Obyektyönlü proqramlaşdırma

C++

Dərs №1

C++ dilindən istifadə etməklə obyektyönlü proqramlaşdırma

Mündəricat

Obyektyönlü proqramlşadırmanın əsasları

Və yenə də xoş gördük! Biz çox ümid edirik ki, siz C dilində proqramlaşdırmadan imtahanı müvəffəqiyyətlə verdiniz və indi də təlimin növbəti mərhələsinə keçə bilərsiniz. Məhz C++ dilində proqramlaşdırmaya. Əvvəlcə bu iki dil ilə bağlı bəzi tarixi faktları xatırladaq. Bu informasiyanı siz artıq birinci dərsdə almışsınız, yenə də:

1972-ci ildə Bell Labs firmasının 31 yaşlı sistem proqramçısı Dennis Ritçi C proqramlaşdırma dilini işləyib hazırladı. Dilin ilk versiyası rus dilinə də tərcümə edilmiş olan B.Kerninq və D.Ritçinin kitabında verilmişdir. Uzun müddət bu versiya sdandart kimi qəbul edilirdi, lakin bəzi hallar C dilinin bir çox addımlarını uzadan dəfələrlə qarşıdurmaya səbəb olurdu. Bu halları ardan qaldırmaq üçün Amerika Milli Standartlaşdırma İnstitutunda (ANSI) C dilinin standartlaşdırılması şurası yaradıldı. 1983-cü ildə C dilinin ANSI C adlanan standartı yaradıldı. 80-ci illərin əvvəlində elə Bell Laboratory-də Bern Straustrup tərəfindən C dilinin əlavələri və genişləndirilməsi nəticəsində demək olar ki, "siniflərlə C" adlanan yeni dil yaradıldı. 1983-cü ildə bu ad C++ olaraq dəyişdirildi.

Xatırladınız? Lakin dillərin yaradılması, bu onlar arasındakı yeganə fərq deyildir :))))).. İş ondadır ki,

proqramlaşdırmaya iki əsas yanaşma mövcuddur: **prosedur və obyektyönlü**. Onların tərifinə baxaq:

Prosedur proqramlaşdırmada hansısa bir obyektə aid olan funksiya və dəyişənlər kodda sərbəst şəkildə yerləşirlər və bir-biri ilə heç bir əlaqələri yoxdur (C dili).

obyekyönlü proqramlaşdırmada hansısa bir obyektə aid olan funksiya və dəyişənlər kodda bir-biriləri ilə müəyyən formada və sıx əlaqədədirlər (C++ dili).

Nəhayət, bugündən etibarən biz obyektyönlü proqramlaşdırma ilə məşğul olacağıq. Bunu qısaca olaraq OYP adlandıracağıq.

OYP – öz növbəsində proqramlaşdırmada əsl inqilab etdi. OYP hesab edir ki, proqramlar hissələrinin daxili quruluşlarından asılı olmayan dəstləri üzərində qurulur. OYP üç əsas prinsip əsasında durulur ki, bunlarla da biz sizi tanış edəcəyik.

İnkapsulyasiya (Encapsulation)

OYP-da verilənlərin müstəqilliyi prinsipi **inkapsulyasiya** adlanır. Bu şəkildə hər bir hissə sistemin digər hissələri tərəfindən əlyetərli olmayan xüsusi verilənllər ehtiva edə bilər. Aydındır ki, bu hissələr tamamilə müstəqil ola bilməzlər, belə ki, onlar öz aralarında qarşılıqlı fəaliyyətdə olmalıdırlar, ortaq verilənlər istifadə etməlidirlər və xüsusi verilənlərini mübadilə etməlidirlər.

Sadə nümunə - hər birinin öz davranışı və xüsusi quruluşu olan, lakin digər hüceyrələrlə qarşılıqlı təsirdə olan və onlarla maddələr mübadiləsi edən çoxsaylı canlı hüceyrədən təşkil olunan orqanizm. Proqramlaşdırmada bu cür canlı orqanizm – proqram, hüceyrə isə - obyektdir, maddələr - verilənlər, qarşılıqlı təsir yolları isə metodlar (funksiyalar) və hadisələrdir.

Obyektin tərifini verməyə çalışaq:

Obyekt – öz dəyişənləri və funksiyaları olan nadir vahiddirki, bu dəyişənlər hasil edilir.

