Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 5 з дисципліни
«Основи програмування-2.
Методології програмування.»

«Успадкування та поліморфізм»

Варіант 28

Виконав студент ІП-11 Сідак Кирил Ігорович
(шифр, прізвище, ім'я, по батькові)

Перевірив _____
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота №5

Мета: вивчити механізм створення і використання класів та об’єктів.

Варіант 28

Створити клас TQuadrangle, який містить координати вершин і методи обчислення площі та периметру. На основі цього класу створити класи-нащадки, які представляють рівносторонні, прямокутник, квадрат, паралелограм (квадрат створити на основі прямокутника). Створити певну кількість чотирикутників кожного виду, щоб їх сумарна кількість дорівнювала n. Обчислити суму площ прямокутників та квадратів і суму периметрів паралелограмів.

Постановка задачі: за умовою задачі треба створити клас TQuadrangle (чотирикутник), який міститиме координати вершин як атрибути та для зручності довжини сторін (які вираховуються на основі цих координат). Також потрібно створити класи-нащадки паралелограм (успадковується від класу чотирикутника), прямокутник (успадковується від класу паралелограма), квадрат (успадковується від класу прямокутника). На основі введеного цілого числа n треба створити сумарно n різних чотирикутників та обчислити суму площ прямокутників та квадратів і суму периметрів паралелограмів.

Програма на C++:

main.cpp

```
#include "Square.h"
#include "input_operations.h"

int main() {
    int n, choice;
    double p, area, sum_squares_rectangles,
sum_p_parallelograms;
    vector<int> coordinates;
    sum_squares_rectangles = 0;
    sum_p_parallelograms = 0;
    cout << "Enter the number of quadrangles to be created:
";
    cin >> n;
    for (int i = 0; i < n; ++i) {
        cout << "Enter 1 for parallelogram, 2 for rectangle,
3 for square: ";
        cin >> choice;
        cout << "Enter the coordinates for the figure in such
format (x, y):" << endl;
        coordinates = input_points();
        if (choice == 1) {
            Parallelogram figure(coordinates);
            p =  figure.get_perimeter();
            sum_p_parallelograms += p;
            cout << "The perimeter of the parallelogram is "
```

```
<< p << endl;
        } else if (choice == 2) {
            Rectangle figure(coordinates);
            area = figure.get_area();
            sum_squares_rectangles += area;
            cout << "The area of the rectangle is " << area
<< endl;
        } else {
            Square figure(coordinates);
            area = figure.get_area();
            sum_squares_rectangles += area;
            cout << "The area of the square is " << area <<
endl;
        }
    }
    cout << "The sum of the areas of all the squares and
rectangles is " << sum_squares_rectangles << endl;
    cout << "The sum of the perimeters of all the
parallelograms is " << sum_p_parallelograms << endl;
    return 0;
}
```

input_operations.h

```
#ifndef LAB_5_C___INPUT_OPERATIONS_H
#define LAB_5_C___INPUT_OPERATIONS_H
#include <vector>
#include <iostream>
using namespace std;
vector<int> input_points();
vector<string> split(string, char);
#endif
```

input_operations.cpp

```
#include "input_operations.h"
vector<int> input_points() {
    vector<int> coordinates;
    string coordinates_str;
    int x, y;
    cin.ignore();
    for (int i = 1; i < 5; ++i) {
        cout << "Point " << i << ":" << endl;
        getline(cin, coordinates_str);
        auto input_str = split(coordinates_str, ',');
        x = stoi(input_str[0]);
        y = stoi(input_str[1]);
        coordinates.push_back(x);
        coordinates.push_back(y);
    }
    return coordinates;
}
```

```
vector<string> split(string str, char sep) {
    vector<string> words;
    string temp_str;
    for (int i = 0; i < str.length(); ++i) {
        if (str[i] == sep || i == str.length() - 1) {
            if (str[i] == sep) {
                words.push_back(temp_str);
                temp_str = "";
            } else {
                temp_str += str[i];
                if (!temp_str.empty())
words.push_back(temp_str);
            }
        } else {
            temp_str += str[i];
        }
    }
    return words;
}
```

