Obyektyönlü proqramlaşdırma

C++

Dərs №12

C++ dili ilə obyektyönlü proqramlaşdırma

Mündəricat

Daxili sinif 4
Daxili sinif ilə işləməyin əsas xüsusiyyətləri 4
Ümumiləşdirmə (aqreqasiya) və tərtib (kompozisiya) 15
Varislilik. Əsas anlayışlar 19
Varisliliyə aid şərti nümunə 20
Varislik zamanı müraciət spesifikatorları və varisliliyin təşkilinin sintaksisi 22
Müraciət spesifikatorları 22
Varislilik sintaksisi 23
Konstruktor və destruktorlar haqqında bir neçə məqamlar . 24
Vahid varisliliyin reallaşdırılmasına aid nümunə 24

Mürəkkəb varislilik 27
Mürrəkəb varisliliyin bəzi xüsusiyyətləri 27
Mürrəkəb varisliliyin istifadəsinə aid nümunə 28
Varisliliyin müsbət və mənfi cəhətlərinin müzakirəsi ... 31
Şablonların varisliliyi 33
Ev tapşırığı 35

Daxili sinif

Salam!!! Bugün biz siniflərin qarşılıqlı təsirinin müxtəlif formalarına (və ya, başqa sözlə yenidən istifadə mexanizmlərinə) baxacağıq. İndi biz sizinlə sadə mövzudan başlayaq – daxili siniflər. Beləliklə, başlanğıc üçün tərifə baxaq:

Daxili sinif (inner class) — tamamilə başqa sinfin daxilində təyin edilən sinifdir. Əgər, adi sinfin obyekti bir qayda olaraq sərbəst mövcud olursa, daxili sinfin obyekti isə, bu sinfin daxil olduğu sinfin obyektinə bağlı olmalıdır. Bir sinfin daxilində olan sinif əhatəli sinif adlanır.

Daxili sinif ilə işin əsas xüsusiyyətləri

Daxili sinif əhatəli sinfin üzvüdür, onun təyini isə əhatəli sinfin istənilən public, private və ya protected bölməsində ola bilər.

Daxili sinfin adı əhatəli sinfin görünmə oblastında aşkardır. Bu o deməkdir ki, əhatəli sinfin görünmə oblastının eyni adı ilə konflikt yaratmır. Məsələn,

```
//sərbəst (qlobal) sinif - sinif A
A class A { /* ... */ };
```



```
class B {
public:

    //Daxili A sinfi B sinfinin görünmə oblastı
    //daxilində inkapsulyasiya edilmişdir
    class A {...};

    //Burda daxili A sinfi istifadə edilir
    A*obj;
};
void main(){
    //Daxili  A  sinfi  verilmiş  görünmə  oblastında
    //görünmür buna görə də burada qlobal A sinfi
    //istifadə edilir
    A A*obj2;
}
```

Daxili sinif üçün daxili olmayan sinifdə olduğu kimi üzv formaları istifadə edilir. Məsələn:

```
class A {

public:
    class B {
        friend class A; //Dostun elan edilməsi
        B( int val=0 ); //Konstruktor
        B *next; //məxsusi sinif göstəricisi
        int value;
    };

private:
    B *obj;
};
```

Xatirladırıq ki, üzv o zaman bağlı adlanır ki, sinfin yalnız müəyyən üzvləri və ya dostları üçün əlyetərlidir. Əhatəli sinfin daxili sinfin bağlı üzvlərinə müraciət haqqı yoxdur.

A sinfinin üzvlərinin təyin edilməsi üçün B sinfinin bağlı üzvlərinə müraciət etmək olar, B sinfi A sinfinin dostudur.

Əvvəlki mülahizənin davamı kimi qeyd etmək lazımdır ki, daxili sinfin də əhatəli sinfin bağlı üzvlərinə heç bir müraciət haqqı yoxdur. Əgər B sinfinə A sinfinin bağlı üzvlərinə müraciət haqqı verilmiş olarsa, onda əhatəli A sinfi daxili sinfi dostu elan etməlidir.

Yuxarıda verilmiş nümunədə bu edilməmişdir, buna görə də B sinfi A sinfinin bağlı üzvlərinə müraciət edə bilməz.