Obyektin yaradılması üçün proqramda istifadə ediləcək müəyyən tip verilənlər təyin etmək lazımdır, bu tipdən olan hər bir verilən (nümunə) obyekti təmsil edəcək. Prinsip etibarilə obyekt kimi struktur verilənlər tipini hesab etmək olar. Ümid edirik ki, siz bunun nə olduğunu hələ də xatırlayırsınız. Strukturda bir obyektə aid olan bütün verilənlər istifadəçi verilənlər tipi adlanan bir ad altında saxlanılır. C++ dilində bu cür verilənlər tipi **sinif** adlanır.

Qara qutu təsəvvür edin. Proqramlaşdırma nöqteyi-nəzərindən bu obyektdir. Qutu obyektdirsə, o zaman sinif nədir? Bu halda sinif bu qutunun hazırlanması üçün çertyojdur. İndi fikirləşək, bu qara qutuya nə aid ola bilər. Tamamilə aydındır ki, bu qara qutunun bəzi xüsusiyyətləri vardır. İlk olaraq onun forması, ikincisi onda saxlanılan və onun bağlanması və açılması, məzmununun çıxarılması üçün mexanizmlər.

Proqramlaşdırmada xüsusiyyət elə xüsusiyyət də adlanır. Bu qutuya tətbiq edilən məzmunun çıxarılması isə metodlar adlanır. Xüsusiyytlər vasitəsilə biz daxili verilənlər ilə işləyir, qiymətləri oxuyur və veririk, metodlar isə obyektin özünün icra etdiyi əməliyyatlardır, yəni, bizim misalda qutunu aç "metodunu çağırmaq" üçün biz qutunun düyməsinə basıb açırıq və qutu sərbəst açılır.

Beləliklə, ümumiləşdirək, inkapsulyasiya – təyin olunmuş bir obyektə aid olan bütün xüsusiyyət və metodların sinif adlanan ümumi ad altında saxlanılmasını təmin edən mexanizmdir.

Varislik (Inheritance)

Əvvəlcə bu terminin tərifini verək:

Varislik – bir obyektin oz xüsusiyyət və metodlarına əlavə olaraq digər obyektin xüsusiyyət və metodlarının varisi olması prosesidir.

Gəlin qara qutunun təhlil edilməsini davam etdirək. Hesab edək ki, qara qutu əsasında biz hədiyyə üçün yaraşıqlı bağlama hazırlayırıq. Bunun üçün biz qutuya hədiyyə bağlamasına xas olan xüsusiyyətlər əlavə edək. Məsələn, rəng. Forma və ölçü, həmçinin, açılma və bağlanma imkanı həm qutu, həmdə bağlamada var. Buna görə də onları yenidən təyin etməyə ehtiyac yoxdur, bunun üçün bağlamanı qutunun varisi etmək olar. Varislik mexanizminin istifadə edilməsinin əsas üstünlüyü də elə bundan ibarətdir.

Biz əvvəlcə hər hansı bir sadə struktur yaradırıq və sonra ona yeni xüsusiyyətlər və metodlar əlavə edirik, bununla da yeni tamamlanmış obyekt alırıq.

Polimorfizm (Polymorphism)

Polimorfizm – obyektin vəziyyətdən asılı olaraq özünü müxtəlif cür aparması və müəyyən əməliyyatlara özünəməxsus cavab verməsi qabiliyyətləridir.

Bayağı hal – siz dükana kolbasa almağa gəlirsiniz. Məhsulu seçirsiniz. Əgər satıcı sizin tanışınızdırsa, o sizə bu məhsulu almağa dəyib dəymədiyini bildirəcək. Əgər tamamilə yad adamdırsa, buna laqeyid qalacaq. Satıcı bu halda vəziyyətdən asılı olaraq rəftar edir. Eləcə də bizim obyektimiz – sərbəst verilənlər tipi xarici təsirlərə müxtəlif cür reaksiya verəcəklər. Hazırda sizin bunun necə baş verəcəyini anlamanız çətindir. Lakin polimorfizm prinsipi ilə siz artıq tanışsınız və funksiyaya yükləmə anlayışını xatırlayırsınız. Funksiya ona ötürülən parametrlərdən asılı olaraq özünün bu və ya digər versiyasını çağırırdı.

