

LUCRAREA DE LABORATOR №4

Tema: Polimorfism. Funcții virtuale. Clase abstracte

Scopul lucrării:

- Studierea polimorfismului;
- Studierea principiilor legăturii întârziate;
- Studierea funcțiilor virtuale;
- Polimorfismul *ad-hoc*;
- Realizarea funcțiilor virtuale;
- Studierea claselor abstracte.

Noțiuni de bază

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

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

În C++ polimorfismul rezultă firesc din :

- Relația "*a fi exemplar*";
- Mecanismul de expediere a mesajelor;
- Moștenire;
- Principiul de substituție.

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

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

Funcțiile polimorfe – reprezintă una din cele mai puternice tehnici de programare obiect orientate. Ele permit concomitent să scrii cod la un nivel înalt de abstractizare și apoi sa-l utilizezi într-o situație concretă. De obicei programatorul execută ajustarea codului cu ajutorul transiterii 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 referințe și nu a pointerilor) i se atribuie o valoare de tipul subclasa, atunci valoarea clasei dinamice se transmite obligatoriu așa, pentru a coincide cu tipul static al variabilelor.

Totuși la utilizarea referințelor sau a pointerilor valoarea salvează tipul său dinamic.

Legături întârziate (*late binding*)

Legătura întârziată reprezintă mecanismul care permite definirea tipului dinamic în timpul executării programului, dar nu în timpul compilării. Un mecanism asemănător sunt descriptorii de fișiere, așa cum fișierele se deschid în timpul executării programului, dar nu în timpul compilării. Acest mecanism este baza polimorfismului, așa cum realizează funcțiile virtuale.

Funcțiile virtuale

Legăturile întârziate rezolvă problema, dar, ea nu are cunoștințe nemijlocite în limbaj. De aceea, pentru referire este necesar de utilizat funcțiile virtuale, care se scriu cu utilizarea cuvântului rezervat *virtual*. Funcțiile virtuale se deosebesc de cele obișnuite numai prin metodele de acces. Dar utilizarea lor are sens numai cu utilizarea referințelor 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 funcțiile globale nu sunt polimorfe, așa cum variabilele se transmit după valoare, ceea ce duce la pierderea caracteristicilor specifice ale clasei derivate, mai mult ca atât, funcția *Say* nu este virtuală. Deci, pentru funcționarea corectă a acestei programe sunt necesare următoarele modificări: funcția *Say* trebuie să fie definită ca virtuală, dar parametrul funcției globale trebuie să fie definit ca o referință sau un pointer.

Polimorfismul *ad-hoc*

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

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

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

Subclasa încearcă să predefinească metoda:

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

Așa cum listele de argumente se deosebesc, atunci a doilea definiție nu se recunoaște 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. Așa erori sunt greu de găsit, deoarece ambele forme de scriere sunt premise, și să speri la o diagnoză de compilator nu trebuie.

Câțiva autori preferă să considere supraîncărcarea, și șabloanele polimorfismului, așa cum des se întîlnesc cu definirea mai multor forme. Totuși trebuie de amintit, că polimorfismul se realizează cu legături târzii, în acel timp când problemele care apar la utilizarea supraîncărcării și

a șabloanelor ce se rezolvă la compilare. De aceea aceste construcții 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 moștenitor. Pentru metodele amânate, totuși, comportamentul pur și 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 conține pur metodele virtuale, de aceea aceste clase sunt numite abstracte. Subclasele trebuie să redefinească aceste metode. Redefinirea numai a metodelor virtuale trebuie să derive din descrierea ei în urmași, pentru care sunt create obiecte reale.

Întrebări de control:

1. Ce fel de mecanisme corespund cu definirea polimorfismului?
2. Și funcțiile obișnuite și cele virtuale pot fi predefinite. Prin ce se deosebesc ele?
3. Cum sunt fizic realizate funcțiile virtuale?
4. Ce reprezintă legătura cu întârziere? Dați exemple.
5. Care sunt condițiile de realizare a polimorfismului?
6. Care variabile sunt polimorfe?
7. Ce reprezintă polimorfismul *ad-hoc*?
8. Prin ce se deosebesc clasele abstracte de cele obișnuite?
9. Cum se definește funcția pur virtuală?

Sarcina

Varianta 1

Creați clasa abstractă de bază *Worker* cu funcția virtuală calcularea salariului. Creați clasele derivate *StateWorker*, *HourlyWorker* și *CommissionWorker*, în care funcția dată este redefinită. În funcția *main* determinați masivul de pointeri la clasa abstractă, căreia i se atribuie adresele obiectelor claselor derivate.

Varianta 2

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

Varianta 3

Creați clasa abstractă de bază *progresia* cu funcția virtuală – suma progresiei. Creați clasele derivate: progresia aritmetică și progresia geometrică. Fiecare clasă conține două câmpuri de tip *double*. Primul – primul membru al progresiei, al doilea (*double*) – restul permanent (pentru aritmetică) și relațiile permanente (pentru geometrică). Definiți funcția de calculare a sumei, unde ca parametru este cantitatea de elemente a progresiei.

