ЎЗБЕКИСТОН РЕСПУБЛИКАСИ АЛОҚА , АХБОРОТЛАШТИРИШ ВА ТЕЛЕКОММУНИКАЦИЯ ТЕХНОЛОГИЯЛАРИ ДАВЛАТ ҚЎМИТАСИ

ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ

«Компьютер» инжиниринг" факультети
«Информатика асослари» кафедраси

«C/C++ тилида дастурлаш» фани бўйича маърузалар матни

4-қисм

Тошкент -2012

Маъруза-25,26 (6 с)

Мавзу: С/С++ НИНГ ГРАФИК ИМКОНИЯТЛАРИ.

Мақсад: ***Талабаларда экраннинг график ҳолати, тасвир рангларидан фойдаланиш, матнавий ҳолатнинг ранг имкониятлари ҳақидаги тушунчани ҳосил қилиш.***

Калит сўзлар: ***Адаптер, график ҳолат, матнавий ҳолат, график ҳолат процедуралари, пиксель, динамик хотира.***

Режа:

1. **Экранни матнавий ҳолатдан график ҳолатга ўтказиш.**
2. **Экранни график ҳолатдан матнавий ҳолатга ўтказиш.**
3. **Тасвирларни ҳосил қилиш.**
4. **Тасвирларни ҳаракатлантириш.**

С++да график ҳолатида ишлаш учун махсус graphics.h файли мавжуд. Бу директива ўзгармаслар, ўзгарувчилар ва турли қисм дастурлардан ташкил топган бўлиб, улар ёрдамида турли график адаптерлар билан ҳар хил тасвирлар чизиш мумкин. Адаптер компьютерда graphics.h файли билан ишлаш имкониятини яратадиган махсус қурилмадир. График ҳолатига ўтилганда экран алоҳида-алоҳида нуқталарга бўлинади.Ҳар бир нуқта ўз координатасига эгадир.

Энг кўп ишлатиладиган адаптерлар:

1. CGA - color graphics Adapter
2. MCGA - multi color graphics array
3. EGA - enhenced graphics Adapter
4. VGA - video graphics array .

Драйверларни кўрсатиш учун қуйидаги ўзгармаслар ишлатилади:

Detect = 0 CGA = 1; MCGA = 2; EGA=3; VGA=9.

Матн ҳолатидан график ҳолатига ўтиш учун махсус процедурадан фойдаланилади: initgraph (&gd, &gm, “ path “); бу ерда:

gd - драйвер номи;

gm - режим номи;

Path - керакли драйвер файлнинг йўли. Кўпинча gd=0 деб олинади.

Драйверлар .bgi файлларида сақланади. Агар драйвер ишчи каталогнинг ўзида жойлашган бўлса, у ҳолда Path = " " (бўш белгиси) бўлади.

График ҳолатидан яна матн ҳолатига ўтиш керак бўлса, closegraph( ) функцияси ишлатилади.

Чизмаларни ҳосил қилиш учун қуйидаги процедура ва функциялар ишлатилади:

1. putpixel (x, y, color) - x ва y координатадаги нуқтани color рангда чизиш;
2. getpixel (x, y) - x ва y координатадаги нуқтанинг рангини аниқлайди;
3. line (x1, y1, x2, y2) - x1 ва y1 координатадаги нуқтадан x2 ва y2 координатадаги нуқтагача кесма чизиш;
4. circle (x, y, r) - маркази x ва y координатада ва радиуси R булган айлана чизиш;
5. rectangle (x1, y1, x2, y2) - юқори чап нуқтаси x1 ва y1 координатада, унг пастки нуқтаси x2 ва y2 координатада бўлган тўғри тўртбурчакни чизиш;
6. setbkcolor (color) - орқа фонга ранг бериш;
7. setcolor (color) - чизиш рангини ўрнатиш (рангли қалам); Бу ерда color - ранг номери ёки номи. Агар ранг номи ёзиладиган бўлса, уни катта харфларда ёзилади.

8. bar (x1, y1, x2, y2) - жорий ранг ва чизиқлар ёрдамида ичи бўялган тўғри тўртбурчак чизиш;
9. fillellpse (x, y, xr, yr) - маркази x ва y да, xr кенгликда ва xr баландликда ичи бўялган рангли эллипс чизади;
10. setfillstyle (style, color) - бўяш усул ва рангни ўрнатиш. Бу ерда style - ўзгармас катталик бўлиб, у қуйидагича бўлиши мумкин:
0 - соҳани фон ранги билан тўлдириш;
1- соҳани ранг билан узлуксиз тўлдириш;
2 - калик горизонтал чизиқлар
3 - ингичка оғма чизиқлар
4 - йўғон оғма чизиқлар
5 - йўғон оғма чизиқлар (бошқа стил)
6 - оғма йўллар
7 - тўртбурчакли чизиқлар
8 - оғма тўртбурчаклар
9 - зич оғма шртихлар
10 - сийрак нуқталар (у ер - бу ерда)
11 - зич нуқталар билан

11. getmaxx - жорий режим ва драйверлар учун нуқталар сонини аниқлаш; getmaxy - жорий режим ва драйверлар учун вертикал нуқталар сони. Бу процедура ёрдамида компьютернинг ўзи экрандаги максимал нуқталар сонини аниқлайди.
12. linerel (x, y) - x ва y координатали нуқтадан жорий нуқтагача кесма чизиш;
lineto (x, y) - жорий нуқтадан x ва y координатали нуқтагача кесма чизиш;
13. bar3D (x1, y1, x2, y2, h, top) - параллелопипед чизади. Бу ерда h - параллелопипеднинг узунлиги; top - юқори қисмини чизиш учун керак. Агар topon - бўлса томи бор, агар topoff - бўлса томи йўк.
14. arc (x, y, a, b, r) - ёй чизиш учун. Бу ерда x ва y - марказнинг координаталари, a - бош бурчак, b - охириги бурчак, r - ёй радиуси. Бурчаклар градусда қабул қилинади.
15. ellipse (x, y, a, b, xr, yr) - ҳудди шу тартибда эллипс ёйини чизади.
16. drawpoly (n, p) - кўпбурчак чизиш учун. Бу ерда n - кўпбурчакнинг учлари сони; p -. Кўпбурчак учларининг координаталари.
17. floodfill (x, y, color) - жорий ранг ва усулдан фойдаланган ҳолда чегараланган соҳани бўяш. Бу ерда x ва y - шу соҳага тегишли бўлган бирор нуқта координатаси. Аввал ранг, стили кейин чизмалар кўрсатилади. Масалан:

```
setcolor (4);            {қизил рангли қалам, чегара ранги}
setfillstyle (1, 2);     {1-стиль билан яшил ранг билан бўяш}
circle (50, 50, 35);     {радуси 35 бўлган айлана чизиш}
floodfill (50, 50, 4); {айлана ичига ранг тўкиш, бўяладиган чегара ранги рангли қалам билан бир хил бўлиши керак}
```

18. setlinestyle (s, a, b) – турли стилдаги чизиқларни чизиш учун; Бу ерда s- style номери; a –фойдаланувчи стилини яратиши мумкин бўлган параметр, одатда a=1 деб олинади; b- чизиқнинг қалинлигини кўрсатадиган параметр
0 – оддий чизиқ
1 – майда пунктир чизиқ
2 – қалин ва узунчоқ пунктир чизиқ
3 – юпқа ва узунчоқ пунктир чизиқ
4 – сийрак нуқтали чизиқ.

1- Мисол:

```
# include <graphics.h>
# include <conio.h>
void  main ( )
{  int  i, j, gd, gm ;
gd= 0;
initgraph (&gd, &gm, " ");
setcolor (14);                                  //  сарик қалам
for  ( i=0; i<=20; i++)
for  ( j=0; j<=20; j++)
circle (i*50, j*30, 55);                        //  сариқ рангли айланалар
rectangle (0, 0, getmaxx, getmaxy);   //экран бўйлаб тўғри тўртбурчак
setcolor (11);                                  //  тўқ феруза рангли қалам
bar3d(200, 300, 100, 150, 30, topon);  //  параллелопипед, ичи оқ
setcolor (CYAN);                                //  оч феруза рангли қалам
fillellipse (350, 360, 135, 90);             //эллипс, ичи оқ рангда
getch( );
closegraph( );  }
```

2-мисол.

```
…….
void main ( )
{  gd=0;
initgraph (&gd, &gm, ' ');
   setbkcolor (BLUE);
   setcolor (14);
 rectangle (120, 130, 240, 250);
setcolor (6);
line (120, 130, 180, 80);
setcolor (2);
line (180, 80, 240, 130);
setcolor (14);
rectangle (160, 160, 200, 250);
setcolor (4);
setfillstyle(7, 9);
circle( 300, 300, 50);
floodfill (300, 300, 4);
getch( ); closegraph ( );   }
```

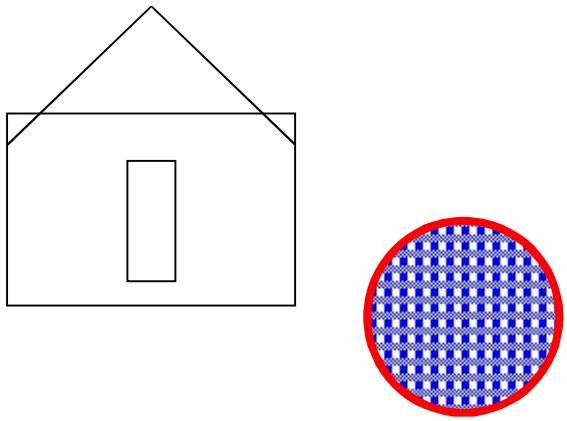


График ҳолатида турли шрифтлардан фойдаланиб матнларни ҳам ёзса бўлади. Шрифтлар .chr кенгайтмали файлларда сақланади. Улар .bgi файллари билан битта каталогда сақланиши шарт.

1. outtextxy (x, y, 'matn'); - матнни ёзиш; бу ерда x ва y матн бошланадиган нуқта координаталари; масалан: outtextxy (10, 10, 'Mirzaev K. 212-07 Aty');
2. settextstyle (sh, n, r); мант шрифтини ўрнатиш; бу ерда sh - шрифт номери (0 - векторли шрифт, 1 - стандарт шрифт); n - шрифт йўналиши (0 - чапдан ўнгга, 1 - қуйидан юқорига ёзиш); r - шрифт размери (оддий шрифтда 1, векторли шрифтда 4 деб олинади);
3. settextjustify (h, v) - ёзилган сатрни текислайди. У outtextxy процедурасидан кейин ёзилади. Бу ерда h - горизонтал текислаш; v - вертикал текислаш; Горизонтал текислаш учун: 0 - чапга; 1- марказга; 2 - унгга. Вертикал текислаш учун: 0 - пастга; 1 - марказга; 2 - юқорига.
4. setusercharsize - вектор шрифтлари учун бир хил символларнинг эни ва бўйини ўрнатади. Масалан: setUserCharSize(x1, y1, x2, y2);

3-мисол. Функцияларнинг графикларини чизиш.

```
# include <graphics.h>
# include <conio.h>
# include <math.h>
void  main ( )
{  int  i, j, gd, gm ;  float x, y;
gd=0;   initgraph (&gd, &gm, "  ");
setcolor (14);
line (320, 0, 320, 480);
line (0, 240, 640, 240);
line (480, 0, 480, 235);
line (325, 120, 635, 120);
line (160, 245, 160, 475);
line (0, 360, 315, 360);
line (480, 245, 480, 475);
line (325, 360, 635, 360);

x =-10;  outtextxy(10, 20, ' y=sin(x) grafigi');
do
{ y = sin(x);
putpixel (160 + 10*x, 120 - y, 5);
 x = x+0.001;   }
while  (x<=10);

x1 = -10;  outtextxy(10, 20, ' y=cos(x) grafigi');
do
{ y = cos(x1);
putpixel (480 + 20*x1, 120 - 20*y1, 6);
 x1 = x1+0.001;   }
while  (x1 < =10);

x2 = -10;  outtextxy(10, 20, ' y=exp(x) grafigi');
do
{ y2 = exp(x2);
putpixel (160 + 10*x2, 360 - 20*y2, 7);
 x2 = x2+0.001;   }
while  (x2 <=10);

x3 = -10;  outtextxy(10, 20, ' y=ln(x) grafigi');
do
{ y3 = ln(x3);
putpixel (480 + 10*x3, 360 - y3, 8);
 x3 = x3+0.001;  }
while  (x3 <=10);  getch( );  closegraph( );   }
```

include <grahpics.h> директивасининг яна шундай процедуралари мавжудки, улар ёрдамида чизмаларни экран бўйлаб ҳаракатга келтириш мумкин. Фигураларни ҳаракатга келтиришнинг бир неча усуллари бор. Улардан бири ҳаракатни такрорланиш буйруғи орқали ташкил қилишдир. Иккинчи усул экранда чизилган чизма жойлашган соҳани массив кўринишида эслаб қолиб, уни махсус процедура ёрдамида экраннинг керакли нуқтасига кўчиришдир. Бунда динамик хотирадан фойдаланилади.

Катта миқдордаги маълумотлар ишлатиладиган масалаларни ечишда, компьютернинг график имкониятларидан фойдаланганимизда хотира ҳажми етишмаслиги мумкин. Бундай холларда динамик хотира жуда қўл келади. Динамик хотира бу компьютернинг дастурга маълумотлар сегментидан ташқари юклатилган тезкор хотирадир. Бу хотира тахминан 200-300 Кбни ташкил қилади. Динамик хотирадан фойдаланиш учун кўрсаткичлар ишлатилади. Бу ўзгарувчиларни (кўрсаткичларни) хотирада жойлаштиришни компилятор амалга оширади. Кўрсаткич шундай ўзгарувчики, унинг қиймати ўзгарувчи қийматига эмас, балки шу ўзгарувчи жойлашган хотира адресига тенгдир.

