21-MA’RUZA. STRUKTURALARNI E’LON QILISH VA INITSIALIZATSIYALASH

1. Strukturalarni e’lon qilish va initsializatsiyalash

Struktura - bu bitta nom bilan birlashtirilgan o‘zgaruvchilar yig‘indisi, bu ma’lumotni birgalikda saqlashning umumiy qabul qilingan usulini ta’minlaydi. Strukturani e’lon qilish natijasida struktura obyektlarini yaratish uchun ishlatiladigan shablon paydo bo‘ladi. Strukturani tashkil etuvchi o‘zgaruvchilar struktura ***a’zolari*** deb ataladi. (Struktura a’zolari ko‘pincha ***a’zo*** yoki ***maydon*** deb ham ataladi.)

Masalan, shaxs haqidagi ma’lumotlarni saqlash uchun, ismingizni, tug‘ilgan kuningizni, bo‘yingizni, vazningizni yoki boshqa ma’lumotlarni kiritishni xohlaysiz:

```
string ism;
int kun;
int oy;
int yil;
int balandlik;
int vazn;
```

Ammo endi sizda 6 ta alohida o‘zgaruvchi mavjud. Agar o‘zingiz haqida ma’lumotni funksiyaga o‘tkazmoqchi bo‘lsangiz, unda har bir o‘zgaruvchini alohida o‘tkazishingiz kerak. Bundan tashqari, agar siz boshqa birov haqida ma’lumot saqlamoqchi bo‘lsangiz, unda har bir kishi uchun qo‘shimcha ravishda yana 6 ta o‘zgaruvchini e'lon qilishingiz kerak bo‘ladi! Bunday amalga oshirish unchalik samarali emasligini oddiy ko‘z bilan ham ko‘rish mumkin.

C++ tili dasturchilarga o‘zlarining shaxsiy ma’lumot turlarini - bir nechta alohida o‘zgaruvchilarni birlashtirgan turlarini yaratishga imkon beradi. Foydalanuvchilar tomonidan aniqlangan ma’lumotlar turlaridan biri bu **strukturadir**. Struktura har xil turdagi o‘zgaruvchilarni bir butunga birlashtirishga imkon beradi.

Strukturani e’lon qilish va aniqlash. Strukturalar dasturchi tomonidan aniqlanganligi sababli, avval kompilyatorga uning qanday ko‘rinishini aytib berishimiz kerak. Buning uchun **struct** kalit so‘zi ishlatiladi:

```
struct Xodim
{
   short id;
   int yosh;
```

```
    double maosh;
};
```

Yuqorida Xodim nomli struktura aniqlandi. U uchta o‘zgaruvchini o‘z ichiga oladi:

1) id – short tipida
2) yosh – int tipida
3) maosh – double tipida

Strukturaning bir qismi bo‘lgan ushbu o‘zgaruvchilar struktura a’zolari (yoki "struktura maydonlari") deb nomlanadi. **Xodim** – bu oddiy struktura e’loni. Garchi biz kompilyatorga uning a’zo o‘zgaruvchilari borligini ko‘rsatgan bo‘lsak-da, hozirda unga xotira ajratilmagan. Odatda struktura nomlarini oddiy o‘zgaruvchi nomlardan ajratish uchun bosh harflar bilan yoziladi.

C ++ da eng oddiy xatolardan biri bu struktura e’loni oxirida nuqtali vergulni (;) unutishdir. Buning natijasida keyingi satrda kompilyatsiya xatosi paydo bo‘ladi. Visual Studio 2010 va undan yangi versiyalari kabi zamonaviy kompilyatorlar sizga oxirida vergulni unutganligingizni aytishadi, lekin eski kompilyatorlar buni unutishi mumkin, chunki bu xatoni topish qiyin.

Employee strukturasidan foydalanish uchun biz faqatgina Employee tipidagi o‘zgaruvchini e'lon qilishimiz kerak:

```
Xodim Anvar;
```

Bu yerda biz Anvar nomidagi Xodim turidagi o‘zgaruvchini aniqladik. Oddiy o‘zgaruvchilardagi kabi, strukturali o‘zgaruvchini aniqlash, ushbu o‘zgaruvchiga xotira ajratilishiga olib keladi.