Beləliklə, nəzəri hissə tamamlandı, praktik hissəyyə keçmək vaxtıdır.

Siniflərlə tanışlıq

Sinfin tərifi və onun obyektinin yaradılması

Sinif – proqramçının mövcud tiplər əsasında təyin etdiyi törəmə struktur tipidir. Başqa sözlə, siniflər mexanizmi tipli verilənlər dəstinin strukturlaşdırılmasına və bu verilənlər üzərində əməliyyatlar dəsini təyin etməyə imkan verir.

Sinfin ümumi sintaksisini növbəti konstruksiya ilə təyin etmək olar:

```
class sinif_adı {üzvlərin_siyahısı};
```

1. **sinif_adı** – ixtiyari seçilən adlardır.
2. **üzvlərin_siyahısı** – sinfə daxil olan tipli verilənlərin təyin və tərif edilməsidir. Sinfin üzvləri verilənlər, funksiyalar, siniflər, sadalanma, bit sahələri və tiplərin adı ola bilər. Başlanğıcda sadəlik üçün hesab edəcəyik ki, sinfin üzvləri tipli verilənlər (əsas və törəmə) və funksiyalardır.
3. Fiqurlu mötərizələrlə əhatə olunmuş üzvlər sinfin gövdəsi adlanır.
4. Sinfin gövdəsindən əvvəl başlıq verilir. Sadə halda sinfin başlığı class sözünü və sinfin adını ehtiva edir.

5. Sinfin təyini həmişə nöqtə vergül ilə tamamlanır.

Beləliklə, sinfə daxil olan funksiyaları **sinfin metodları** və ya **üzv funksiyalar** adlandıracağıq. Sinfin verilənləri – **üzv verilənlər** və ya **sinfin verilənlər elementləri** adlandıracağıq. Yenidən tərifə qayıdaq: sinif – proqramçı tərəfindən daxil edilən tipdir. Hər bir tip obyektlərin təyin edilməsi üçün istifadə edildiyi üçün sinif obyektinin yaradılması sintaksisini təyin edək:

```
sinif_adı obyekt_adı;
```

Sinif obyektinin təyin edilməsi yaddaş sahəsinin ayrılması hər biri verilmiş sinfin ayrı üzvlərini əks etdirən obyektin ayrı elementlərinə uyğun gələn fraqmentlərə ayrılmasını nəzərdə tutur.

Sinif üzvlərinə müraciət üsulları

Sinifin üzvlərinə bir neçə müraciət səviyyəsi mövcuddur. Bunlardan əsas ikisini nəzərdən keçirək:

public - sinfin üzvləri xaricdən müraciətə açıqdırlar.

private - sinfin üzvləri xaricdən müraciətə bağlıdırlar.

Gəlin bunları şərh edək.

Susmaya görə sinfə aid olan bütün dəyişənlər və funksiyalar bağlı (*private*) kimi təyin edilir. Bu o deməkdir ki, onlar yalnız üzv funksiyaların daxilində istifadə edilə bilərlər. Proqramın digər hissələri üçün, məsələn main() kimi, bağlı üzvlərə müraciət qadağan edilib. ***Yeri gəlmişkən, sinfin strukturdan yeganə fərqi – onun bütün üzvləri susmaya görə public-dir.***

public müraciət spesifikatorunu istifadə etməklə proqramın bütün funksiyaları (sinfin daxilindəki və onun xaricindəki) tərəfindən əlyetərli olan sinfin açıq üzvlərini yaratmaq olar.

Verilmiş sinfin müəyyən obyektinə müraciət etmək üçün sintaksis (strukturda olduğu kimi) növbəti şəkildədir:

```
obyekt_adı.sinfin_üzv_adı;
```