Динамик хотира соҳасидан жой ажратиш учун **new** оператори ишлатилади. Бу сўздан кейин хотирага жойлаштириладиган объект тоифаси аниқланади. Масалан: new int; деб ёзсак, динамик хотирадан 2 байт жой ажратган бўламиз. Масалан: int *p;

p = new int;

ёки int *p = new int ;

Ажратилган хотира соҳасига бирор қийматни жойлаштириш мумкин: *p = 750 ;

Бу ёзувни қуйидагича ўқилади: « p кўрсаткичида адреси сақланаётган хотирага 750 сонини ёзинг ».

Динамик хотира соҳаси чегараланган, у тўлиб қолганда new оператори орқали жой ажратиш хатоликка олиб келади. Бу холни биз хотиранинг тўлиб кетиши ёки оқиб кетиши деймиз (утечка памяти). Шунинг учун хотира бошқа керак бўлмаса уни бўшатиш зарурдир. Буни **delete** оператори ёрдамида бажарилади. Масалан: delete p;

Экранда чизмаларни ҳаракатлантириш учун керак бўладиган процедуралар:

1.i**magesize (x1, y1, x2, y2)** – экраннинг чап юқори нуқтаси ва унг пастки нуқтаси координаталаридан тўғри тўртбурчакли соҳани сақлаш учун керак бўладиган хотиранинг ўлчами (байтларда олинади);

2. g**etimage (x1, y1, x2, y2, p)** – динамик хотиранинг берилган p майдонида тўғри тўртбурчакли тасвирни сақлаш. Бу ерда p – тасвир сақланадиган жойнинг адресини сақлайдиган ўзгарувчи, яъни кўрсаткич.

3. **putimage (x, y, p, m)** – экраннинг берилган жойига тасвирни чиқариш; бу ерда x ва y – хотиранинг p майдонидаги тасвирдан нусха кўчириладиган экран майдонининг чап юқори нуқтаси; m – тасвирни экранга чиқариш ҳолати. Агар:

m = 0 (NormalPut) - тасвирни кўчириш. Бунда эскичи ўчиб, янгиси пайдо бўлади (худди юриб кетаётгандек)

m = 1 (XorPut)

m = 2 (Orput) –

m = 3 (AndPut)

```
# include < graphics.h >
# include <conio.h >
# include <dos.h >
void main ( )
{  int  gd = 0, gm, I, j, s;  int  *a;
 initgraph(&gd,&gm,"");
  setcolor ( 4 );
  circle ( 30, 30, 20 );  putpixel ( 30, 30, 2);
  rectangle ( 10, 10, 50, 50);
  s = imagesize ( 9, 9, 51, 51);
   *a = new  int;   *a = s;
  getimage (9, 9, 51, 51, a);
  for ( i = 0; i < = 585; i + +)
  { putimage ( i, 10, a, 0);  sound (20);   delay (10); nosound ( ); }
  for ( j = 10; j < = 420;  j + +)
  { putimage(585, j, a, 0 );  sound ( 30 );  delay (10);  nosound( );}
  for (i = 585;  i > = 10; i - -)
  { putimage(i, 420, a, 0);  delay ( 10 ); }
  for (j = 420;  j >10; j - -)
  { putimage(10, j, a, 0);  delay( 10 ); }
  delete  a;
  getch();  //closegraph( );
  }
```

Назорат саволлари:

1. C++ тилининг график имкониятлари.
2. Тасвирларни ҳосил қилувчи функциялар.
3. Тасвирларни ҳаракатлантириш.
4. C/C++ тилида графика.
5. initgraph процедураси.
6. График адаптерлар ҳақида
7. Экраннинг фон рангини ўзгартириш
8. Тўғри чизиқ, айлана, тўғритўртбурчак, эллипс, арка, параллелопипедларни ва ҳакозоларни чизиш асослари
9. Бўяш усуллари.
10. Чегараланган соҳани бўяш алгоритми
11. Чизиш стиллари
12. Функцияларни чизиш асослари
13. График режимида матнларнинг ишлатилиши

Тестлар:

1. initgraph (&gd, &gm, " path ") процедурасининг вазифаси?
   A. Матнавий ҳолатни ўрнатади.
   B. График ҳолатни ўрнатади.
   C. Драйверни аниқлайди.

2. circle (30, 30, 20); процедурасининг вазифаси?

A. Тўғри чизиқ чизади.

B. Айлана чизади.

C. Тўғри тўртбурчакни чизади..

Маъруза-27. (4 с)

Мавзу: САТРЛАР, ЛИТЕРАЛ (БЕЛГИЛАР). САТРЛАР УЧУН СТАНДАРТ ФУНКЦИЯЛАР. САТРЛИ ВА СИМВОЛЛИ ЎЗГАРУВЧИЛАР.

Мақсад: ***Талабаларга символлар билан ишлаш ва сатрлар устида амаллар бажариш асосларини ўргатиш.***

Калит сўзлар: ***Символ, символли массив, сўзларга ишлов бериш, сатр, сатрларни улаш, сатрларни саралаш*****.**

Режа:

1. **Символли массивлар.**
2. **Символли массивни киритиш.**
3. **Матнга ишлов бериш.**
4. **Сатрларни ташкил этиш**
5. **Дастурда сатрлардан фойдаланиш.**
6. **Сатрларга ишлов берувчи стандарт функциялар.**

C/C++ тилида сатрлар (1та сўз) символли массив сифатида ишлатилади. Символли массивлар қуйидагича ифодаланади:

```
char  soz[10];
```

Символли массивлар қуйидагича инициализация қилинади:

```
char shahar[ ]="TOSHKENT ";
```

Бу ҳолда массив элементларининг сони автоматик равишда аниқланади ва массив охирига курсорни кейинги қаторга кўчирадиган \0 белгиси қўйилади.

Юқоридаги мисолни қуйидагича инициализация қилиш ҳам мумкин:

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

бу ҳолда сўз охирига \0 белгисини қўйиш шарт.

Масалан: Берилган сўздаги бир символни олиб ташлаш дастури.

```
# include <iostream.h>
void  main ( )
{
char s[100], c; int i, j;
cin >> s;                  // киритилаётган сўз
cin >>c;                   // киритилаётган символ
for (i=j=0; s[i]!='\0'; i++)
if (s[i] != c) s[j++] = s[i];
```

```
// кир. символнинг барчасини олиб ташлайди
s[j]='\0';
cout <<"yangi soz="<<s;
}
```

Сўзлар массивлари билан ишлаш

C/C++ тилида сўзлар массивлари икки ўлчамли массив сифатида ишлатилади. Масалан: char nom [4][6]; бу таъриф ёрдамида ҳар бири 6та символдан иборат 4та сўзли массив берилган. Сўзли массивлар қуйидагича инициализация қилиниши мумкин:

```
char ismlar [3][10] = {"Anvar", "Mirkomil", "Abdulla"};
```

Ҳар бир сўз учун хотирадан 10 байт жой ажратилади ва ҳар бир сўз охирига \0 белгиси қўйилади.

Сўзли массивлар инициализация қилинганида сўзлар сони кўрсатилмаслиги ҳам мумкин, бу ҳолда сўзлар сони берилганларга қараб автоматик равишда аниқланади. Масалан:

```
char comp[ ][9]={"kompyuter","printer","kartrij"};
```

1-Мисол: берилган символдан (ҳарфдан) бошланадиган сўзлар рўйхатини чиқаринг.

```
# include <iostream.h>
void  main ( )
{
char a[] [10], b;   int i, n;
cout<<"so'zlar sonini kiriting=";   cin>>n;
cout<<"so'zlarni kiriting=";
for (i=0; i<=n; i++)
cin >> a[i];
cout << "simvolni kiriting=";   cin >> b;
for (i = 0; i <= n;  i++)
if (a[i][0] = = b)   cout  << a[i] << endl;
}
```

```
Сўзлар сони:  3
Сўзларни киритинг:
Nodir
Bobur
Sobir
Символни киритинг:
B
Натижа:
Bobur
```

2-мисол. Талабалар рўйхати берилган (фамилияси). Ita фан номи ва шу фандан талабаларнинг олган баҳолари киритилган. Ёмон баҳо олган талабалар рўйхатини чиқариш дастури тузилсин.

```
# include <iostream.h>
main ( )
{ char a[20][10], s[10];  int k[20], i;
cout <<"fan nomini kiriting=";    cin >>s;
cout <<"familiyalar:"<<endl;
for (i=0; i<20; i++)
cin >>a[i];
cout <<"baholar:"<<endl;
for (i=0; i<=20; i++)
cin >>k[i];
for (i=0; i<=20; i++)
if (k[i] = =2) cout <<"Yomon baho olganlar:"<<a[i]<<endl;
}
```

Массивларни ташкил этишда кўрсаткичлардан фойдаланилса ҳам бўлади. Бунда массивлар қуйидагича эълон қилинади:

```
тоифа  *массив номи [сон];
```

Кўрсаткичларни символли массивларга ишлатиш қулайдир. Масалан:

char fam[][10] = {"Olimov","Rahimov","Ergashev"} деб ёзилса, хотирада 30 элементдан иборат массив ҳосил бўлади, чунки ҳар бир фамилия 10та ҳарфга етмаса ҳам охири 0(ноль)лар билан тўлдирилади. Уни кўрсаткичлар ёрдамида таърифланса, яъни
char *fam[][10] = {"Olimov","Rahimov","Ergashev"} деб ёзилса, хотирада 24та элементдан иборат бўлган массив ҳосил бўлади, чунки фамилиялардаги ҳар бир ҳарфлар саналади. Шунда хотира ҳам тежалади.
Изоҳ: ҳар бир фамилия охирига \0 белгиси қўйилади.

Сатрлар устида амаллар

С++ тилида сатрли маълумотлар устида бир қанча амалларни бажариш мумкин. Бунинг учун аввало **# include <string.h>** ни улаш керак бўлади.
1. а) 2та сатрли ўзгарувчиларни бир-бирига улаш
strcat (s1, s2); яъни s2 сатрни s1 сатрга улаш.
б) 2-сатрнинг n та символини 1-сатрга улаш
strncat (s1, s2, n); яъни s2 сатрнинг n та символини s1 сатрга улаш.
Натижа доим 1-кўрсатилган сатрга ўзлаштирилади.
Масалан;

```
1)  # include < string.h >
    # include < iostream.h>
    void main( )
    {  char  s1[10] = ”Gul”,  s2[10] = “bahor”;
    strcat (s1, s2);
    cout << s1 << endl;    //  yoki   cout << strcat (s1, s2)<< endl;
    }
```

Натижа; Gulbahor
Агар cout << strcat (s2, s1)<< endl;
деб ёзсак натижа bahorGul чиқади.

```
2)  # include < string.h >
    # include < iostream.h>
    void main( )
    {  char  s1[10] = ”Gul”,  s2[10] = “bahor”;
    strncat (s1, s2,4);
    cout << s1 << endl;    //  yoki   cout << strncat (s1, s2,4)<< endl;
    }
```

Натижа; Gulbaho

2. а) 2-сатрнинг нусхасини 1-сатрга қўйиш
strcpy (s1, s2);
b) 2-сатрдан n та символни 1-сатрга нусхасини қўйиш
strncpy (s1, s2, n);
Масалан:

```
# include < string.h >
 # include < iostream.h>
 void main( )
 {  char  s1[10] = ” Dilorom ”,  s2[10] = “ Gul ”;
 strncpy (s1, s2, 3);
 cout << s1 << endl;    //  yoki   cout << strncpy (s1, s2, 3)<< endl;
 }
```

Натижа: Dilorom
Агар strncpy (s2, s1, 3); деб ёзилса, натижа Dil чиқади.

3. 2та сатр ўзгарувчисини солиштириш.

a) **strcmp (s1, s2);**

Агар s1 > s2; натижа + сон

Агар s1 < s2 ; натижа – сон

Агар (s1=s2) улар айнан тенг бўлсалар 0 чиқади.

Масалан;

```
char  s1[10] = " Gulnora ",  s2[10]=" Guli ";  int  n;
n=strcmp ( s1, s2 );    n +
n=strcmp ( s2, s1 );    n –
```

yoki:

```
char  s1[10] = " Guli ",  s2[10]=" Gulnora ";  int  n;
n=strcmp ( s1, s2 );    n -
n=strcmp ( s2, s1 );    n +
```

b) **strncmp (s1, s2, n);** - 1-сатрнинг n та символини 2-сатр билан солиштириш.

4. Сатр узунлигини ҳисоблаш **strlen (s)**;

Масалан:

```
char s = "Dastur";   int  n;
n = strlen (s);
cout << n << endl;
```

Натижа ; n=6

5. а) Сатр ўзгарувчисининг бошидан символларини берилган символ билан алмаштириш:

strset (s, c);

s – берилган сатр ўзгарувчиси; c – символ

б) Сатр ўзгарувчисининг бошидан n та символини берилган символ билан алмаштириш:

strnset (s, c, n);

Масалан:

```
char  s = "Hilola",  c = 'Z';  int  n = 1;
        strnset ( s, c, n);
        cout << s<< endl;
```

Натижа; Zilola

```
char  s = "Hilola",  c = 'Z';
        strset ( s, c);
        cout << s<< endl;
```

Натижа; ZZZZZZ

1-мисол. Палиндром масаласи. Палиндром сўз деб ўнгига ҳам, тескарисига ҳам бир хил ўқиладиган сўзга айтилади.

```
# include <iostream.h>
# include <string.h>
void main ( )
{ char  s[20], sp[20]=" ";  int n, i;
cin >> s ;                      // berilgan so'z
n = strlen (s);                 // so'z  uzunligi
for ( i=0; i < n; i++)
sp [(n-1) – i] = s [ i ];
if ( strcmp( s, sp) = = 0)   //  so'zlar  bir hil bo'lsa
cout <<"palindrom so'z="<< sp <<endl;
else  cout <<"palindrom so'z  emas ="<< sp <<endl;
```

}

2-мисол.

```
# include <iostream.h>
# include <string.h>
void main ( )
{ char d[20],  *b = “  “,  *c = “Dilafruz”,  *a = “Aripiva”;
strcpy ( d, a );
strcat ( d, b );
strcat ( d, c );
cout << d << endl;
}
```

Натижа: Aripova Dilafruz

Юқоридаги функциялардан ташқари яна сатр тоифасидаги ўзгарувчилар устида қуйидаги функцияларни ҳам ишлатиш мумкин ва уларни ишлатиш учун **# include <stdlib. h>** ни улаш керак.