Siz bir xil strukturaning bir nechta o‘zgaruvchilarini e'lon qilishingiz mumkin:

```
Xodim Anvar;
Xodim Botir;
```

Struktura a’zolariga kirish va initsializatsiyalash. Xodim Anvar kabi struktura o‘zgaruvchisini e'lon qilganimizda, Anvar butun strukturani anglatadi. Alohida a’zolarga kirish uchun a’zolarni tanlash operatori (.) ishlatiladi. Masalan, quyidagi kodda, strukturaning har bir a’zosini initsializatsiya qilish uchun a’zoni tanlash operatoridan foydalanamiz:

```
Xodim Anvar;
Anvar.id = 8;
```

```
Anvar.yosh = 27;
Anvar.maosh = 32.17;

Xodim Botir;
Botir.id = 9;
Botir.yosh = 30;
Botir.maosh = 28.35;
```

Oddiy o‘zgaruvchilar singari, strukturaning a’zo o‘zgaruvchilari avtomatik ravishda initsializatsiya qilinmaydi. Yuqoridagi misolda qaysi o‘zgaruvchi Anvar, qaysi biri Botir ekanligini aniqlash oson. Bu an'anaviy yagona o‘zgaruvchilarga qaraganda ancha yuqori darajadagi samaradorlikni ta’minlaydi.

Struktura a’zolarining o‘zgaruvchilari oddiy o‘zgaruvchilar bilan bir xil ishlaydi, shuning uchun ular bo‘yicha oddiy arifmetik va taqqoslash amallarini bajarishingiz mumkin:

```
int umumiyYosh = Anvar.yosh+ Botir.yosh;
   if (Anvar.maosh > Botir.maosh)
        cout << "Anvar Botirdan ko‘ra ko‘proq pul ishlaydi\n";
   else if (Anvar.maosh < Botir.maosh)
        cout << "Botir Anvardan kam pul ishlab topadi\n";
   else
        cout << "Anvar va Botir bir xil miqdordagi pulni ishlab topadi\n";
   // Anvarning maoshini oshirish
   Anvar.salary += 3.75;
   // Bugun Botirning tug‘ilgan kuni
   ++Botir.yosh; // Botirning yoshini 1 yoshga oshirish uchun oldindan o‘sish yordamida
```

Strukturalarni initsializatsiyalash. Har bir a’zoga qiymatlarni tartib bilan belgilash orqali strukturani initsializatsiyalash juda qiyin vazifa (ayniqsa, agar ular ko‘p bo‘lsa), shuning uchun C++ tilida strukturalarni initsializatsiyalashning tezkor usuli mavjud – initsializatorlar ro‘yxati. Bu struct turi o‘zgaruvchisini e'lon qilish paytida strukturaning ba’zi yoki barcha a’zolarini initsializatsiya qilish imkoniyati mavjud:

```
#include <iostream>
using namespace std;
int main()
{
```

```
    struct Xodim
    {
       short id;
       int yosh;
       double maosh;
    };

    Xodim Anvar = { 5, 27, 45000.0 };
  // Anvar.id = 5, Anvar.age = 27, Anvar.salary = 45000.0
    Xodim Botir = { 6, 29};
  // Botir.id = 6, Botir.age = 29, Botir.salary = 0.0 (jimlik bo‘yicha initsializatsiya)
     return 0;
  }
```

C++11 standarti bo‘yicha uniform initsializatsiyasidan foydalanishingiz mumkin:

```
Xodim Anvar { 5, 27, 45000.0 };
Xodim Botir { 6, 29 };
```

Agar boshlang‘ich ro‘yxatida bir yoki bir nechta element yetishmayotgan bo‘lsa, u holda ularga standart qiymatlar beriladi (jimlik bo‘yicha 0). Yuqoridagi misolda james.salary a’zosiga jimlik bo‘yicha 0.0 qiymati berilgan, chunki initsializatsiyalash paytida hech qanday qiymat berilmadi.

C++ 11/14: nostatik a’zolarni initsializatsiyalash. C++11 standarti strukturaning nostatik a’zolariga standart qiymatlarni belgilash qobiliyatini qo‘shdi. Masalan:

```
#include <iostream>
struct Triangle
{
   double length = 2.0;
   double width = 2.0;
};
int main()
{
   Triangle z;
   z.length = 3.0;
   return 0;
```

```
}
```

Strukturalar a’zolariga qiymatlarni belgilash. C++11 dan oldin, agar biz struktura a’zolariga qiymatlarni belgilashni xohlasak, uni har bir a’zo uchun alohida-alohida bajarishimiz kerak edi:

```
struct Xodim
{
   short id;
   int yosh;
   double maosh;
};

Xodim Anvar;
Anvar.id = 5;
Anvar.yosh = 27;
Anvar.maosh = 45000.0;
```