B sinfini A sinfinin açıq üzvü (public bölməsində) o deməkdir ki, daxili sinifi bütün proqramda, həmçinin sinfin üzvlərinin və dostlarının təyini xaricində istifadə etmək olar. Məsələn:

```
void main(){
    A::B *ptr;
}
```

Bu üsul daxili sinif üçün geniş görünmə oblastı verir. Lakin daxili sinif adətən əhatəli sinfin xüsusi ehtiyacları üçün istifadə edilir və bütün proqramda əlyetərli olmamalıdır. Buna görə də daxili B sinfini A sinfinin qapalı üzvü kimi növbəti şəkildə elan etmək lazımdır:

```
class A {

public:
    //...
private:
   class B {
      // ...
   };
   B *obj;

};
```


İnti B tipi yalnız A sinfinin üzv və dostlarının təyinində əlyetərlidir, buna görə də B sinfinin bütün üzvlərini açıq elan etmək olar.

Bu cür yanaşmada A sinfini B sinfinin dostu kimi elan etməyə ehtiyac olmur. Aşağıda A sinfinin yeni təyini verilmişdir:

```
class A{

public:
    //...
private:

    //Artıq B qapalı daxili tipdir
    class B {
    //onun üzvləri isə açıqdır
    public:
       B( int val=0 );
       B *next;
       int value;
    };

    B *obj;

};
```

Diqqətinizə çatdırırıq ki, B sinfinin konstruktoru (əvvəlki nümunədə) sinfin təyininin daxilində daxili kimi verilməmişdir və nəticə olaraq onun xaricində təyin edilməlidir.

Belə bir sual açıq qalır: "Harada?". B sinfinin konstruktoru A sinfinin üzvüdür və deməli, sonuncunun gövdəsində təyin edilə bilməz. Nəticə olaraq, onu əhatəli sinfin təyinini ehtiva edən qlobal görünmə oblastında təyin etmək lazımdır. Burda belə bir qayda meydana gəlir:

Daxili sinfin üzv-funksiyası gövdədə daxili kimi təyin edilmədiyi zaman o əhatəli sinfdən xaricdə təyin edilməlidir.

B sinfinin konstruktorunun təyini belə görünməlidir. Lakin aşağıda verilmiş sintaksis qlobal görünmə oblastında düzgün deyil:

```
class A {
public:

private:

    class B {

    public:
        B( int val=0 );

    };
};

//səhv: Görünmə oblastının xaricində
B:: B( int val ) { ... }
```

Problem ondan ibarətdir ki, B sinfinin adı qlobal görünmə oblastında yoxdur. Onun bu şəkildə istifadəsi zamanı B sinfinin A sinfinin görünmə oblastında daxili sinif olduğunu göstərmək lazımdır.

Bu növbəti şəkildə edilir:

```
//daxili sinfin adına onun əhatəli sinfi vasitəsilə
//müraciət edirik
//bu qlobal görünmə oblastında baş verir
A::B::B( int val ) {
   value = val;
   next = 0;
}
```

Əgər B sinfi daxilində statik üzv elan edilərsə, onda onun təyinini qlobal görünmə oblastına yerləşdirmək lazımdır. Bu üzvün qiymətləndirilməsi (inisializasiyası) növbəti şəkildə ola bilər:

```
int A::B::static_mem = 1024;
```

Qeyd: Diqqət edin ki, üzv-funksiyalar və statik üzv-verilənlər daxili sinfin açıq üzvləri olmaya bilər, bu da onun üçündür ki, onu sinfin gövdəsinin xaricində təyin etmək mümkün olsun. B sinfinin qapalı üzvləri, həmçinin qlobal görünmə oblastında təyin edilirlər.

Daxili sinfi əhatəli sinfin xaricində təyin etməyə icazə verirlir. Məsələn, B sinfinin təyini qlobal görünmə oblastında ola bilər:

```
class A {

public:
   //...
```

```
private:
   //Burada yenilənmə vacibdir
   class B;
   B *obj;
};


//Daxili sinfin adı əhatəli sinfin adı ilə qeyd
//edilmişdir
class A::B {
public:
   B( int val=0 );
   B *next;
   int value;
};
```

Qlobal təyində daxili B sinfinin adı əhatəli A sinfinin adı ilə qeyd edilməlidir. Diqqət edin ki, B sinfinin A sinfi gövdəsində yenilənməsinə yol vermək olmaz. Daxili sinfin təyini, əgər əhatəli sinfin üzvü tərəfindən elan edilməzsə, qlobal görünmə oblastında verilə bilməz. Lakin bu zaman daxili sinifin əhatəli sinfin açıq üzvü olması vacib deyil.