1. Сатр тоифасидаги сонни бутун сон тоифасига ўтказиш

char → int **atoi (s);**

```
int  n;  char  *s = “12345”;
n = atoi ( s );
cout <<  n <<endl;
H :  n = 12345;
```

```
int n;  char *s= “12345.67”;
n = atoi ( s );
cout <<  n <<endl;
H :  n = 12345;
```

2. Сатр тоифасидаги сонни ҳақиқий сон тоифасига ўтказиш

char → float **atof (s);**

```
float  n;  char  *s = “12345.67”;
n = atof ( s );
cout <<  n <<endl;
H :  n = 12345.67;
```

```
float  n; char  *s = “123e+30 ”;
n = atof ( s );
cout <<  n <<endl;
H :  n = 1.23e+32;
```

3. Сатр тоифасидаги сонни узун бутун сон тоифага ўтказиш

char → long **atol (s);**

```
long  n;  char  *s = “1234567890”;
n = atol ( s );
cout <<  n <<endl;
H :  n = 1234567890;
```

```
long  n; char  *s = “123456789.67 ”;
n = atol ( s );
cout <<  n <<endl;
H :  n = 123456789;
```

4. Бутун тоифадаги ўзгарувчини сатр тоифасига ўтказиш

float **→** char **itoa (a, b, c);**

Бу ерда:
a – int тоифасидаги ўзгарувчи
b – char тоифасидаги ўзгарувчи
c – с.с. асоси

```
int  a = 1234, c = 10;  char  *b;
itoa (a, b, c);
cout <<  b <<endl;
Н :  b = 1234;
int  a = 1234, c = 8;  char  *b;
itoa (a, b, c);
cout <<  b <<endl;
Н :  b = 30071;
```

```
int  a = 12.34, c = 10;  char  *b;
itoa (a, b, c);
cout <<  b <<endl;
Н :  b = 12;
```

5. Узун бутун тоифасидаги ўзгарувчини сатр тоифасига ўтказиш
long → char **ltoa (a, b, c);**
Бу ерда:
a – long тоифасидаги ўзгарувчи
b – char тоифасидаги ўзгарувчи
c – с.с. асоси

```
long  a = 123456789, c = 10;
char  *b;
ltoa (a, b, c);
cout <<  b <<endl;
Н :  b = 123456789;
```

7. Ишорасиз узун бутун тоифасидаги ўзгарувчини сатр тоифасига ўтказиш
unsigned long → char **ultoa (a, b, c);**
Бу ерда:
a – ишорасиз узун тоифасидаги ўзгарувчи
b – char тоифасидаги ўзгарувчи
c – с.с. асоси

```
unsigned long  a = 1234567890, c = 10;  char  *b;
ultoa (a, b, c);
cout <<  b <<endl;
Н :  b = 1234567890;
```

Назорат саволлари:

1. Символли массивлар.
2. Символли массивни киритиш.
3. Матнга ишлов бериш.
4. Сатрларни ташкил этиш
5. Дастурда сатрлардан фойдаланиш.
6. Сатрларга ишлов берувчи стандарт функциялар.

.

Тестлар:

1. Бутун натижа қайтариб ҳақиқий аргумент қабул қилувчи функция сарлавҳаси аниқлансин.

+A) int func1 (float a);
B) int func1 (int a);

C) float (float a);
D) float func1 (float a);
E) long func1 (float a).

2. Бутун натижа қайтариб бутун аргумент қабул қилувчи функция сарлавҳаси аниқлансин.

A) int func1 (float a);
+B) int func1 (int a);
C) float (float a);
D) float func1 (float a);
E) long func1 (float a).

Маъруза-28 (4 с)

Мавзу: ХОТИРАНИНГ ТАҚСИМЛАНИШИ. АВТОМАТИК ВА СТАТИК ЎЗГАРУВЧИЛАР. ДИНАМИК ХОТИРА.

Мақсад: ***Талабаларда хотира турлари ва хотира турлари билан ишлаш ҳақидаги тушунчаларни шакллантириш.***

Калит сўзлар: ***Глобал ўзгарувчилар соҳаси, динамик хотира, регистрли хотира, дастур сегменти, стекли хотира.***

Режа:

1. **Хотира турларидан фойдаланиш.**
2. **Динамик хотира соҳасидан жой ажратиш.**
3. **Адрес билан ишлаш.**

Дастурчи ўз дастури устида иш бошлаши билан операцион система (DOS, WINDOWS) компилятор талабига мувофик хотира соҳасидан жой ажратади. Мавжуд хотира 5та соҳага ажратилади:

1. Глобал ўзгарувчилар соҳаси
2. Бўш ёки динамик хотира
3. Регистрли хотира
4. Дастур сегменти
5. Стекли хотира

Локал ўзгарувчилар ва функция параметрлари стекли хотирага жойлаштирилади. Стек – бу дастурдаги функция чақирилганда ундаги маълумотлар учун талаб қилинадиган хотиранинг махсус соҳасидир. Унинг стек деб аталишига сабаб, «охириги келган 1-кетади» жараёни асосида ишлашидир. Агар бир функция иккинчисини чақирса, иккинчи функция ўз ишини тугатганидан сўнг 1-функция ўз ишини якунлайди, яъни охириги чақирилган функция биринчи бўлиб ишини тугатади. (тахланган тарелкалар каби). Локал ўзгарувчиларнинг ўзига хос хусусияти шундан иборатки, дастур бошқаруви унга тегишли функциядан чиқиши билан унга ажратилган хотира соҳаси бўшатилади, яъни бу ўзгарувчилар ўчирилади.

Дастур коди, яъни барча операторлар ва амаллар кетма-кетлиги дастур сегментида жойлаштирилади.

Регистрли хотира дастурдаги ўзгарувчиларни оператив хотирада эмас, балки марказий процессорнинг бирор регистрида сақлаш учун хизмат қилади.

Глобал ўзгарувчилар дастур ишга туширилганидан бошлаб токи ўз ишини якунлагунича хотирадан ўчирилмайди. Иҳтиёрий жойда ундан фойдаланиш мумкин, яъни

бу ўзгарувчилар хотирадан доимий жой олиб туради. Уларни ўчириб бўлмайди. Бу муаммони динамик хотира ҳал этади.

Динамик хотирани ахборотлар ёзилган ячейкаларнинг номерланган тўплами сифатида қараш мумкин. Бу хотира ячейкаларини номлаш мумкин эмас. Уларга мурожаат керакли ячейка адресини ўзида сақловчи кўрсаткичлар орқали амалга оширилади.

Динамик хотира дастур ишини тугатганида ҳам бўшатилмайди, бундай хотирани бўшатиш билан дастурчининг ўзи шуғулланади.

Хўш, динамик хотирани бўшатиш нима учун керак?

Ўзгарувчи учун ажратилган хотира соҳаси бўшатилмагунича бу жойдан фойдаланиш мумкин эмас. Агар динамик хотира функция билан ишлаш жараёнида ажратилган бўлса, функция ишини тугатганидан кейин ҳам биз ундан бемалол фойдаланишимиз мумкин. Динамик хотиранинг яхши томони ундаги маълумотларга фақатгина унинг кўрсаткичига мурожаат қилиш орқалигина бўлади. Бу эса бизга жорий маълумотларни ўзгартиришни қатъий назорат қилиш имкониятини беради.

Динамик хотира соҳасидан жой ажратиш учун **new** оператори ишлатилади. Бу сўздан кейин хотирага жойлаштириладиган объект тоифаси аниқланади.

Масалан: new int; деб ёзсак, динамик хотирадан 2 байт жой ажратган бўламиз.

new оператори натижа сифатида белгиланган хотира ячейкасининг адресини қайтаради. Бу адрес кўрсаткичга ўзлаштирилади.

Масалан: int *p;
p = new int;

ёки int *p = new int ;

Ажратилган хотира соҳасига бирор қийматни жойлаштириш мумкин:
*p = 750 ;

Бу ёзувни қуйидагича ўқилади: « p кўрсаткичида адреси сақланаётган хотирага 750 сонини ёзинг ».

Динамик хотира соҳаси чегараланган, у тўлиб қолганда new оператори орқали жой ажратиш хатоликка олиб келади. Бу холни биз хотиранинг тўлиб кетиши ёки оқиб кетиши деймиз (утечка памяти). Шунинг учун хотира бошқа керак бўлмаса уни бўшатиш зарурдир. Буни delete оператори ёрдамида бажарилади. Масалан: delete p;

Бунда кўрсаткич ўчирилмайди, балки унда сақланаётган адресдаги хотира соҳаси бўшатилади. Белгиланган хотиранинг бўшатилиши кўрсаткичга таъсир қилмайди, унга бошқа адресни ўзлаштириш ҳам мумкин.

Масалан:

```
# include < iostream.h >
void main ( )
{ int t = 5, h = 7 ;
  int  *p = &t ;
  cout << t << endl;
  cout << *p << endl;
  cout << &h << endl;
```

```
  p = new  int ;   *p=9;
  cout << *p << endl;
  cout << t << endl;
  delete  p;  }
```

Натижада: t=5; *p = 5;
&h = 0xfff2;
*p = 9; t = 5;

Назорат саволлари:

1. Динамик хотира тушунчаси.
2. Жой ажратиш оператори.
3. Адрес билан ишлаш.
4. Кўрсаткичларни қўллаш.
5. Динамик хотирани нима учун бўшатиш керак?

Тестлар:

1. Қайси сўз ёрдамида динамик хотира ажратилади?
   A) delete
   B) new
   C) void
   D) break
   E) return

2. Қайси сўз ёрдамида динамик хотира ўчирилади?
   +A) delete
   B) new
   C) void
   D) break
   E) return

3. Дастур натижаси ?

```
  p = new  int ;   *p=9;
cout << *p << endl;
cout << t << endl;
delete  p;  }
```

Маъруза-29 (4 с)

Мавзу: ФОЙДАЛАНУВЧИНИНГ ТОИФАСИНИ ЯРАТИШ. ТУЗИЛМАЛАР, БИРЛАШМАЛАР.

Мақсад: ***Талабаларда фойдаланувчи тоифасини яратиш, тузилмалар билан ишлаш, аралаш тоифаларни бирлаштириш тушунчаларини ҳосил қилиш.***

Калит сўзлар: ***Тузилма, фойдаланувчи тоифаси.***

Режа:

1. **Фойдаланувчининг директиваларини (include) яратиш.**
2. **Тузилмаларни ташкил этиш**
3. **Фойдаланувчи тоифасини ташкил этиш.**
4. **Массивлар- тузилма элементлари.**

Фойдаланувчининг директиваларини (include) яратиш

C/C++ тилида фойдаланувчи ўзининг include ини яратиши ва ундан керакли вақтларда кенг фойдаланиши имконияти берилган. Бунинг учун керакли амаллар алгоритми функция ва процедура сифатидаги дастури блокнотга ёзилади. Сўнгра бу дастурга иҳтиёрий ном (лотинча) берилади, файл номидан сўнг .h кенгайтмаси берилади. h - header - сарлавҳа деган маънони билдиради. Бу файлни INCLUDE папкасига сақлаш шарт! Фойдаланувчи бу include да аввалдан мавжуд бўлган include лардан, исталган ўзгарувчи, ўзгармаслар, функция ва процедуралардан фойдаланиши мумкин. Дастурчи ўзининг шахсий include ини ишлатиши учун асосий дастурда (C/C++ муҳитида) уларни эълон

қилиши ва ундаги функция ва процедураларнинг номлари ҳамда ҳақиқий параметрларининг ўрнини билиши керак. Шахсий include ларни чақиришда < > ёки " " белгилари ишлатилиши мумкин.
Масалан: arccos x = arctg ( ) формуласи ёрдамида ташкил этувчи include яратамиз ва ундан фойдаланамиз.
Блокнотдаги дастури:

```
# include < math.h >
float   acos (float x)
{ float  y;
y = atan( sqrt ( 1-x*x) / x );
return y;
}
```

Бу файлга иҳтиёрий ном берамиз, масалан: farruh . h сўнгра ундан фойдаланиб дастур тузамиз: (C/C++ муҳитида)

```
# include  < iostrream.h >
# include < conio.h >
# include < farruh.h >
void main ( )
{ float  x, y;
cin >> x;
y = acos (x);
cout << "y="<< y  << endl;
getch ( );
}
```

Одатда яратилаётган include ларга барча керакли функциялар гуруҳлаб жойлаштирилади ва бир йўла чақириб ишлатилади. Масалан:

```
# include  < math. h >
float acos ( float  x)
{  ……     }

float   asin ( float  x )
{ ……. }

float  sh ( float  x )
{  ……... }
```

ва х.к.

2-мисол. Учбурчак ва бешбурчак чизадиган процедура учун include яратинг ва ундан фойдаланинг.

