 с.с. коди	16 с.с. коди	Клавиатура тугмаси	Мазмуни
nul	0	00	^@	Нол
soh	1	01	^A	Сарлавҳа бошланиши
stx	2	02	^B	Матн бошланиши
etx	3	03	^C	Матн тугаши
eot	4	04	^D	Узатишнинг тугаши
enq	5	05	^E	Сўров
ack	6	06	^F	Тақиқлаш
bel	7	07	^G	Сигнал (товуш)
bs	8	08	^H	Орқага қадам
ht	9	09	^I	Горизонтал табуляция
lf	10	0A	^J	Янги сатрга ўтиш
vt	11	0B	^K	Вертикал табуляция
ff	12	0C	^L	Янги саҳифага ўтиш
cr	13	0D	^M	Кареткани қайтариш
soh	14	0E	^N	Суришни ман этиш
si	15	0F	^O	Суришга рухсат бериш
dle	16	10	^P	Берилганлар боғлаш калити
dc1	17	11	^Q	1-қурилмани бошқариш
dc2	18	12	^R	2-қурилмани бошқариш
dc3	19	13	^S	3-қурилмани бошқариш
dc4	20	14	^T	4-қурилмани бошқариш
nak	21	15	^U	Таққослаш инкори
syn	22	16	^V	Синхронизация
etb	23	17	^W	Узатилган блок охири
can	24	18	^X	Рад қилиш
em	25	19	^Y	Соҳа тугаши
sub	26	1A	^Z	Алмаштириш
esc	27	1B	^[	Калит
fs	28	1C	^\	Файллар ажратувчиси
qs	29	1D	^]	Гуруҳ ажратувчи
rs	30	1E	^^	Ёзувлар ажратувчиси
us	31	1F	^_	Модуллар ажратувчиси

6и-жадвал. Аксланувчи белгилар (32-127)

Белги	10 с.с. коди	16 с.с. коди	Белги	10 с.с. коди	16 с.с. коди	Белги	10 с.с. коди	16 с.с. коди
	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

7и-жадвал. Аксланувчи белгилар (128-255)(Windows-1251)

Белги	10 с.с. коди	16 с.с. коди	Белги	10 с.с. коди	16 с.с. коди	Белги	10 с.с. коди	16 с.с. коди
Ђ	128	80	«	171	AB	Ц	214	D6
Ѓ	129	81	¬	172	AC	Ч	215	D7
‚	130	82	-	173	AD	Ш	216	D8
ѓ	131	83	®	174	AE	Щ	217	D9
„	132	84	Ї	175	AF	Ъ	218	DA
…	133	85	°	176	B0	Ы	219	DB
†	134	86	±	177	B1	Ь	220	DC
‡	135	87	І	178	B2	Э	221	DD
€	136	88	і	179	B3	Ю	222	DE
‰	137	89	ґ	180	B4	Я	223	DF
Љ	138	8A	µ	181	B5	а	224	E0
‹	139	8B	¶	182	B6	б	225	E1
Њ	140	8C	·	183	B7	в	226	E2
Ќ	141	8D	ё	184	B8	г	227	E3
Ћ	142	8E	№	185	B9	д	228	E4
Џ	143	8F	є	186	BA	е	229	E5
ђ	144	90	»	187	BB	ж	230	E6
‘	145	91	ј	188	BC	з	231	E7
’	146	92	S	189	BD	и	232	E8
“	147	93	s	190	BE	й	233	E9
”	148	94	ї	191	BF	к	234	EA
•	149	95	А	192	C0	л	235	EB
-	150	96	Б	193	C1	м	236	EC
—	151	97	В	194	C2	н	237	ED
□	152	98	Г	195	C3	о	238	EE
™	153	99	Д	196	C4	п	239	EF
љ	154	9A	Е	197	C5	р	240	F0
›	155	9B	Ж	198	C6	с	241	F1
њ	156	9C	З	199	C7	т	242	F2
ќ	157	9D	И	200	C8	у	243	F3
ћ	158	9E	Й	201	C9	ф	244	F4
џ	159	9F	К	202	CA	х	245	F5
	160	A0	Л	203	CB	ц	246	F6
Ў	161	A1	М	204	CC	ч	247	F7
ў	162	A2	Н	205	CD	ш	248	F8
Ј	163	A3	О	206	CE	щ	249	F9
¤	164	A4	П	207	CF	ъ	250	FA
Ґ	165	A5	Р	208	D0	ы	251	FB
¦	166	A6	С	209	D1	ь	251	FC
§	167	A7	Т	210	D2	э	253	FD
Ё	168	A8	У	211	D3	ю	254	FE
©	169	A9	Ф	212	D4	я	255	FF
Є	170	AA	Х	213	D5			

3-илова

8и-жадвал. Математик функциялар кутубхонаси (math.h)

Функция прототипи	Бажарадиган амали
int abs(int i)	i сонни абсолют қийматини қайтаради
double acos(double x)	Радианда берилган x аргументни аркко-синус қийматини қайтаради
double asin(double x)	Радианда берилган x аргументни арк-синус қийматини қайтаради
double atan(double x)	Радианда берилган x аргументни арк-тангенс қийматини қайтаради
double atan2(double x, double y)	Радианда берилган x/y нисбатнинг арк-тангенси қийматини қайтаради
double ceil(double x)	Ҳақиқий x қийматини унга энг яқин катта бутун сонгача айлантиради ва уни ҳақиқий кўринишда қайтаради
double cos(double x)	x радианга тенг бўлган бурчакни косинусини қайтаради
double cosh(double x)	x радианга тенг бўлган бурчакни гиперболик косинусини қайтаради
double exp(double x)	e^x қийматни қайтаради
double fabs(double x)	Ҳақиқий сонни абсолют қийматини қайтаради
double floor(double x)	Ҳақиқий x қийматни энг яқин кичик сонга айлантиради ва уни ҳақиқий сон кўринишида қайтаради
double fmod(double x, double y)	x сонини y сонига бўлиш натижасидаги қолдиқни қайтаради. % амалига ўхшаган, фақат ҳақиқий сон қайтаради
double frexpr(double x, int *expptr)	x сонни мантиссасини ва даражасини ажратиб, мантисса қийматини қайтаради ва даражасини кўрсатилган expptr адресига жойлаштиради
double hypot(double x, double y)	Тўғри учбурчакни катетлари бўйича гипотенузани ҳисоблайди
long int labs(long int num)	num узун бутун соннинг абсолют қийматини қайтаради
double ldexp(double x, int exp)	$X*2^{exp}$ қийматни қайтаради
double log(double x)	x сонининг натурал логарифмини қайтаради
double log10(double x)	x сонинг 10 асосли логарифмини қайтаради
double modf(double x, double *intptr)	x сонининг каср қисмини қайтаради ва бутун қисмини intptr адресга жойлайди

double poly(double x, int n, double c[])	$c[n]x^n+c[n-1]x^{n-1}+ \ldots+c[1]x+c[0]$ полиномни қийматини ҳисоблайди
double pow(double x, double y)	x^y ҳисоблайди
double pow10(int p)	10^p ҳисоблайди
double sin(double x)	x радианга тенг бўлган бурчакни синусини қайтаради
double sinh(double x)	x радианга тенг бўлган бурчакни гиперболик синусини қайтаради
double sqrt(double x)	x сонининг квадрат илдизини қайтаради
double tan(double x)	x радианга тенг бўлган бурчакни гиперболик косинусини қайтаради
double tanh(double x)	x радианга тенг бўлган бурчакни гиперболик косинусини қайтаради