Nümunəyə baxaq.

```
#include  <iostream>
using      namespace
std; class Test{
    //müraciət spesifikatoru verilmədiyi üçün
    //verilmiş dəyişən susmaya görə bağlıdır
    //sinif xaricindən müraciət üçün(private)
    int one;
    //public müraciət spesifikatoru
    //bundan sonra gələn bütün üzvlər
    //xaricdən müraciət üçün açıq olacaq
public:
    //Sinifdə dəyişənlərin qiymətləndirilməsi
    //yaradılarkən qadağan edilmişdir,
    //buna görə də biz bu əməliyyatı yerinə
    //yetirmək üçün metod təyin edirik

    void Initial(int o,int
        t){ one=o;
        two=t;
    }
    //sinfin dəyişənlərinin qiymətini
    //ekranda göstərən metod
    void Show(){
        cout<<"\n\n"<<one<<"\t"<<two<<"\n\n";
    }
    int two;
};
```



```
void main(){
     //Test tipində obyekt yaradır
     Test obj;
     //funksiyanı onun parametrini qiymətləndirərək

     //çağırır

     obj.Initial(2,5);

     //ekrana vermə
     obj.Show(); // 2 5
     //açıq two dəyişəninə birbaşa yazma
     //one dəyişəni üçün belə yazmaq olmaz,
     //çünki ona müraciət bağlıdır
     obj.two=45;


     //yenidən ekrana vermə
     obj.Show(); //2 45
}
```

Yuxarida şərh edilmiş misal ümumilikdə sadədir, lakin bir məqama diqqət edək – **sinfin dəyişənlərini parametr kimi sinfin metoduna göndərməyə ehtiyac yoxdur, belə ki, onlar avtomatik görünür.**

Konstruktorlar və destruktorlar

Sinfin obyektlərinin ilkin qiymətləndirilməsi. Konstruktorlar

Bəzən obyektin yaradılması zamanı onun elementlərinə başlanğıc qiymət mənimsətmək lazım gəlir. Bildiyiniz kimi, sinfin təyini zamanı bunu birbaşa etmək olmaz. Bu problemi həll etmək üçün başlanğıc qiymətləri sinfin dəyişənlərinə mənimsədən funksiya yazmaq və bu funksiyanı hər dəfə obyekti elan edərkən çağırmaq lazımdır, bunu biz dərsin əvvəlki bölməsində müvəffəqiyyətlə nümayiş etdirdik.

Lakin, C++ dilində bu problemi başqa bir formada həll etməyə imkan verən mexanizm mövcuddur. Bu - konstruktordur.

Konstruktor (*construct - layihələndirmək*) – sinfin adı ilə eyni adda elan edilən, xüsusi üzv funksiyadır. Konstruktor üçün qaytarma tipi təyin eilmir. Hətda void!!! Konstruktorun yaradılmasına aid nümunəyə baxaq:

```
#include  <iostream>
using      namespace
std; class Test{
     //müraciət spesifikatoru göstərilmədiyi üçün
     //verilmiş dəyişən susmaya görə bağlıdır
     //sinif xaricindən müraciət üçün(private)
     int one;
     //public müraciət spesifikatoru
     //ondan sonra gələn bütün üzvlər
     //xaricdən müraciətə açıq olacaqlar
public:
     Test(){
          one=0;
          two=0;
     }
     //sinifdə dəyişənlərin qiymətləndirilməsi
     //yaradılan zaman qadağan olunduğu üçün biz
     //bunu etmək üçün metod təyin edirik.
     void Initial(int o,int t){
          one=o;
          two=t;
     }
     //sinfin dəyişənlərinin qiymətlərini ekranda
     //göstərən metod
     void Show(){
          cout<<"\n\n"<<one<<"\t"<<two<<"\n\n";
     }
     int two;
};
void main(){
     //Test tipində obyekt yaradılır
     Test obj; //(burda konstruktor işləyir)
     //ekranda göstərmə
     obj.Show(); // 0 0
     //parametrləri qiymətləndirilən funksiya
     //çağırılır
     obj.Initial(2,5);
     //ekranda göstərmə
     obj.Show(); //2 5
    //açıq two dəyişəninə birbaşa yazma
```

```
      //one dəyişəninə bunu tətbiq etmək mümkün
      //deyil, belə ki, ona müraciət bağlıdır
      obj.two=45;

      //yenidən ekranda göstərmə
       obj.Show(); //2 45
}
```

1. Konstruktor obyekt yaradılarkən avtomatik çağırılır, yəni, xüsusi çağırış lazım deyil.
2. Konstruktorun əsas təyinatı – obyektin qiymətləndirilməsidir.
3. Konstruktor həmişə public olmalıdır!!!

Parametrlər vasitəsilə konstruktora sinfin obyektlərini qiymətləndirmək üçün istənilən verilənləri ötürmək olar.