Point.h

```
#ifndef LAB_5_C___POINT_H
#define LAB_5_C___POINT_H
#include <cmath>
class Point {
    int x, y;
public:
    Point() = default;
    Point(int x, int y) { this -> x = x; this -> y = y; }
    int get_x() const { return x; }
    int get_y() const { return y; }
    friend double get_distance(Point, Point);
};
#endif
```

Point.cpp

```
#include "Point.h"
double get_distance(Point point_1, Point point_2) {
    return sqrt(pow(point_2.x - point_1.x, 2) + pow(point_2.y
- point_1.y, 2));
}
```

TQuadrangle.h

```
#ifndef LAB_5_C___TQUADRANGLE_H
#define LAB_5_C___TQUADRANGLE_H
#include "Point.h"
#include <vector>
using namespace std;
```

```
class TQuadrangle {
protected:
    Point point_1, point_2, point_3, point_4;
    double side_1, side_2, side_3, side_4;
public:
    explicit TQuadrangle(vector <int>);
    virtual double get_perimeter() = 0;
    virtual double get_area() = 0;
};
#endif
```

TQuadrangle.cpp

```
#include "TQuadrangle.h"

TQuadrangle::TQuadrangle(vector<int> coordinates) {
    point_1 = Point(coordinates[0], coordinates[1]);
    point_2 = Point(coordinates[2], coordinates[3]);
    point_3 = Point(coordinates[4], coordinates[5]);
    point_4 = Point(coordinates[6], coordinates[7]);
    side_1 = get_distance(point_1, point_2);
    side_2 = get_distance(point_2, point_3);
    side_3 = get_distance(point_3, point_4);
    side_4 = get_distance(point_1, point_4);
}
```

Parallelogram.h

```
#ifndef LAB_5_C___PARALLELOGRAM_H
#define LAB_5_C___PARALLELOGRAM_H
#include "TQuadrangle.h"
class Parallelogram: public TQuadrangle {
public:
    explicit Parallelogram(vector <int> coordinates):
TQuadrangle(coordinates) {}
    double get_perimeter() override { return 2 * (side_1 +
side_2); }
    double get_area() override;
};
#endif
```

Parallelogram.cpp

```
#include "Parallelogram.h"
double Parallelogram::get_area() {
    auto vector_1 = Point(point_2.get_x() - point_1.get_x(),
point_2.get_y() - point_1.get_y());
    auto vector_2 = Point(point_4.get_x() - point_1.get_x(),
point_4.get_y() - point_1.get_y());
    return abs(vector_1.get_x() * vector_2.get_y() -
vector_1.get_y() * vector_2.get_x());
}
```

Rectangle.h

```
#ifndef LAB_5_C___RECTANGLE_H
#define LAB_5_C___RECTANGLE_H
#include "Parallelogram.h"
class Rectangle: public Parallelogram {
public:
    explicit Rectangle(vector <int> coordinates):
Parallelogram(coordinates) {}
    double get_area() override { return side_1 * side_2; }
};
#endif
```

Square.h

```
#ifndef LAB_5_C___SQUARE_H
#define LAB_5_C___SQUARE_H
#include "Rectangle.h"
class Square: public Rectangle {
public:
    explicit Square(vector <int> coordinates):
Rectangle(coordinates) {};
    double get_perimeter() override { return 4 * side_1; }
    double get_area() override { return pow(side_1, 2); }
};
#endif
```

Програма на Python:

main.py

```
from quadrangles import Parallelogram, Rectangle, Square
from input_operations import input_points

if __name__ == '__main__':
    n = int(input("Enter the number of quadrangles to be
created: "))
    sum_squares_rectangles = 0
    sum_p_parallelograms = 0
    for i in range(n):
        choice = int(input("Enter 1 for parallelogram, 2 for
rectangle, 3 for square: "))
        print(f"Enter the coordinates for the figure in such
format (x, y): ")
        coordinates = input_points()
        if choice == 1:
            figure = Parallelogram(coordinates)
            p = figure.get_perimeter()
            sum_p_parallelograms += p
            print(f"The perimeter of the parallelogram is
```

```
{p}")
        elif choice == 2:
            figure = Rectangle(coordinates)
            area = figure.get_area()
            sum_squares_rectangles += area
            print(f"The area of the rectangle is {area}")
        else:
            figure = Square(coordinates)
            area = figure.get_area()
            sum_squares_rectangles += area
            print(f"The area of the square is {area}")
    print(f"The sum of the areas of all the squares and
rectangles is {sum_squares_rectangles}")
    print(f"The sum of the perimeters of all the
parallelograms is {sum_p_parallelograms}")
```

