Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра АСОІУ

**ЗВІТ**

про виконання комп’ютерного практикуму № 3

з дисципліни

“ООП”

Тема: «Поліморфізм С++»

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Головченко Максим Миколайович |  | студент 2-го курсу  гр. ІП-51 ФІОТ  Зарічковий Олександр Анатолійович |

Київ – 2016

**ЗМІСТ:**

[1 Мета роботи 3](#_Toc463896091)

[2 Постановка задачі (варіант 7, рівень Б) 4](#_Toc463896092)

[3 Діаграма класів 5](#_Toc463896093)

[4 Код програми 6](#_Toc463896094)

[5 Приклади виконання програми 10](#_Toc463896095)

[6 Висновок 11](#_Toc463896096)

# Мета роботи

Мета роботи - вивчити основні концепції об'єктно-орієнтованого програмування. Вивчити особливості віртуальних функцій, абстрактних класів і поліморфізму.

# Постановка задачі (варіант 7, рівень Б)

Спроектувати ієрархію класів: клас геометричні фігури і його спадкоємці: ромб, квадрат, паралелограм. Визначити в базовому класі і перевизначити в спадкоємців методи обчислення площі та периметра фігури. Елементи-дані фігур оголошуються в базовому класі, а ініціалізуються в спадкоємців (елементи дані: сторони фігур).

Виконати завдання рівня А, з умовою, роботи з спадкоємцями через об'єкт базового абстрактного класу. Визначені в базових класах методи повинні бути чисто віртуальними.

# Діаграма класів

Діаграма класів наведена на рисунку 4.1:

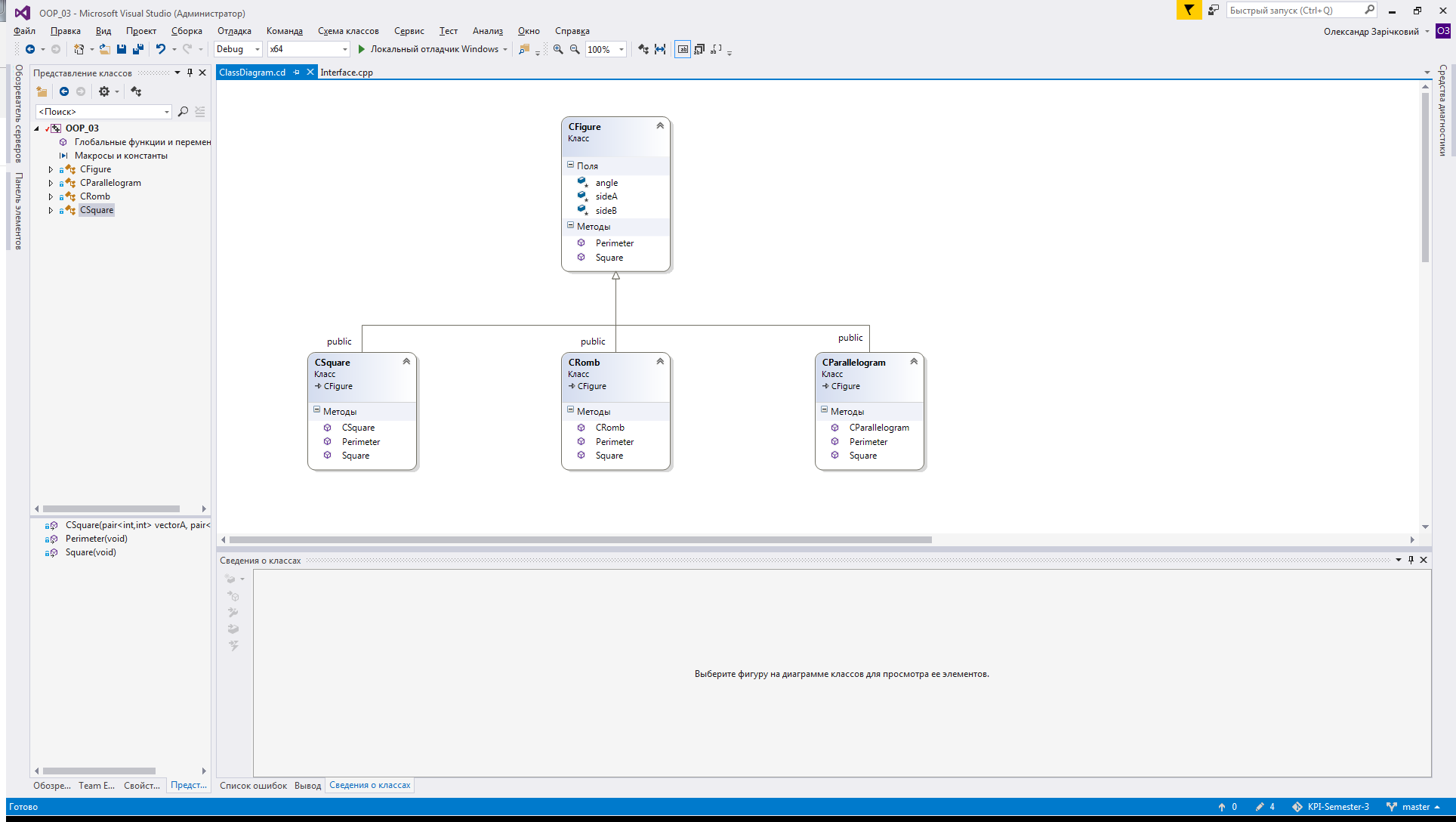


Рисунок 4.1 – Діаграма класів

# Код програми

“main.cpp”

#include "Interface.h"

int main(void) {

while (outputMenu());

}

“Interface.h”

#pragma once

#include <utility>

using std::pair;

bool outputMenu(void); // ћеню

pair <int, int> inputVector(char\* str); // Введення вектора

“Interface.cpp”

#include "Interface.h"

#include "CFigure.h"

#include "CParallelogram.h"

#include "CSquare.h"

#include "CRomb.h"

#include <cstdio>

#include <cstdlib>

// Виведення меню

bool outputMenu(void) {

system("cls");

printf("OOP lab work #3\nVariant - 7, Level - C\nTask: Define figures using polymorphism\nWriter: Alexander ALEXKIRNAS Zarichkovyi\n\n");

printf("Menu:\n1. Start program\n2. Exit\n\nMake your choice ");

int t;

scanf("%d", &t);

switch (t) {

case 1: {

pair <int, int> vectorA = inputVector("first");

printf("\n");

pair <int, int> vectorB = inputVector("second");

printf("\n");

CFigure\* objects[3] = {

new CRomb(vectorA, vectorB),

new CSquare(vectorA, vectorB),

new CParallelogram(vectorA, vectorB)

};

printf("Squares:\n");

for (int i = 0; i < 3; i++) printf("%d - %.3Lf\n", i + 1, objects[i]->Square());

printf("\n\nPerimeter:\n");

for (int i = 0; i < 3; i++) printf("%d - %.3Lf\n", i + 1, objects[i]->Perimeter());

printf("\n\nWhere:\n1 - Romb\n2 - Square\n3 - Parellelogram\n\n");

system("pause");

break;

}

case 2: return false;

}

return true;

}

// Введення вектора

pair <int, int> inputVector(char\* str) {

int x1, x2, y1, y2;

printf("Input %s vector:\n", str);

printf("X1: "); scanf("%d", &x1);

printf("Y1: "); scanf("%d", &y1);

printf("X2: "); scanf("%d", &x2);

printf("Y2: "); scanf("%d", &y2);

return std::make\_pair(x2 - x1, y2 - y1);

}

“CFigure.h”

#pragma once

#include <cmath>

#include <utility>

#define M\_PI 3.14159265358979323846 // pi

using std::pair;

// Абстрактний базовий клас

class CFigure {

protected:

double sideA, sideB; // Довжина сторін

double angle; // Кут між сторонами

public:

virtual double Square (void) = 0;

virtual double Perimeter (void) = 0;

};

“CRomb.h”

#pragma once

#include "CFigure.h"

// Клас ромба

class CRomb : public CFigure {

public:

CRomb(pair <int, int> vectorA, pair <int, int> vectorB);

double Square(void);

double Perimeter(void);

};

“CRomb.cpp”

#include "CRomb.h"

CRomb::CRomb(pair <int, int> vectorA, pair <int, int> vectorB) {

sideA = sqrt(static\_cast<double> (vectorA.first \* vectorA.first + vectorA.second \* vectorA.second));

sideB = sqrt(static\_cast<double> (vectorB.first \* vectorB.first + vectorB.second \* vectorB.second));

angle = acos((vectorA.first \* vectorB.first + vectorA.second \* vectorB.second) / (sideA \* sideB));

