LUCRAREA DE LABORATOR №6

Tema: Polimorfism. Funcţii virtuale

Scopul lucrării:

- Studierea polimorfismului;
- Studierea principiilor legăturii întîrziate;
- Studierea funcţiilor virtuale;
- Polimorfismul *ad-hoc*;
- Realizarea funcţiilor virtuale;
- Studierea claselor abstracte.

Noţiuni de bază

Cuvântul polimorfism provine din greacă şi aproximativ se traduce ca „*multe forme*" (*poly* – multe, *morphos* - formă). În viaţă tipurile polimorfe – sunt acelea, care se caracterizează printr-o diversitate de forme sau caracteristici. În chimie legăturile polimorfe se pot cristaliza, cel puţin, în două forme diferite (de exemplu, carbonul are două forme cristaline –grafitul şi diamantul). Din alt punct de vedere, inginerul TI este în primul rând om, şi apoi este inginer (principiul de substituţie).

În limbajele de programare un obiect polimorf – este o entitate (variabilă, argumentul funcţiei), care păstrează, în timpul executării programului, valorile diferitor tipuri. Funcţiile polimorfe – sunt acele funcţii, care au argumente polimorfe.

În C++ polimorfismul rezultă firesc din :

- Relaţia *"a fi exemplar"*;
- Mecanismul de expediere a mesajelor;
- Moştenire;
- Principiul de substituţie.

Unul din avantajele principale a programării orientate pe obiecte, constă în posibilitatea de a combina aceste mijloace. În rezultat se obţine un set bogat de exemple tehnice împreună cu utilizarea repetată a codului.

Variabila polimorfă: conţine valoarea, care se referă la diferite tipuri de date. Variabilele polimorfe realizează principiul de substituire. Cu alte cuvinte, cu toate că pentru asemenea variabile există un tip de date aşteptat, tipul real poate fi un subtip a tipului aşteptat. În C++ variabilele polimorfe există numai ca referinţe şi pointeri.

Funcţiile polimorfe – reprezintă una din cele mai puternice tehnici de programare obiect orientate. Ele permit concomitent să scrii cod la un nivel înalt de abstractizare şi apoi sa-l utilizezi într-o situaţie concretă. De obicei programatorul execută ajustarea codului cu ajutorul transmiterii mesajelor adăugătoare destinatarului, care utilizează metoda. Aceste metode adăugătoare, deseori nu sunt legate cu clasa la nivelul abstractizării metodei polimorfe. Ele sunt metode virtuale, care se definesc pentru clasele de nivel mult mai jos.

Când unei adevărate variabile (adică nu unei referinţe şi nu a pointerilor) i se atribuie o valoare de tipul subclasa, atunci valoarea clasei dinamice se transmite obligatoriu aşa, pentru a coincide cu tipul static al variabilelor.

Totuşi la utilizarea referinţelor sau a pointerilor valoarea salvează tipul său dinamic.

Legături întîrziate (*late binding*)

Legătura întîrziată reprezintă mecanismul care permite definirea tipului dinamic în timpul executării programului, dar nu în timpul compilării. Un mecanism asemănător sunt descriptorii de fişiere, aşa cum fişierele se deschid în timpul executării programului, dar nu în timpul compilării. Acest mecanism este baza polimorfismului, aşa cum realizează funcţiile virtuale.

Funcţiile virtuale

Legăturile întîrziate rezolvă problema, dar, ea nu are cunoştinţe nemijlocite în limbaj. De aceea, pentru referire este necesar de utilizat funcţiile virtuale, care se scriu cu utilizarea cuvântului rezervat *virtual*. Funcţiile virtuale se deosebesc de cele obişnuite numai prin metodele de acces. Dar utilizarea lor are sens numai cu utilizarea referinţelor sau a pointerilor.

Aducem un exemplu:

```
#include<iostream.h>
class Animal{
public:
   void Say(){ cout<<"!!!\n";}
};
class Dog: public Animal{
public:
   void Say(){ cout<<"GAV\n";}
};
class Cat: public Animal{
public:
   void Say(){ cout<<"MIAU\n";}
};
void FunSay(Animal a){
 a.Say();
}
void main(){
 Animal a;
 Dog    d;
 Cat    c;
 FunSay(a);
 FunSay(d);
 FunSay(c);
}
```

În acest exemplu funcţiile globale nu sunt polimorfe, aşa cum variabilele se transmit după valoare, ceea ce duce la pierderea caracteristicilor specifice ale clasei derivate, mai mult ca atât, funcţia *Say* nu este virtuală. Deci, pentru funcţionarea corectă a acestei programe sunt necesare următoarele modificări: funcţia Say trebuie să fie definită ca virtuală, dar parametrul funcţiei globale trebuie să fie definit ca o referinţă sau un pointer.