Bu murakkab jarayon, ayniqsa strukturada ko‘plab a’zolar mavjud bo‘lganda. C++11 da initsializatorlar ro‘yxati yordamida strukturalar a’zolariga qiymatlarni belgilashingiz mumkin:

```
#include <iostream>
using namespace std;
struct Xodim
{
   short id;
   int yosh;
   double maosh;
};

int main()
{
   Xodim Anvar;
   Anvar = { 5, 27, 45000.0 };
   cout<<Anvar.yosh;

   return 0;
}
```

2. Struktura va funksiyalar

Alohida o‘zgaruvchidan emas, balki strukturalardan foydalanishning katta afzalligi bu butun strukturani uning a’zolari bilan ishlashi kerak bo‘lgan funksiyaga o‘tkazish qobiliyatidir:

```
#include <iostream>
using namespace std;

struct Xodim
{
   short id;
   int yosh;
   double maosh;
};

void printInformation(Xodim Anvar)
{
   std::cout << "ID: " << Anvar.id << "\n";
   std::cout << "Yosh: " << Anvar.yosh << "\n";
   std::cout << "Maosh: " << Anvar.maosh << "\n";
}

int main()
{
   Xodim Anvar = { 21, 27, 28.45 };
   Xodim Botir= { 22, 29, 19.29 };

   // Anvar haqidagi ma’lumotlarni chiqarish
   printInformation(Anvar);

         cout << "\n";

   // Botir haqidagi ma’lumotlarni chiqarish
   printInformation(Botir);

   return 0;
}
```

Yuqoridagi misolda biz Employee strukturasini printInformation () funksiyasiga o‘tkazdik. Bu bizga har bir o‘zgaruvchini alohida o‘tkazmaslikka imkon berdi.

Dasturni bajarish natijasi:

ID: 21
Yosh: 27
Maosh: 28.45

ID: 22
Yosh: 29
Maosh: 19.29

Funksiya strukturani ham qaytarishi mumkin (bu funksiya bir nechta o‘zgaruvchini qaytarishi mumkin bo‘lgan kam holatlardan biri). Masalan:

```
#include <iostream>
using namespace std;
struct Point3d
{
   double x;
   double y;
   double z;
};

Point3d getZeroPoint()
{
   Point3d temp = { 0.0, 0.0, 0.0 };
   return temp;
}

int main()
{
   Point3d zero = getZeroPoint();

   if (zero.x == 0.0 && zero.y == 0.0 && zero.z == 0.0)
      cout << "Nuqta nolga teng\n";
   else
      cout << "Nuqta nolga teng emas\n";
```

```
    return 0;
}
```

Ichki strukturalar. Ba’zi strukturalar boshqa strukturalarni o‘z ichiga olishi mumkin. Masalan:

```
struct Xodim
{
    short id;
    int yosh;
    double maosh;
};

struct Kompaniya
{
    Xodim Anvar; // Employee – bu Company tarkibidagi struktura
    int XodimNomeri;
};

Kompaniya Samsung;
```

Bunday holda, agar CEO ning (ijrochi direktorning) ish haqi qancha ekanligini bilmoqchi bo‘lsak, unda a’zolarni tanlash operatoridan ikki marta foydalanishimiz kerak bo‘ladi:

```
Samsung.Anvar.maosh;
```

Birinchidan, Samsung strukturasidan Anvar maydonini, so‘ngra Xodim strukturasidan **maosh** maydonini tanlaymiz.

Ichki tuzilishga ega bo‘lgan ichki initsializator ro‘yxatlaridan foydalanishingiz mumkin:

```
struct Xodim
{
    short id;
    int yosh;
    float maosh;
};

struct Kompaniya
```

```
{
    Xodim Anvar; // Employee Company strukturasining ichidagi strukturadir
    int Xodim nomeri;
};

Kompaniya Samsung = {{ 3, 35, 55000.0f }, 7 };
```

Struktura hajmi. Odatda, strukturaning kattaligi uning barcha a’zolari kattaliklarining yig‘indisidir, lekin har doim ham emas! Masalan, Xodim strukturasini ko‘rib chiqing. Ko‘pgina platformalarda short 2 bayt, int 4 bayt, double - 8 bayt. Demak, Xodim 2 + 4 + 8 = 14 bayt bo‘lishi kutilmoqda. Xodim strukturasining aniq hajmini bilish uchun biz sizeof operatoridan foydalanishingiz mumkin:

```
sizeof(Xodim)
```