LUCRARE DE LABORATOR NR. 5
LA DISCIPLINA „PROGRAMAREA ÎN LIMBAJUL C++”

Tema: **Moştenirea multiplă**

Scopul lucrării:

- Studierea regulilor de determinare a moştenirii multiple;
- Studierea avantajelor şi neajunsurilor moştenirii multiple;
- Probleme legate de utilizarea moştenirii multiple;
- Studierea rezolvării problemelor;

Noţiuni de bază

Motivarea

Moştenirea multiplă, reprezintă prin sine moştenirea de la două sau mai multe clase. Pentru a înţelege pentru ce ne trebuie moştenirea multiplă trebuie să ne amintim, că moştenirea simplă nu rezolvă toate problemele, aşa cum câteodată ne obligă să alegem dintre două clase de bază potrivite.

Este interesantă părerea lui Buch referitor la acest mecanism: „Moştenirea multiplă este ca o paraşută: de regulă, nu ne trebuie, dar, când întâmplător o să ne trebuiască, o să ne pară rău dacă nu o să fie la îndemână” .

În aşa fel, acest mecanism este foarte necesar. Dar, el nu se foloseşte în toate limbajele, dar este realizat în C++. Ca de exemplu: trebuie de descris clasa „Fereastra” cu buton şi denumire, însă clasele Fereastra, Fereastra cu buton, Fereastra cu denumire, sunt deja create şi reprezintă prin sine următoarea ierarhie.

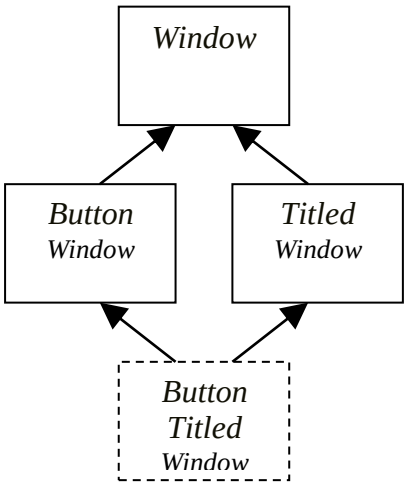

Figura 1 Moştenirea de la două clase înrudite.

Ambele clase pot fi utilizate în calitate de clasă de bază, pentru noua clasă, încă ne creată. În lipsa moştenirii multiple ar fi fost necesar de ales şi de completat funcţionalităţile care nu ajung. Nu numai că utilizatorul este pus în faţa unei alegeri, completarea codului, prin sine contrazice principiului de minimizare a codului şi utilizarea lui repetată.

Moştenirea multiplă are loc şi în viaţă:

- Practicantul este student şi colaborator vremelnic. El primeşte notă pentru lucrul său, ca student, şi în acelaşi moment, execută funcţiile de colaborator, şi posibil primeşte şi salariu.

- Tehnicianul dentist este un lucrător sanitar, aşa cum a absolvit universitatea de medicină, în acelaşi moment el execută sarcini mai mult specifice fierarilor sau giuvaerilor.
- Ornitorincul hrăneşte copiii săi cu lapte, însă depune ouă.
- Acţiunile reprezintă hârtii, ori valori de preţ.
- Clasa *iostream* moşteneşte clasele: *istream* şi *ostream*.

Ultimul exemplu nu este tocmai din viaţă, şi iarăşi din programare.

Definirea

Moştenirea multiplă se declară în felul următor:

```
class Student{
public:
  int mark;
...
};
class Worker{
public:
  int salary;
};
class Practicant: public Student, public Worker{
};
void PutMark(Student& s, int mark){
 s.mark = mark;
}
void PutSalary(Worker& w, int salary){
 w. salary = salary;
}

void main(){
Practicant p;
      PutMark(p,5);
      PutSalary(p,200);
}
```

Din exemplu se vede că, clasa derivată importă comportamentul ambelor clase de bază, aşa cum, corespunde ca parametru pentru ambele funcţii globale, adică principiul de substituţie rămâne în vigoare! Desigur că, este valabilă şi posibilitatea de utilizare a formelor de moştenire, se poate de moştenit *public* de la una şi protejat de la alta.

Dificultăţi

Ca şi orice alt instrument puternic şi frumos, moştenirea multiplă are neajunsurile sale, care au devenit motivul de eliminare a moştenirii multiple din multe limbaje moderne. Problemele apar din cauza apariţiei ambiguităţilor. Presupunem, că în ambele clase de bază există câmpuri cu unul şi acelaşi nume.

```
class A{
public:
  int x;
};
class B{
public:
  int x;
};

class C: public A, public B{

};
void main(){
 C c;
```

```
 c.x = 10;
}
```

În aşa fel, clasa C conţine două variabile cu unul şi acelaşi nume, aşa cum se moştenesc ambele, deoarece pentru fiecare clasa de bază variabila poate să aibă un sens propriu. Compilatorul nu poate decide, cărei din variabilele moştenite să-i atribuie valoarea nouă. Aceeaşi situaţie are loc şi în cazul funcţiilor.

