**Ministerul Educației al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**Departamentul Ingineria Software și Automatică**

**Raport**

Lucrarea de laborator nr.6

Disciplina Programarea orietntată pe obiecte:

Tema: Polimorfism. Funcţii virtuale

Varianta 5

**Efectuat**: st.gr. TI-207 Buenscu Gabriel

**Verificat lect.univ.Lisnic Inga**.

Chișinău 2021

**Scopul lucrării:**

1. Studierea polimorfismului;
2. Studierea principiilor legăturii târzii;
3. Studierea funcțiilor virtuale;
4. Polimorfismul **ad-hoc**;
5. Realizarea funcțiilor virtuale;
6. Studierea claselor abstracte;

**Sarcina:**

**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

Noțiuni de bază:

Cuvântul polimorfism provine şi aproximativ se traduce ca „*multe forme*” (*poly* – multe, *morphos* - formă). Cuvântul *morphos* are legătură cu zeul grec Morphus, care putea să apară în vis oamenilor în orişice formă în care el doreşte.

În viaţă tipurile polimorfe – sunt acelea, care se caracterizează după o cantitate de diferite forme şi 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, dar apoi este inginer (principiul de substituire).

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

În С++ polimorfismul este o urmările obişnuită:

1. Relaţia *"a fi exemplar"*;
2. Mecanismul de expediere a mesajelor;
3. Moştenirea;
4. Principiul de substituire.

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

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);

}

**Mersul lucrării:**

**Listingul programului:**

#include <iostream>

#include <cmath>

using namespace std;

class Lines {

public:

Lines() {

}

public:

double a,b,x,y;

virtual bool fx(){return 0;};

};

class StraightLine:public Lines{

//y = ax + b;

public:

StraightLine(double x,double y,double a,double b) {

this->a = a;

this->b = b;

this->x = x;

this->y = y;

}

bool fx() {

if(y == a \* x + b) {

cout << "\n este linie";

return true;

} else {

cout << "\n nu este linie";

return false;

}

}

};

class Ellipse:public Lines{

//x^2/a^2 + y^2/b^2 = 1

public:

Ellipse(double x,double y,double a,double b) {

this->a = a;

this->b = b;

this->x = x;

this->y = y;

}

bool fx() {

if(pow(x,2)/pow(a,2) + pow(y,2)/pow(b,2) == 1) {

cout << "\n este elipsa";

return true;

} else {

cout << "\n nu este elipsa";

return false;

}

}

};

class Hyperbola:public Lines{

//x^2/a^2 - y^2/b^2 = 1

public:

Hyperbola(double x,double y,double a,double b) {

this->a = a;

this->b = b;

this->x = x;

this->y = y;

}

bool fx() {

if((pow(x,2)/pow(a,2) - pow(y,2)/pow(b,2)) == 1) {

cout << "\n este hiperbola";

return true;

} else {

cout << "\n nu este hiperbola";

return false;

}

}

};

int main() {

Lines \*\*a;

for(int i = 0; i < 3; i++) {

a[i] = new Lines();

}

StraightLine dreapta(1,2,1,1);

Ellipse elips(1,2,3,4);

Hyperbola hiperbola(1,1,1,1);

a[0] = &dreapta;

a[1] = &elips;

a[2] = &hiperbola;

a[0]->fx();

a[1]->fx();

a[2]->fx();

}

****

**Concluzie:**

Elaborând această lucrare de laborator eu am obținut cunoștințe despre moștenire multiplă, și că aceasta reprezintă prin sine moștenirea de la două sau mai multe clase, am studiat unde se utilizează, care sunt avantajele acestei moșteniri, care este diferență între moștenirea multiplă și cea simplă.