Блокнотда: (номи chiz . h булсин)

```
# include  < graphics. h >
int  gd=0, gm;  initgraph (&gd, &gm, " ");
void  uchb ( int  x1, int  y1, int  x2, int  y2, int  x3, int  y3)
{ line (x1, y1, x2, y2);
  line  (x1, y1, x3, y3);
  line  (x2, y2, x3, y3 ); }

void  besh ( int a[ ] )
{ drawpoly ( 6, a );}
```

Энди бу include дан фойдаланамиз:

```
# include  < chiz. h >
# include  < conio. h >
{  void  main ( )
setcolor (4);
uchb ( 10, 20, 100, 80, 250, 150);
int a[ ] = { 10, 200, 30, 50, 70, 50, 90, 200, 50, 350, 10, 200);
besh (a);
getch ( );
}
```

Бир неча хил тоифадаги ўзгарувчиларни битта ном билан бирлаштириб ишлатиш тузилма дейилади. Тузилмалар мураккаб катталикларни ўзаро боғлаб, ягона тоифа остида дастур тузишда ёрдам беради. Тузилмалар қуйидагича таърифланади:

```
struct тузилма номи
{
  элементлар таърифи;
}
```

Тузилма номи иҳтиёрий (лотинча) символлар билан белгиланади.

Мисол учун омбордаги молларни тасвирловчи тузилма ҳосил қиламиз. Унда: мол номи (char[10]); сотиб олиш нархи (long); устама хақ, фоиз(float); моллар сони (int); мол келиб тушган сана (char[9]). Бу тузилма дастурда қуйидагича таърифланади:

```
struct  ombor
{
char name[10];  // мол номи 10та символдан ошмайди
long  nn;          // мол нархи
float f;           //  фоизи
int  i;            //   моллар сони
char date[9];    //   сана 9та символдан иборат
}
```

Тузилмалар таркибида иҳтиёрий тоифадаги ўзгарувчилар, массивлар қатнашишлари мумкин.

Тузилмалар таърифланганда конкрет рўйхатни киритиш мумкин, яъни

```
struct тузилма номи
{
  элементлар таърифи;
} конкрет рўйхат;
```

Масалан: талабалар рўйхати учун тузилма ҳосил килиш:

```
struct  student
{ char name[10];
  char fam[10];
  int ty;} student1, student2,student3;
```

Бу ҳолда student тузилмали тоифа билан бирга 3та конкрет рўйхат киритилган ва ҳар бири учун исми, фамилияси, туғилган йили берилади.

Тузилмали тоифа таърифланганда тузилма номи кўрсатилмасдан, тўғридан тўғри конкрет рўйхат кўрсатилиши ҳам мумкин:

```
struct
{
  элементлар таърифи;
} конкрет рўйхат;
```

Масалан: юқоридаги ёзувни қуйидагича ёзиш мумкин:

```
struct
{ char name[10];
  char fam[10];
  int ty;} student1, student2,student3;
```

ёки

```
struct
{ char prossesor[10];
  char matplat[10];
  int memory;
  int disk; } IBM_486, Pentium_4,Compaq;
```

Тузилма элементларига қуйидагича мурожаат қилиш мумкин:
тузилма номи. элемент номи;
Бу ерда нуқта амали тузилма элементига мурожаат қилиш амали дейилади. Бу амал қавс амалига ўхшаб юқори устиворликка эгадир.
Масалан: student1.name; student2.ty; student3.fam;

```
student1.name = "Anvar"; student2.ty = 1990; student3.fam = "Solijonov";
```

Тузилмаларни ишлатишда кўпинча typedef сўзидан фойдаланилади. Бундан мақсад тузилмадаги катталиклардан янги тоифа яратиб, ундан дастурда кенг фойдаланиш имкониятини яратишдир. Уни фойдаланувчининг (дастурчининг) тоифаси деб юритилади. typedef да янги ном ишлатилади ва шу ном кейинчалик бошқа ўзгарувчиларни белгилашда ишлатилади. Мисол:

```
typedef  struct
{ float f;
  char st[10];
  double d, d1;}  bbb;
bbb   nnn[25];
```

Мисол учун қуйидаги дастурни кўрамиз. Унда комплекс сонларни қўшиш амали қурилган. Комплекс сонлар ҳақиқий ва мавхум қисмлардан ташкил топади. Улар алоҳида-алоҳида қўшилади.

```
# include <iostream.h>
# include <conio.h>
typedef  struct
{ double real;
  double imag; } complex;
void main( )
{  complex  x, y, z;
cin >>x.real;  cin >>x.imag;
cin >>y.real;  cin >>y.imag;
z.real = x.real + y.real;
z.imag = x.imag + y.imag;
cout << z.real;  cout << z.imag;
getch( );
}
```

Массивларни ҳам тузилма элементлари сифатида ишлатиш мумкин. Массив элементларининг сон қийматларини инициализация қилиш ҳам мумкин. Масалан:

```
struct  { float a; int mas[4] } pp = { 5.34, {1,2,3,4}};
```

Масалан: гуруҳда 20та талаба бор. Талабаларнинг фамилияси ва информатика фанининг 3та модулидан олган бахолари берилган. Шу фан бўйича қарздор талабалар ҳақида маълумот берувчи дастур тузинг.

```
# include <iostream.h>
# include <conio.h>
void main ( )
{
typedef struct
{ char fam[15]; int  mod1, mod2, mod3;} gr;
gr  gr211_08[20]; int i, f;    clrscr( );
cout <<"malumotlarni kiriting:\n";
for (i = 1; i<=20; i++)
{ cout <<"familiyasini kiriting: \n";
cin >>gr211_08[i].fam;
cout <<"ballarini kiriting:\n";
{ cin >> gr211_08[i].mod1;
cin >> gr211_08[i].mod2;
cin >> gr211_08[i].mod3;}  }
f=0;
for(i=1; i<=20; i++)
if (gr211_08[i].mod1= =2 || gr211_08[i].mod2= =2 || gr211_08[i].mod3= =2)
{
cout <<"Qarzdor talaba:"<<gr211_08[i].fam<<endl;
f=f+1; }
cout <<"qarzdor talabalar soni:"<<f<<endl;
getch();  }
```

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 тур-даги х- абцисса ва у- ордината жуфтлиги - (х,у) кўринишида бери-лади. 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.

С++ 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 і-майдо-нининг (ном$_i$) эълони.

Бошқача айтганда, структура эълон қилинган ўзгарувчилардан (майдонлардан) ташкил топади. Унга ҳар хил турдаги берилган-ларни 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 тури-нинг 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 ўзга-рувчилар 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 майдон-лари 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 майдон-ларига 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 эгаллаган-лигини 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 қийма-тини 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 структура-сини 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 Yo`nalish[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 тартибла-ниши 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 ишлаши-нинг 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 |

Назорат саволлари:

1. Тузилмани таърифлаш.
2. Тузилмалар билан ишлаш асослари
3. Тузилма элементларига мурожаат.
4. Массивларнинг тузилма элементи сифатида қўлланиши.
5. Фойдаланувчининг типини яратиш.
6. Тузилмада фойдаланувчи тоифасини ташкил этиш.

Тестлар:

1. Қайси сўз ёрдамида тузилма таърифланади?

A) switch
B) throw
C) public
+D) struct
E) for

2. Қайд этиш (перечисление) қайси калитли сўздан бошланади?

+A) enum
B) struct
C) typedef
D) union
E) class

3. Бирлашма (объединение) қайси калитли сўздан бошланади?

A) enum
B) struct
C) typedef
+D) union
E) class

4. Тузилма майдонига мурожаат этганда (.) дан чапда қайси операнд жойлашади?
A) тузилма майдони
B) тузилма исми
+C) тузилмали ўзгарувчи
D) тузилманинг калитли сўзи
E) тузилмага кўрсаткич

Маъруза-30, 31 (6 с)

Мавзу: ФАЙЛЛИ ОҚИМЛАР. ФАЙЛЛАРНИ КИРИТИШ ВА ЧИҚАРИШ. ДИСКДАГИ ФАЙЛЛАР.

Мақсад: Талабаларда **файллар билан ишлаш кўникмаларини ҳосил қилиш.**

Калит сўзлар: ***Файл, оқим, фойдаланувчи директиваси.***

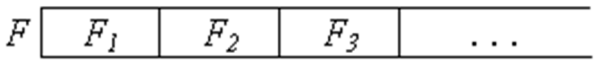
Режа:

1. **Матн ва бинар файллар**
2. **Ўқиш-ёзиш оқимлари. Стандарт оқимлар**
3. **Файлларни ташкил этиш**
4. **Символларни ўқиш-ёзиш функциялари**
5. **Файлларни оқим кўринишида ташкиллаштириш.**
6. **Фойдаланувчи директивасини ташкил этиш.**

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 компоненталари сонини аниқлаш имкони йўқ. Улар масалани ечиш жараёнида аниқланадива 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 ходимларининг 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.

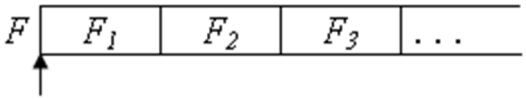
Файл - бу бир хил турдаги қийматлар жойлашган ташқи хотирадаги номланган соҳадир.

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 қўшилиши 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 жойлаштиришмумкин. 12.2-расмда файл кўрсаткичи файл бошини кўрсатмоқда.

Файл ёзувларига мурожаат кетма-кет равишда амалга оширилади: n- ёзувга мурожаат қилиш 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 С++ 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”.

С++ 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 амал-лар 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 Си 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 сатр-нинг 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 компонеталари *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-satruzunroq[13][10]

[13][10]

4- satr[13][10]

[26]

Матндаги [n] - n-кодли бошқарув белгисини билдиради. Одатда матн таҳрирлари бу белгиларни кўрсатмайди.

Бинарфайллар- 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 бажа-рилмайди.

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, формат-лашган 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 жўна-тишни операцион системанинг ўзи таъминлайди.

Символларни 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 с белгини чиқаради. Функция қайтарувчи қиймати сифатида int турига айлантирилган с 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 пайти-да 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 Си 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 аниқлаштирувчи-сига 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 акслантирилади ва ўзгарувчи билан номланган хотира бўлагига жойлаштирилади (сақланади). Функция оқимдан берилганларни ўқиш жараёнини «тўлдирувчи белгини» учратганда ёки оқим тугаши натижасида тўхтатиши мумкин. Оқимдан берилганларни ўқиш муваффаққиятли бўлса, функция муваффаққиятли 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 белгилар 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 тўлдирувчи белги ёки алмаштирилмайдиган белги учраса 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	Қўзғалувчи нуқтали сон	float * arg
F	Қўзғалувчи нуқтали сон	float * arg
G,g	Қўзғалувчи 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 бажарилмайди, ‘%’ 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 ўнгга текислашни, соннинг ишорасини, ўнлик каср нуқтасини, охирдаги нолларни, саккизлик ва ўн олтилик сонларнинг аломатларни чоп этишни бошқаради. Масалан, ‘-‘ байроғи қийматни ажратилган ўрингa 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) кўри-нишини, адрес турини аниқлайди.
<тур белгиси>	Ҳа	Аргумент қиймати алмаштириладиган тур аломати белгиси

Форматли ёзиш функцияси printf() қуйидаги прототипга эга:

intprintf(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	Қўзғалувчи нуқтали сон	[-]dddd.dddd кўринишидаги қўзғалувчи нуқтали сон
E	Қўзғалувчи нуқтали сон	[-]d.dddd ёки e[+/-]ddd кўринишидаги қўзғалувчи нуқтали сон
G	Қўзғалувчи нуқтали сон	Кўрсатилган аниқликка мос e ёки f шаклидаги қўзғалувчи нуқтали сон
E, G	Қўзғалувчи нуқтали сон	Кўрсатилган аниқликка мос e ёки f шаклидаги қўзғалувчи нуқтали сон. e формат учун ‘E’ чоп этилади.
Белгилар		
S	Сатрга кўрсаткич	0-белгиси учрамагунча ёки кўрса-тилган 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 қўзғалувчи 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 ASatr

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 - modeпараметри файлни очиш режимини аниқлайди. У қабул қилиши мумкин бўлган қийматлар 12.5- жадвалда келтирилган.

12.5-жадвал. Файл очиш ҳолатлари

Mode қиймати	Файл очилиш ҳолати тавсифи
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+	Янгифайл 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 аргу-ментда 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. Белгиларниўқиш тугагандан кейин s сатр охирига, сатр тугаш аломати ‘\0’ белгиси қўшилади. Агар сатрни ўқиш муваф-фақиятли бўлса, функция s аргумент кўрсатадиган сатрни қайтаради, акс ҳолда NULL.

Файл оқимига сатрни fputs() функцияси ёрдамида чиқариш мум-кин. Буфункцияпрототипи

int fputs (const char *s, FILE *stream);

кўринишида аниқланган. Сатр охиридаги янги сатрга ўтиш белгиси ва терминаторлароқимга чиқарилмайди. Оқимга чиқариш муваффа-қиятли 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--;
}
}
}
returnM;
 }
```

Программада берилганларни оқимдан ўқиш ёки оқимга чиқаришда файлдан форматли ўқиш - fscanf() ва ёзиш - fprintf() функцияларидан фойдаланилган. Буфункцияларнинг мос равишда scanf() ва printf() функцияларидан фарқи - улар берилганларни биринчи аргумент сифатида бериладиган матн файлдан ўқийди ва ёзади.

Номи фойдаланувчи томонидан киритиладиган f1 файлдан бутунсонлар 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 тўрган 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 тўрган 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 тузилмаси боғланади. Бу тузилма 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 yozishamalga 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 тузилмалар 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 тузилмалар массивидаги мос ўриндаги тузилманинг Familiya майдони клавиа-турадан киритилган янги сатр билан ўзгартирилади. oqim2 файл кўрсаткичи файл бошидан k*sizeof(Shaxs) байтга сурилади ва shaxs2 массивнинг k - тузилмаси (shaxs2+k) шу ўриндан бошлаб файлга ёзилади. Кейин oqim2 файли кўрсаткичи ўзгартириш киритилган ёзув бошига қайтарилади ва бу ёзув тузилмасига ўқилади ҳамда экранга чоп этилади.

Масала. Ҳақиқий сонлар ёзилган f файли берилган. f файлдаги элементларнинг ўрта арифметигидан кичик бўлган элементлар миқдори 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 бўшбўлмаса,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;
}
```

Дастурда 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 топилади ва функция натижаси сифатида қайтарилади.