Nə qədər ki, kompilyator daxili sinfin təyinini görməyib, onun göstəricisini və istinadını elan etməyə icazə verilir. A sinfinin obj üzvünün elan edilməsi (yuxarıda göstərilmiş nümunədəki kimi) düzgündür, baxmayaraq ki, B sinfi qlobal görünmə oblastında təyin edilmişdir, belə ki, bu üzv göstəricidir. Əgər onlardan biri obyekt olsa idi, onda onun A sinfində elan edilməsi kompilyasiya xətası verəcəkdi:


```
class A {

public:
   //...

private:
   //Elan vacibdir
   class B;
   B *obj;
   B x;   //səhv: təyin edilməmiş daxili sinif

};
```

Qeyd: Əhatəli sinfin gövdəsi xaricində nəyə görə daxili sinif təyin etmək lazımdır? Mümkündür ki, o B sinfinin reallaşdırılmasının bəzi məqamlarını dəstəkləyir, bizə isə onu A sinfinin istifadəçilərindən gizlətmək lazımdır. Buna görə də biz daxili sinfin təyinini A sinfini ehtiva edən başlıq faylına yerləşdiririk. Beləliklə, B sinfinin təyini A sinfinin və onun üzvlərinin reallaşdırılmasını ehtiva edən cari faylın daxilində ola bilər.

Daxili sinfi əvvəlcə elan etmək, sonra isə əhatəli sinfin gövdəsində təyin etmək olar. Bu daxili siniflərdə bir-birinə istinad edən üzvlərin olmasına imkan verir:

```
class A {

public:
   //...
private:

   // A::B elanı
   class B;
```

```
    class Ref {
       //pli A::B* tipindədir
       B *pli;
    };

    // A::B təyini
    class B {
       //pref A::Ref* tipindədir
       Ref *pref;
    };

};
```

Əgər B sinfi Ref sinfinin təyinindən əvvəl elan edilməmişdirsə, onda pli üzvünün elanı səhvdir.

Daxili sinif əhatəli sinfin statik üzvlərinə, hətta onlar açıq olsa belə, birbaşa müraciət edə bilməz.

İstənilən belə müraciət göstərici, istinad və ya əhatəli sinfin obyekti vasitəsilə edilə bilər. Məsələn:

```
class A {

public:
   int init( int );

private:

   class A::B {

   public:

      B( int val=0 );
      void mf( const A & );
      int value;

   };
};
```



```
A::B::B { int val )
{
   //A::init() – A sinfinin statik olmayan üzvü
   //A  sinfinin  obyekt  və  ya  tip  göstəricisi
   //vasitəsi ilə istifadə edilə bilər

   value = init( val ); //səhv: init-in səhv
                        //istifadəsi
};
```

Sinfin statik olmayan üzvlərinin istifadəsi zamanı kompilyasiyanın bu cür üzvun aid olduğu obyekti identifikasiya etmə imkanı olmalıdır. B sinfinin üzv funksiyası daxilində this göstəricisi yalnız onun üzvlərinə qeyri-aşkar tətbiq edilir. Qeyri-aşkar this göstəricisinə görə biz bilirik ki, value üzvü konstruktorun çağırıldığı obyektə aiddir. B konstruktorunun daxilində this göstəricisi B* tipindədir. init() üzv-funksiyasına müraciət etmək üçün A tipində obyekt və ya A* tipində göstərici lazımdır.

Növbəti mf() üzv-funksiyası problemin həlli üçündür, belə ki, init() funksiyasına istinad-parametr vasitəsilə müraciət edir. Beləliklə, init() funksiyanın arqumentinə ötürülən obyekt üçün çağırılır:

```
void A::B::mf( A & i1 ) {

   // init() funksiyasına istinada görə müraciət edir
   memb. = i1.init();
}
```

Son olaraq qeyd etmək lazımdır ki, daxili sinif əsas sinfin iş prosesində hansısa lokal məsələlərin yerinə yetirilməsi üçündür.

Buna görə də, ondan məqsədsiz istifadə etməməyi məsləhət görürük. Sizin sinfi gələcəkdə başqa bir yeni sinifdə istifadə etmək lazım gələ bilər, və sadəcə obyekt yaratmaq yerinə hər şeyi yenidən yazmaq lazım gələcək.

