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

ЛАБОРАТОРНАЯ РАБОТА №1

по курсу “Объектно-ориентированное программирование”
I семестр, 2021/22 учебный год

Студент: *Абросимов Алексей Дмитриевич, группа М8О-207Б-20*
Преподаватель: *Дорохов Евгений Павлович, каф. 806*

Задание:

Спроектировать и запрограммировать на языке C++ классы трёх фигур. Классы должны удовлетворять следующим правилам:

- Должны быть названы как в вариантах задания и расположены в раздельных файлах;

- Иметь общий родительский класс Figure;
- Содержать конструктор, принимающий координаты вершин фигуры из стандартного потока `std::cin`, расположенных через пробел (например: `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)`, с переводом строки в конце).

Вариант №3:

- Фигура 1: Прямоугольник
- Фигура 2: Трапеция
- Фигура 3: Ромб

Описание программы:

Исходный код разделён на 8 файлов:

- `figure.h` – описание класса фигуры
- `rectangle.h` – описание класса прямоугольника (наследуется от фигуры)

- `rectangle.cpp` – реализация класса прямоугольника
- `rhombus.h` – описание класса квадрата (наследуется от прямоугольника)
- `rhombus.cpp` – реализация класса квадрата
- `trapezoid.h` – описание класса трапеции (наследуется от фигуры)
- `trapezoid.cpp` – реализация класса трапеции
- `main.cpp` – основная программа

Дневник отладки:

Вывод:

Выполнение лабораторной работы позволило мне ознакомиться с основами ООП. Я создал несколько классов с наследованием и перегруженными операциями ввода и вывода.

Исходный код:

figure.h:

```
#ifndef FIGURE_H
#define FIGURE_H
#include <iostream>
class Figure {
 public:
    virtual void Print(std::ostream&os)=0;
    virtual double Area()=0;
    virtual size_t VertexesNumber()=0;
   virtual bool isit()=0;
};
#endif // FIGURE_H
```

rectangle.h:

```
#ifndef RECTANGLE_H
#define RECTANGLE_H
#include "figure.h"
#include <iostream>
```

```
class Rectangle:public figure{
public:
   Rectangle();
   Rectangle(std::istream&is);
   bool isit();
   void Print(std::ostream&os);
   size_t VertexesNumber();
   double Area();
   ~Rectangle();

private:
   double x1;
   double y1;
   double x2;
   double y2;
   double x3;
   double y3;
   double x4;
   double y4;
};

#endif // RECTANGLE_H
```

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;
}
void Rectangle::Print(std::ostream&os){
   os << "Rectangle " << "(" <<x1<<" "<<y1<<")"<< "(" <<x2<<" "<<y2<<")"<< "(" <<x3<<" "<<y3<<")"<< "("
<<x4<<" " <<y4<<")" <<std::endl;
}
size_t Rectangle::VertexesNumber(){
   return 4;
}
bool Rectangle::isit(){
double perp;
double perp2;
perp=(x4-x1)*(x2-x1)+(y4-y1)*(y2-y1);
perp2=(x3-x4)*(x3-x2)+(y3-y4)*(y3-y2);
if((perp+perp2)==0) return true;
else return false;

}
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;
}
```

rhombus.h:

```
#ifndef RHOMBUS_H
#define RHOMBUS_H
#include "figure.h"
#include <iostream>

class Rhombus:public figure
{
public:
    Rhombus();
    Rhombus(std::istream&is);
    bool isit();
    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
```

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;
}
void Rhombus::Print(std::ostream&os){
    os << "Rhombus " << "(" <<x1<<" "<<y1<<")"<< "(" <<x2<<" "<<y2<<")"<< "(" <<x3<<" "<<y3<<")"<< "("
<<x4<<" " <<y4<<")" <<std::endl;
}
size_t Rhombus::VertexesNumber(){
```

```
    return 4;
}
bool Rhombus::isit(){

if((sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2))==sqrt((x2-x3)*(x2-x3)+(y2-y3)*(y2-y3)))&&(sqrt((x3-x4)*(x3-x4)+(y3-
y4)*(y3-y4))==sqrt((x1-x4)*(x1-x4)+(y1-y4)*(y1-y4)))) return true;
}
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;
}
```

trapezoid.h:

```
#ifndef TRAPEZOID_H
#define TRAPEZOID_H
#include "figure.h"
#include <iostream>

class Trapezoid:public figure
{
public:
    Trapezoid(std::istream&is);
    bool isit();
    void Print(std::ostream&os);
    size_t VertexesNumber();
    double Area();
    ~Trapezoid();

private:
    double x1;
    double y1;
    double x2;
    double y2;
    double x3;
    double y3;
    double x4;
    double y4;
};

#endif // TRAPEZOID_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;
```

```
}
void Trapezoid::Print(std::ostream&os){
   os << "Trapezoid " << "(" <<x1<<" "<<y1<<")"<< "(" <<x2<<" "<<y2<<")"<< "(" <<x3<<" "<<y3<<")"<< "("
<<x4<<" " <<y4<<")" <<std::endl;
}
size_t Trapezoid::VertexesNumber(){
   return 4;
}
bool Trapezoid::isit(){
double k=(y1-y4)/(x1-x4);
double k1=(y2-y3)/(x2-x3);
if(k==k1) return true;
else return false;
}
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;
}
```

main.cpp:

```
#include <iostream>
#include "rectangle.cpp"
#include "rhombus.cpp"
#include "trapezoid.h"
int main()
{
while(true){
std::cout<<"set Trapezoid coords"<<std::endl;
figure *a=new trapezoid(std::cin);
if(a->isit()){
a->Print(std::cout);
std::cout<<"S="<< a->Area()<<std::endl;
std::cout<< "Vertex number is "<<a->VertexesNumber()<<std::endl;
delete a;
break;
}
else std::cout<<"This is not trapezoid"<<std::endl;
}

while(true){
std::cout<<"set Rectangle coords"<<std::endl;
figure *b=new rectangle(std::cin);
if(b->isit()){
b->Print(std::cout);
std::cout<<"S="<< b->Area()<<std::endl;
std::cout<< "Vertex number is "<<b->VertexesNumber()<<std::endl;
delete b;
break;
}
else std::cout<<"This is not rectangle"<<std::endl;
}
```

```
std::cout<<"set Rhombus coords"<<std::endl;
while(true) {
figure *c=new rhombus(std::cin);
  std::cout<<c->isit()<<std::endl;
if (c->isit()) {
    std::cout<<"S="<< c->Area()<<std::endl;
    std::cout<< "Vertex number is "<<c->VertexesNumber()<<std::endl;
    delete c;
    break;
}
else std::cout<<"This is not rhombus" <<std::endl;;
}

return 0;
}
```