ШАГ компьютерная академия

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ С

Dərs №16

Strukturlar

Mündəricat

1. Struktur. Strukturun tərifi

C dilini öyrənməyə başladıqda siz verilənlərin tipi anlayışı ilə tanış oldunuz. Gəlin bu anlayışı genişləndirməyə çalışaq. Belə ki, standart tiplərdən başqa, proqramçı struktur adı altında öz şəxsi düzəltmə verilənlər tipini yarada bilər.

Struktur - müxtəlif tiplərə malik olub bir ad altında birləşmiş, bir və ya daha çox obyektdən ibarət çoxluqdur.

Struct verilənlər tipi – C dilində verilənlərin əsas qurma blokudur. O, bir-biri ilə məntiqi əlaqə ilə bağlı olan, müxtəlif elementlərin rahat birləşmə üsuludur.

Nümunə əsasında strukturlar üzrə işin xüsusiyyətləri ilə tanış olaq.

```
#include <iostream>
using namespace std;
/*
  İstifadəçi verilənlər tipinin (strukturun)
  yaradılması.
  .
  Bir obyektə aid olan bütün verilənlər bir yerə
  yığılır.
*/
struct date
{
    int day; // gün int month; //ay int year;//il
```

```
    int weekday; // həftənin günü
    char mon_name[15];// ayın adı
};

void main(){
    // date tipli obyektin yaradılması və date my

    yaradarkən onun qiymətləndirilməsi

    birthday={20,7,1981,1,"July"};

    // Obyektin mahiyyətinin ekranda göstərilməsi
    // Strukturun ayrılıqda bir üzvünə müraciət
    // nöqtə (.)operatoru vasitəsi ilə yerinə
    yetirilir
    cout<<"________MY BIRTHDAY __________\n\n";
    cout<<"DAY "<<my_birthday.day<<"\n\n";
    cout<<"MONTH "<<my_birthday.month<<"\n\n";
    cout<<"YEAR "<<my_birthday.year<<"\n\n";
    cout<<"WEEK DAY
    "<<my_birthday.weekday<<"\n\n"; cout<<"MONTH
    NAME "<<my_birthday mon_name<<"\n\n";

    //boş obyektin yaradılması və onun klaviaturadan
    //date your_birthday ilə doldurulması
    cout<<"DAY ? "; cin>>your_birthday.day;
    cout<<"MONTH ?"; cin>>your_birthday.month;
    cout<<"YEAR ?";
    cin>>your_birthday.year; cout<<"WEEK DAY ?";
    cin>>your_birthday.weekday; cout<<"MONTH NAME
    ?"; cin>>your_birthday.mon_name;
    cout<<"______YOUR BIRTHDAY_____\n\n"; cout<<"DAY
          "<<your_birthday.day<<"\n\n";
    cout<<"MONTH "<<your_birthday.month<<"\n\n";
    cout<<"YEAR "<<your_birthday.year<<"\n\n";
    cout<<"WEEK DAY "<<your_birthday.weekday<<"\n\n";
    cout<<"MONTH NAME "<<your_birthday.mon_name<<"\n\n";
}
```

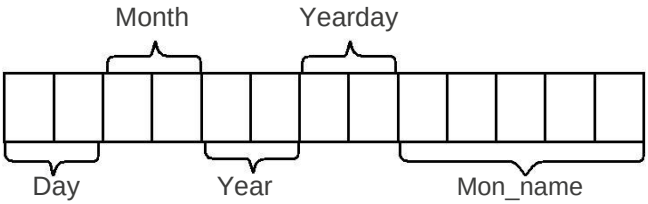
Strukturun xüsusiyyətləri

1. Strukturun təsviri struct açar sözündən başlanır. Ondan sonra strukturun adı (burada date) gəlir. Struktur tipinin bu adı gələcəkdə konkret obyektin yaradılmasında istifadə olunur.
2. Struktur tipinin adından sonra fiqurlu mötərizədə hər elementin tipi təyin edildikdən sonra strukturun elementlərinin (strukturun elementi dəyişən, massiv və ya struktur ola bilər) siyahısı gəlir. Strukturun elementləri bir birindən nöqtəli vergül ilə ayrılır. Məsələn:

```
struct date
{
    int day; int month; int
    year; int yearday;
    char mon_name[5];
};
```

