Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа
по курсу «Объектно-ориентированное программирование»
III Семестр

Задание 3
Вариант 27
Наследование, полиморфизм

Студент:	Чурсина Н. А.
Группа:	М8О-208Б-18
Преподаватель:	Журавлев А.А.
Оценка:	
Дата:	

Москва 2019

1. Код программы на языке C++

Main.cpp

```
#include <iostream>
#include "Figures.hpp"
#include <string>



double sum(std::vector<Figure*> v) {
   double res= 0;
   for (auto i = v.begin(); i!= v.end(); ++i){
      res += (*i)->area();
   }
   return res;
}

void remove_elem(int index, std::vector<Figure*>& v){
   v.erase(v.begin() + index);
}

void print(std::vector<Figure*>& v){
   for (auto i = v.begin(); i!= v.end(); ++i) {
      std::cout << (**i) << "center : " << (*i)->center()<< "\narea : " << (*i)->area()<< "\n";
   }
}


int main() {
   std::cout << "\nPlease, enter coordinates of Figure, beginning with down left point\n";
   std:: cout<< "Available input:\n1 - input Trapezoid\n2 - input Rectangle\n3 - input Rhombus\n4 - Print Figures\n5 - Remove
Figure\n6 - Print Sum\n7 - Exit\n";
   std::vector<Figure*> v;

   while (1){
      std::cout << "\n---Enter new command---\n\n";
      int input;
      std::cin >> input;
      int i = 0;
      switch (input) {
         case 1: v.push_back(new Trapezoid());
            std::cin >> (*(v.back()));
            break;
         case 2: v.push_back(new Rectangle());
            std::cin >> (*(v.back()));
            break;
         case 3: v.push_back(new Rhombus());
            std::cin >> (*(v.back()));
            break;
         case 4: print(v);
            break;
```

```
            case 5: std::cout << "Enter index to delete\n";
                std::cin >> i;
                if (i>= v.size()) {
                    std::cout << "incorrect index\n";
                    break;
                }
                remove_elem(i, v);
                break;
            case 6: std::cout<< sum(v) << "\n";
                break;
            case 7: return 0;
        }
    }
    return 0;
}
```

Figures.cpp

```
#include "Figures.hpp"
#include <cmath>
#
//проверить скалярные произведения у прямоугольника , проверить авенство сторон у ромба и у трапеции параррллельность

double Figure::edge_length(const Pair& a, const Pair& b) const {
    return std::sqrt(std::pow(b.first - a.first, 2) + std::pow(b.second - a.second, 2));
}

Pair Figure::edge_center(const Pair& a, const Pair& b) const {
    return Pair((a.first + b.first)/2, (a.second + b.second)/2);
}

double Quadrangle::area() const {
    double sum = this->a[3].first*this->a[0].second - this->a[0].first*this->a[3].second;
    for (int i = 0; i < 3; ++i) sum += this->a[i].first*this->a[i+1].second - this->a[i+1].first*this->a[i].second;
    return std::abs(sum)/2.;
}

std::istream& Quadrangle::input(std::istream& is) {
    return is >> a[0] >> a[1] >> a[2] >> a[3];
}
std::ostream& Quadrangle::output(std::ostream& os) const {
    return os << a[0] << "   " << a[1] << "   " << a[2] << "   " << a[3] << "\n";
}


std::istream& Rectangle::input(std::istream& is) {
    std::cout << "Input Rectangle coordinates:\n";
    return Quadrangle::input(is);
}
std::ostream& Rectangle::output(std::ostream& os) const {
```

```
   if(( (a[1].first - a[0].first) * (a[2].first - a[1].first) + (a[1].second - a[0].second) * (a[2].second - a[1].second)) == 0) {
   os << "Rectangle coordinates:\n";
      return Quadrangle::output(os);
   }else {
      return os << "It is not a Rectangle!  \n ";
      }
}

Pair Rectangle::center() const {
    return edge_center(a[0], a[2]);
}
double Rectangle::area() const {
   return edge_length(a[0], a[1]) * edge_length(a[2], a[3]);
}
//double Multiplication(const Vector &vec1, const Vector &vec2){
//    Vector temp;
//    temp.X = vec1.X * vec2.X;
//    temp.Y = vec1.Y * vec2.Y;
//    return temp.X + temp.Y;
//}
//
//void Rectangle::true_(const ) {
//
//}

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

   std::cout<< "Input Rhombus coordinates:\n";
   return Quadrangle::input(is);

}

std::ostream& Rhombus::output(std::ostream& os) const {
   if ( edge_length(a[0], a[1]) == edge_length(a[1], a[2])== edge_length(a[2], a[3])){
      os << "Rhombus coordinates:\n";
      return  Quadrangle::output(os);
   }else {
      return os << "It is not a Rhombus !  \n ";
   }
}

double Rhombus::area() const {
   return (edge_length(a[0], a[2]) * edge_length(a[1], a[3]))/2;
}

Pair Rhombus::center() const  {
   return edge_center(a[0], a[2]);
}

std::istream& Trapezoid::input(std::istream& is) {
   std::cout<< "Input Trapezoid coordinates:\n";
   return Quadrangle::input(is);
}
```

```
std::ostream& Trapezoid::output(std::ostream& os) const {
   if (( (a[3].second - a[0].second) * (a[2].first - a[1].first) ) == ( (a[2].second - a[1].second) * (a[3].first - a[0].first))) {
   os << "Trapezoid coordinates:\n";
   return Quadrangle::output(os);
   }else{
      return os << "It is not a Trapezoid !  \n ";
   }
}

double Trapezoid::area() const {
   return Quadrangle::area();
}

Pair Trapezoid::center() const {
   double x = 0, y = 0;
   for (int i = 0; i < 4; ++i) {
      x += a[i].first;
      y += a[i].second;
   }
   return Pair(x/4, y/4);
}
```