Бош функцияда файлни яратиш муваффақиятли кечганлиги текширилади ва шунга мос хабар берилади.

Олдиндан берилган катталикларни - объектларни киритиш -чиқариш

C++ тилида киритиш-чиқариш оқимларининг синфлари мавжуд бўлиб, улар киритиш-чиқариш стандарт кутубхонасининг объектга мўлжалланган эквивалентидир. Улар қуйидагилар:

istream - киритиш оқими
ostream - чикариш оқими
iostream - киритиш/чиқариш оқими

Сатрли оқимлар хотирада жойлаштирилган сатрли буферлардан маълумотларни киритиш-чиқариш учун хизмат қилади.

istrstream - сатрли киритиш
ostrstream - сатрли чиқариш
strstream - сатрли киритиш/чиқариш

Қуйидаги файлли окимлар файллар билан ишлаш учун хизмат қилади.

ifstream - файлли киритиш
ofstream - файлли чиқариш
fstream - файлли киритиш/чиқариш

Одатда бу оқимлар include < …..> сифатида ёзилади.

ifstream, ofstream ва fstream оқимлари дастурда файллар ҳосил қилиш, улардаги маълумотлардан фойдаланиш учун ишлатилади. Уларнинг қўлланилиши қуйидагича:

ofstream name (" path\ file_name"); - маълумотли файл ҳосил қилиш, яъни маълумотлар базаси учун очиш;

Масалан: ofstream farruh("c:\ tcpp\bin\d11.dat");
ofstream alibek ("nnn.txt");

Бу ерда

name - иҳтиёрий ном (лотинча);яъни оқим номи. Кейинчалик файлдаги маълумотларни ёзиш ёки ўкиш учун шу номдан фойдаланамиз.
d11.dat ва nnn.txt биз ҳосил қилган файл номлари бўлиб, улар оқим номлари билан боғлангандир.
Энди ҳосил бўлган маълумотлардан фойдаланиш учун уни очишни кўрамиз:
ifstream name ("path");
Масалан: ifstream farruh ("c:\ tcpp\bin\d11.dat");
ifstream alibek("nnn.txt");
Очилган файлларни албатта ёпиш керак! Бу жараённи қуйидагича амалга оширилади: name.close( );
Масалан, farruh.close( ); ёки alibek.close( );
Демак, farruh билан d11.dat, alibek билан nnn.txt номлари ўзаро маълумот алмашинувини таъминлайди.
Масалан:
2та бутун сонни ва уларнинг йиғиндисини ўзида сақловчи файл ҳосил қилинг ва ундан кейинги дастурда фойдаланинг.
Сонларни a, b, йигиндини s, файлни ttt.dat, оқим номини jasur деб атаймиз.

```
# include <iostream.h>
# include <fstream.h>
# include <conio.h>
void main ( )
{
int a=12, b=13, s;
ofstream jasur ("ttt.dat");
s = a + b;
cout <<”s=” << s << endl;
jasur << s <<endl;
jasur.close ( );
getch ( );
}
```

Энди ундан фойдаланамиз:

```
// # include <iostream.h>
# include <fstream.h>
# include <conio.h>
# include <math.h>
void main ( )
{
int   s; float s1;
ifstream jasur ("ttt.dat");
jasur >>s;
s1 = sin (s);
cout <<"s1="<<s1<<endl;
jasur.close ( );
getch( );   }
```

2-мисол. Матрица ва векторлар берилган. Сон қийматлари иҳтиёрий. Ушбу қийматлардан фойдаланиб, матрицани векторга кўпайтириш, матрицанинг изини ҳисоблаш ва векторнинг йиғиндисини ҳисоблаш дастурини тузинг.
Аввал матрица ва векторларнинг сон қийматларини ўзида сақловчи файл ҳосил қиламиз.
Сўнгра бу маълумотлардан фойдаланамиз.

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

```
# include <stdlib.h>
# include <time.h>
void main ( )
{ srand (time (0));
int  a [3][3], b[3], i, j;
ofstream said ("akbar.txt");
for ( i=0; i<3; i++)
{ for (j=0; j<3; j++)
{ a[i][j] = rand( );
said <<a[i][j]; } }
for (i=0; i<3; i++)
{ b[i] = rand( );
said << b[i];  }
said.close ( );
}
# include <iostream.h>
# include <fstream.h>
void main ( )
{
int a[3][3], b[3], i, j, c[3], s1=0, s2=0;
ifstream said ("akbar.txt");
for ( i=0; i<3; i++)
for (j=0; j<3; j++)
said >>a[i][j];
for ( i=0; i<3; i++)
said >> b[i];
for ( i=0; i<3; i++)
{  c[i] = 0;
for (j=0; j<3; j++)
c[i] = c[i] + a[i][j] * b[i];
cout << "c="<<c[i]<<endl; }
for ( i=0; i<3; i++)
s1 = s1 + a[i][i];
for ( i=0; i<3; i++)
s2=s2 + b[i];
cout << "s1="<<s1<<"    s2="<<s2<<endl;
}
```

Назорат саволлари:

1. Файллар билан ишлаш асослари.
2. Символларни ўқиш-ёзиш функциялари
3. Файлдан ўқиш-ёзиш функциялари
4. Файл кўрсаткичини бошқариш функциялари
5. Керакли include ларни кўрсатинг
6. Фойдаланувчининг директивасини яратиш

Тестлар:

1. Файллар билан ишлаш учун дастурга қайси сарлавҳали файлни қўшиш лозим?
   A) stdio.h
   B) iostream.h
   C) conio.h

+D) fstream.h
E) string.h

2. Файл охирини аниқлаш учун қандай функция ишлатилади?
A) bad()
B) fail()
C) flush()
D) tellg()
+E) eof()

3. Файл билан ишлашда хатони аниқлаш учун қандай функция ишлатилади?
A) bad()
+B) fail()
C) flush()
D) tellg()
E) eof()

4. Файлни қўшиш режимида очиб файл кўрсаткичини файл охирига жойлаштирувчи очиш режими қийматини кўрсатинг:
+A) ios::app
B) ios::ate
C) ios::in
D) ios::nocreate
E) ios::noreplace

Маъруза-32 (8 с)

Мавзу: ОБЪЕКТГА ЙЎНАЛТИРИЛГАН ДАСТУРЛАШ АСОСЛАРИ. ОБЪЕКТЛАР ВА СИНФЛАР. СИНФЛАР ВА ТУЗИЛМАЛАР.

Мақсад: Талабаларга объектга йўналтирилган дастурлаш асосларини ўргатиш.

Калит сўзлар: Инкапсуляция, меросхўрлик, полиморфизм, синф, синф хоссалари, синф усуллари.

Режа:
1. **Объектга йўналтирилган программалаш тамойиллари.**
2. **Жойлаштириладиган (inline) функциялар–аъзолар**
3. **Конструкторлар ва деструкторлар**
4. **Синфнинг контанста объектлари ва константа функция-аъзолари**
5. **Синфларни ташкил этиш, объектлар.**
6. **Синф усуллари ва амаллари.**
7. **Синфларни эълон қилиш, очиқ ва ёпиқ синфлар.**
8. **Синф усулларининг аниқланиши.**
9. **Синфнинг статик аъзолари**

Объектга йўналтирилган дастурлаш (ОЙД) – бу программалашга янги бир ёндашувдир. Ҳисоблаш техникасининг ривожланиши ва ечилаётган масалаларни тобора мураккаблашуви программалашнинг турли моделларини (парадигмаларини) юзага келишига сабаб бўлмоқда. Биринчи компиляторларда (масалан, FORTRAN тили) программалашнинг функциялардан фойдаланишга асосланган процедура моделини қўллаб қувватлаган. Бу модел ёрдамида программа тузувчи бир нечта минг қаторли программаларни ёзиши мумкин эди. Ривожланишнинг кейинги босқичида программаларнинг структурали модели пайдо бўлди ва ALGOL, Pascal ва Си тиллар компиляторларида ўз аксини топди. Структурали программалашнинг моҳияти – программани ўзаро боғланган процедуралар (блоклар) ва улар қайта ишлайдиган берилганларнинг мажмуаси деб қарашдан иборат. Ушбу модел программа блоклари кенг қўллашга, GOTO операторидан имкон қадар кам фойдаланишга таянган ва унда программа тузувчи ўн минг қатордан ортиқ программаларни ярата олган. Яратилган программани процедурали моделга нисбатан созлаш ва назорат қилиш осон кечган.

Мураккаб масалаларни ечиш учун программалашнинг янги услубига зарурат пайдо бўлдики, у ОЙП моделида амалга оширилди. ОЙП модели бир нечта таянч концепцияларга асосланади.

Берилганларни абстракциялаш – берилганларни янги турини яратиш имконияти бўлиб, бу турлар билан худди берилганларнинг таянч турлари билан ишлагандек ишлаш мумкин. Одатда янги турларни берилганларнинг абстракт тури дейилади, гарчи уларни соддароқ қилиб «фойдаланувчи томонидан аниқланган тур» деб аташ мумкин.

Инкапсуляция – бу берилганлар ва уларни қайта ишловчи кодни бирлаштириш механизмидир. Инкапсуляция берилганлар ва кодни ташқи таъсирдан сақлаш имконини беради.

Юқоридаги иккита концепцияни амалга ошириш учун C++ тилида *синфлар* ишлатилади. *Синф* термини билан объектлар тури аниқланади. Синфнинг ҳар бир вакили (нусхаси) *объект* деб номланади. Ҳар бир объект ўзининг алоҳида ҳолатига эга бўлади. Объект ҳолати унинг унинг *берилганлар-аъзоларнинг* айни пайтдаги қиймати билан аниқланади. Синф вазифаси унинг *функция-аъзоларининг* синф объектлари устида бажарадиган амаллар имконияти билан аниқланади.

Берилган синф объектини яратиш *конструктор* деб номланувчи махсус функция-аъзо томонидан, ўчириш эса деструктор деб номланувчи махсус функция–аъзо орқали амалга оширилади.

Синф ички берилганларини мурожаатни чеклаб қўйиши мумкин. Чеклов берилганларни очиқ (public), ёпиқ (private) ва ҳимояланган (protected) деб аниқлаш билан тайинланади.

Синф, шу турдаги объектнинг ташқи дунё билан ўзаро боғланиши учун қатъий мулоқат шартларини аниқлайди. Ёпиқ берилганларга ёки кодга фақат шу объект ичида мурожаат қилиш мумкин. Бошқа томондан, очиқ берилганларга ва кодларга, гарчи улар объект ичида аниқланган бўлса ҳам, программанинг ихтиёрий жойидан мурожаат қилиш мумкин ва улар объектни ташқи олам билан мулоқатни яратишга хизмат қилади. Яратилган объектларни, уларни функция–аъзоларига оддийгина мурожаат орқали амалга оширилувчи *хабарлар* (ёки *сўровлар*) ёрдамида бошқариш мумкин. Кейинчалик Windows хабарлари билан адаштирмаслик учун сўров термини ишлатилади.

Ворислик – бу шундай жараёнки, унда бир объект бошқасининг хоссаларини ўзлаштириши мумкин бўлади. Ворислик орқали мавжуд синфлар асосида ҳосилавий синфларни қуриш мумкин бўлади. Ҳосилавий синф (*синф–авлод*) ўзининг она синфидан (*синф-аждод*) берилганлар ва функцияларни ворислик бўйича олади, ҳамда улар қаторига фақат ўзига хос бўлган қирраларни амалга оширишган имкон берувчи берилган ва функцияларни қўшади. Аждод синфдаги ҳимояланган берилган-аъзоларга ва функция-аъзоларга аждод синфда мурожаат қилиш мумкин бўлади. Бундан ташқари, ҳосилавий

синфда она синф функциялари қайта аниқланиши мумкин. Демак, ворислик асосида бир-бири билан «она-бола»муносабатидаги синфлар шажарасини яратиш мумкин.*Таянч синф* термини синфлар шажарасидаги она синф синоними сифатида ишлатилади. Агар объект ўз атрибутларини (берилганлар-аъзолар ва функциялар–аъзолар) фақат битта она синфдан ворислик билан олса, *якка (*ёки *оддий) ворислик* дейилади. Агар объект ўз атрибутларини бир нечта она синфлардан олса, *тўпламли ворислик* дейилади.

Полиморфизм – бу коднинг, бажарилиш пайтидан юзага келадиган ҳолатга боғлиқ равишда ўзини турлича амал қилиш хусусиятидир. Полиморфизм – бу фақат объектлар хусусияти бўлмасдан, балки функциялар-аъзолар хусусиятидир ва улар хусусан, битта номдаги функция-аъзони, ҳар хил турдаги аргументларга эга ва бажаридаган амали унга узатиладиган аргументлар турига боғлиқ бўлган функциялар учун (ўрнида) фойдаланиш имкониятида намоён бўлади. Бу ҳолатга *функцияларни қайта юклаш* дейилади. Полиморфизм амалларга ҳам қўлланиши мумкин, яъни амал мазмуни (натижаси) операнд (берилган) турига боғлиқ бўлади. Полиморфизмнинг бундай турига *амалларни қайта юклаш* дейилади.

Полиморфизм яна бир таърифи қуйидагича: полиморфизм – бу таянч синфга кўрсаткичларнинг (мурожаатларнинг), уларни виртуал функцияларни чақиришдаги турли шакл (қийматларни) қабул қилиш имкониятидир. C++ тилининг бундай имконияти *кечиктирилган боғланиш* натижасидир. Кечиктирилган боғланишда чақириладиган функция-аъзолар адреслари программа бажарилиши жараёнида динамик равишда аниқланади. Анъанавий программалаш тилларида эса бу адреслар статик бўлиб, улар компиляция пайтида аниқланади (*олдиндан боғланиш*). Кечиктирилган боғланиш фақат виртуал функциялар учун ўринли.