Ümumiləşdirmə (aqreqasiya) və tərtib (kompozisiya)

Ümumiləşdirmə (aqreqasiya) və tərtib (kompozisiya). İndi biz sizinlə siniflər arasında münasibət quran daha bir mexanizmə baxacağıq. Bu mexanizm, bir çox dil texnologiyalarında, həmçinin C++ dilində də geniş istifadə edilir. O ümumiləşdirmə (və ya aqreqasiya) adlanır.

Ümumiləşdirmə — bir sinfin obyektinin (obyektlərinin) digər sinfin obyektinin tərkibinə daxil edilməsidir.

İndi biz sizə ümumiləşdirməyə aid misal göstərəcəyik, siz isə məmuniyyətlə heyrətlənəcəksiniz. Məlum olduğu kimi, siz belə bir şeyi əvvəl sərbəst etmişdiniz.

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

// "nıqtə" sinfi
class Point{

    //koordinatlar
    int X;
    int Y;
public:

    //konstruktor
    Point(){
        X=Y=0;
    }
   //koordinatların verilməsi
```

```
    void SetPoint(int iX,int iY){
        X=iX;
        Y=iY;
    }

    //koordinatlarin nümayişi
    void Show(){
        cout<<"----------------------------\n\n";
        cout<<X<<"\t"<<Y<<"\n\n";
        cout<<"----------------------------\n\n";
    }
};

//fiqur sinfi
class Figura{

    //ümumiləşdirmə nöqtəsi
    //(künclərin koordinatları)
    Point*obj;

    //künclərin sayı
    int count;
    //fiqurun rəngi
    int color;

    public:

    //konstruktor
    Figura(){
        count=color=0;
        obj=NULL;
    }

    //fiqurun yaradılması
    void CreateFigura(int cr,int ct){
        //əgər künclərin sayı üçdən azdırsa, bu fiqur
        //deyil
        if(ct<3) exit(0);
```



```
        //rəngin və künclərin sayının
        //qiymətləndirilməsi
        count=ct;
        color=cr;
        //nöqtə massivi üçün yaddaşın ayrılması
        obj=new Point[count];
        if(!obj) exit(0);

        //nöqtələrin koordinatlarının verilməsi
        int tempX,tempY;
        for(int i=0;i<count;i++){
            cout<<"Set X\n";
            cin>>tempX;
            cout<<"Set Y\n";
            cin>>tempY;
            obj[i].SetPoint(tempX,tempY);
        }
    }

    //fiqurun göstərilməsi
    void ShowFigura(){
        cout<<"----------------------------\n\n";
        cout<<"Color"<<color<<"\n\nPoints —
              "<<count<<"\n\n";
        for(int i=0;i<count;i++){
            obj[i].Show();
        }
    }

    //fiqur vardırsa yaddaşı boşaltmalı
    ~Figura(){
        if(obj!=NULL) delete[]obj;
    }
};