Progresia aritmetică $a_j=a_0+jd, j=0,1,2,\dots$

Suma progresiei aritmetice: $s_n=(n+1)(a_0+a_n)/2$

Progresia geometrică: $a_j=a_0r^j, j=0,1,2,\dots$

Suma progresiei geometrice: $s_n=(a_0-a_nr)/(1-r)$

Varianta 4

Creați clasa abstractă de bază *Mammal* – mamifere cu funcția virtuală - descrierea. Creați clasele derivate *Animal* și *Human*. Pentru clasa *Animal* determinați clasele derivate *Dog* – câinele și *Cow* – vaca, în care funcția se predefinește.

Varianta 5

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

Varianta 6

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

Varianta 7

Creați clasa abstractă de bază *Container* cu funcțiile virtuale de inserare și extragere. Creați clasele derivate *Stack* și *Queue*, în care funcțiile date sunt predefinite. În funcția *main*

determinați masivul de pointeri la clasa abstractă, cărora li se atribuie adresele obiectelor claselor derivate.

Varianta 8

Creați clasa abstractă de bază *Figure* cu funcția virtuală – aria suprafeței. Creați clasele derivate paralelipiped, tetraedru, sferă în care funcția dată este predefinită.

Aria suprafeței paralelipipedului: $S=6xy$.

Aria suprafeței sferei: $S=4\pi r^2$.

Aria suprafeței tetraedrului: $S=a^{2\sqrt{3}}$

Varianta 9

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

Varianta 10

Creați clasa abstractă de bază *Instituție de învățământ* cu funcția virtuală - descriere. Creați clasele derivate *instituție de învățământ preșcolară*, *instituție de învățământ medie* și *instituție de învățământ superioară* în care funcția dată este predefinită. În funcția *main* determinați masivul de pointeri la clasa abstractă, cărora li se atribuie adresele diferitor obiecte.

Varianta 11

Creați clasa abstractă de bază *dirijarea* cu funcțiile virtuale –dirijarea rădăcinilor. Creați clasele derivate *dirijare liniară* și *dirijare pătratică*, în care funcția dată este predefinită.

Varianta 12

Creați clasa abstractă de bază *Figure* cu funcțiile virtuale - volum. Creați clasele derivate paralelipiped, piramidă, tetraedru, sferă în care funcția dată este predefinită. În funcția *main* determinați 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: $V=4\pi r^3/3$.

Varianta 13

Creați clasa abstractă de bază *Worker* cu funcția virtuală calcularea salariului. Creați clasele derivate *StateWorker*, *HourlyWorker* și *CommissionWorker*, în care funcția dată este redefinită. În funcția *main* determinați masivul de pointeri la clasa abstractă, cărora li se atribuie adresele obiectelor claselor derivate.

Varianta 14

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

Varianta 15

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

Varianta 16

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

Varianta 17

Creați clasa abstractă de bază *Figure* cu funcția virtuală – aria suprafeței. Creați clasele derivate paralelipiped, tetraedru, sferă în care funcția dată este predefinită.

Aria suprafeței paralelipipedului: $S=6xy$.

Aria suprafeței sferei: $S=4\pi r^2$.

Aria suprafeței tetraedrului: $S=a^{2\sqrt{3}}$.

Varianta 18

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

Varianta 19

Creați clasa abstractă de bază *Figure* cu funcția virtuală - aria. Creați clasele derivate *Square*, *Circle*, *Triangle*, *Rectangle* în care funcția dată este predefinită. În funcția *main* determinați masivul de pointeri la clasa abstractă, căreia i se atribuie adresele obiectelor claselor derivate.

Varianta 20

Creați clasa abstractă de bază *Mammal* – mamifere cu funcția virtuală - descrierea. Creați clasele derivate *Animal* și *Human*. Pentru clasa *Animal* determinați clasele derivate *Dog* – câinele și *Cow* – vaca, în care funcția se predefinește.

Varianta 21

Creați clasa abstractă de bază *Instituție de învățământ* cu funcția virtuală - descriere. Creați clasele derivate *instituție de învățământ preșcolară*, *instituție de învățământ medie* și *instituție de învățământ superioară* în care funcția dată este predefinită. În funcția *main* determinați masivul de pointeri la clasa abstractă, cărora li se atribuie adresele diferitor obiecte.

Varianta 22

Creați clasa abstractă de bază *progresia* cu funcția virtuală – suma progresiei. Creați clasele derivate: *progresia aritmetică* și *progresia geometrică*. Fiecare clasă conține două câmpuri de tip *double*. Primul – primul membrul al progresiei, al doilea (*double*) – restul permanent (pentru aritmetică) și relațiile permanente (pentru geometrică). Definiți funcția de calculare a sumei, unde ca parametru este cantitatea de elemente a progresiei.

Progresia aritmetică $a_j = a_0 + jd, j=0,1,2,\dots$

Suma progresiei aritmetice: $s_n = (n+1)(a_0 + a_n)/2$

Progresia geometrică: $a_j = a_0 r^j, j=0,1,2,\dots$

Suma progresiei geometrice: $s_n = (a_0 - a_n r)/(1-r)$