Moştenire în C++

Obiective:

- Clase derivate
- Mostenirea
- Controlul accesului
- Mostenirea multiplă
- Supraincarcarea functiilor membre
- Constructorii si destructorul clasei derivate

C++ permite mostenirea ceea ce înseamnă că putem deriva o clasa din alta clasa de baza sau din mai multe clase. Sintaxa este următoarea:

class Clasa_Derivata : [modificatori de acces] Clasa_de_Baza { …. } ;

sau

class Clasa_Derivata : [modificatori de acces] Clasa_de_Baza1, [modificatori de acces] Clasa_de_Baza2, [modificatori de acces] Clasa_de_Baza3 ……..

Clasa de baza se mai numeste clasa parinte sau superclasa, iar clasa derivata se mai numeste subclasa sau clasa copil.

class A : **public** B { /* declaratii */};

class A : **protected** B { /* declaratii */};

class A : **private** B { /* declaratii */};

Dacă lipseste modificatorul de acces, atunci e considerat implicit private.

Functiile membre din clasa derivată au acces doar la membrii publici si protected din clasa de bază, la cei privati nu au acces.

Tabelul din curs de acces din clasa derivată si din exterior la membrii din clasa de bază:

Base class	Access control	Access in derived class	External access
private protected public	private private private	not accessible private private	not accessible not accessible not accessible
private protected public	protected protected protected	not accessible protected protected	not accessible not accessible not accessible
private protected public	public public public	not accessible protected public	not accessible not accessible accessible

Studiati următorul exemplu, comentati si argumentati erorile:

```
#include <iostream.h>
class Unu
{
    int x;
    protected:
    int y;
    public:
    int z;
};

class Doi:Unu
{
    void f()
    {
            //x=10; //err
            y=20;
            z=30;
    }
};

class Trei:public Unu
{};

class Patru:Trei
{
    void f2();
};

void f()
{
    Unu u;
    Doi d;
    Trei t;
    u.x=11; //err
    u.y=22; //err
    u.z=33;

    d.x=11; //err
    d.y=22; //err
    d.z=33; //err

    t.x=11; //err
    t.y=22;//err
    t.z=33;
}

void Patru::f2()
{
```

```
    x=11;//???
    y=22;  //??
    z=33;  //??
}
```

Clasa de baza sau clasa părinte este cea originală, iar clasa derivată va mostenii caracteristicile clasei de bază.

Derivarea se face in modul următor:

```
class FisaBiblioteca:public Carte
{
    char autor[], editura[];
    public:
    FisaBiblioteca(char *titlu, char *autor, char *editura):Carte(titlu)
    {
            strcpy(FisaBiblioteca::autor,autor);
            strcpy(FisaBiblioteca::editura,editura);
    }
};
```

In clasa de bază Carte avem definit si initializat deja titlu, el este mostenit in clasa FisaBibliotecă, nu mai trebuie redefinit.

Cand se va creea o nouă instantă a clasei FisaBibliotecă se va apela mai întai constructorul clasei Carte si apoi constructorul clasei FisaBibliotecă. Constructorul clasei Carte trebuie specificat, are un singur parametru, dacă nu compilatorul va genera eroare. Dacă, constructorul clasei Carte nu are nici un parametru, atunci compilatorul nu va genera eroare, nu se va impune sa specificăm acest constructor.

Membri protejati din clasa de bază sunt accesibili in clasa derivată si pot fi modificati.

Exemplu:

```
class Carte
{
    char titlu[64];
    public:
    Carte(char *titlu)
    {
            strcpy(Carte::titlu,titlu);
    }
    void afis_carte()
    {
            cout<<"titlul cartii este "<<titlu<<endl;
    }
    protected:
    float cost;
    void afis_cost()
    {
            cout<<"cartea costa "<<cost<<endl;
    }
};
```

```
class FisaBiblioteca:public Carte
{
    char autor[64], editura[64];
    public:
    FisaBiblioteca(char *titlu, char *autor, char *editura):Carte(titlu)
    {
            strcpy(FisaBiblioteca::autor,autor);
            strcpy(FisaBiblioteca::editura,editura);
            cost=49.98;
    }
    void afis_biblio()
    {
            afis_carte();
            afis_cost();
            cout<<"autor: "<<autor<<" editura "<<editura<<endl;
    }
};
void main()
{
FisaBiblioteca fisa("C++ Programming", "Stroustrup", "Editura");
fisa.afis_biblio();
}
```

Cand folosim o derivare protected din clasa de baza, atunci toti membri clasei de baza vor fi privati pentru clasa derivată. La derivarea privată o functie de interfata ne va permite sa accesam membrii privati din clasa de baza, dar nu ii putem modifica.