void main(){


    Figura f;
```

```
    f.CreateFigura(255,3);
    f.ShowFigura();
}
```

Qeyd etmək lazımdır ki, ümumiləşdirmənin özünün də xüsusi halı vardır ki, bu da tərtib adlanır. Bu halın tərifini verək:

Tərtib — hər bir ümumiləşdirilmiş obyektin özünün ümumi olduğu ümumiləşdirmə formasıdır.

Başqa sözlə, əgər bizim nümunəyə fiqur-arqumentlər dəstini yığan "şəkil" sinfi əlavə edəriksə, bu elə tərtib deməkdir.

Tərtib və ümumiləşdirmə bizim növbəti söhbət açacağımız varislilik anlayışı ilə forma və üslubuna görə rəqabət aparan kifayət qədər güclü vasitədir. Bunların mənfi və müsbət cəhətlərinə bu dərsin növbəti bölməsində baxacağıq.

Varislik. Əsas anlayışlar

Varislik məsələsinə baxmağa başlayarkən, qeyd etmək lazımdır ki, istənilən obyekt həll edilən məsələnin bəzi fraqmentinin xassələrini və davranışını, bu fraqmentə aid verilənləri və metodları vahid tam əlaqələndirərək bir yerə yığmaq üçündür.

Xatırlayaq. Hər bir obyekt sinfin konkret nümayəndəsidir. Eyni sinfin obyektləri müxtəlif adla, lakin tiplərinə və verilənlərin daxili adına görə eyni olurlar. Eyni sinfin obyektlərinə öz verilənlərini emal etmək üçün sinfin eyni funksiyaları və obyektləri ilə iş ücün təyin edilmiş eyni əməliyyatlar əlyetərlidir. Beləliklə, adı proqramçının öz istəyinə görə seçdiyi lazım olan sayda obyektlərin daxil edilməsinə imkan verən tip rolunda çıxış edir.

Lakin, müxtəlif siniflərin obyektləri və siniflərin özləri siniflərin əvvəl baxılan ierarxiyasına uyğun obyektlərin ierarxiyasının formalaşdığı varislilik ilə münasibətdə ola bilər.

Siniflərin ierarxiyası artıq mövcud olan siniflərin əsasında yeni siniflər təyin etməyə imkan verir. Mövcud siniflər adətən baza (valideyn) sinifləri adlanır. Baza sinifləri əsasında formalaşan yeni siniflər isə - törəmə siniflər (axın-siniflər, oğul siniflər) adlanır.

Törəmə siniflər varisliliyə görə öz baza sinfindən verilənləri və metodları alır, özünün məxsusi komponentləri ola bilər.

Bu zaman, faris üzvlər əjdadda təsvir edilirlər və yalnız baza sinfində qalırlar.

Verilmiş baza sinfinin metodlarının bəzi adları törəmə sinifdə yenidən təyin edilə bilər. Bu halda baza sinfinin uyğun komponentləri törəmə sinifdən əlyetərli deyil.

Törəmə sinifdən adları bu sinifdə yenidən təyin edilmiş baza sinfinin komponentlərinə müraciət etmək üçün görünmə oblastının göstərilməsi (dəqiqləşdirilməsi) operatorundan - '::' istifadə edilir.

İstənilən törəmə sinif, öz növbəsində başqa bir sinif üçün baza sinfi ola bilər və beləliklə siniflərin və obyektlərin ierarxiyası formalaşdırılır. İerarxiyada törəmə obyekt bütün baza obyektlərinin varisliliyi üçün varis olur. Başqa sözlə, obyektin özünün bütün baza siniflərinə müraciət imkanı olur.

Bizim bəhs etdiyimiz varislilik tipi – vahid varislilik adlanır. Lakin C++ dilində daha bir varislilik forması - mürəkkəb varisliliyə icazə verilir. Mürəkkəb varislilik – sinfin baza siniflərinin bir-biri ilə heç bir əlaqəsi olmayan üzvlərinin varislilik imkanıdır. Bu tip haqqında biz daha sonra ətraflı danışacağıq.

Varisliliyə aid şərti nümunə

Gəlin əvvəlcə "sözdə" adlanan şərti varisliliyə aid nümunəyə baxaq.

Beləliklə, tutaq ki, "ekranda nöqtə (mövqe)" sinfi var. Bunu baza sinfi hesab edəcəyik və onun əsasında "ekranda pəncərə" sinfini quraq.

Bu sinfin verilənləri baza sinfindən nöqtələrin koordinatlarını və pəncərənin genişliyi və yüksəkliyi kimi iki xüsusi dəyişənləri miras alacaq. Nəslin ehtimal edilən özəl və miras aldığı metodlara baxaq:

- Pəncərəni X oxu boyunca DX qədər sürüşdürən funksiya. ÖZƏL
- Pəncərəni Y oxu boyunca DY qədər sörüşdürən finksiya. ÖZƏL
- Yuxarı sol güncün X koordinatını verən funksiya VALİDEYN
- Yuxarı sol güncün Y koordinatını verən funksiya VALİDEYN
- Pəncərənin X oxu boyunca ölçüsünü (genişliyini) verən funksiya. ÖZƏL
- Pəncərənin Y oxu boyunca ölçüsünü (hündürlüyünü) verən funksiya. ÖZƏL
- Konstruktor — Pəncərənin sol yuxarı küncünü təyin edən verilmiş parametrə görə verilmiş adla yaradılmış pəncərəni ekranda yaradır. ÖZƏL
- Destruktor — verilmiş adda pəncərəni silir. ÖZƏL

Sadədir, elə deyilmi? Lakin bu yalnız nəzəriyyədir. Gəlin nə öyrəndiyimizin dəqiq tərifini verək. Beləliklə:

Varislilik — bir obyektin digər obyektin xassələrini alıb yalnız onun üçün xarakterik olan xüsusiyyətlər əlavə edən mexanizmdir.

Varislilik zamanı müraciət spesifikatorları və varisliliyin təşkilinin sintaksisi

Müraciət spesifikatorları

Siniflərin varisliliyi zamanı sinif üzvlərinə müraciət oblastı əsas rol oynayır. İstənilən sinif üçün onun bütün üzvləri onun fəaliyyət oblastında yerləşirlər. Funksiyaya məxsus istənilən funksiya istənilən verilənlər istifadə edə və sinfə məxsus istənilən funksiyanı çağıra bilər. Sinif xaricində ümumi halda yalnız public statusunda olan komponentlər əlyetərlidir.

Xatırladırıq ki, sinif üzvlərinə növbəti müraciət spesifikatorları mövcuddur:

1. ***Özəl*** (*private*) — metod və verilənlər yalnız təyin olunduqları sinfin daxilində əlyetərlidirlər.
2. ***Qorunmuş*** (*protected*) — metod və verilənlər təyin olunduqları sinfin daxilində əlyetərlidirlər və əlavə olara bütün törəmə siniflər üçün əlyetərlidirlər.
3. ***Ümumi*** (*public*) — sinfin komponentləri proqramın istənilən yerindən əlyetərlidirlər, yəni qlobaldlrlar.

Yuxarıda verilən bütün müraciət spesifikatorları sinfin yalnız konkret üzvləri üçün göstərilmirlər, həmçinin eyni zamanda varisliliyin statusunun təyin edilməsi üçün də istifadə edilə bilərlər. Bunun üçün törəmə sinfin təyini zamanı spesifikator birbaşa baza sinfinin əvvəlində qoyulur.

Baza və törəmə siniflərin müxtəlif birləşməsində müraciətin statusları haqqında razılaşmalar növbəti cədvəldə verilmişdir

Baza sinfində müraciət	Baza sinfindən əvvəl müraciət spesivikatoru	Törəmə sinfində müraciət
public	yoxdur	private
protected	yoxdur	private
private	yoxdur	private
public	public	public
protected	public	protected
private	public	private
public	protected	protected
protected	protected	protected
private	protected	private
public	private	private
protected	private	private
private	private	private

Varisliliyin sintaksisi

Və nəhayət, son olaraq varis-sinfin yaradılmasının sintaksisi.