input_operations.py

```
def input_points():
    coordinates = []
    for i in range(1, 5):
        point_str = input(f"Point {i}: ").split(',')
        x, y = int(point_str[0]), int(point_str[1])
        coordinates.append(x)
        coordinates.append(y)
    return coordinates
```

point.py

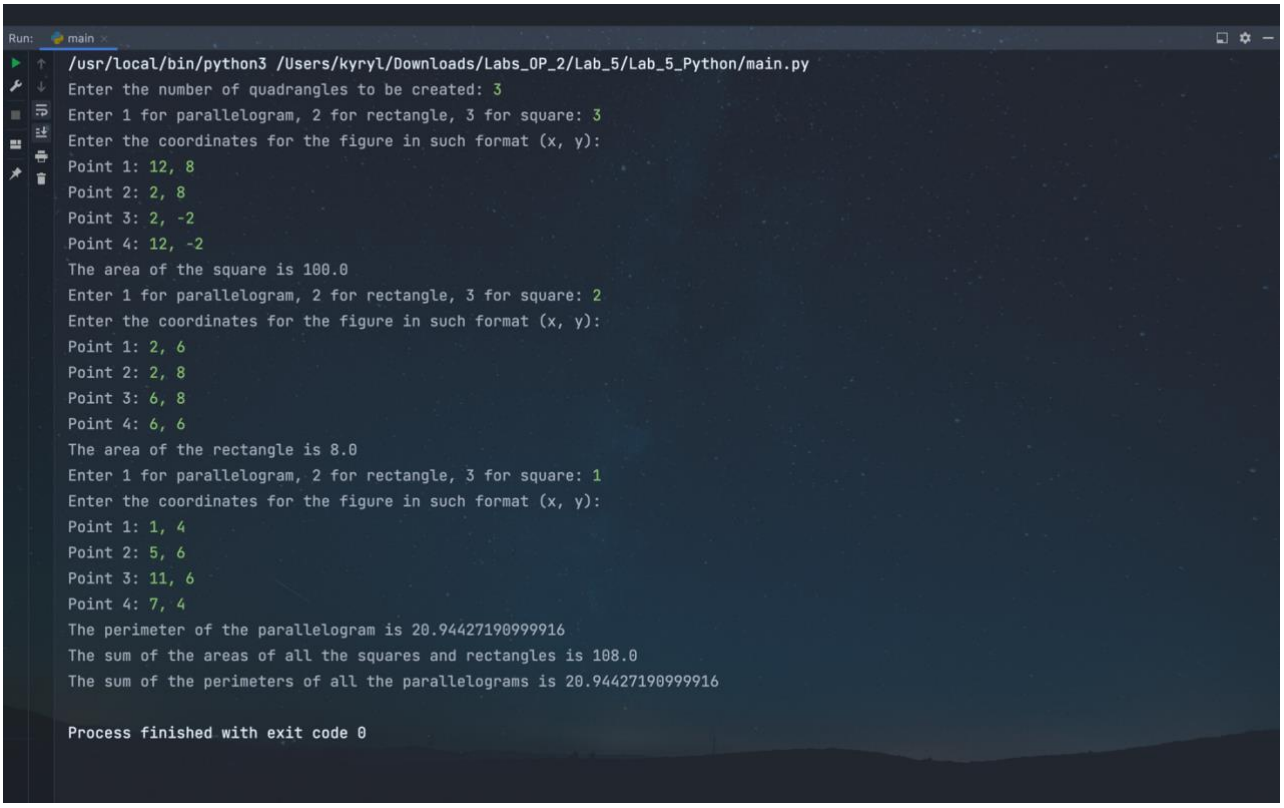
```
import math


class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y


def get_distance(point_1, point_2):
    return math.sqrt((point_2.x - point_1.x) ** 2 +
(point_2.y - point_1.y) ** 2)
```