Синфлар

Синф тушунчаси C++ тилидаги энг муҳим тушунчалардан биридир. Синф синтаксиси структура синтаксисига ўхшашдир ва унинг кўриниши қуйидагича:

```
class <синф номи>
 {
 // синфнинг ёпиқ берилганлар–аъзолари ва функциялар–
 // аъзолари
   public:
 // синфнинг очиқ берилганлар–аъзолари ва функциялар–
// аъзолари
}
<объектлар рўйхати>
```

Одатда синф тавсифида <объектлар рўйхати> қисми шарт эмас. Синф объектлари кейинчалик, зарурат бўйича эълон қилиниши мумкин. Гарчи <синф номи> қисми ҳам мажбурий бўлмаса ҳам, унинг бўлгани маъқул. Чунки <синф номи> берилганларнинг турининг янги номи бўлиб, унинг ёрдамида шу синф объектлари аниқланади.

Синф ичида эълон қилинган функция ва берилганлар шу синф аъзолари ҳисобаланди. Синф эълонининг ичида эълон қилинган ўзгарувчилар *берилганлар-аъзолар*, синф ичида эълон қилинган функциялар *функциялар–аъзолар* дейилади. Келишув бўйича синф ичидаги барча функция ва ўзгарувчилар шу синф учун ёпиқ ҳисобланади, яъни уларни фақат шу синф аъзолари ишлатиши мумкин. Синфнинг очиқ аъзоларини эълон қилиш учун public калит сўзи ва «:» белгисидан фойдаланилади. Синф эълонидаги public сўзидан кейин эълон қилинган функциялар ва ўзгарувчиларга синфнинг бошқа аъзолари ва программанинг шу синф ишлатилган ихтиёрий жойидан мурожаат қилиш мумкин бўлади.

Синф эълонига мисол:

```
class Sinf_1
```

```
{
// синфнингёпиқэлементи
int a;
public:
int get_a();
void set_a(int_num);
}
```

Гарчи int get_a()ва void set_a(int_num) функциялари Sinf_1 синф ичида эълон қилинган бўлса ҳам, улар ҳали аниқлангани йўқ. Функцияни аниқлаш учун синф номи ва «::» белгиларини ёзиш орқали амалга оширилади. Бу ерда «::» – *кўриш соҳасини кенгайтириш амали* дейилади. Функция-аъзони аниқлашнинг умумий шакли қуйидагича:

```
<тур><синф номи>::<функция номи> (<параметрлар рўйхати>)
{
  // функция танаси
}
```

Юқорида эълон қилинган Sinf_1 синфнинг int get_a() ва void set_a(int_num) функция-аъзолари аниқлашга мисол келтирилган:

```
int Sinf_1 :: get_a()
{ return a;}
void Sinf_1:: set_a(int num) {a=num;}
```

Sinf_1 синфини эълон қилиш шу синф туридаги объектларини юзага келтирмайди. Синф объектларини юзага келтириш учун синф номини берилганлар тури спецификатори сифатида ишлатиш зарур бўлади. Масалан,

```
Sinf_1 obj1,obj2;
```

Синф объекти яратилгандан кейин «.» ёрдамида синфнинг очиқ аъзоларига мурожаат қилиш мумкин бўлади. Мисол учун

```
obj1.set_a(20);
obj2.set_a(50);
```

мурожаатлар орқали obj1 ва obj2 объектларнинг a ўзгарувчиларига қийматлар берилади. Ҳар бир объект синфда эълон қилинган ўзгарувчиларнинг ўз нусхаларига эга бўлади. Шу сабабли, obj1 объектидаги a ўзгарувчи obj2 объектдаги a ўзгарувчидан фарқ қилади.

Синф элементларига кўрсаткичлар орқали ҳам амалга ошириш мумкин. Қуйидаги мисол буни намоён этади.

```
class Nuqta
{
 public:
int x,y;
void Koord_Qiymat_Berish( int _x, int _y);
};
voidNuqta::Koord_Qiymat_Berish(int _x, int _y)
{
 x= _x; y=_y;
}
int main()
{
Nuqta  x0y;
Nuqta * Koord_kurssatgich= & x0y;
 //…
 x0y.x=0;
 Koord_kursatgich –> y=0;
```

```
 Koord_Kursatgich –> Koord_Qiymat_Berish(10,15);
 //…
return 0;
}
```

Шуни қайд қилиш керакки, синф объектларига “.” ва ”–>” орқали мурожаат қилишнинг компиляция нуқтаи-назаридан ҳеч бир фарқи йўқ. Компилятор “.” билан мурожаатни ”–>” билан алмаштиради. Масалан,

obj1.set_a(20);

кўрсатмаси компилятор томонидан

(&obj1)–> set_a(20);

кўринишидаги кўрсатма билан алмаштирилади.

С++ тили синфнинг берилганлар-аъзоларига маълум бир чекланишлар қўяди:

- берилганлар–аъзолар auto, extern ёки register модификаторлари билан аниқланиши мумкин эмас;
- синфнинг берилганлар–аъзолари шу синф туридаги объект бўлиши мумкин эмас, лекин улар шу синфга кўрсаткич ёки мурожаат(&) бўлиши, бошқа синф объекти бўлиши мумкин.

Синф, унинг аъзолари ишлатилишидан олдин эълон қилинган бўлиши керак. Бироқ, айрим ҳолларда синфда ҳали эълон қилинмаган синфга кўрсаткич ёки мурожаат (&) эълон қилишга зарурат бўлиши мумкин. Бу ҳолда синфнинг тўлиқ бўлмаган эълонидан фойдаланишга тўғри келади. Синфнинг тўлиқ бўлмаган эълони қуйидаги кўринишга эга:

class <синф номи>;

Мисол кўрайлик.

```
class Sinf2; // синфнинг тўлиқмас эълони
class Sinf1
{
int x;
  Sinf2 * sinf2;  // sinf2 синфига кўрсаткич
public:
Sinf1(int _x) {x=_x;}
};
int main()
{
 //…
return 0;
}
class Sinf2  // Sinf2 синфининг тўлиқ эълони
{
int a;
public:
Sinf2();
};
```

Шуни қайд этиш керакки, синф эълони структура эълонига ўхшаш, фарқли равишда:

- синф эълонида public, protected ёки private мурожаат модификаторлари ишлатилади;
- struct калит сўзи ўрнида class ёки union калит сўзлари ишлатилиши мумкин;
- одатда синф таркибида берилганлардан ташқари функция-аъзолар киради;
- синф конструктор ёки деструктор деб номланувчи махсус функция-аъзоларига эга бўлади.

Қуйида struct ва union калит сўзлари билан аниқланган синфларга мисол келтирилган.

```
struct Nuqta
{
private:
int x; int y;
public:
int Olish_X();
int Olish_Y();
void Qiymat_Berish_X(int _x);
void Qiymat_Berish_Y(int _y);
};
union Bit
{
Bit(unsigned int n);
void Bit_Chop_Qilish();
unsigned int num;
unsigned char c [sizeof(unsigned int)];
};
```

Мурожаат спецификатори, ундан кейин жойлашган барча синф элементларига қўлланилади, токи бошқа спецификатор учрамагунча ёки синф эълони тугамагунча.

Қуйида спецификаторлар тавсифи келтирилган.

Синфга мурожаат спецификаторлари:

private - берилганлар-аъзоларга ва функциялар-аъзоларга фақат шу синф функциялар-аъзолари мурожаат қилиши (ишлатиши) мумкин;

protected-берилганлар-аъзоларга ва функциялар-аъзоларга фақат шу синф ваш у синфдан ҳосил бўлган синфлар функциялар-аъзолари мурожаат қилиши (ишлатиши) мумкин;

public - берилганлар-аъзоларга ва функциялар-аъзоларга фақат шу синф функция-аъзолари ва синф объекти мавжуд бўлган программа функциялари мурожаат қилиши (ишлатиши) мумкин;

С++ тилида структура ва бирлашмалар синф турлари деб қаралади. Структура ва синфлар бир-бирига ўхшаш, фақат келишув бўйича мурожаат билан фарқ қилади: структурада келишув бўйича барча элементлар public мурожаатига эга бўлса, синфда улар private мурожаатида бўлади. Бирлашмада ҳам, худди структурадек келишув бўйича элементлар public мурожаатда бўлади.

Структура ва бирлашмалар ўз конструктор ва деструкторларига эга бўлиши мумкин. Шу билан биргаликда бирлашмаларни синф сифатида ишлатишга маълум бир чекловлар мавжуд. Биринчидан, улар бирорта синф вориси бўлиши мумкин эмас, ўзлари ҳам бошқа синфлар учун она синф бўла олмайди. Улар static атрибутли аъзоларга, ҳамда конструктор ва деструкторли объектларга эга бўлиши мумкин эмас.

Бирлашмани синф сифатида ишлатилишига мисол кўрайлик.

```
#include <iostream.h>
union Bit
{
Bit(unsigned int n);
void Bit_Chop_Qilish();
unsigned int num;
unsigned char c[sizeof(unsigned int)];
};
Bit::Bit(unsigned int n){num=n;};
void Bit::Bit_Chop_Qilish( )
```

```
{
int i,j;
for (j=sizeof(unsigned int)-1; j>=0; j--)
 {
 cout<<"Байтнинг иккилик кўриниши "<<j<<':';
for (i=128; i; i>>=1)
  {
if (i & c[j]) cout<<'1';
else cout<<'0';
}
cout<<endl;  }
}

int main( )
 {
 Bit bit(2000);
  Bit.Bit_Chop_Qilish();
return 0;
 }
```

Бу мисолда объектга берилган ишорасиз бутун қийматнинг иккилик кўриниши чоп этилади.

Конструкторлар ва деструкторлар

Объектни яратишда уни инициализациялаш керак.Бу мақсадда C++ тилида конструктор деб номланувчи махсус функция-аъзо аниқланган. Синф конструктори ҳар сафар синф объекти яратилиши пайтида чақирилади. Конструктор номи ўзи аъзо бўлган синф номи билан устма–уст тушади ва қайтарувчи қийматга эга бўлмайди. Масалан,

```
#include <iostream.h>
class Sinf
{
  int var;
 public:
  Sinf(); // Конструктор
  void Chop_etish_var();
 };
Sinf::Sinf()
{
 cout<< "Конструктор ишлади \n";
 var=0;
}
Sinf::Chop_etish_var(){cout<<var;}
int main()
{
 Sinf ob;
ob.Chop_Etish_var();
//…
 return 0;
}
```

Бу мисолда Sinf конструктори экранга хабар чиқаради ва ёпиқ var ўзгарувчини инициализациялайди.

Шуни қайд этиш керакки, программа тузувчи конструктор чақирадиган код ёзмаслиги керак.Барча зарур ишни компилятор амалга оширади. Юқорида қайд қилингандек конструктор у тегишли синф объекти яратилаётган пайтда чақирилади.Ўз навбатида объект бу объектни эълон қилувчи оператор бажарилишида яратилади. Шунинг учун ҳам С++ тилида ўзгарувчини эълон қилувчи оператор бажарилувчи оператор ҳисобланади.

Глобал объектлар учун конструктор программа бажарилиши бошланганда чақирилади. Локал объектлар учун конструктор ўзгарувчи эълонининг ҳар бир бажарилишида чақирилади.

Конструкторга нисбатан тескари амал бажарадиган функция-аъзоларга деструкторлар дейилади. Бу функция-аъзо объект ўчирилишида чақирилади. Одатда деструктор объект томонидан эгалланган хотирани бўшатиш учун хизмат қилади.Унинг номи синф номи билан мос тушади, фақат олдига ‘~’ белгиси қўйилади.

Қуйида деструктор аниқланган синфга мисол келтирилган.

```
#include <iostream.h>
class Sinf;
{
int var;
 public:
  Sinf(); // Конструктор
 ~Sinf(); // Деструктор
  void Chop_etish_var();
  Sinf::Sinf()
  {
 cout<< “Конструктор ишлади \n”;
   var=0;
}
  Sinf::~Sinf( )
  {
 cout<< “Деструктор ишлади \n”;  }
void Sinf::Chop_etish_var( )
{
 cout<<var<<endl;
}

int main()
{
 Sinf ob;
ob.Chop_Etish_var();
//…
 return 0;
}
```

Деструктор объект ўчирилишида чақирилади. Глобал объектлар программа тугашида ўчирилади. Локал объектлар – уларни кўриш соҳасидан чиқишда ўчирилади.

Шуни алоҳида қайд этиш керакки, конструктор ва деструкторларга кўрсаткичлар ҳосил қилиш мумкин эмас.

Агар синф ўзгарувчиларини инициализация қилиш зарур бўлса, параметрли конструктор ишлатилди. Юқорида келтирилган мисолга ўзгартириш киритамиз.

```
#include <iostream.h>
class Sinf
{
  int a, b;
```

```
 public:
  Sinf(int x, int y);     // Конструктор
 ~Sinf();                          // Деструктор
  void Chop_etish_var();
}
Sinf::Sinf(int x, int y)
{
 cout<< “Конструктор ишлади \n”;
a=x; b=y;
}
Sinf::~Sinf()
{
 cout<< “Деструктор ишлади \n”;
}

void Sinf::Chop_etish_var()
{
 cout<<a<<b<<endl;
}
int main()
{
Sinf ob(5,10);
ob.Chop_Etish_var();
//…
 return 0;
}
```

Бу ерда ob объекти эълонида конструкторга узатилган қийматлар синф таркибидаги a ва b ёпиқ ўзгарувчиларни инициализация қилишда ишлатилади.