3. Elementlər siyahısının sonundakı sağ fiqurlu mötərizədən sonra obyektlər siyahısı gələ bilər. Məsələn, struct date {...} x, y, z; operatoru x, y, z dəyişənlərini təsvir olunmuş tipin strukturu kimi təyin edir və yaddaşın ayrılmasına gətirir.

date tipli x strukturunun yaddaşının bölüşdürülməsini təsvir edək:



4. Strukturun ardınca obyektlərinin siyahısının təsviri yoxdursa, bu yaddaş bölünməsinə gətirib çıxarmır (yuxarıdakı proqramda olduğu kimi); bu yalnız strukturun formasını təyin edir. Əgər belə təsvirdə tipin adı qeyd olunursa (məsələn, date), bu ad sonradan strukturun obyektlərini təyin edən zaman istifadə edilə bilər (proqramda your_birthday və my_birthday strukturlarının təyini).
5. Strukturun təsvirindən sonra fiqurlu mötərizəyə alınmış başlanğıc qiymətləri verməklə onu qiymətləndirmək olar. Proqramda bu şəkildə my_birthday strukturu qiymətləndirilmişdir.

```
date my_birthday={20,7,1981,1,"July"};
```

6. Strukturun müəyyən bir üzvünə müraciət aşağıdakı şəkildə olur:

```
<strukturun adı>.<elementin adı>
```

7. Strukturlar bir-birinin daxilində ola bilər.

Məsələn, işçinin şəxsi kartı belə ola bilər:

```
struct date
{
    int day; // Gün.
    char month[10]; // Ay.
    int year; // İl.
};

struct person
{
    char name[namesize]; // Adı,soyadı,atasının adı.
    char address[adrsize]; // Ev ünvanı.
    int zipcode[2]; // Poçt indeksi.
    int s_number [2]; // Sosial müdafiə kodu.
    int salary[4]; // Əmək haqqı.
    date birthdate; // Doğum tarixi.
    date hiredate; // İşə daxil olma tarixi.
};
```

Person strukturunda date tipli iki struktur vardır və bizim proqram date strukturu üçün şablona malik olmalıdır.

Nick dəyişənini aşağıdakı şəkildə təyin etsək:

```
struct person Nick;
```

onda Nick.birthdate.month doğum ayını göstərəcək. Strukturun üzvünə çıxış əməliyyatı soldan sağa yerinə yetirilir.

İndi isə əsaslara baxaraq strukturlarla nə etmək mümkün olmasına keçək. Bunu dərsin növbəti bölüməsində gözdən keçirək.

2. Strukturlar üzərində əməllər

Strukturlar üzərində hansı əməliyyatların mümkün olmasını müəyyənləşdirək.

1. «.» (nöqtə) opraturu vasitəsi ilə strukturun elementinə müraciət.
2. «->» operatoru vasitəsilə ilə göstəriciyə görə strukturun elemtinə müraciət növbəti şəkildə olur: <strukturun göstəricisi> «->» <-strukrurun_elementi>. Beləliklə, pd->year və (*pd). year müraciətləri ekvivalentdir. Dairəvi mötərizələr (*pd) zəruridir, belə ki, «.» elementə müracitə operatoru prioritetı görə «*» operatorundan yüksəkdir.
3. «&» əməliyyatının köməyi ilə strukturun ünvanının təyini.

```
 #include <iostream>
 using namespace std;
//date adlı strukturun təyini
struct date
{
     int day;
     int month;
     int year;
     char mon_name[12];
};

//Strukturun obyektinin yaranması və inisializəsi.
 date d = { 2,5,1776,"July" };
//d - date tipinin dəyişəni.
 void main ()
{
//p strukturun göstəricisi.
```

```
    struct date *p;
    // Strukturun məzmununun ekrana çıxarılması
    // (obyektdən müraciət)
    cout<<d.day << " ";
    cout<<d.year << " ";
    cout<<d.month << " ";
    cout<<d.mon_name << "\n\n";

    // Struktur obyektinin ünvanının p = &d göstəricisinə
       yazılması;
    // Strukturun məzmununun ekrana çıxarılması
    // (göstəricidən müraciət)
    cout << p->day << " ";
    cout << p->month << " ";
    cout << p->year << " ";
    cout << p->mon_name << "\n\n";
}
```