}

double CRomb::Square(void) {

if (sideA != sideB) return 0;

else return sideA \* sideA \* sin(angle);

}

double CRomb::Perimeter(void) {

if (sideA != sideB) return 0;

else return 4 \* sideA;

}

“CSquare.h”

#pragma once

#include "CFigure.h"

// Клас прямокутника

class CSquare : public CFigure {

public:

CSquare (pair <int, int> vectorA, pair <int, int> vectorB);

double Square(void);

double Perimeter(void);

};

“CSquare.cpp”

#include "CSquare.h"

CSquare::CSquare(pair <int, int> vectorA, pair <int, int> vectorB) {

sideA = sqrt(static\_cast<double> (vectorA.first \* vectorA.first + vectorA.second \* vectorA.second));

sideB = sqrt(static\_cast<double> (vectorB.first \* vectorB.first + vectorB.second \* vectorB.second));

angle = acos((vectorA.first \* vectorB.first + vectorA.second \* vectorB.second) / (sideA \* sideB));

}

double CSquare::Square(void) {

if (sideA != sideB || angle != M\_PI / 2) return 0;

else return sideA \* sideA;

}

double CSquare::Perimeter(void) {

if(sideA != sideB || angle != M\_PI / 2) return 0;

else return 4 \* sideA;

}

“CParsllelogram.h”

#pragma once

#include"CFigure.h"

class CParallelogram : public CFigure {

public:

CParallelogram (pair <int, int> vectorA, pair <int, int> vectorB);

double Square(void);

double Perimeter(void);

};

“CParsllelogram.cpp”

#include "CParallelogram.h"

CParallelogram::CParallelogram(pair <int, int> vectorA, pair <int, int> vectorB) {

sideA = sqrt(static\_cast<double> (vectorA.first \* vectorA.first + vectorA.second \* vectorA.second));

sideB = sqrt(static\_cast<double> (vectorB.first \* vectorB.first + vectorB.second \* vectorB.second));

angle = acos((vectorA.first \* vectorB.first + vectorA.second \* vectorB.second) / (sideA \* sideB));

}

double CParallelogram::Square(void) {

return sideA \* sideB \* sin(angle);

}

double CParallelogram::Perimeter(void) {

return 2 \* (sideA + sideB);

}

# Приклади виконання програми

Приклад виконання програми наведений на рисунку 5.1:

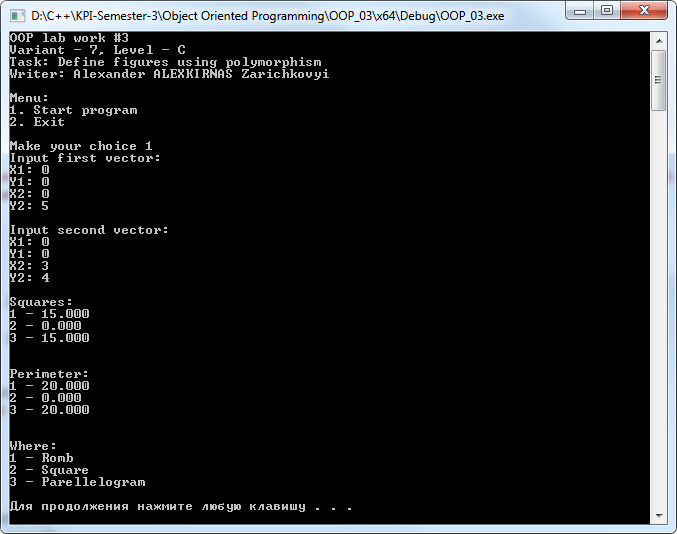


Рисунок 5.1 – Приклад виконання програми

# Висновок

Отже, дана програма створює об’єкт класа CFigure, який є абстрактним класом та від якого породжуються класи CRomb, CSquare, CParallelogram. Для кожного з цього класу викликаються полімормні фунції знаходження площі та переметру. Оскільки дана програма реалізована за модульним принципом і кожен модуль програми створювався й налагоджувався автономно, то дана програма буде видавати очікуваний результат на всіх наборах вхідних даних.