Параметрли конструкторга қиймат узатиш синтаксиси қуйидаги ифоданинг қисқарган шакли ҳисобланади:

“Sinf ob(5,10); “

ифодаси

“Sinf(5,10);“

ифодаси билан эквивалент. Аксарият ҳолларда қисқартирилган шакл ишлатилади.

Конструктордан фарқли равишда деструктор параметрга эга бўлиши мумкин эмас, чунки ўчирилаётган объект ўзгарувчиларига қиймат бериш маъноga эга эмас.

Конструктор учун аниқланган бир нечта қоидаларни келтирамиз:

- конструктор учун қайтарилувчи қиймат тури кўрсатилмайди;
- конструктор қиймат қайтармайди;
- конструктор ворислик билан ўтмайди;
- конструктор const, volatile, static ёки virtual модификаторлари билан эълон қилинмайди.

Агар синф аниқланишида конструктор эълон қилинмаса, компилятор ўзи келишув бўйича параметрсиз конструкторни ҳосил қилади.

Деструктор учун қуйидаги қоидалар аниқланган:

- деструктор параметрларга бўлиши мумкин эмас;
- деструктор қиймат қайтармайди;
- деструктор ворислик билан ўтмайди;
- синф биттадан ортиқ деструкторга эга бўлиши мумкин эмас;
- деструктор const, volatile, static ёки virtual модификаторлари билан эълон қилинмайди.

Агар синфда деструктор эълон қилинмаса, компилятор ўзи келишув бўйича конструкторни ҳосил қилади.

Одатда синф берилганлари-аъзолари конструктор танасида инициализацияланади. Лекин инициализациянинг бошқа усул билан – *элементларни инициализациялаш рўйхати* орқали амалга ошириш мумкин. Элементларни инициализациялаш рўйхати функция сарлавҳаси аниқланишидан кейин икки нуқта (‘:’) кейин жойлашади ва унда вергул билан ажратилган ҳолда берилганлар-аъзолар ва таянч синфлар ёзилади. Ҳар бир элемент учун қавс ичида инициализацияда ишлатиладиган бир ёки бир нечта параметрлар кўрсатилади. Қуйидаги мисолда элементларни инициализациялаш рўйхати орқали синф ўзгарувчиларини инициализация қилиш кўрсатилган.

```
        class Sinf
        {
int a, b;
         public:
          Sinf(int x, int y); // Конструктор
        };
        Sinf:: Sinf( int x, int y): a(x), b(y);
   {
    cout<< “Конструктор ишлади \n”;
   }
        //…
```

Келтирилган программа бўлагида конструктор бажарадиган иш олдинги мисолдаги конструктор иши билан эквивалент.

Албатта, синф элементларини инициализация қилишнинг қайси шаклини қўллаш программа тузувчига боғлиқ. Бироқ, шундай ҳолатлар бўладики, унда элементларни инициализациялаш рўйхатидан фойдаланмасликнинг иложи йўқ: синфнинг берилганлар–константаларига ва кўрсаткичларига бошланғич қиймат беришда; синф аъзоси объект бўлганда ва бу объект конструктори бир ёки бир нечта параметрларга қиймат беришни талаб қилган ҳолларда.

Келишув бўйича конструкторларни ишлатишда коллизиядан (бир нарсани икки хил тушунишдан) қочиш керак, яъни синфда бир нечта конструктор бўлганда компилятор қачон уларнинг қайси бири чақирилишини аниқ билиши керак. Қуйидаги мисолда бу ҳолат билан боғлиқ хато кўрсатилган.

```
        class S
        {
    public:
     S();                  // Келишув бўйича конструктор
          S(int i=0);      // Келишув бўйича конструктор ўрнида
                                //ишлатилиши мумкин бўлган конструктор
        };
        int main()
        {
   S ob1(10); // S::S(int) конструктори ишлатилади.
        Sob2;            //Нотўғри. S::S(int)ёки S::S()
                                // конструкторларининг қайси бири
                                // чақирилиши номаълум.
```

Бу зиддиятни ҳал қилиш йўли – бу синф эълонидан келишув бўйича конструкторни ўчиришдир.

Нусхалаш конструктори

Нусхалаш конструктори синф объектини яратади ва шу синфнинг мавжуд объектларидан берилганларни (уларнинг қийматини) нусхасини олади.Шу сабабли у синф

объектига константа кўрсаткич (const S&) ёки оддийгина кўрсаткич (S&) бўлган ягона параметрга эга бўлади. Параметрларнинг биринчисини ишлатиш маъқул ҳисобланади, у константа объектларни нусхалаш имконини беради.

Қуйидаги мисолда нусхалаш конструкторини ишлатиш кўрсатилган.

```
classNuqta
{
int x,y;
public:
Nuqta(const Nuqta& koor);
Nuqta(int x0, int y0);
void Abstissa();
void Ordinata();
void Uzgartirish (int delta_x, int delta_y);
};
Nuqta::Nuqta(const Nuqta& koor)
{
 x = koor.x;  y = koor.y;
};
Nuqta::Nuqta(int x0, int y0){x = x0; y = y0;};
voidNuqta::Abstissa(){cout<<"x="<<x<<endl;};
voidNuqta::Ordinata(){cout<<"y="<<y<<endl;};
voidNuqta::Uzgartirish(int delta_x, int delta_y)
{
 x+=delta_x;  y+=delta_y;
};
int main( )
{
Nuqtakoord1(5,10);
Nuqta koord2=koord1;
Nuqtakoord3(koord1);
 koord1.Uzgartirish (3, -2);
 koord2.Uzgartirish (1, 2);
 cout<<” koord1 объект берилган-аъзолари қиймати:\n”;
koord1.Abstissa();
koord1.Ordinata();
 cout<<” koord2 объект берилган-аъзолари қиймати:\n”;
koord2.Abstissa();
koord2.Ordinata();
 cout<<” koord3 объект берилган-аъзолари қиймати:\n”;
koord3.Abstissa();
koord3.Ordinata();
return 0;
}
```

Программа ишлаши натижасида экранга қуйидагилар чоп этилади:

koord1объект берилган-аъзолари қиймати:

x=8

y=8

koord2 объект берилган-аъзолари қиймати:

x=6

y=12

koord3 объект берилган-аъзолари қиймати:

x=5
y=10

this кўрсаткичи

С++ тилидаги ҳар бир объект компилятор томонидан яратиладиган ва объектга кўрсатувчи this деб номланувчи махсус кўрсаткичга эга. this кўрсаткичининг тури S* бўлиб, бу ердаги S – ушбу объект синфи туридир. this кўрсаткичи синфда аниқланганлиги учун унинг амал қилиш соҳаси ўзи аниқланган синф бўлади. Бошқача айтганда, this кўрсаткичини компилятор томонидан қўшилувчи синфнинг яширинган параметри деб қараш мумкин. Синф функция-аъзоси чақирилганда унга this кўрсаткичи, гўёки биринчи аргумент сифатида узатилади. Яъни функция-аъзони чақиришнинг қуйидаги кўриниши

Obekt1.Funktsiya(arg1,arg2);

компилятор томонидан

Obekt1.Funktsiya(&Obekt1, arg1,arg2);

кўринишида талқин қилинади.

Функция чақирилганда унинг қавс ичидаги аргументлари стекка ўнгдан чапга томонга қараб жойлаштирилди. Биринчи аргумент (this) стекка энг охирда жойлашади. Функция–аъзо ичида объектлар адреси this орқали аниқланади. Қуйида this кўрсаткичини ишлатиш билан боғлиқ программа матнини келтирилган.

```
#include <iostream.h>
#include <string.h>
class S
{
char Ism[20];
public:
S(char*);
void Salom();
};
S::S(char * _Ism)
{
strcpy(Ism, _Ism);
Salom();                    // учта мурожаат
this -> Salom();      // ўзаро
 (*this).Salom();     // эквивалент
}
void S::Salom()
{
cout<<"Salom "<<Ism<<'\n';
cout<<"Salom "<<this ->Ism<<'\n';
}
int main( )
{
S obekt("Muqumov Rustam");
return 0;
}
```

Программа матнидан кўриниб турибдики, функция-аъзо ичида бошқа функция-аъзоларга ва берилганлар–аъзоларга бевосита уларнинг номлари билан ёки this кўрсаткичи орқали мурожаат қилиш мумкин. Шу сабабли амалиётда this кўрсаткичидан кам фойдаланилади. Асосан, this кўрсаткичи функция қайтарувчи қиймати сифатида

(return this; ёки return *this;) ва операторларни қайта юклаш масалаларида кенг қўлланилади.

Жойлаштириладиган (inline) функциялар–аъзолар

Функциялар бўлимида inline функциялар ҳақида маълумот берилган эди.Уларнинг бошқа функциялардан фарқи – компилятор программада бундай функциялар чақирилган жойга функцияни чақириш буйруғини эмас, балки функция танасини қўяди. inline функцияларнинг афзаллиги шунда эдики, программада оддий функцияни чақиришдаги аргументларни узатиш, стекка қийматларни жойлаштириш ва қайта олиш билан боғлиққўшимча амаллар бажарилмайди. Ноқулай томони, inline функцияга мурожатлар кўп бўлганда программа ҳажми ошиб кетади. Бу ўринда inline функциялар қўлланишини макросларни ишлатишга ўхшатиш мумкин. Шу сабабли, одатда жуда содда функциялар inline функциялар сифатида ишлатилади. Функциялар аниқланишда inline спецификаторини қўйишни компиляторга функция чақирилган жойларга функция танасини қўйишга талаб деб қаралади. Агар компилятор функция танасини қўйишни амалга ошира олмаса, inline функция оддий функция сифатида ишлатилади. Компилятор қуйидаги ҳолатларда inline функцияни чақирув жойига қўя олмайди, агарда функция:

- танасида такрорлаш операторлари ишлатилган бўлса (for, while, do while);
- танасида switch, goto операторлари бўлса;
- static ўзгарувчиларни ишлатса;
- рекурсив бўлса;
- void туридан фарқи қайтарувчи турга эга ва танасида return оператори бўлмаса;
- танасида ассемблер кодли бўлаклар мавжуд бўлса.

Жойлаштириладиган функциялар сифатида нафақат оддий функциялар, балки синфнинг функция– аъзолари ҳам аниқланиши мумкин. Бунинг учун синф эълонида функция–аъзони эълонига унинг аниқланишини қўйиш етарли ёки синфдан ташқаридаги функция аниқланишида, функция сарлавҳаси олдига inline спецификаторини қўйиш зарур бўлади. Қуйида келтирилган мисолда жойлаштирилувчи функция – аъзоларининг икки хил вариантдаги эълони кўрсатилган.

```
class Nuqta
  {
        int x;
int y;
public :
   Get_x(){return x;}
   Get_y(){return y;}
void Set_x(int _x);
void Set_y(int _y);
 }
inline void Nuqta::Set_x(int _x) {x= _x;}
inline void Nuqta::Set_y(int _y) {y= _y;}
```

Синфнинг статик аъзолари

Синф аъзолари static модификатори билан эълон қилиниши мумкин. Синф статик аъзосини синф соҳаси чегарасида мурожаат қилиш мумкин бўлган глобал ўзгарувчи ёки функция деб қараш мумкин. Синфнинг static деб эълон қилинган берилганлар-аъзолари синфнинг барч аобъектлари томонидан биргаликда ишлатилади,чунки бундай ўзгарувчининг ягона нусхаси бўлади. Амалда синфнинг статик берилганлари учун

хотирадан жой ажратилади, ҳаттоки синфнинг бирорта объекти бўлмасаҳам. Шу сабабли синф статик берилганини эълон қилиб қолмасдан, уни аниқлаш шарт. Масалан:

```
        class Sinf
{
public:
 Sinf();
static int Sanagich;//статик берилган–аъзо эълони
    }
        int Sinf::Sanagich=0; // статик берилган–аъзо эълони
```

Бу мисолда, гарчи Sanagich статик берилган – аъзо public бўлимида эълон қилинган синф объекти номини ишлатиш ёрдамида мурожаат қилиш мумкин.

```
        Sinf  sinf1;
        Sinf1.Sanagich++;
        Sinf sinf2;
        Sinf2->Sanagich--;
```

Статик берилганлар – аъзоларга синф номи орқали мурожаат қилган маъқул бўлади.

```
        Sinf::Sanagich++;
```

Бу ҳолат Sanagich статик берилган–аъзо барча синф объектлари учун ягона эканлигини таъкидлайди.

Агарда статик берилганлар ёпиқ деб эълон қилинган бўлса, уларга функциялар – аъзолар орқали мурожаат қилиш мумкин.

Умуман олганда статик берилганлар–аъзоларни ишлатишда қуйидаги таклифларни бериш мумкин:

- статик берилганлар-аъзоларни бир нечта синф объектлари томонидан биргаликда ишлатиш учун аниқланг;
- статик берилганлар-аъзоларини private, protected модификаторлар билан эълон қилиш орқали уларга мурожаатни чекланг.

Синфнинг статик берилган-аъзосини ишлатишга мисол.

```
        class S;
        {
public:
S() {ob_soni++;}
    ~S() {ob_soni--;}
static int ob_soni ;
private:
int x;
  };
int S::ob_soni=0;
int main ()
  {
   S* p_ob=new S[5];
   cout<<"Sinfning "<<S::ob_soni<<" ob'ekti mavjud./n";
delete [] p_ob;
return 0;
  }
```

Программа ишлаш натижасида экранга

Sinfning 5 ob'ekti mavjud.

Синфнинг статик функцияларни ишлатишнинг ўзига хослиги шундаки, улар ҳам ягона нусхада аниқланади ва бирорта синф объектининг "шахсий" функцияси бўлмайди. Шу сабабли, бу функцияларга this кўрсаткичи узатилмайди. Статик функцияларнинг бундай хусусиятидан Windows учун программалашда кенг фойдаланилади.