```
class sinfin_adı: varislilik_spesifikatoru
 baza_sinfini_adı{
    sinfin_təsviri;
};
```

Sintaksisi daha ətraflı gözdən keçirək:

- **sinfin_adı** — yeni yaradılan sinfin adı.
- **varislilik_spesifikatoru** — varis üzvə müraciət spesifikatoru.
- **baza_sinfinin_adı** — miras əldə ediləcək sinif.
- **sinfin_təsviri** — yeni sinfin gövdəsi.

Konstruktor və destruktorlar haqqında bəzi məqamlar

Burada biz varislilik zamanı konstruktor və destruktorların işi ilə bağlı bəzi mülahizələr gətirəcəyik.

Baza sinfinin konstruktorları həmişə törəmə sinfin konstruktorunun icrasından əvvəl çağırılır və icra edilir.

Baza sinfinin destruktorları törəmə sinfin təyinindəki siniflərin əks düzülüşünə görə icra edilirlər. Bu şəkildə, obyektin ləğv edilməsi onun yaradılması ilə tərs mütənasibdir.

Sinfin obyektləri və baza sinfinin obyektləri üçün destruktorların çağırılması qeyri-aşkar icra edilir və proqramçının heç bir fəaliyyətini tələb etmir.

Beləliklə müəyyən nəzəri biliklər əldə etdik. Artıq təcrübədən keçirmək vaxtıdır.

Vahid varisliliyin reallaşdırılmasına aid nümunə

İndi isə bundan əvvəlki bölmələrin birində yaratdığımız konstruksiyanı reallaşdıraq.


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

// "nöqtə" sinfi
class Point{
protected:
    int x;
    int y;
public:
    Point(){
        x=0;
        y=0;
    }
    // x-in əldə edilməsi
    int&GetX(){
        return x;
    }
    // yin əldə edilməsi
    int&GetY(){
        return y;
    }
};

class MyWindow: public Point{
    int width;
    int height;

public:
    MyWindow(int W,int H){
        width=W;
        height=H;
    }
    //genişliyin əldə edilməsi
    int&GetWidth(){
        return width;
    }
```

```
    //yüksəkliyin əldə edilməsi
    int&GetHeight(){
        return height;
    }
    //sürüşdürmə funksiyası
    void MoveX(int DX){
        x+=DX;
    }
    void MoveY(int DY){
        y=DY;
    }
    //ekranda göstərmə
    void Show(){
        cout<<"--------------\n\n";
        cout<<"X = "<<x<<"\n\n";
        cout<<"Y = "<<y<<"\n\n";
        cout<<"W = "<<width<<"\n\n";
        cout<<"H = "<<height<<"\n\n";
        cout<<"--------------\n\n";
    }
};
void main(){
    //obyektin yaradılması
    MyWindow A(10,10);
    A.Show();
    //parametrlərin dəyişdirilməsi
    A.GetX()=5;
    A.GetY()=3;
    A.GetWidth()=40;
    A.GetHeight()=50;
    A.Show();
    // "pəncərənin" sürüşdürülməsi
    A.MoveX(2);
    A.MoveY(7);
    A.Show();
}
```


