Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

ЗВІТ

про виконання комп’ютерного практикуму № 3

з дисципліни

“Об’єктно-орієнтоване програмування”

Тема: ПОЛІМОРФІЗМ C++

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Головченко М.М. |  | студент 2-го курсу  гр. ІП-52 ФІОТ  Набоков Е.М. |

Київ 2016

ЗМІСТ

[ЗМІСТ 2](#_Toc463982426)

[1 ПОСТАНОВКА ЗАДАЧІ 3](#_Toc463982427)

[2 ТЕОРЕТИЧНІ ВІДОМОСТІ 4](#_Toc463982428)

[3 АЛГОРИТМ 5](#_Toc463982429)

[3.1 Знаходження площі квадрату 5](#_Toc463982430)

[3.2 Знаходження периметру квадрату 5](#_Toc463982431)

[3.3 Знаходження площі прямокутника 5](#_Toc463982432)

[3.4 Знаходження периметру прямокутника 5](#_Toc463982433)

[3.5 Знаходження площі трикутника 5](#_Toc463982434)

[3.6 Знаходження периметру прямокутника 6](#_Toc463982435)

[4 ДІАГРАМА КЛАСІВ 7](#_Toc463982436)

[5 ВИХІДНИЙ КОД 8](#_Toc463982437)

[6 ВІДЕОКОПІЯ РЕЗУЛЬТАТУ 12](#_Toc463982438)

[7 ВИСНОВОК 13](#_Toc463982439)

1 ПОСТАНОВКА ЗАДАЧІ

Спроектировать иерархию классов: класс функция от одной переменной и его наследники: синус и его производная, косинус и его производная. Определить в базовом классе и переопределить в наследниках методы вычисления значения функции для заданного значения переменной . Элементы-данные фигур объявляются в базовом классе, а инициализируются в наследниках (элементы данные: значение переменной). Определяемые в базовых классах методы должны быть чисто виртуальными.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ

Производная sin(x) = cos(x)

Производная cos(x) = - sin(x)

4 ДІАГРАМА КЛАСІВ

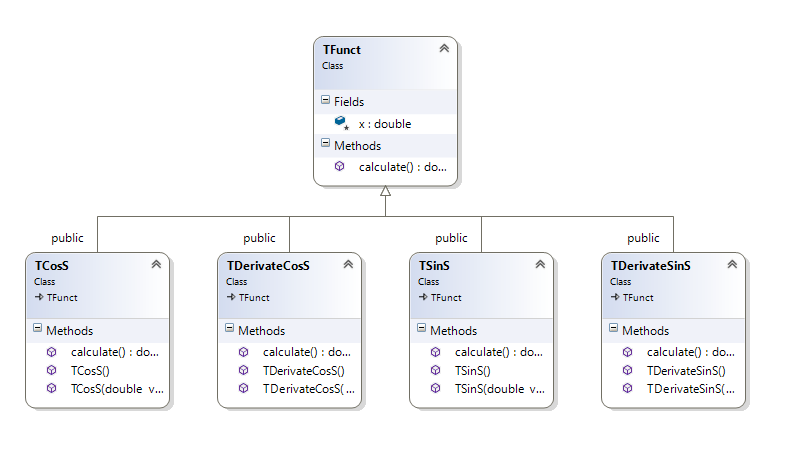


Рисунок 4.1 — Діаграма класів програми

5 ВИХІДНИЙ КОД

“main.cpp”

#include <iostream>

#include <cmath>

#include "TSinS.h"

#include "TCosS.h"

#include "TDerivateSinS.h"

#include "TDerivateCosS.h"

using namespace std;

void header();

void menu();

void checkChoice(int &, int min, int max);

int main()

{

system("color F0");

TFunct \*a, \*b, \*c, \*d;

header();

int choice;

double x;

do {

menu();

checkChoice(choice, 1, 6);

switch (choice)

{

case 1:

{ TSinS sines(1.57); a = &sines; cout << "sin(1.57) = " << a->calculate() << endl; break;}

case 2:

{TCosS cosines(3.14); b = &cosines; cout << "cos(3.14) = " << b->calculate() << endl; break;}

case 3: {TDerivateSinS derSin(0); c = &derSin; cout << "sin'(0) = " << c->calculate() << endl; break;}

case 4: {TDerivateCosS derCos(5); c = &derCos; cout << "cos'(5) = " << c->calculate() << endl; break;}

case 5:

{

cout << "[1] -> Sin()\n[2] -> Cos()\n[3] -> Sin'()\n[4] -> Cos'()\n";

checkChoice(choice, 1, 4);

cout << "Enter x: "; cin >> x;

switch (choice)

{

case 1: { TSinS sines1(x); a = &sines1; cout << "sin(" << x << ") = " << a->calculate() << "\n"; break;}

case 2: {TCosS cosines1(x); b = &cosines1; cout << "cos(" << x << ") = " << b->calculate() << "\n"; break;}

case 3: {TDerivateSinS derSin1(x); c = &derSin1; cout << "sin'(" << x << ") = " << c->calculate() << "\n"; break;}

case 4: {TDerivateCosS derCos1(x); d = &derCos1; cout << "cos'(" << x << ") = " << d->calculate() << "\n"; break;}

}

break;

}

case 6: exit(0);

}

system("pause");

} while (true);

}

void header()

{

cout << "Eduard Nabokov\nIP-52\n\"Polymorphizm\"" << endl;

}

void menu()

{

cout << "[1] -> Test 1 (Function: sin(x); x = 1.57)\n[2] -> Test 2 (Function cos(x); x = 3.14)\n[3] -> Test 3 (Function sin'(x); x = 0)\n[4] -> Test 4 (Function cos'(x); x = 5)\n[5] -> Customer\n[6] -> Exit\n";

}

void checkChoice(int &ch, int min, int max)

{

bool flag = 1;

do

{

if (!flag)

{

cout << "\nTry again!\n";

}

cout << "Enter choice (1 - " << max << ")";

cin >> ch;

flag = 1;

} while (ch > max || ch < min);

}

“TFunct.h”

#pragma once

#include <cmath>

#include <iostream>

class TFunct {

protected:

double x;

public:

virtual double calculate() = 0;

};

“TSinS.h”

#pragma once

#include "TFunct.h"

class TSinS : public TFunct {

public:

TSinS();

TSinS(double var);

double calculate();

};

“TSinS.cpp”

#include "TSinS.h"

TSinS::TSinS() {}

TSinS::TSinS(double var)

{

x = var;

}

double TSinS::calculate() {

return sin(x);

}

“TCosS.h”

#pragma once

#include "TFunct.h"

class TCosS : public TFunct {

public:

TCosS();

TCosS(double var);

double calculate();

};

“TCosS.cpp”

#include "TCosS.h"

TCosS::TCosS() {}

TCosS::TCosS(double var)

{

x = var;

}

double TCosS::calculate()

{

return cos(x);

}

“TDerivateSinS.h”

#pragma once

#include "TFunct.h"

class TDerivateSinS : public TFunct {

public:

TDerivateSinS();

TDerivateSinS(double var);

double calculate();

};

“TDerivateSinS.cpp”

#include "TDerivateSinS.h"

TDerivateSinS::TDerivateSinS() {}

TDerivateSinS::TDerivateSinS(double var)

{

x = var;

}

double TDerivateSinS::calculate()

{

return cos(x);}

“TDerivateCosS.h”

#pragma once

#include "TFunct.h"

class TDerivateCosS : public TFunct {

public:

TDerivateCosS();

TDerivateCosS(double var);

double calculate();};

“TDerivateCosS.cpp”

#include "TDerivateCosS.h"

TDerivateCosS::TDerivateCosS() {}

TDerivateCosS::TDerivateCosS(double var) {

x = var;

}

double TDerivateCosS::calculate() {

return -sin(x);}

6 ВІДЕОКОПІЯ РЕЗУЛЬТАТУ

На рисунку 6.1 показано результат виконання програми.