Юқорида айтилган фикрлардан бир нечта муҳим хулосалар келиб чиқади:

– статик функция –аъзоларни синфнинг бирорта ҳам вакили (объекти) мавжуд бўлмаса ҳам чақириш мумкин;
– синфнинг статик функцияси фақат синфнинг статик берилганларини қайта ишлаши мумкин ва фақат синфнинг статик функция–аъзоларини чақириши мумкин;
– статик функция-аъзо virtual модификатори билан эълон қилиниши мумкин эмас.

Қуйида келтирилган программа функция–аъзони ишлатишга мисол бўлади:

```
#include <iostream.h>
class S
 {
public:
S() {sanagich++};
          ~S() {sanagich--};
          //..
        static int Sinf_Sanagichi(){return sanagich;}
private:
int x;
static int sanagich;
 };
int S::sanagich=0;

int main()
 {
  S * pOb= new S[10];
cout<<"S sinfning "<<S::Sinf_Sanagichi()
<<" ob'ekti mavjud"<<endl;
delete [] pOb;
return 0;
 }
```

Синфнинг контанста объектлари ва константа функция-аъзолари

Синфнинг функция-аъзолари параметрлар рўйхатидан кейин келувчи const модификатори билан эълон қилиниши мумкин. Бундай функция синф берилганлар-аъзолари қийматларини ўзгартира олмайди ва синфнинг константа бўлмаган функция–аъзоларини чақириши мумкин эмас.Константа функция-аъзоси бўлган синфга мисол келтирамиз.

```
classNuqta
 {
int x,y;
        public:
Nuqta(int _x, int _y);
void Qiymat_Berish(int x0, int y0);
//константафункция-аъзо
void Qiymat_Olish(int xx, int yy) const;
};
Nuqta::Nuqta(int _x, int _y)
{ x=_x; y=_y;}
voidNuqta::Qiymat_Berish(int x0, int y0)
{ x=x0; y=y0;}
voidNuqta::Qiymat_Olish(int &xx, int &yy)const
{ xx=x; yy=y; }
```

Худди шундай, константа объектлар яратиш мумкин. Бунинг учун объект эълони олдига const модификатори қўйилиши керак.

const Nuqta koord(3,6);

const калит сўзи компиляторга ушбу объектнинг ҳолати ўзгармаслиги кераклигини билдиради. Шу сабабли объект берилган-азъолари қийматини ўзгартирадиган функция–аъзосини чақириши учраса, компилятор хато ҳақида хабар беради. Бу қоидага константа функция-аъзоларни чақириш риоя қилмайди, чунки ўз мазмунига кўра улар берилган–аъзоларни қийматини ўзгартира олмайди. Юқорида эълон қилинган Nuqta синфини константа объект ишлатишига мисол.

```
// Nuqta синф эълони ва аниқланиши
int main()
 {
Nuqtanuqta1(3,7);
constNuqta nuqta2(8,10);//Константаобъект
int a,b;
  nuqta1.Qiymat_Olish (a,b);
  nuqta2. Qiymat_Berish(2,3);// Хато
nuqta2. Qiymat_Olish(a,b); // Тўғри
  return 0;
 }
```

Константа функция–аъзоларнинг берилганлар–аъзолар билан ишлаш билан боғлиқ чекловларни “айланиб ўтиш” учун mutable калит сўзи аниқланган. Бу калит сўз синфнинг қайси берилган-аъзоси константа функция-аъзолар томонидан ўзгартирилиши мумкин эканлигини кўрсатади. Статик ва константа берилганлар-аъзоларига mutable калит сўзини ишлатиш мумкин эмас, у берилганлар туринг модификатори сифатида ишлатилади.

Мисол.

```
#include <iostream.h>
class Sinf
{
mutable int count;
mutable const int * intPtr;//Ўринли, гарчи
// кўрсаткич константа бутун сонга кўрсатса
// ҳам ўзи константа эмас
public:
int Funktsiya(int i=0) const
  {
count=i++;
intPtr =&i;
cout<<*intPtr;
return count;
  }
};
int main()
{
 Sinf S;
S.Funktsiya();
return 0;
}
```

Программа ишлаши натижасида экранга

1

чоп этилади.

Шу вактгача тузган дастурларимиз берилган маълумотлар устида бирор-бир амалларни бажарувчи процедуралар кетма-кетлигидан иборат эди. Процедура ва функциялар ҳам ўзида аниқланган кетма-кет командалар тўпламидан иборатдир.

Объектга йўналтирилган дастурлашнинг асосий мақсади берилганлар ва улар устида амал бажарувчи процедураларни ягона объект деб қарашдан иборатдир. Масалан: нон, автомобиль, одам … объектлари.

С++ тили Объектга йўналтирилган дастурлаш принципларини қўллаб қувватлайди. Бу принциплар қуйидагилардан иборат:

1. Инкапсуляция
2. Меросхўрлик
3. Полиморфизм

Инкапсуляция. Агар мухандис ишлаб чиқаришда диод, триод ёки резисторни ишлатса, у бу элементларни янгитдан иҳтиро қилмайди, балки дўкондан сотиб олади. Демак, мухандис уларнинг қандай тузилганлигига эътиборини қаратмайди, бу элементлар яхши ишласа етарли. Айнан шу ташқи конструкцияда ишлайдиган яширинлик ёки автономлик хоссаси инкапсуляция дейилади.

Меросхурлик. Янги объект яратилаётган бўлса, иккита вариантдан бири танланади: мутлақо янгисини яратиш ёки мавжуд моделнинг конструкциясини такомиллаштиришдир. Кўпинча 2-вариант танланади, демак, баъзи хусусиятлари ўзгартирилади холос. Бу нарса меросхўрлик принципига асос солади. Янги синф олдин мавжуд бўлган синфни кенгайтиришдан ҳосил бўлади. Бунда янги синф олдинги синфнинг меросхўри деб аталади.

Полиморфизм. Поли – кўп, морфе – шакл деган маънони билдиради. С++ тили бир хил номдаги функция турли объектлар томонидан ишлатилганда турли амалларни бажариш имкониятини таъминлайди. Полиморфизм – шаклнинг кўп хиллигидир.

Дастурда ишлатиладиган ҳар бир ўзгарувчи ўз тоифасига эга ва у қуйидагиларни аниқлайди:

1. хотирадаги ўлчовини;
2. унда сакланаётган маълумотларни;
3. унинг ёрдамида бажарилиши мумкин булган амалларни.

С++ тилида дастурчи ўзига керакли иҳтиёрий тоифани ҳосил қилиши мумкин. Бу янги тоифа ички тоифаларнинг хоссалари ва уларнинг функционал имкониятларини ўзида ифодалайди. Янги тоифа синфни эълон қилиш орқали тузилади. Синф бу – бир-бири билан функционал боғанган ўзгарувчилар ва усуллар (функциялар) тўпламидир.

Масалан: Мушук номли синф тузмоқчимиз. Бу ерда унинг ёши, оғирлиги каби ўзгарувчилар ва миёвлаш, сичқон тутиш каби функциялардан ишдатилади. Ёки Машина синфи ғилдирак, эшик, ўриндиқ, ойна каби ўзгарувчилар ва хайдаш, тўхтатиш каби функциялардан иборат.

Синфдаги ўзгарувчилар – синф аъзолари ёки синф хоссалари дейилади.

Синфдаги функциялар одатда ўзгарувчилар устида бирор бир амал бажаради. Уларни синф усуллари (методлари) деб ҳам аталади.

Синфни эълон килиш учун **class** сўзи , { } ичида эса шу синфнинг аъзолари ва усуллари келтирилади. Масалан:

```
class  non
{  int  ogirlik ;
   int  baho ;
```

```
    void  yasash ( );
    void  yopish ( );
    void  eyish ( );
}
```

Синфни эълон қилишда хотира ажратилмайди. Синф эълон қилинганда компилятор фақат шундай (non) синф борлигини, ҳамда унда қандай қийматлар (ogirlik, baho) сақланиши мумкинлигини, улар ёрдамида қандай амалларни (yasash, yopish, yeyish) бажариш мумкинлиги ҳақида хабар беради. Бу синф объекти ҳаммаси бўлиб 4 байт жой эгаллайди (2та int).

Объект синфнинг бирор бир нусхаси ҳисобланади.

С++ тилида тоифаларга қиймат ўзлаштирилмайди, балки ўзгарувчига ўзлаштирилади. Шунинг учун тўғридан-тўғри int = 55 деб ёзиб бўлмаганидек non.baho=1200 деб ҳам бўлмайди. Ўзлаштиришда хатоликка йўл қўймаслик учун олдин non синфига тегишли patir объектини ҳосил қиламиз кейин эса унга керакли қийматларни берамиз.

Масалан:

```
int  a;          // бутун тоифали а ўзгарувчиси, объекти
non  patir;  //
```

Энди non синфининг реал объекти аниқланганидан сўнг унинг аъзоларига мурожаат

```
patir.baho = 1200;
patir.ogirlik = 500;
patir.yasash ( ) ;
```

Синфни эълон қилишда қуйидагилардан фойдаланилади:

public - очик

private – ёпик

Синфнинг барча усул ва аъзолари бошланғич ҳолда автоматик равишда ёпиқ бўлади.ёпиқ аъзоларга эса фақат шу синфнинг усуллари орқалигини мурожаат қилиш мумкин. Объектнинг очик аъзоларига эса дастурдаги барча функциялар мурожаат қилиши мумкин. Лекин синф аъзоларига мурожаат қилиш анча мушкул иш ҳисобланади. Агар тўғридан тўғри:

```
non  patir;
patir.baho = 1200;
patir.og`irlik = 500;
```

деб ёзсак хато бўлади.

Аъзоларга мурожаат қилишдан олдин уни очиқ деб эълон қилиш керак:

```
# include < iostream.h >
class  non
{  public :
    int  baho;
    int  og`irlik;
    void yasash ( );    };
void  main ( )
{ non  patir;
patir.baho = 1200;   patir.og`irlik = 500;
cout <<"men olgan patir "<<patir.baho <<" so'm"<<endl;
cout <<" uning og'irligi ="<<patir.og`irlik <<endl ;     }
```

Синф усулларининг аниқланиши.

Синф усулини аниқлаш учун синф номидан сўнг иккита икки нуқта (::) белгиси, функция номи ва унинг параметрлари кўрсатилади.

Масалан: non синфининг patir объектидаги усулларни аниқлаш дастури:

```
# include < iostream.h >
class  non
{  public :
   int  baho;
   int  og`irlik;
   void yasash ( );   };
void  non :: yasash ( );
{  cout << "hamir qoriladi, zuvala tutiladi, doira holiga keltiriladi"<< endl;  }
void  main ( )
{ non  patir;
patir.baho = 1200;   patir.og`irlik = 500;
cout <<"men olgan patir "<<patir.baho <<" so'm"<<endl;
cout <<" uning og'irligi ="<<patir.og`irlik <<endl ;
cout <<" u quyidagich yasaladi:";
patir . yasash ( );   }
```

Назорат саволлари :

1. Инкапсуляция нима?
2. Полиморфизм ҳақида тушунча.
3. Ворисликнинг қўлланиши.
4. Коструктурлик ва деструктурлик?
5. Синфнинг статик аъзолари
6. Синфлар ?
7. Синф хоссалари ва усуллари?
8. Синф усулларининг аниқланиши.

Тестлар:

1. Ворислик бу:
   A) бир объектга бошқа объект нусхасини қўшиш
   B) бир объектга бошқа объектга илова қўшиш
   +C) бир синфга бошқа синф функционаллигини қўшиш
   D) синф усулларини қайта таърифлаш
   E) ягона объектда маълумотлар ва функцияларни жамлаш

2. Инкапсуляция бу:
   +A) ягона объектда маълумотлар ва функцияларни жамлаш
   B) бир объектга бошқа объект нусхасини қўшиш
   C) ягона объектда маълумотлар ва шу маълумотларга кўрсаткичларни жамлаш
   D) ягона объектда маълумотлар ва шу маълумотларга иловаларни жамлаш
   E) бир синф усулларини бошқасида қайта таърифлаш

3. Ҳамма объектлар учун умумий бўлган синф аъзолари қайси сўз ёрдамида таърифланади?
   +A) static
   B) protected
   C) private
   D) friend
   E) public

4. Синфдан ташқарида таърифланган функцияга синф ёпиқ элементларига мурожаат ҳуқуқи кайси сўз ёрдамида берилади?

A) static
B) protected
C) private
+D) friend
E) try

5. Қайси сўз ёрдамида фақат синф ичида ёки унинг авлодларида синфнинг аъзоларидан эркин фойдаланиш ҳуқуқини бериш мумкин?

A) static
+B) protected
C) private
D) friend
E) public

6. Берилган синфлар коррект ва бир хилми?

```
class cA {
  int value;
  void method();
}
struct sA {
  int value;
  void method();
}
```

A) коррект ва бир хил
B) cA коррект эмас, sA коррект
C) cA коррект, sA коррект эмас
+D) коррект, лекин бир хил эмас
E) бир хил лекин коррект эмас

7. Берилган синфлар коррект ва бир хилми?

```
class cA {
  float s;
   public:
   float show(return s;);
}
struct sA {
private:
 float s;
  public:
  float show(return s;);
}
```

+A) коррект ва бир хил
B) cA коррект эмас, sA коррект
C) cA коррект, sA коррект эмас
D) коррект, лекин бир хил эмас
E) бир хил лекин коррект эмас

8. Агар demo синфи қуйидагича таърифланган бўлса:

```
class demo{
   int x;
```

```
public:
demo(int i){x=i;}
};
```

шу синфга тегишли объектлар массиви тўғри таърифини кўрсатинг:

A) class demo a[20];

+B) class demo b[2]={demo(10),demo(100)};

C) class demo b[2]={10,100};

D) class demo b[1]={demo(10),demo(100)};

E) тўғри жавоб йўк