Mürəkkəb varislililk

C++ dilində törəmə sinif bir neçə baza sinfindən birbaşa törəyə bilər. Törəmə sinfin birdən çox baza sinfinin olması mürəkkəb varislilik adlanır. Mürəkkəb varislilik sintaksis olaraq vahid varislilikdən birdən çox element ehtiva edən baza siyahısına görə fərqlənir. Məsələn, belə:

```
class A
{
    //A sinfinin təsviri
};

class B
{
    // B sinfinin təsviri
};

class C : public A, public B
{
    // C sinfinin təsviri
};
```

Mürəkkəb varisliliyin bəzi xüsusiyyətləri

Törəmə sinfin üzv-obyektlərinin yaradılması zamanı baza siyahısında baza siniflərinin ardicilliği konstruktorların çağırılma ardıcıllığını susmaya görə təyin edir.

Daha əhəmiyyətlisi məhdudiyyətdir ki, sinfin eyni adı törəmə sinfin elanı zamanı baza siyahısında birdən artıq ola bilməz. Bu odeməkdir ki, törəmə sinfinin formalaşdırılmasında iştirak edən birbaşa baza sinifləri siyahısında təkrarlana elementlər olmamalıdır.

İndi isə, nəzəriyyə üzərində çox dayanmadan gəlin dərsin növbəti bölməsində verilmiş praktik nümüunəyə baxaq.

Mürəkkəb varisliliyin istifadəsinə aid nümunə

Nəsildən nəsilə ötürülən klassik nümunə. Varisliliyin sadə strukturunu yaradaq. Canlı – "sığın"ı götürək.

```
#include <iostream>
#include <string.h>
using namespace std;

// "buynuz" sinfi
class Roga{

protected:

    char color[25];
    int wes;
```



```
public:
    Roga(){

        strcpy(color,"Dirty");
        wes=20;
    }
    Roga(char  *c,int w){
        strcpy(color,c);
        wes=w;
    }
};

// "dırnaqlılar" sinfi
class Kopyta{
protected:

    char forma[25];
    int razmer;

public:
    Kopyta(){
        strcpy(forma,"Big");
        razmer=10;
    }

    Kopyta(char  *c,int w){

        strcpy(forma,c);
        razmer=w;
    }
};

// "buynuz" və "dırnaqlılar" sinfindən törəyən
//"sığın" sinfi
class Los:public Roga,public Kopyta{

    public:
    char name[255];
```

```
    Los(char  *c){
        strcpy(name,c);
    }
    //Nəslin funksiyaları hər iki baza
    //sinfinin elementlərinə müraciət
    //edə bilərlər
    void DisplayInfo(){

        cout<<"Name "<<name<<"\n";
       cout<<"Forma "<<forma<<"\n";
       cout<<"Color "<<color<<"\n";
       cout<<"Wes rogov "<<wes<<"\n";
       cout<<"Razmer kopyt "<<razmer<<"\n";
    }
};

void main()
{
    //nəsil-sinfinin obyektinin yaradılması
    Los l("Vasya");
    l.DisplayInfo();
}
```


Varisliliyin müsbət və mənfi cəhətlərinin müzakirəsi

Beləliklə, bugün biz sizinlə siniflərin öz aralarındakı qarşılıqlı münasibətinin bir neçə üsuluna baxdıq. Bu üsulların özünün müsbət və mənfi cəhətləri var. Bunları gözdən keçirək:

Varislilik kompilyasiya mərhələsində statik olaraq təyin edilir, onu istifadə etmək asandır, belə ki, o birbaşa proqramlaşdırma dili tərəfindən dəstəklənir.