Polimorfismul *ad-hoc*

Particularitatea confuză a redefinirii metodelor în limbajul C++ constă în deosebirea între redefinirea metodelor virtuale şi nevirtuale.

Mai multe confuzii apar, dacă programatorul încearcă să redefinească funcţia virtuală în subclasă, indicînd (posibil, din greşeală) alt tip de argumente. De exemplu, clasa de bază conţine descrierea:

```
virtual void display (char *, int);
```

Subclasa încearcă să predefinească metoda:

```
virtual void display (char *, short);
```

Aşa cum listele de argumente se deosebesc, atunci a doilea definire nu se recunoaşte ca o predefinire, dar ca supraîncărcare. De aceea, de exemplu, la apelarea în forma tipului părintelui se va alege prima metodă, darn u a doua. Aşa erori sunt greu de găsit, deoarece ambele forme de scriere sunt premise, şi să speri la o diagnoză de compilator nu trebuie.

Câţiva autori preferă să considere supraîncărcarea, şi şabloanele polimorfismului, aşa cum des se întâlnesc cu definirea mai multor forme. Totuşi trebuie de amintit, că polimorfismul se realizează cu legături târzii, în acel timp când problemele care apar la utilizarea supraîncărcării şi

a şabloanelor ce se rezolvă la compilare. De aceia aceste construcţii a limbajului uneori se numesc polimorfismul *ad-hoc*.

Clasele abstracte

Metodele de amânare (câteodată se numesc metode abstracte, dar în C++ metode virtuale) pot fi privite ca generalizarea predefinirii. În ambele cazuri comportamentul clasei părinte se modifică pentru moştenitor. Pentru metodele amânate, totuşi, comportamentul pur şi simplu nu este definit. Orice activitate folositoare este dată în clasa fiică.

```
class Shape {
public:
  ...
  virtual void draw() = 0;
  ...
};
```

Compilatorul nu permite utilizatorului crearea unui exemplar a clasei, care conţine pur metodele virtuale, de aceia aceste clase sunt numite abstracte. Subclasele trebuie să redefinească aceste metode. Redefinirea numai a metodelor virtuale trebuie să derive din descrierea ei în urmaşi, pentru care sunt create obiecte reale.

Întrebări de control:

1. Ce fel de mecanisme corespund cu definirea polimorfismului?
2. Şi funcţiile obişnuite şi cele virtuale pot fi predefinite. Prin ce se deosebesc ele?
3. Cum sunt fizic realizate funcţiile virtuale?
4. Ce reprezintă legătura cu întârziere? Daţi exemple.
5. Care sunt condiţiile de realizare a polimorfismului?
6. Care variabile sunt polimorfe?
7. Ce reprezintă polimorfismul *ad-hoc*?
8. Prin ce se deosebesc clasele abstracte de cele obişnuite?
9. Cum se defineşte funcţia pur virtuală?

Sarcina
Varianta 1

Creaţi clasa abstractă de bază *Worker* cu funcţia virtuală calcularea salariului. Creaţi clasele derivate *StateWorker, HourlyWorker* şi *CommissionWorker,* în care funcţia dată este redefinită. În funcţia *main* determinaţi masivul de pointeri la clasa abstractă, cărei i se atribuie adresele obiectelor claselor derivate.

Varianta 2

Creaţi clasa abstractă de bază *Figure* cu funcţia virtuală - aria. *Square, Circle, Triangle, Trapeze* în care funcţia dată este predefinită. În funcţia *main* determinaţi masivul de pointeri la clasa abstractă, cărei i se atribuie adresele obiectelor claselor derivate. Aria trapezului: $S=(a+b)h/2$.