Mostenire multiplă

O clasa poate mostenii caracteristicile a mai multor clase. Sintaxa am vazut-o mai sus.

Constructorii si destructorii claselor de baza si ai celei derivate:

```
#include <iostream.h>
class Carte
{
    char titlu[64];
    float static const cost2=100;
    protected:
    void afis_cost2()
    {
            cout<<"costul2 "<<cost2<<endl;
    }
    public:
    Carte(char *titlu)
    {
            strcpy(Carte::titlu,titlu);
            cout<<"constructorul clasei de baza1"<<endl;
    }
    void afis_carte()
    {
```

```
            cout<<"titlul cartii este "<<titlu<<endl;
        }
        ~Carte()
        {
            cout<<"destructorul clasei de baza1"<<endl;
        }
        protected:
        float cost;
        void afis_cost()
        {
            cout<<"cartea costa "<<cost<<endl;
        }
    };

    class Pagini
    {
        protected:
        int linii;
        public:
        Pagini(int linii)
        {
            Pagini::linii=linii;
          cout<<"constructor clasa de baza2"<<endl;
        }
        void afis_pagini()
        {
            cout<<"cartea are nr de pagini "<<linii<<endl;
        }
        ~Pagini()
        {
            cout<<"destructorul clasei de baza2"<<endl;
        }

    };

    class FisaBiblioteca:Carte,Pagini
    {
        char autor[64], editura[64];
        public:
        FisaBiblioteca(char *titlu, char *autor, char *editura):Carte(titlu),Pagini(166)
        {
            strcpy(FisaBiblioteca::autor,autor);
            strcpy(FisaBiblioteca::editura,editura);
            cost=49.98;
            cout<<"Constructorul clasei derivate"<<endl;
        }
        void afis_biblio()
        {
            afis_carte();
            afis_cost();
```

```
            afis_cost2();
            cout<<"autor: "<<autor<<" editura "<<editura<<endl;
            afis_pagini();
        }
        ~FisaBiblioteca()
        {
            cout<<"Destructorul clasei derivate"<<endl;
        }
};
void main()
{
FisaBiblioteca fisa("C++ Programming", "Stroustrup", "Editura");
fisa.afis_biblio();
}
```

Probleme:

1. Folositi-vă imaginatia si creati un program în C++ pentru a ilustra cat mai bine conceptul de mostenire, mostenire multiplă. (De exemplu: intr-o familie părintii au 2 copii, o fata si un băiat, fiecare dintre acestia mosteneste cate ceva diferit de la mama, de la tata, de la bunici, de la verisori, cum ar fi culoarea ochilor, culoarea parului, ….).

2. Atunci cand programele noastre derivează două sau mai multe obiecte dintr-o clasă de bază comună pot apărea erori de ambiguitate. Ce putem face pentru a preveni aparitia mai multor copii ale clasei de bază intr-un obiect dat, adică ce putem face pentru a preveni ca programele nostre să nu mostenească mai multe copii ale unei clase de bază date? Justificati răspunsul dat printr-un program concret!

3. Cum rezolvăm conflictele de nume dintre clasele de bază si cele derivate?

Răspuns problema 3:

-utilizăm operatorul de rezolutie pentru a fi clar pentru compilatorul nostru pe care functie să o utilizeze.

Exemplu:

```
class Baza
{
  public:
  void afisare()
  {
        cout<<"Suntem in clasa de baza"<<endl;
  }
};

class Derivata:public Baza
{
  public:
  void afisare()
  {
        cout<<"Suntem in clasa derivata"<<endl;
  }
};
void main()
{
  Derivata obiect;
  obiect.afisare(); //se va utiliza functia din clasa Derivata
  obiect.Baza::afisare(); //se va utiliza functia din clasa Baza
}
```