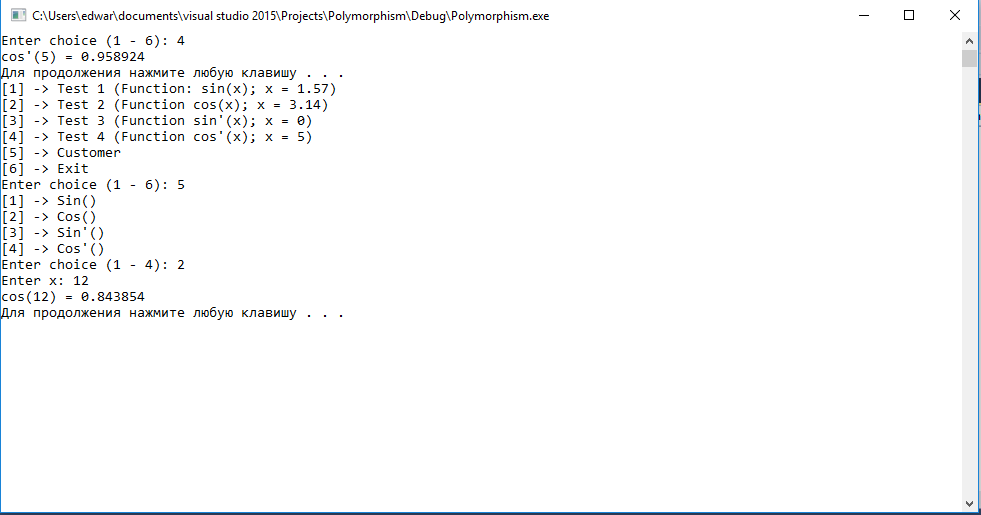
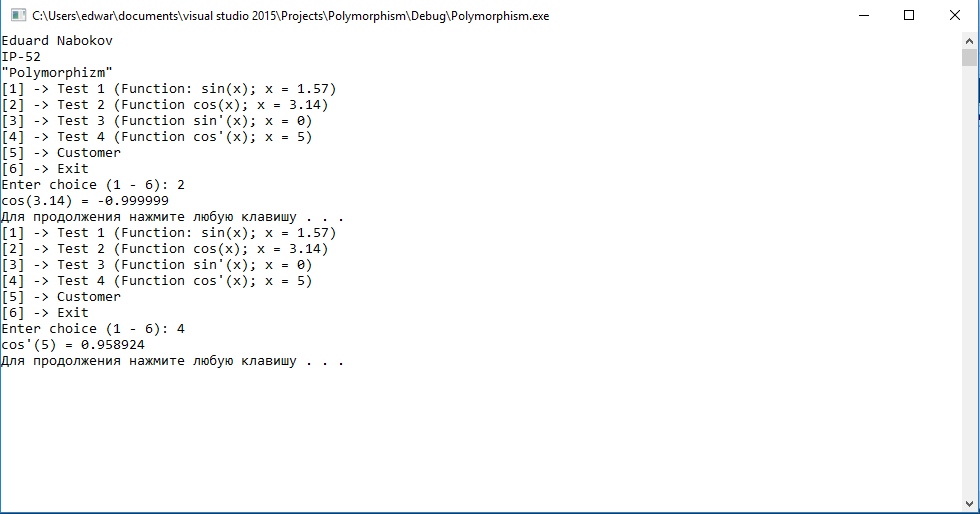
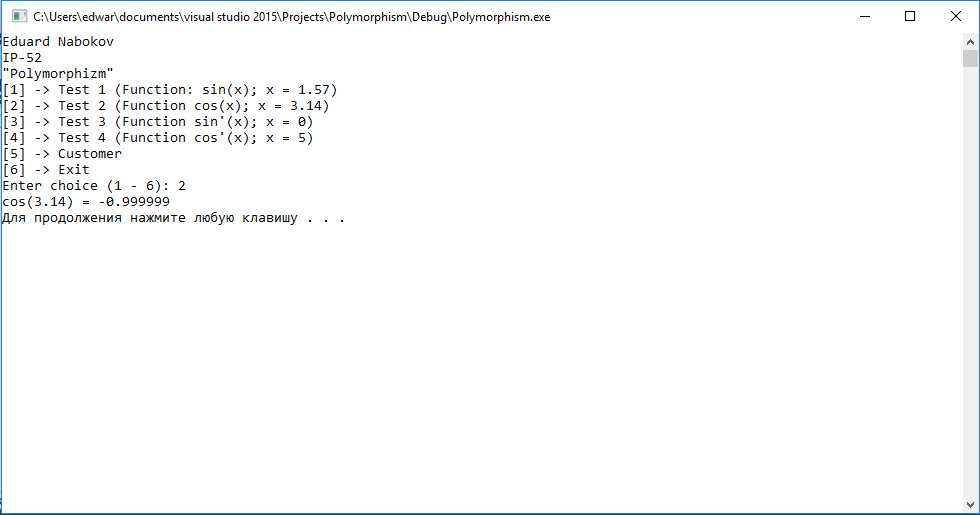


Рисунок 6.1 – Відеокопія результату виконання програми

ВИСНОВОК

У рамках цієї лабораторної роботи я зіштовхнувся з задачею використання наслідування абстрактних класів для реалізації виведення інформації про тригонометричні функції. За допомогою наслідування абстрактного класу, до нащадків можна звертатися як до об’єктів базового класу, що може бути корисним, наприклад, при об’єднанні об’єктів классів – нащадків у масив.