**Түзүлүштөр**

***Тузулуш***-деп компоненттери (элементтери) ар түрдүү тип болгон курама объектини айтабыз.

***Тузулуш (структура)*** - бир канча объектилердин биригишин берет. Массив сыяктуу эле маалыматтардын жыйындысын берет. Массив менен түзүлүштүн айырмасы: массивтин элементтери бир типте жана ирээттелген болот, ал эми түзүлүштүн элементтери ар түрдүү типте болуп жана ирээтелбеши мүмкүн. Паскаль тилинде бул түшүнүк аралаш типтер же жазылыштар деп берилет жана төмөндөгүдөй жазылат.

Type <жазылыш ысымы>=RECORD

<1-талаа> : <тип-1>;

<2-талаа> : <тип-2>;

. . . . . .

<k-талаа>:<тип-n>; END;

C++ тилинде түзүлүштөр 3жол менен берилет. Түзүлүштөрдү баяндоодо struct деген кызматчы сөз колдонулат.

*1-жолу:*

struct {

<баяндоолордун тизмеси>

} <түзүлүштүн аталыштары>;

<баяндоолордун тизмеси>-түзүлүштүн элементтерин (компоненттерин бөлүктөрүн) баяндайт.

<түзүлүштүн аталыштары>-өзгөрмөлөрдүн, массивтердин, көрсөткүчтөрдүн жана функциялардын ысымдары.

*Мисалы*:

```
struct {
          double x, y;
        } a, b, c[9];
```

Бул түзүлүштө a, b өзгөрмөлөрүнүн ар бири эки компоненттен турган түзүлүш катары аныкталып жатат, ал эми c- массив катары элементтери эки компоненттен турат. a=xa+ya*i, b=xb+yb*i, c[0]=x0+y0*i, c[1]=x1+y1*i ж.у.с. struct {

```
          int ai, күн, gil;
        } data1, data2;
```

*2-жолу:*

Түзүлүштүн тибин колдонуучу да аныкташы мүмкүн. Бул учурда typedef-кызматчы сөзү колдонулат бул типтин ысымын өзгөрмөлөрдү аныктоодо колдонсо болот. Жалпы жазылышы:

```
typedef struct {
          <баяндоолордун  тизмеси>;
          }<типтин ысымы>;
```

*Мисалы*:

```
typedef struct {
                    char name [30]; int d;
                  } cmp;
```

cmp e1, e2; /* e1, e2 өзгөрмөлөрү cmp тибиндеги түзүлүштөр*/

*3-жолу:*

Кандайдыр бир үлгүнү же белгини (метка) колдонууга негизделген. Жалпы жазылышы:

struct <белги> {

<баяндоолордун тизмеси>

};

<белги>-бул кандайдыр бир ысым

*Мисалы:*

```
struct student {
                    char name [20]; int
                    kurs;
                    char gruppa [10];
                    } s1,s2;
```

Түзүлүштүн компоненттерине (бөлүктөрүнө) жетиш үчүн же иштетиш үчүн <түзүлүштүн ысымы> <компоненттин ысымы> struct student s1, s2; бул мисалда student <белги> катары, ал эми s1, s2 тиби student белгиси менен берилген түзүлүштөр s1.name [20], s1.kurs, s1.gruppa[10]