Varianta 3

Creaţi clasa abstractă de bază *progresia* cu funcţia virtuală – suma progresiei. Creaţi clasele derivate: progresia aritmetică şi progresia geometrică. Fiecare clasă conţine două câmpuri de tip double. Primul – primul membrul al progresiei, al doilea (double) – restul permament (pentru aritmetică) şi relaţiile permanente (pentru geometrică). Definiţi funcţia de calculare a sumei, unde ca parametru este cantitatea de elemente a progresiei.
Progresia aritmetică $a_j=a_0+jd$, $j=0,1,2,\ldots$
Suma progresiei aritmetice: $s_n=(n+1)(a_0+a_n)/2$
Progresia geometrică: $a_j=a_0r^j$, $j=0,1,2,\ldots$
Suma progresiei geometrice: $s_n=(a_0-a_nr)/(1-r)$

Varianta 4

Creaţi clasa abstractă de bază *Mammal* – mamifere cu funcţia virtuală - descrierea. *Animal* şi *Human*. Pentru animal determinaţi clasele derivate *Dog* – câinele şi *Cow* – vaca, în care funcţia se predefineşte.

Varianta 5

Creaţi clasa abstractă de bază *Lines* cu funcţia virtuală *f(x)*. Creaţi clasele derivate *StraightLine, Ellipse, hyperbola* în care funcţia dată este predefinită. În funcţia *main* determinaţi masivul de pointeri la clasa abstractă, căreia i se atribuie adresele obiectelor diferitor obiecte. Ecuaţia dreptei: $y=ax+b$, elipsei: $x^2/a^2+y^2/b^2=1$, hiperbolă: $x^2/a^2-y^2/b^2=1$

Varianta 6

Creaţi clasa abstractă de bază *Figure* cu funcţia virtuală - perimetru. Creaţi clasele derivate - *Rectangle, Circle, Triangle, Rhomb* în care funcţia dată este predefinită. În funcţia *main* determinaţi masivul de pointeri la clasa abstractă, căreia i se atribuie adresele obiectelor diferitor obiecte.

Varianta 7

Creaţi clasa abstractă de bază *Container* cu funcţiile virtuale de inserare şi extragere. Creaţi clasele derivate *Stack* şi *Queue,* în care funcţiile date sunt predefinite. În funcţia *main* determinaţi masivul de pointeri la clasa abstractă, cărora li se atribuie adresele obiectelor claselor derivate.

Varianta 8

Creaţi clasa abstractă de bază *Figure* cu funcţia virtuală – aria suprafeţei. Creaţi clasele derivate paralelipiped, tetraedru, sferă în care funcţia dată este predefinită.
Aria suprafeţei paralelipipedului: $S=6xy$.
Aria suprafeţei sferei: $S=4\pi r^2$.
Aria suprafeţei tetraedrului: $S=a^{2\sqrt{3}}$

Varianta 9

Creaţi clasa abstractă de bază *Number* cu funcţia virtuală - media. Creaţi clasele derivate *Complex, Vector* din 10 elemente, *Matrix* 2 pe 2, în care funcţia dată este predefinită. În funcţia *main* determinaţi masivul de pointeri la clasa abstractă, cărora li se atribuie adresele diferitor obiecte.

Varianta 10

Creaţi clasa abstractă de bază *Instituţie de învăţământ* cu funcţia virtuală - descriere. Creaţi clasele derivate *instituţie de învăţământ preşcolară*, *instituţie de învăţământ medie* şi *instituţie de învăţământ superioară* în care funcţia dată este predefinită. În funcţia *main* determinaţi masivul de pointeri la clasa abstractă, cărora li se atribuie adresele diferitor obiecte.

Varianta 11

Creaţi clasa abstractă de bază *dirijarea* cu funcţiile virtuale –dirijarea rădăcinilor. Creaţi clasele derivate dirijare liniară şi dirijare pătratică, în care funcţia dată este predefinită.

Varianta 12

Creaţi clasa abstractă de bază *Figure* cu funcţiile virtuale - volum. Creaţi clasele derivate paralelipiped, piramidă, tetraedru, sferă în care funcţia dată este predefinită. În funcţia *main* determinaţi masivul de pointeri la clasa abstractă, cărora li se atribuie adresele diferitor obiecte.
Volumul paralelipipedului - $V=xyz$ (*x,y,z – стороны).*
Volumul piramidei: $V=xyh$ (*x,y,* - стороны, *h - высота).*
Volumul tetraedrului: $V= a^{3\sqrt{}}2/12$.

Volumul sferei: $V=4\pi r^3/3$.