quadrangles.py

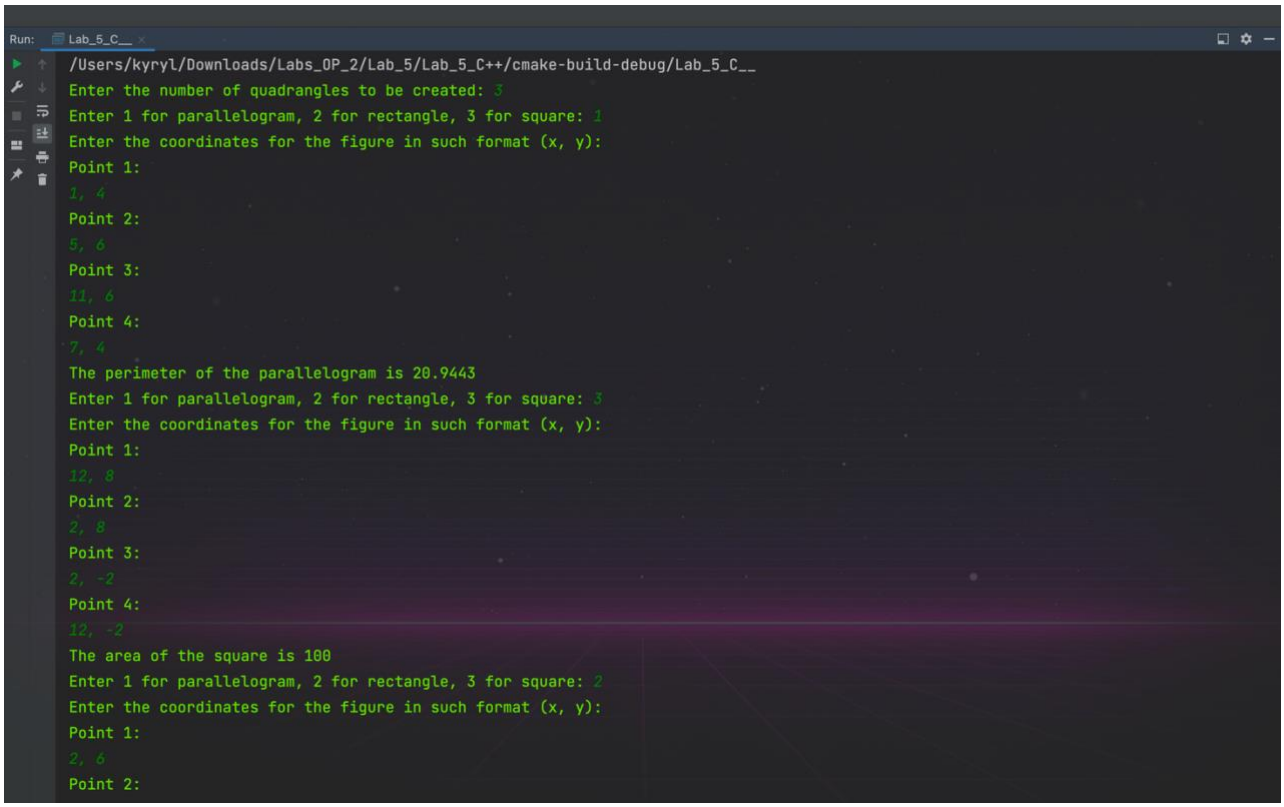
```
from abc import ABC, abstractmethod
from point import Point, get_distance


class TQuadrangle(ABC):
    def __init__(self, coordinates):
        self.point_1 = Point(coordinates[0], coordinates[1])
        self.point_2 = Point(coordinates[2], coordinates[3])
```