*Мисалы*: Машинанын маркасы, чыккан жылы жана баасы берилген. Берилген акчадан ашпаган, жана берилген жылдан эски болбогон жеңил машиналардын тизмесин чыгаргыла.

```
# include <stdio.h> #
include <conio.h>
 main ( )
{
struct {
char marka [10]; int gil;
int baasi;
} avto [20];
int n, i, y,m;
clrscr();
printf (“\n n<20 болгондой машиналардын санын киргиз:”); scanf (“%d”, &n);
printf (“%d машинадан турган тизмени киргиз: \n”, n); for (i=0; i<n; i++)
{fflush (stdin);
printf (“Машинанын маркасын киргиз: ”); gets(avto [i]. marka);
printf (“Машинанын жылын киргиз: ”); scanf (“%d”,
&avto [i]. gil);
printf (“Машинанын баасын киргиз: ”); scanf (“%d”,
&avto [i]. baasi);
}
printf (“Керектүү жылды жана бааны үтүр менен киргиз:”); scanf (“%d,%d”, &y,
&m);
printf (“Сизге ылайык келүүчү машина:”);
for (i=0; i<n; i++)if (avto [i]. gil>=y&&avto [i]. baasi <=m)
printf (“\n%12S%nd%7.3f”, avto [i]. marka, avto [i]. gil, avto [i]. baasi);
```

}

Жыйынтык:

n<20 болгондой машиналардын санын киргиз: 3 3 машинадан
турган тизмени киргиз: Машинанын маркасын киргиз: mmmm
Машинанын жылын киргиз: 2001
Машинанын баасын киргиз: 56000 Машинанын маркасын
киргиз: zzzz Машинанын жылын киргиз: 2002 Машинанын
баасын киргиз: 65000

Машинанын маркасын киргиз: ffff Машинанын жылын
киргиз: 2003 Машинанын баасын киргиз: 85000
Керектүү жылды жана бааны үтүр менен киргиз:2002,67000 Сизге ылайык келүүчү машина:

zzzz 2002 65000

Бул программа жеңил машина сатып алуучунун акчасынын баасынан чоң эмес, берилген жылдан эски болбогон жеңил автомашиналардын маркаларынын чыккан жылдарын жана бааларынын тизмелерин чыгарып берет. Түзүлүштүн элементтерине жетүү көрсөткүчтүн жардамы менен да ишке ашырылат. Бул учурда түзүлүштүн элементине кайрылууда "→" жебе белгиси колдонулат.

*Мисалы*: жогорудагы дата түзүлүшүн колдонуучунун тибин аныктоочу typedef-кызматчы сөзүн колдонуп аныктайлы.

```
typedef struct
{ int күн; char
  ai[10]; int gil ;
 }data;  data
d1, d2; data
*kor; kor=&
d1;
kor→күн=19; kor→ai="сентябрь";
kor→gil=1959;
```

Бул программанын бөлүгүндө түзүлүшкө болгон көрсөткүч жана түзүлүштүн элементерине көрсөткүч аркылуу жетүү баяндалган. Көрсөткүчтү колдонуп түзүлүштүн элементерине жетүүнүн башкача жолу.

```
typedef struct
{ int күн; char
ai[10]; int gil ;
}data;
data d1, d2; data
*kor; kor=& d1;
(*kor).күн=19;
(*kor).ai="сентябрь";
(*kor).gil=1959;
```

Түзүлүштүн элементтерин иштетүүдө массивти колдонууга мисал карайлы. Мисалы: студенттин тизмедеги номери боюнча ал жөнүндө информация чыгарып берүүчү программа.

```
# include <stdio.h> #
include <conio.h> main ( )
{int i, k; typeaef
```

```
struct
{int num;  char fam
[20];
char isim [15];
}student; student
mas[3]; clrscr();
printf (“\n Студенттердин номерин, фамилиясын жана ысымын киргиз”);
for (i=0; i<3; i++)
{printf (“\n %d-студенттин номерин киргиз: ”); scanf (“%d”,
&mas[i].num);
printf (“\n %d-студенттин фамилиясын киргиз: ”); scanf (“%s”,
&mas[i].fam)
printf (“\n %d-студенттин ысымын киргиз : “); scanf (“%s”,
&mas[i]. isim)}
printf (“Студенттин тизмедеги номерин киргиз\n”); scanf (“%d”, &n);
for (i=0; i<3; i++); if
(mas[i].num==k);
printf (“Студент: %s %s\n”, mas[i].fam, mas[i].isim);
}
```

Жыйынтык:

Студенттердин номерин, фамилиясын жана ысымын киргиз

1-студенттин номерин киргиз: 555

1-студенттин фамилиясын киргиз: Максатбеков

1- студенттин ысымын киргиз : Нуртегин

2- студенттин номерин киргиз: 777

2-студенттин фамилиясын киргиз: Эркинбаева

2- студенттин ысымын киргиз : Айзат

3- студенттин номерин киргиз: 999

3-студенттин фамилиясын киргиз: Ибраева

3-студенттин ысымын киргиз : Гулмира Студенттин тизмедеги номерин киргиз: 555 Студент: Максатбеков Нуртегин

Түзүлүштүн элементтери анык бир маанини кармап, ар кайсы учурларда өзгөрбөгөн формага ээ болсо анда мындай түзүлүштү- туракту деп айтабыз.

**Бирикме**

Си тилинде учурдагы эсте бир канча ар түрдүү типтеги өзгөрмөлөрдү жайгаштыруу үчүн өзүнчү тип каралган. Бул типти бирикме деп айтабыз. Бирикме union кызматчы сөзү менен

берилет жана түзүлүшкө окшош эле жарыяланат. Түзүлүштөн бирикменин айырмасы болуп бирикменин баардык элементтери эстен бир эле оорунду ээлейт. Ал эми түзүлүштүн ар бир элементтери үчүн учурдагы эстен орун бөлүнүп берилет. Бирикменин элементтеринин өлчөмү чоңуна карата учурдагы эстен орун берилет, калган элементтери ушул орунга кабатталып жайланышкан сыяктанат. Бирикменин элементтерине кайрылуу түзүлүшкө окшош эле чекит же жебе менен ишке ашырылат. Бирикме төмөнкүдөй баяндалат:

```
union {
        <1-элементтин баяндалышы>
        <2-элементтин баяндалышы>
                . . . . . .
        <n-элементтин баяндалышы>
      }<бирикменин аталышы>;
```

Бирикмени биринчиден кандайдыр бир убакыттын ичинде көп объектилердин ичинен бир гана объект активдүү болсо, эсти үнөмдөөдө экинчиден негизги бир типтеги объектиге башка бир типти ыйгаруу үчүн колдонушат.

Бирикмеге *мисал* карайлы.

```
union {
    float radius;        /*Айлана*/ float a[2];
                         /*Тик бурчтук*/ int
    b[3];                /*үч бурчтук*/
    orun p;              /*p-чекит,  orun-колдонуучунун  тиби*/
} geomfig ;
```

Жогорудагы мисалда кандайдыр бир элемент маниге ээ болсо ал активдүү элемент деп аталат. Калгандары каралбайт.

Мисалы radius элементине маани берилсе ошол эле учурда а[2] же в[3] ж.у.с. элементтерине кайрылууга болбой калат.

**Өзгөрмөлүү түзүлүштөр**

Түзүлүштөрдү колдонуп программа түзүүдө Си тилинде өзгөрмөлүү түзүлүш түшүнүгү да колдонулат. Өзгөрмөлүү түзүлүштөр деп –түзүлүштүн элементтери ар кайсы учурда, ар түрдүү маанилерди кармаган түзүлүштү айтабыз. Өзгөрмөлүү түзүлүш жалпы компоненттеринин жыйындыларын кармайт. Жалпы учурда өзгөрмөлүү түзүлүш үч бөлүктөн турат. Жалпы компоненттердин жыйындысы, активдүү компоненттин белгиси жана өзгөрмөлүү компоненттеринин бөлүктөрү. Өзгөрмөлүү түзүлүштөрдү ишке ашырууда түзүлүштөрдүн жана бирикмелердин комбинациясы колдонулат. Өзгөрмөлүү түзүлүштүн жалпы берилиши.

```
struct { <жалпы компоненттер>;
        <активдүү компоненттин белгиси>; union { <1-
  компоненттин баяндалышы>;
        <2-компоненттин  баяндалышы>;
                . . . . . .
        <n-компонентин  баяндалышы>;
        } <бирикменин аталышы>;
        } <өзгөрмөлүү түзүлүштүн ысымы>;
```