Rezolvarea acestei ambiguităţi constă în utilizarea precizării numelui de variabilă. Numele variabilelor pot coincide, dar numele claselor nu coincid. Aşa că, pentru indicarea variabilei care este utilizată trebuie de indicat clasa, de la care este moştenită variabila:

```
c.A::x = 10;
c.B::x = 5;
```

Mai dificil este cînd clasele A şi B sunt înrudite, adică provin de la aceleaşi clase, aşa cum este prezentat în figura 1, cu toate că poate exista un caz mai dificil. În această configurare există două variabile identice, cu unul şi acelaşi sens. Se pot deosebi, aşa cum este prezentat mai sus. Dar problema constă în aceea că, ele şi după sens sunt identice, iar funcţiile definite la nivelul doi al ierarhiei, vor lucra cu propriile copii ale variabilelor, care des duc la greşeli semantice greu de găsit. Mai mult ca atât, în acest caz, constructorul clasei de bază este chemat de două ori. Pentru rezolvarea acestei probleme se utilizează ultima formă de moştenire: virtuală.

```
class A{
public:
  int x;
  A(int x){this->x=x;}
};
class B: virtual public A{
public:
  B(int x):A(x){}
};
class C: virtual public A{
public:
  C(int x):A(x){}
};
class D: public B, public C{
public:
  D(int x):A(x),B(x),C(x){}
}
```

Aşa cum se vede, clasele *B* şi *C* trebuie ambele să moştenească clasa A. Imediat se rezolvă problema despre existenţa a două variabile identice. Mai mult ca atât, în acest caz este necesar de chemat constructorul clasei de bază manual, aşa cum este arătat în exemplu. Deoarece nu întotdeauna ierarhia este proiectată de un elaborator, şi unele clase deja pot fi compilate, în asemenea cazuri rezolvarea problemei este, practic, imposibilă.

Întrebări de control:

1. Care sunt avantajele moştenirii multiple?
2. Care sunt problemele la realizarea moştenirii multiple?
3. Clasificaţi problemele legate de moştenirea multiplă?
4. Cum se rezolvă problemele legate de moştenirea multiplă?
5. De ce este necesară moştenirea virtuală?
6. Cum este realizată moştenirea virtuală?
7. Cum lucrează constructorii la moştenirea multiplă şi virtuală?

Sarcina

Pentru toate variantele este necesar de creat două programe, care ar ilustra ambele exemple date mai sus de moştenire multiplă.

Varianta 1

a) Să se creeze, o ierarhie de moştenire: student, colaborator - practicant.
b) Să se creeze, o ierarhie de moştenire: om - student, colaborator - practicant.

Varianta 2

- Să se creeze, o ierarhie de moştenire: mamifere, reptile – ornitorinc.

a) Să se creeze, o ierarhie de moştenire: animale - mamifere, reptile – ornitorinc.

Varianta 3

a) Să se creeze, o ierarhie de moştenire: televizor, dispozitiv digital – monitor.
b) Să se creeze, o ierarhie de moştenire: dispozitiv electric - televizor, dispozitiv digital – monitor.

Varianta 4

a) Să se creeze, o ierarhie de moştenire: stilou, creion – condei de ardezie.
b) Să se creeze, o ierarhie de moştenire: rechizite de birou - stilou, creion – condei de ardezie.

Varianta 5

a) Să se creeze, o ierarhie de moştenire: barca cu motor, motocicleta – motocicleta de apa.
b) Să se creeze, o ierarhie de moştenire: transport – barca cu motor, motocicletă – motocicleta de apa.

Varianta 6

a) Să se creeze, o ierarhie de moştenire: divan, pat – divan-pat.
b) Să se creeze, o ierarhie de moştenire: mobilă - divan, pat – divan - pat.

Varianta 7

a) Să se creeze, o ierarhie de moştenire: transport aerian, transport de pasageri – lainer *Boing 747*
b) Să se creeze, o ierarhie de moştenire: transport - transport aerian, transport de pasageri - lainer *Boing 747*

Varianta 8

a) Să se creeze, o ierarhie de moştenire: undiţă, telescop – undiţă telescopică.
b) Să se creeze, o ierarhie de moştenire: obiect - undiţă, telescop – undiţă telescopică.

Varianta 9

a) Să se creeze, o ierarhie de moştenire: hârtie, valori – acţiuni.
b) Să se creeze, o ierarhie de moştenire: obiect - hârtie, valori – acţiuni.

Varianta 10

a) Să se creeze, o ierarhie de moştenire: vehicul, camion – maşina de teren.
b) Să se creeze, o ierarhie de moştenire: automobil - vehicul, camion – maşină de teren.

Varianta 11

a) Să se creeze, o ierarhie de moştenire: carte, caiet – caiet de notiţe.
b) Să se creeze, o ierarhie de moştenire: hârtie - carte, caiet – caiet de notiţe.

Varianta 12

a) Să se creeze, o ierarhie de moştenire: avion, corabie – elicopter de apă.
b) Să se creeze, o ierarhie de moştenire: transport - elicopter, corabie - elicopter de apa.