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»

«Успадкування та поліморфізм»

Варіант 28

Виконав студент ІП-13, Петров Ігор Ярославович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вєчерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота № 5
Варіант 28

28. Створити клас TQuadrangle, який містить координати вершин і методи обчислення площі та периметру. На основі цього класу створити класи-нащадки, які представляють 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.

Код C++

main.cpp

```
#include "quadrangle.h"
#include "func.h"

int main() {
    int n;
    cout << "Number of quadrangles: "; cin >> n;
    double s_sum = 0, p_sum = 0;
    vector <TQuadrangle*> quadrangles;
    for (size_t i = 0; i < n; i++)
    {
        int choice = make_choice();
        cout <<"Quadrangle #" << i+1 << "\nEnter coordinates clockwise, starting with the left upper
corner." << endl;
        vector<vector<int>>points = enter_points();
        if (choice == 1) {
            p_sum += Parallelogram(points).get_p();
            quadrangles.push_back(new Parallelogram(points));
        }
        else if (choice == 2) {
            s_sum += Rectangle(points).get_s();
            quadrangles.push_back(new Rectangle(points));
        }
        else {
            s_sum += Square(points).get_s();
            quadrangles.push_back(new Square(points));
        }
    }
    for (size_t i = 0; i < n; i++)
    {
        cout << "\nQuadrangle #" << i + 1 << "\nP: " << quadrangles[i]->get_p() << "\nS: " <<
quadrangles[i]->get_s();
    }
    cout << "\nSum of P of all the parallelograms: " << p_sum << endl;
    cout << "Sum of S of all the squares and rectangles: " << s_sum << endl;
 }
```

quadrangle.cpp

```
#include "quadrangle.h"

TQuadrangle::TQuadrangle(vector<vector <int>> points) {
       point1 = Point(points[0]);
       point2 = Point(points[1]);
       point3 = Point(points[2]);
       point4 = Point(points[3]);
}

Point::Point(vector<int> vec) {
       x = vec[0];
```

```
	y = vec[1];
}

double Parallelogram::get_p() {
	return (side_distance(point1, point2) + side_distance(point2, point3)) * 2;
}

double Parallelogram::get_s() {
	vector<int>vec1;
	vec1.push_back(point2.get_x() - point1.get_x());
	vec1.push_back(point2.get_y() - point1.get_y());
	vector<int>vec2;
	vec2.push_back(point4.get_x() - point1.get_x());
	vec2.push_back(point4.get_y() - point1.get_y());
	return abs(vec1[0] * vec2[1] - vec1[1] * vec2[0]);
}

double Rectangle::get_s() {
	return side_distance(point1, point2) * side_distance(point2, point3);
}

double Square::get_p() {
	return side_distance(point1, point2) * 4;
}

double Square::get_s() {
	return pow(side_distance(point1, point2), 2);
}

double side_distance(Point first_point, Point second_point) {
	return sqrt(pow((second_point.get_x() - first_point.get_x()), 2) + pow((second_point.get_y() -
first_point.get_y()), 2));
}
```

quadrangle.h

```
#pragma once
#include <iostream>
#include <vector>
#include <string>

using namespace std;

class Point {
private:
	double x, y;
public:
	Point() = default;
	Point(vector<int> vec);
	double get_x() { return x; };
	double get_y() { return y; };
};

class TQuadrangle {
protected:
	Point point1, point2, point3, point4;
public:
	TQuadrangle(vector <vector <int>>);
	virtual double get_p() = 0;
	virtual double get_s() = 0;
};

class Parallelogram : public TQuadrangle {
public:
	Parallelogram(vector <vector <int>> points) : TQuadrangle(points) {};
```

```
	double get_p() override;
	double get_s() override;
};

class Rectangle : public Parallelogram {
public:
	Rectangle(vector <vector <int>> points) : Parallelogram(points) {};
	double get_s() override;
};

class Square : public Rectangle {
public:
	Square(vector <vector <int>> points) : Rectangle(points) {};
	double get_p() override;
	double get_s() override;
};

double side_distance(Point, Point);
```

func.h

```
#pragma once
#include <iostream>
#include <vector>
#include <string>

using namespace std;

int make_choice();
vector <vector <int>> enter_points();
vector <int>
split(string,char
sep' ');
```