Активдүү компоненттин белгисинин учурдагы мааниси бирикменин өзгөрмөлүү компоненттеринин кайсынысы ошол учурда активдүү болоорун көрсөтүп турат. Мисал катары 4.5. Бирикме бөлүмүндөгү геометриялык фигуралар жөнүндө маселени карайлы. Бул маселеде фигуралар үч параметр менен берилсин деп алалы: аянт, периметр жана өлчөм. Үч фигура үчүн жалпы мүнөздүү параметрлер болуп аянт жана периметр эсептелинет. Ал эми өлчөм болсо тегерек үчүн радиусу, тик бурчтук үчүн анын эки жагы, үч бурчтук үчүн үч жагы эсептелет. Программанын фрагментин карайлы:

```
typedef struct {
  float aiant, perimetr; int t;
```

```
union {float r; float
a[2]; float b[3];
} geomfig;
} fig; fig f;
```

бул жерде f өзгөрмөсү fig тибиндеги өзгөрмө болуп эсептелет.

Жогорудагы мисалда fig тибиндеги ар бир өзгөрмө үч туруктуу aiant, perimeter жана t деген компоненттерден турат. t компоненти geomfig бирикмесинин кайсыл компонентти активдүү болоорун көрсөтүп турат.

*Мисалы:* f.t=1 болсо анда fig тибиндеги f өзгөрмөсүнүн t компоненти 1ге барабар болду дегенди билдирет жана геометриялык фигуралардын ичинен айлананы иштетүү активдештирилди деп эсептелет жана f.geomfig.r=5.0 же айлананын радиусуна башка деле маанилер берилип айлана иштетиле баштайт. Ушул эле сыяктуу f.t=2 болсо тик бурчтук иштетилет жана ага тиешелүү эки жагы мисалы: f.geomfig.a[0]=2.0 жана

f.geomfig .a[1]=3.0 дон же башка деле маанилер берилип тик бурчтук иштетиле берет. Ошондой эле f.t=3 болсо анда үч бурчтук иштетилет.

**Маанилештирүү**

Программада өзгөрмөлөрдүн маанилерин берүү программанын текстинде же клавиатура аркылуу киргизүү сабында же маалыматтар файлы аркылуу киргизүү менен жүргүзүлөт. Өзгөрмөлөрдүн маанилерин ал өзгөрмөлөрдү баяндоодо баштапкы маанилери менен кошо берүү – маанилештирүү деп аталат жана өзгөрмөнүн маанисин программанын текстинде берүү жолун билдирет. Маанилештирүүнү жөнөкөй өзгөрмөлөргө, массивдерге жана түзүлүштөргө жүргүзүүгө болот.

*Мисалы:*

```
int i=0, j=1;
int mas[5]={1,3,5,7,9};
int mat [2][3]={0,2,4,6,8,10}
```

Мындай маанилештирүү төмөнкү менчиктөө операторлорунун тобуна тең күчтүү:

```
mas [0]=1; mas [1]=3; mas 2]=5; mas [3]=7; mas [4]=9;
mat [0][0]=0; mat [0][1]=2; mat [0][2]=4; mat [1][0]=6; mat
[1][1]=3; mat [1][2]=10;
```

Көп өлчөмдүү массивтерди маанилештирүүдө, анын кабатталган түрү колдонулат.

*Мисалы:*

```
int mat [2][3]={{0,2,4},{6,8,10}}
```

Түзүлүштөрдү маанилештирүүнү карайлы:

```
typedef struct {
        char atalysh [30];
        char avtor [20]; int
        baasy; } kitep;
main( )
{static kitep sap={“Айкөл Манас”, “Ж. Садыков”, 930}
}
```

kitep тибиндеги sap өзгөрмөсүн маанилештирүү жүргүзүлдү