**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 4**

Тема: Основы метапрограммирования

Студент: Савров Никита

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. Постановка задачи

Разработать шаблоны классов согласно варианту задания. Параметром шаблона должен являться скалярный тип данных задающий тип данных для оси координат. Классы должны иметь публичные поля. Фигуры являются фигурами вращения. Для хранения координат фигур необходимо использовать шаблон std::pair.

Создать набор шаблонов, создающих функции, реализующие:

1. Вычисление геометрического центра фигуры;

2. Вывод в стандартный поток вывода std::cout координат вершин фигуры;

3. Вычисление площади фигуры;

Параметром шаблона должен являться тип класса фигуры ( например Square<int>). Помимо самого класса фигуры, шаблонная функция должна уметь работать с tuple. Например, std::tuple<std::pair<int,int>, std::pair<int,int>, std::pair<int,int>> должен интерпретироваться как треугольник. std::tuple<std::pair<int,int>, std::pair<int,int>, std::pair<int,int>, std::pair<int,int>> - как квадрат. Каждый std::pair<int,int> - соответствует координатам вершины фигуры вращения.

Создать программу, которая позволяет:

• Вводить из стандартного ввода std::cin фигуры, согласно варианту задания (как в виде класса, так и в виде std::tuple).

• Вызывать для нее шаблонные функции (1-3).

1. Описание программы

Для реализации задачи реализуем три шаблонных класса: Rectangle, Rhomb, Trapeze в которых будем хранить координаты вершин и перегрузим операторы >> и << для ввода фигур. Вне классов реализуем функции Square и Center для нахождения площади и геометрического центра фигур. Так-же реализуем функцию Set для ввода координат и сохранения их в поле класса. В функции main реализуем интерфейс с помощью которого будем выбирать нужное действие.

1. Набор testcases

test\_01

1

0 0

6 5

4

5

6

2

30 4

4

5

6

3

0 0

0 5

6 5

7 0

4

5

6

7

0 0

0 4

4 4

4 0

8

9

a

0

test\_02

1

0 0

7 2

4

5

6

2

90 5

4

5

6

3

0 0

0 5

7 5

8 0

4

5

6

7

0 0

0 7

7 7

7 0

8

9

a

0

test\_03

1

0 0

9 2

4

5

6

2

60 5

4

5

6

3

0 0

0 5

9 5

10 0

4

5

6

7

0 0

0 8

8 8

8 0

8

9

a

0

1. Результаты выполнения тестов.

test\_01

program for work with figures

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

1

0 0

6 5

4

5

6

2

30 4

4

5

6

3

0 0

0 5

6 5

7 0

4

5

6

7

0 0

0 4

4 4

4 0

8

9

a

0

input cordinats(2 vertex)

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

square for figure = 30

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

center = 3 2.5

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

0 5

6 5

6 0

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

input rhomb(angle and side)

lol'

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

square for figure = 8.00736

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

center = 2.73186 2.73186

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

1.99908 3.46463

5.46371 5.46371

3.46463 1.99908

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

input cordinats(4 vertex)

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

square for figure = 32.5

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

center = 3.25641 2.4359

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

0 5

6 5

7 0

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

enter coordinats (4 vertex)

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

161 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

2 2

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

0 4

4 4

4 0

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

test\_02

program for work with figures

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

1

0 0

7 2

4

5

6

2

90 5

4

5

6

3

0 0

0 5

7 5

8 0

4

5

6

7

0 0

0 7

7 7

7 0

8

9

a

0

input cordinats(2 vertex)

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

square for figure = 14

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

center = 3.5 1

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

0 2

7 2

7 0

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

input rhomb(angle and side)

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

square for figure = 25

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

center = 2.5 2.5

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

0 5

5 5

5 0

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

input cordinats(4 vertex)

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

square for figure = 37.5

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

center = 3.75556 2.44444

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

0 5

7 5

8 0

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

enter coordinats (4 vertex)

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

491 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

3.5 3.5

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

0 7

7 7

7 0

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

test\_03

program for work with figures

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

1

0 0

9 2

4

5

6

2

60 5

4

5

6

3

0 0

0 5

9 5

10 0

4

5

6

7

0 0

0 8

8 8

8 0

8

9

a

0input cordinats(2 vertex)

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

square for figure = 18

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

center = 4.5 1

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

0 2

9 2

9 0

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

input rhomb(angle and side)

lol'

