МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСТИТЕТ)

**ЛАБОРАТОРНАЯ РАБОТА №1**

по курсу объектно-ориентированное программирование 3 семестр, 2021/22 уч. Год

Студент Абросимов Алексей Дмитриевич, группа М8О-207Б-20

Преподаватель Дорохов Евгений Павлович

**Условие**

Задание:Вариант 3: Прямоугольник, Ромб, Трапеция.

Необходимо спроектировать и запрограммировать на языке C++ классы трех фигур, согласно варианту задания. Классы должны удовлетворять следующим правилам:  Должны быть названы также, как в вариантах задания и расположенны в раздельных файлах: отдельно заголовки (имя\_класса\_с\_маленькой\_буквы.h), отдельно описания методов (имя\_класса\_с\_маленькой\_буквы.cpp); 

Иметь общий родительский класс Figure; 

Содержать конструктор по умолчанию;  Содержать конструктор, принимающий координаты вершин фигуры из стандартного потока std::cin, расположенных через пробел. Пример: 0.0 0.0 1.0 0.0 1.0 1.0 0.0 1.0  Содержать набор общих методов:

◦ size\_t VertexesNumber() - метод, возвращающий количество вершин фигуры;

◦ double Area() - метод расчета площади фигуры;

◦ void Print(std::ostream& os) - метод печати типа фигуры и ее координат вершин в поток вывода os в формате: Rectangle: (0.0, 0.0) (1.0, 0.0) (1.0, 1.0) (0.0, 1.0)\n

**Описание программы**

Исходный код лежит в 7 файликах:

1. main.cpp

2. rectangle.cpp

3. rectangle.h

4. rhombus.cpp

5. rhombus.h

6. trapezoid.cpp

7. trapezoid.h

**Дневник отладки**

Результат работы программы при тестовых данных: 0 0 1 1 2 1 3 0 0 0 1 0 1 1 0 1 0 0 1 0 1 1 0 1

Trapezoid (0 0)(1 1)(2 1)(3 0)

S=2

Vertex number is 4

Rectangle (0 0)(1 0)(1 1)(0 1)

S=1

Vertex number is 4

Rhombus (0 0)(1 0)(1 1)(0 1)

S=1

Vertex number is 4

**Недочёты**

**Выводы**

Данная лабораторная работа позволила мне ознакомиться с абстрактными классами и наследованием. Я снова убедился в удобности и практичности такого подхода к программированию, как ООП.

Ссылка на гитхаб: https://github.com/yungalexxxey/oop\_labs/tree/main/lab1

**Исходный код**

main.cpp

#include <iostream>

#include <vector>

#include "rectangle.h"

#include "rhombus.h"

#include "trapezoid.h"

//#include "Figure.h"

int main()

{

std::vector<Figure \*> vect;

Figure \*a = new Trapezoid(std::cin);

vect.push\_back(a);

Figure \*b = new Rectangle(std::cin);

vect.push\_back(b);

Figure \*c = new Rhombus(std::cin);

vect.push\_back(c);

for (int i = 0; i < vect.size(); i++)

{

vect[i]->Print(std::cout);

std::cout << "S=" << vect[i]->Area() << std::endl;

std::cout << "Vertex number is " << vect[i]->VertexesNumber() << std::endl;

}

delete (Trapezoid \*)a;

delete (Rectangle \*)b;

delete (Rhombus \*)c;

return 0;

}

rectangle.cpp

#include "rectangle.h"

#include <math.h>

Rectangle::Rectangle(std::istream&is){

std::cout <<"set x1 and y1:";

is >> x1 >> y1;

std::cout <<"set x2 and y2:";

is >> x2 >> y2;

std::cout <<"set x3 and y3:";

is >> x3 >> y3;

std::cout <<"set x4 and y4:";

is >> x4 >> y4;

}

Rectangle::~Rectangle(){

}

void Rectangle::Print(std::ostream&os){

os << "Rectangle " << "(" <<x1<<" "<<y1<<")"<< "(" <<x2<<" "<<y2<<")"<< "(" <<x3<<" "<<y3<<")"<< "(" <<x4<<" " <<y4<<")" <<std::endl;

}

size\_t Rectangle::VertexesNumber(){

return 4;