4. Strukturun qiymətləndirilməsi.

```
#include <iostream>
using namespace std;
struct date
{
    int day;
    int month;
    int year;
    char mon_name[12];
};
date a = { 14,7,1954,"July" }; date b;
void main ()
{// a obyektinin məzmunun göstərilməsi
 cout<< a.day << " ";
    cout<< a.year << " ";

    cout<< a.month << " ";
```

```
    cout<< a.mon_name << "\n\n";

    // b     obyektinin     a     obyekti     tərəfindən
       qiymətləndirilməsi
    b = a;
    // b obyektinin məzmununun göstərilməsi
    cout<<  b.day  <<  "  ";
    cout<<  b.year  <<  "  ";
    cout<< b.month << " ";
    cout<< b.mon_name << "\n\n";
}
```

5. Strukturun funksiya parametri kimi ötürülməsi və funksiyanın işinin nəticəsində onun geri qayıtması.

```
#include  <iostream>
using      namespace
std; struct date
{
    int day;
    int month;
    int year;
    char mon_name[12];
};

void Show(date a){
// a obyektinin məzmununun göstərilməsi
    cout<< a.day << " ";
    cout<< a.year << " "; cout<<
    a.month << " "; cout<<
    a.mon_name << "\n\n";}
date Put(){// date temp obyektinin qurulması;
cout<<"DAY ? ";
    cin>>temp.day;

    cout<<"MONTH ? ";
```

```
    cin>>temp.month;
    cout<<"YEAR ? ";
    cin>>temp.year;
    cout<<"MONTH NAME ? ";
    cin>>temp.mon_name;
    return temp;
}

date a = { 14,7,1954,"July" }; date b;

void main ()
{
    // Obyektin funksiyaya ötürülməsi
    Show(a);

    // Obyektin qaytarılma qiyməti kimi qəbulu
    b=Put();

    // b obyektinin məzmunun göstərilməsi
    Show(b);
    }
```

6. Strukturun müəyyən üzvlərinin funksiya arqumenti kimi ötürülməsi. Məsələn, day_of_year1 funksiyası üçün:

```
#include <iostream>

using namespace std;

// ilin aylarındakı günlərin sayı üçün köməkçi massiv
 int day_tab[2][13]= {0,31,28,31,30,31,30,31,31,30,31,
                      30,31,0,31,29,31,30,31,30,31,31,
                      30,31,30,31};

struct date
{
```

```
    int day; int
    month; int
    year; int
    dayyear;
    char mon_name[12];
};

void Show(date a){
    // a obyektinin məzmunun göstərilməsi
    cout<< a.day << " ";
    cout<< a.year << " "; cout<<
    a.month << " "; cout<<
    a.dayyear << " "; cout<<
    a.mon_name << "\n\n";
}

int day_of_year1 (int day,int month,int year)
{
    // İl və ayın köməyi ilə günün hesablanması
    int i, leap;
    leap = year%4==0 && year%100!=0 ||
    year%400==0; for (i=1; i <; month; i++)
    day += day_tab[leap][i];
    return (day);
}

date a = {20,7,1981,0,"July"};

void main ()
{
    //Ayrı-ayrı üzvlərin funksiyaya ötürülməsi

    a.dayyear=day_of_year1(a.day,a.month,a.year);

    //a obyektinin məzmunun göstərilməsi
Show(a);}
```

İndi isə, aşağıdakı nümunədə strukturun elementləri üzərində daha cox istifadə olunan bəzi əməllərin yerinə yetirilmə sırasına baxaq:

```
struct
{
    int x;
    int *y;
} *p; // p - struktura göstərici.
```