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

square for figure = 21.654

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

center = 3.06163 3.06163

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

1.29345 4.8298

6.12326 6.12326

4.8298 1.29345

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

input cordinats(4 vertex)

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

square for figure = 47.5

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

center = 4.75439 2.45614

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

0 5

9 5

10 0

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

enter coordinats (4 vertex)

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

641 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

4 4

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0 0

0 8

8 8

8 0

1 - enter rectangle

2 - enter rhomb

3 - enter trapeze

4 - square for figure

5 - geomcenter for figure

6 - print figure

7 - enter tuple

8 - square tuple

9 - center tuple

a - print tuple

0 - exit

0

5.Листинг программы

// Лабораторная работа №4 Савров Никита М80-207Б-18

// Разработать шаблоны классов согласно варианту задания. Параметром шаблона должен являться скалярный тип данных задающий тип

// данных для оси координат. Классы должны иметь публичные поля. Фигуры являются фигурами вращения.

// Для хранения координат фигур необходимо использовать шаблон std::pair.

// Создать набор шаблонов, создающих функции, реализующие:

// 1. Вычисление геометрического центра фигуры;

// 2. Вывод в стандартный поток вывода std::cout координат вершин фигуры;

// 3. Вычисление площади фигуры;

// Параметром шаблона должен являться тип класса фигуры ( например Square<int>). Помимо самого класса фигуры, шаблонная функция должна уметь работать с tuple. Например, std::tuple<std::pair<int,int>, std::pair<int,int>, std::pair<int,int>> должен интерпретироваться как треугольник. std::tuple<std::pair<int,int>, std::pair<int,int>, std::pair<int,int>, std::pair<int,int>> - как квадрат. Каждый std::pair<int,int> - соответствует координатам вершины фигуры вращения.

// Создать программу, которая позволяет:

// • Вводить из стандартного ввода std::cin фигуры, согласно варианту задания (как в виде класса, так и в виде std::tuple).

// • Вызывать для нее шаблонные функции (1-3).

// Фигуры, согласно варианту задания: Прямоугольник, трапеция, ромб.

#include <iostream>

#include <vector>

#include <cmath>

#include <tuple>

using namespace std;

template<class T>

class Rectangle{

public:

pair <T, T> a,b,c,d;

using type = T;

int number\_vertex = 4;

Rectangle(){

a.first = 0;

a.second = 0;

b.first = 0;

b.second = 0;

c.first = 0;

c.second = 0;

d.first = 0;

d.second = 0;

}

void Set (vector <pair<T,T>> cord){

a.first = cord[0].first;

a.second = cord[0].second;

b.first = cord[1].first;

b.second = cord[1].second;

c.first = cord[2].first;

c.second = cord[2].second;

d.first = cord[3].first;

d.second = cord[3].second;

}

void Set (tuple<pair <int, int>, pair <int, int>, pair <int, int>, pair <int, int>> t){

a = get<0>(t);

b = get<1>(t);

c = get<2>(t);

d = get<3>(t);

}

friend istream& operator>>(istream& in, Rectangle<T> &obj){

vector <pair <T,T>> v(4);

in >> v[0].first >> v[0].second >> v[2].first >> v[2].second;

v[1].first = v[0].first;

v[1].second = v[2].second;

v[3].first = v[2].first;

v[3].second = v[0].second;

obj.Set(v);

return in;

}

friend ostream& operator<<(ostream& out, Rectangle<T> &obj){

out << obj.a.first << " " << obj.a.second << '\n';

out << obj.b.first << " " << obj.b.second << '\n';

out << obj.c.first << " " << obj.c.second << '\n';

out << obj.d.first << " " << obj.d.second << '\n';

return out;

}

};

template<class T>

class Rhomb{

public:

pair <T, T> a,b,c,d;

using type = T;

int number\_vertex = 4;

Rhomb(){

a.first = 0;

a.second = 0;

b.first = 0;

b.second = 0;

c.first = 0;

c.second = 0;

d.first = 0;

d.second = 0;

}

void Set (vector <pair<T,T>> cord){

a.first = cord[0].first;

a.second = cord[0].second;

b.first = cord[1].first;

b.second = cord[1].second;

c.first = cord[2].first;

c.second = cord[2].second;

d.first = cord[3].first;

d.second = cord[3].second;

}

void Set (tuple<pair <int, int>, pair <int, int>, pair <int, int>, pair <int, int>> t){

a = get<0>(t);

b = get<1>(t);

c = get<2>(t);

d = get<3>(t);