Varislilik zamanı mövcud reallaşdırmanın dəyişdirilməsi məsələsi də sadələşir. Əgər nəsil yalnız bəzi əməliyyatları əvəz edirsə, onda digər əməliyyatlarda istifadə edilmiş olacaq, belə ki, onlar əvəz edilmişləri çağırırlar.

Lakin varisliliyin mənfi xüsusiyyətləri də vardır:

Birincisi, icra zamanı varisliliyin reallaşdırılmasını dəyişdirmək olmaz.

İkincisi, valideyn sinif bəzən özünün altsiniflərinin fiziki təsvirini hissə-hissə təyin edir. Valideyn və törəmə siniflərin reallaşdırılması möhkəm əlaqəlidir.

Obyektlərin təsviri (kompozisiyası) icra zamanı obyektlərin digər obyektə müraciətinin əldə edilməsi hesabına dinamik təyin edilir. Təsviri obyektlərin bir-biri ilə əlaqəyə riayət etməsi zamanı tətbiq etmək olar. Proqramın icrası zamanı istənilən obyekti başqası ilə əvəzləmək olar, yetər ki, onlar eyni tipdə olsunlar.

Bundan başqa, belə ki, obyektin reallaşdırılması zamanı ilk növbədə onun interfeysi reallaşdırılır, ona görə də reallaşdırmadan asılılıq kəskin düşür.

Lakin mexanizmin seçilməsi, həmişə olduğu kimi qoyulan məsələdən asılıdır.

Şablonların varisliliyi

Sizə məmuniyyətlə bildirmək istəyirəm ki, siniflərin şablonları siniflərin özü kimi varislilik mexanizmini dəstəkləyirlər. Varisliliyin bütün əsas ideyası bu zaman dəyişməz qalır, bu da siniflərin ierarxiyasına uyğun olaraq şablonların ierarxik strukturunu qurmağa imkan verir.

Tamamilə adi nümunəyə baxaq. Bu nümunə başqa şablondan törəyən yeni şablonun necə yaradılmasını nümayiş etdirir.

```
//valideyn-sinf
template <class T>
class Pair
{
    T a;
    T b;
    public:
    Pair (T t1, T
    t2); //...
};

//valideyn sinfinin konstruktoru
template <class T>
Pair <T>::Pair (T t1, T t2) : a(t1), b(t2)
{}

//nəsil-sinif

template <class T>
class Trio: public Pair <T>
{
```

```
      T c;
    public:
    Trio (T t1, T t2, T
    t3); //...
};

//Diqqət edin ki, valideyn konstruktorun
//çağırılması, T tipinin parametr kimi
//ötürülməsi ilə müşayiət edilir.
template <class T>
Trio<T>::Trio (T t1, T t2, T t3): Pair <T> (t1,
t2), c(t3)
{
}
```


Ev tapşırığı

1. Tələbə haqqında informasiya ehtiva edəcək Student sinfini yaradın. Varislilik mexanizmi ilə Aspirant (aspirant – namizadlik işini müdafiə etməyə hazırlaşan tələbədir) sinfini yaradın.
2. Azərbaycan vətandaşı haqqında informasiya ehtiva edəcək Passport sinfini yaradın. Varislilik mexanizmi vasitəsilə Passport sinfindən törəyən ForeignPassport (xarici pasport) sinfini reallaşdırın. Xatırladırıq ki, xarici pasport pasport məlumatlarından başqa, həmçinin viza, xarici pasportun nömrəsi kimi məlumatları da ehtiva edir.
3. Mürəkkəb varislilik anlayışından istifadə edərək "Kvadrat daxilinə çəkilmiş çevrə" sinfini yaradın.
4. Dərsin “Ümumiləşdirmə (aqreqasiya) və tərtib (kompozisiya)” bölməsində “nöqtə” və “fiqur” siniflərinin birliyinə aid nümunə verilmişdir. Sizin vəzifəniz bu proqrama kompazisiya yaratmaqla viqur sinfinin bir neçə obyektini ehtiva edəcək sinif əlavə etməkdən ibarətdir.