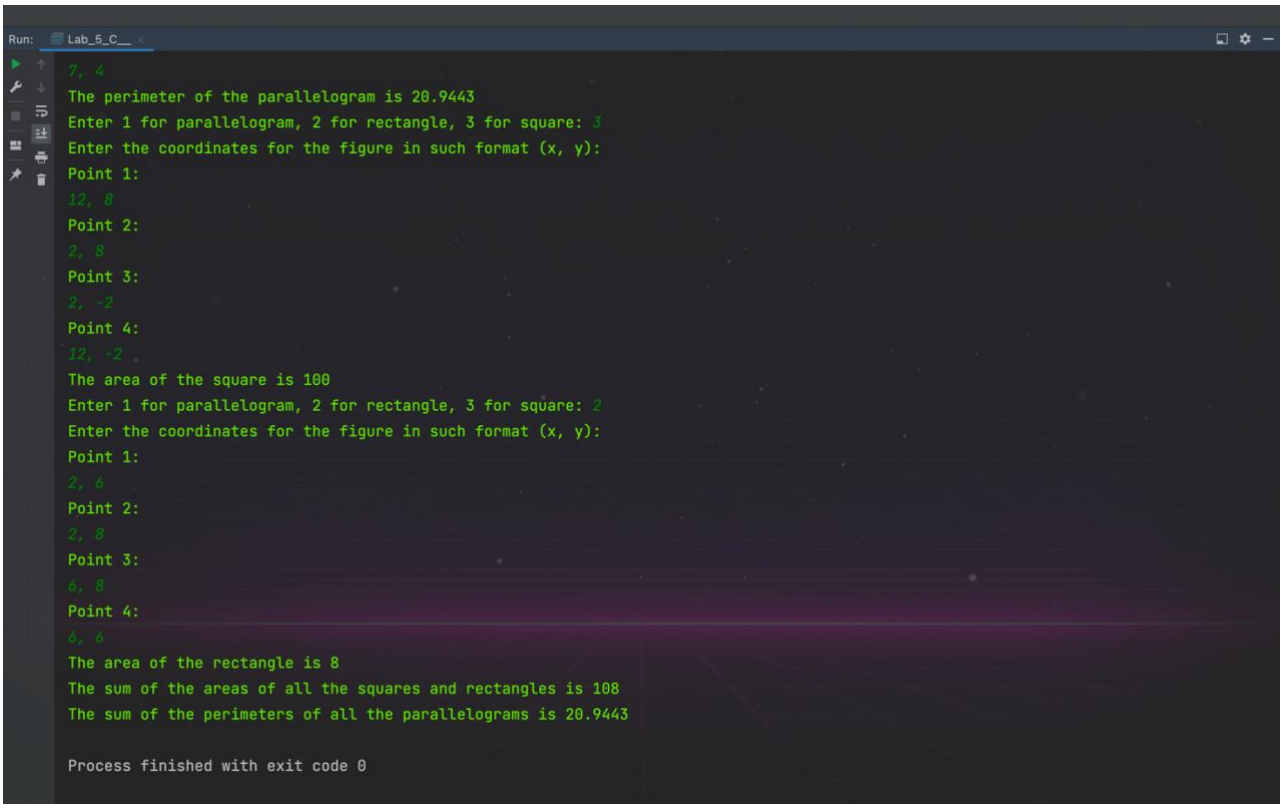
```
        self.point_3 = Point(coordinates[4], coordinates[5])
        self.point_4 = Point(coordinates[6], coordinates[7])
        self.side_1 = get_distance(self.point_1,
self.point_2)
        self.side_2 = get_distance(self.point_2,
self.point_3)
        self.side_3 = get_distance(self.point_3,
self.point_4)
        self.side_4 = get_distance(self.point_1,
self.point_4)

    @abstractmethod
    def get_perimeter(self):
        pass

    @abstractmethod
    def get_area(self):
        pass


class Parallelogram(TQuadrangle):
    def get_perimeter(self):
        return (self.side_1 + self.side_2) * 2

    def get_area(self):
        vector_1 = Point(self.point_2.x - self.point_1.x,
self.point_2.y - self.point_1.y)
        vector_2 = Point(self.point_4.x - self.point_1.x,
self.point_4.y - self.point_1.y)
        return abs(vector_1.x * vector_2.y - vector_1.y *
vector_2.x)


class Rectangle(Parallelogram):
    def get_area(self):
        return self.side_1 * self.side_2


class Square(Rectangle):
    def get_perimeter(self):
        return self.side_1 * 4

    def get_area(self):
        return self.side_1 ** 2
```

Результат на Python:



Результат на C++:





Висновок

Отже, я вивчив механізм створення і використання класів та об’єктів, а також механізм успадкування класів, перевизначення методів, тобто використав поліморфізм. Використавши віртуальні методи у C++ та абстрактні методи у Python, створивши таким чином абстрактний клас чотирикутника з декількома атрибутами (координатами вершин та довжинами сторін) й конструктором з параметрами, та створивши декілька нащадків (паралелограм, прямокутник, нащадок паралелограма, та квадрат, нащадок прямокутника), які реалізували ці методи та викликали конструктор цього базового класу, я отримав коректний результат.