func.cpp

```
#include "func.h"

int make_choice() {
	int choice;
	cout << "Enter 1 for parallelogram, 2 for rectangle or 3 for square: "; cin >> choice;
	while (choice != 1 and choice != 2 and choice != 3) {
		cout << "Incorrect input. Enter 1 for parallelogram, 2 for rectangle or 3 for square: ";
cin >> choice;
	}
	return choice;
}

vector <vector <int>> enter_points() {
	vector <vector <int>> final_vec;
	string temp;
	cin.ignore();
	for (size_t i = 0; i < 4; i++)
	{
		cout << "Point " << i + 1 << " in format x y: ";
		getline(cin, temp);
		final_vec.push_back(split(temp));
	}
	return final_vec;
}

vector<int> split(string line, char sep) {
	vector<int> res;
	string slice = "";
	line += sep;
	for (int i = 0; i < int(line.length()); i++) {
```

```
            if (line[i] == sep) {
                    if (slice.length() > 0) res.push_back(stoi(slice));
                    slice = "";
            }
            else slice += line[i];
        }
        return res;
}
```

Python

main.py

```python
from quadrangle import Parallelogram, Rectangle, Square
from func import enter_points, make_choice


def main():
    n = int(input("Number of quadrangles: "))
    s_sum = 0
    p_sum = 0
    quadrangles = list()
    for i in range(n):
        choice = make_choice()
        print(f"Quadrangle #{i + 1}")
        print("Enter coordinates clockwise, starting with the left upper corner.")
        points = enter_points()
        if choice == 'p':
            obj = Parallelogram(points)
            p_sum += obj.get_p()
        elif choice == 'r':
            obj = Rectangle(points)
            s_sum += obj.get_s()
        else:
            obj = Square(points)
            s_sum += obj.get_s()
        quadrangles.append(obj)
    for i in range(n):
        print(f"Quadrangle #{i + 1}\nP:{quadrangles[i].get_p()}\nS:{quadrangles[i].get_s()}\n")
    print(f"Sum of P of all the parallelograms: {p_sum}")
    print(f"Sum of S of all the squares and rectangles: {s_sum}")


if __name__ == "__main__":
    main()
```

quadrangle.py

```
from abc import ABC, abstractmethod
import math


class TQuadrangle(ABC):

    def __init__(self, points: list):
        self._point1 = Point(points[0])
        self._point2 = Point(points[1])
        self._point3 = Point(points[2])
        self._point4 = Point(points[3])

    @abstractmethod
    def get_p(self):
        pass

    @abstractmethod
    def get_s(self):
        pass


class Parallelogram(TQuadrangle):

    def get_p(self):
        return (side_distance(self._point1, self._point2) + side_distance(self._point2,
self._point3))*2

    def get_s(self):
        vec1 = [self._point2.get_x() - self._point1.get_x(), self._point2.get_y() -
self._point1.get_y()]
        vec2 = [self._point4.get_x() - self._point1.get_x(), self._point4.get_y() -
self._point1.get_y()]
        return abs(vec1[0] * vec2[1] - vec1[1] * vec2[0])
```

```
class Rectangle(Parallelogram):

    def get_s(self):
        return side_distance(self._point1, self._point2) * side_distance(self._point2, self._point3)


class Square(Rectangle):

    def get_p(self):
        return side_distance(self._point1, self._point2) * 4

    def get_s(self):
        return side_distance(self._point1, self._point2) ** 2

class Point:
    def __init__(self, coordinates: list):
        self.__x = coordinates[0]
        self.__y = coordinates[1]

    def get_x(self):
        return self.__x

    def get_y(self):
        return self.__y


def side_distance(first_point: Point, second_point: Point):
    return math.sqrt((second_point.get_x() - first_point.get_x()) ** 2 + (second_point.get_y()
- first_point.get_y()) ** 2)
```

func.py

```
def enter_points():
    final_list = list()
    for i in range(0, 4):
        point = [int(c) for c in input(f"Point {i + 1} in format x y: ").split()]
```

```
        final_list.append(point)
    return final_list


def make_choice():
    choice = (input("Enter 'p' for parallelogram, 'r' for rectangle or 's' for square: "))
    while choice not in ['p', 'r', 's']:
        choice = (input("Incorrect input. Enter 'p' for parallelogram, 'r' for rectangle or 's'
for square: "))
    return choice
```

Результат роботи