}

double Rectangle::Area(){

double r1 = sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

double r2 = sqrt((x2 - x3) \* (x2 - x3) + (y2 - y3) \* (y2 - y3));

double r3 = sqrt((x1 - x3) \* (x1 - x3) + (y1 - y3) \* (y1 - y3));

double p=(r1+r2+r3)/2;

double s= 2\*sqrt((p \* (p - r1) \* (p - r2) \* (p - r3)));

return s;

}

rectangle.h

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

#include "figure.h"

#include <iostream>

class Rectangle:public Figure{

public:

Rectangle();

Rectangle(std::istream&is);

void Print(std::ostream&os);

size\_t VertexesNumber();

double Area();

virtual ~Rectangle();

private:

double x1;

double y1;

double x2;

double y2;

double x3;

double y3;

double x4;

double y4;

};

#endif // RECTANGLE\_H

rhombus.cpp

#include "rhombus.h"

#include <math.h>

Rhombus::Rhombus(std::istream&is){

std::cout <<"set x1 and y1:";

is >> x1 >> y1;

std::cout <<"set x2 and y2:";

is >> x2 >> y2;

std::cout <<"set x3 and y3:";

is >> x3 >> y3;

std::cout <<"set x4 and y4:";

is >> x4 >> y4;

}

Rhombus::~Rhombus(){

}

void Rhombus::Print(std::ostream&os){

os << "Rhombus " << "(" <<x1<<" "<<y1<<")"<< "(" <<x2<<" "<<y2<<")"<< "(" <<x3<<" "<<y3<<")"<< "(" <<x4<<" " <<y4<<")" <<std::endl;

}

size\_t Rhombus::VertexesNumber(){

return 4;

}

double Rhombus::Area(){

double d1 = sqrt((x1 - x3) \* (x1 - x3) + (y1 - y3) \* (y1 - y3));

double d2 = sqrt((x2 - x4) \* (x2 - x4) + (y2 - y4) \* (y2 - y4));

double s=d1\*d2/2;

return s;

}

rhombus.h

#ifndef RHOMBUS\_H

#define RHOMBUS\_H

#include "figure.h"

#include <iostream>

class Rhombus:public Figure

{

public:

Rhombus();

Rhombus(std::istream&is);

void Print(std::ostream&os);

size\_t VertexesNumber();

double Area();

~Rhombus();

private:

double x1;

double y1;

double x2;

double y2;

double x3;

double y3;

double x4;

double y4;

};

#endif // RHOMBUS\_H

trapezoid.cpp

#include "trapezoid.h"

#include <math.h>

Trapezoid::Trapezoid(std::istream&is){

std::cout <<"set x1 and y1:";

is >> x1 >> y1;

std::cout <<"set x2 and y2:";

is >> x2 >> y2;

std::cout <<"set x3 and y3:";

is >> x3 >> y3;

std::cout <<"set x4 and y4:";

is >> x4 >> y4;

}

Trapezoid::~Trapezoid(){

}

void Trapezoid::Print(std::ostream&os){

os << "Trapezoid " << "(" <<x1<<" "<<y1<<")"<< "(" <<x2<<" "<<y2<<")"<< "(" <<x3<<" "<<y3<<")"<< "(" <<x4<<" " <<y4<<")" <<std::endl;

}

size\_t Trapezoid::VertexesNumber(){

return 4;

}

double Trapezoid::Area(){

double h=sqrt((y2-y1)\*(y2-y1));

double os1=sqrt((x4-x1)\*(x4-x1)+(y1-y4)\*(y1-y4));

double os2=sqrt((x3-x2)\*(x3-x2)+(y3-y2)\*(y3-y2));

double s=(os1+os2)\*h/2;

return s;

}

trapezoid.h

#ifndef TRAPEZOID\_H

#define TRAPEZOID\_H

#include "figure.h"

#include <iostream>

class Trapezoid:public Figure

{

public:

Trapezoid(std::istream&is);

void Print(std::ostream&os);

size\_t VertexesNumber();

double Area();

virtual ~Trapezoid();

private:

double x1;

double y1;

double x2;

double y2;

double x3;

double y3;

double x4;

double y4;

};

#endif // TRAPEZOID\_H