Misal. *Nöqtə təyin edən sinif:*

```
#include <iostream>
using namespace std;
//sinfin təyini
Point class Point {
      int x, y;
      //nöqtənin koordinatları susmaya görə
      //private müraciət səviyyəsindədir
public:
      //konstruktor sinfin x və y dəyişənlərinə uyğun
      //olaraq başlanğıc qiymətləri mənimsədir
      Point(int x0, int y0)
      {
            x = x0;
            y = y0;
      }
```



```
      //nöqtənin koordinatları ekrana verilir
      void ShowPoint()
      {
            cout << "\nx = " << x;
            cout << "\ny = " << y;
       }
};

void main()
{
      Point A(1,3); //koordinatları x = 1, y = 3 olan
      //A nöqtəsi (Point tipində obyekt) yaradılır
      //konstruktor çağırılır
      Point(1, 3)
      A.ShowPoint(); //A nöqtəsinin koordinatları
      //ekrana verilir
}
```

Qeyd: Obyekt yaradılarkən parametrlərin qiymətləri konstruktora funksiyanın adi qaydada çağırılmasına uyğun sintaksisdən istifadə etməklə ötürülür.

Konstruktor haqqında daha bir neçə məlumat

1. Parametrsiz konstruktor susmaya görə konstruktor adlanır. Bucür konstruktorlar adətən sinfin üzv dəyişənlərinə çox istifadə edilən qiymətləri mənimsədirlər.

```
Point()
{
      x = 0;
      y = 0;
}
```

2. Hər bir sinif üçün yalnız bir susmaya görə konstruktor təyin edilə bilər.

3. Əgər sinif üçün heç bir konstruktor təyin edilməmişdirsə, kompilyator konstruktoru susmaya görə yaradır. Bu cür konstruktor heç bir başlanğıc qiymət qaytarmır, o sadəcə mövcud olur :))).

Destruktorlar

Destruktor konstruktora görə əks funksiyanı yerinə yetirir. **Деструктор** (*destruct* - *yoxedici*) – obyektin ləğv edilməsi zamanı avtomatik olaraq çağırılan xüsusi funksiyadır – məsələn, obyekt görünmə sahəsindən çıxdığı zaman. Obyektin ləğv edilməsi zamanı destruktor istənilən məsələni həll edə bilər. Məsələn, əgər konstruktorda yaddaş dinamik ayrılmışdırsa, onda destruktor sinfin obyektinin ləğv edilməsindən əvvəl bu yaddaşı boşaltmalıdır.

Destruktorla işin əsas xüsusiyyətləri

1. Destruktor heç bir parametr qəbul etmir və heç bir qiymət qaytarmır.
2. Sinif yalnız bir destruktor ehtiva edə bilər.

Və son olaraq konstruktor və destruktorun çağırılma ardıcıllığını əks etdirən misala baxaq.

```
#include <iostream>
using namespace std;
//CreateAndDestroy sinfinin təyin edilməsi
class CreateAndDestroy
{
public:
     CreateAndDestroy(int value) //konstruktor
     {
          data = value;
          cout << " Object " << data << " constructor";
     }

     ~CreateAndDestroy() //destruktor
     {
          cout << " Object " << data <<
             " destructor" << endl;
     }
private:
     int data;
};

void main ()
{
     CreateAndDestroy one(1);
     CreateAndDestroy two(2);
}

PROGRAM OUTPUT

Object 1 constructor
Object 2 constructor
Object 2 destructor
Object 1 destructor
```

Ev tapşırığı

1. Rəqəmsal sayğac məhdud diapazonlu dəyişəndir. Onun tam qiyməti təyin edilmiş maksimum qiymət həddinə (məsələn, k 0-100 diapazonunda qiymət alır) çatdığı zaman sıfırlanır. Bu cür sayğacın parametri kimi rəqəmsal saatı, kilometrölçən sayğacı nümunə göstərmək olar. Bu cür sayğac üçün sinfi təyin edin. Maksimal və minimal qiymətlərin verilməsini, sayğacın qiymətinin 1 vahid artırılmasını, cari qiymətin qaytarılmasını təmin edin.
2. Tələbələr qrupunu əks etdirən sinif təyin edin. Tələbə həmçinin uyğun sinif vasitəsilə reallaşdırılır.