Figures.hpp

```
#ifndef Figures_hpp
#define Figures_hpp

#include <iostream>
#include <vector>

#endif /* Figures_hpp */



struct Pair {
   double first;
   double second;
   Pair() = default;
   Pair(double a, double b) : first(a), second(b) {}

   friend inline std::ostream& operator<<(std::ostream& os, const Pair& obj){
      return os <<"x : "<< obj.first << " y : "<< obj.second<< '\n';
   }

   friend inline std::istream& operator>>(std::istream& is, Pair& obj){
       return is >> obj.first >> obj.second;
   }
};

class Figure {
public:
```

```
    virtual double area() const = 0;
    virtual Pair center() const = 0;

    friend inline std::ostream& operator <<(std::ostream& os, const Figure& obj) {
        return obj.output(os);
    }

    friend inline std::istream& operator >>(std::istream& is, Figure& obj){
        return obj.input(is);
    }
protected:
    virtual std::istream& input(std::istream& is) = 0;
    virtual std::ostream& output(std::ostream& os) const = 0;

    double edge_length(const Pair& a, const Pair& b) const;
    Pair edge_center(const Pair& a, const Pair& b) const;
};

class Quadrangle: public Figure {
protected:
    Pair a[4];
public:
    double area() const;

    std::istream& input(std::istream& is);
    std::ostream& output(std::ostream& os) const;
};

class Rectangle: public Quadrangle {
public:
    std::istream& input(std::istream& is);
    std::ostream& output(std::ostream& os) const;

    Pair center() const;
    double area() const;
};

class Rhombus: public Quadrangle {
public:
    std::istream& input(std::istream& is);
    std::ostream& output(std::ostream& os) const;

    double area() const;
    Pair center() const;
};

class Trapezoid: public Quadrangle {
public:
    std::istream& input(std::istream& is);
    std::ostream& output(std::ostream& os) const;

    double area() const;
    Pair center() const;
};
```

2. Ссылка на репозиторий на Github

https://github.com/kwk18/oop_exercise_03

3.Набор testcases

1. Входные данные:
1
0 0
2 3
4 3
6 0
4

2. Входные данные:
2
2 2
2 6
5 6
2 5
4

3. Входные данные:
3
-2 3
0 5
2 3
0 1

4

4. Результат выполнения тестов

1.
Trapezoid coordinates:
x : 0 y : 0
x : 2 y : 3
x : 4 y : 3
x : 6 y : 0

center : x : 3 y : 1.5

area : 12

2.
Rectangle coordinates:
x : 2 y : 2
x : 2 y : 6
x : 5 y : 6
x : 2 y : 5

center : x : 3.5 y : 4

area : 12

3.
Rhombus coordinates:
x : -2 y : 3
x : 0 y : 5
x : 2 y : 3
x : 0 y : 1

center : x : 0 y : 3

area : 8

5. Объяснение результатов программы

Наследование - это создание нового объекта на основании старого или создание нового класса на основании старого.

Полиморфизм – это возможность дополнять объект функционалом. Возможность выступать объекту в разных формах. Классический полиморфизм - замещение, переопределение методов.

В программе реализованы:
структура Pair – для координат точек, они представлены в виде двух полей типа double;
класс Figure – с основными методами для фигур;
класс Quadrangle – наследник Figure, общий подкласс четырехугольников;
класс Rhombus – наследник Quadrangle, класс для расчета ромба ;
класс Rectangle – наследник Quadrangle, класс для расчета прямоугольника;
класс Trapezoid – наследник Quadrangle, класс для расчета трапеции ;

Доступные функции:

1.double sum(std::vector<Figure*> v) - считает сумму всех площадей фигур

2. void remove_elem(int index, std::vector<Figure*>& v) – удаляет фигуру

3. void print(std::vector<Figure*>& v) – функция для вывода фигур

Вывод: Проделав данную работу я изучила парадигмы ООП на C++. Улучшила навык работы с GitHub и CMake.