- ***(++p)->x*** —x-a müraciətə qədər p-ni artırır;
- ***(p++)->x*** —x-a müraciətdən sonra p-ni artırır (dairəvi mötərizələr mütləq deyil, belə ki, prioritetinə görə əvvəlcə «->» əməliyyatı yerinə yetirəcək);
- ****p->y*** — y-in göstərdiyi obyektin məzmu seçilir;
- ****p->y++*** — y-in göstərdiyi obyekt emal olunduqdan sonra y artır (analoji olaraq *s++);
- ****p++->y*** — y-in göstərdiyi seçiləndən sonra p-ni artırir ;
- ***(*(*p).y)++*** — y-in göstəricisini artırır.

Strukturların istifadəsində bir şeyi qeyd etmək lazımdır: yeni verilənlər tiplərinin yaradılması. Elə verilənlər tipləri vardır ki, onlar müəyyən məsələlərin həllində massiv və strukturlara nisbətən daha səmərəlidirlər. Bunlar növbələr, ikilik ağaclar, cədvəllər və qrafalardır. Bu tiplərin çoxu «bağlı» strukturlardan yaranır. Adətən hər bir belə struktur bir və ya iki verilənlər tipi üstəgəl bir və ya iki bu tipdən olan struktur göstəricisinə malik olur. Göstəricilər bir strukturun başqası ilə əlaqəsini yaratmaq və bütün struktur boyu axtarışa imkan vermək üçün yol təmin etməyə xidmət edirlər .

3. sizeof operatoru

C dilində sizeof adlanan xüsusi unar əməliyyat vardır ki, o, öz operandının ölçüsünü baytlarla qaytarır. sizeof əməliyyatının operandı istənilən ifadə ola bilər:

```
sizeof(İfadə);
```

sizeof əməliyyatının nəticəsi int tipinə malikdir. Obyektin ölçüsünün alınması.

```
#include <iostream>
using namespace std;
void main ()
{
    int a; char
    b; unsigned
    int *p;
    /* ---------------------------- */
    cout<<"sizeof(a)="<<sizeof(a)<<"\n";
    cout<<"sizeof(b)="<<sizeof(b)<<"\n";
    cout<<"sizeof(c)="<<sizeof(c)<<"\n";
    cout<<"sizeof(p)="<<sizeof(p)<<"\n";
    cout<<"sizeof(int)="<<sizeof(int)<<"\n";
    cout<<"sizeof(int *)="<<sizeof(int *)<<"\n";
}
Proqramın işinin nəticəsi:
sizeof(a)=4
sizeof(b)=1
sizeof(c)=4
sizeof(p)=4
sizeof(int)=4
sizeof(int *)=4
```

Strukturun ölçüsü

Yəgin ki, siz fikirləşirsiniz ki, strukturun ölçüsü onun üzvlərinin ölçüləri cəminə bərabərdir. Bu belə deyil. Strukturda müxtəlif uzunluqlu obyektlərin bərabərləşdirilməsi nəticəsində adsız «boşluqlar» əmələ gələ bilər. Məsələn, əgər, char tipli dəyişən bir bayt, int isə 4 bayt yer tutursa, onda struktur üçün:

```
#include <iostream>
using namespace std;
struct Test
{
    char c;
    int i;
};

void main ()
{
    Test d={'#',78};
    cout<<sizeof(Test)<<" "<<sizeof(d)<<"\n\n";
}
```

5 deyil, 8 bayt tələb oluna bilər. sizeof əməliyyatı düzgün qiyməti qaytarır.

4. Ev tapşırığı

1. Videomaqazin strukturunu aşağıdakı sahələrlə yaradın:
   - Filmin adı
   - Rejissor
   - Janr
   - Populyarlıq reytinqi
   - Diskin qiyməti

   Aşağıdakı əməliyyatları həyata keçirmək:
   - Ada görə axtarış
   - Janra görə axtarış
   - Rejissora görə axtarış
   - Janrda ən populyar film
   - Yazıların göstərişi və əla