}

friend istream& operator>>(istream& in, Rhomb<T> &obj){

vector <pair <T,T>> v(4);

int angle;

double side;

cin >> angle >> side;

v[0].first = 0;

v[0].second = 0;

if (angle <= 90){

int alpha = (90 - angle) / 2;

v[3].second = sin(alpha) \* side;

v[3].first = cos(alpha) \* side;

v[1].first = sin(alpha) \* side;

v[1].second = cos(alpha) \* side;

v[2].second = sin(alpha) \* side + v[1].second;

v[2].first = cos(alpha) \* side + v[1].first;

} else if (angle < 180) {

int alpha = (270 - angle) / 2;

v[3].second = cos(alpha) \* side;

v[3].first = sin(alpha) \* side;

v[1].first = cos(alpha) \* side;

v[1].second = sin(alpha) \* side;

double d = sqrt(2 \* side \* side - 2 \* side \* side \* (cos(180 - alpha)));

v[2].first = cos(45) \* side;

v[2].second = cos(45) \* side;

} else {

for (int i = 0; i < 4; i++){

v[i].first = 0;

v[i].second = 0;

}

}

obj.Set(v);

return in;

}

friend ostream& operator<<(ostream& out, Rhomb<T> &obj){

out << obj.a.first << " " << obj.a.second << '\n';

out << obj.b.first << " " << obj.b.second << '\n';

out << obj.c.first << " " << obj.c.second << '\n';

out << obj.d.first << " " << obj.d.second << '\n';

return out;

}

};

template<class T>

class Trapeze{

public:

pair <T, T> a,b,c,d;

using type = T;

int number\_vertex = 4;

Trapeze(){

a.first = 0;

a.second = 0;

b.first = 0;

b.second = 0;

c.first = 0;

c.second = 0;

d.first = 0;

d.second = 0;

}

void Set (vector <pair<T,T>> cord){

a.first = cord[0].first;

a.second = cord[0].second;

b.first = cord[1].first;

b.second = cord[1].second;

c.first = cord[2].first;

c.second = cord[2].second;

d.first = cord[3].first;

d.second = cord[3].second;

}

void Set (tuple<pair <int, int>, pair <int, int>, pair <int, int>, pair <int, int>> t){

a = get<0>(t);

b = get<1>(t);

c = get<2>(t);

d = get<3>(t);

}

friend istream& operator>>(istream& in, Trapeze<T> &obj){

vector <pair <T,T>> v(4);

for (int i = 0; i < 4; i++){

in >> v[i].first >> v[i].second;

}

obj.Set(v);

return in;

}

friend ostream& operator<<(ostream& out, Trapeze<T> &obj){

out << obj.a.first << " " << obj.a.second << '\n';

out << obj.b.first << " " << obj.b.second << '\n';

out << obj.c.first << " " << obj.c.second << '\n';

out << obj.d.first << " " << obj.d.second << '\n';

return out;

}

};

template <class T>

typename enable\_if <is\_same<T, tuple<pair <int, int>, pair <int, int>, pair <int, int>, pair <int, int>>>::value,istream&>::type operator>>(istream& in, T &obj){

pair <int, int> a,b,c,d;

in >> a.first >> a.second >> b.first >> b.second >> c.first >> c.second >> d.first >> d.second;

tuple<pair <int, int>, pair <int, int>, pair <int, int>, pair <int, int>> t(a,b,c,d);

obj = t;

return in;

}

template <class T>

typename enable\_if <is\_same<T, tuple<pair <int, int>, pair <int, int>, pair <int, int>, pair <int, int>>>::value,ostream&>::type operator<<(ostream& out, T &obj){

out << get<0>(obj).first << " " << get<0>(obj).second << '\n';

out << get<1>(obj).first << " " << get<1>(obj).second << '\n';

out << get<2>(obj).first << " " << get<2>(obj).second << '\n';

out << get<3>(obj).first << " " << get<3>(obj).second << '\n';

return out;

}

template<class T>

typename enable\_if <is\_same<T, tuple<pair <int, int>, pair <int, int>, pair <int, int>, pair <int, int>>>::value,int>::type Square(T shape){

int x0 = get<0>(shape).first;

int x1 = get<1>(shape).first;

int x2 = get<2>(shape).first;

int x3 = get<3>(shape).first;

int x4 = x0;

int y0 = get<0>(shape).second;

int y1 = get<1>(shape).second;

int y2 = get<2>(shape).second;

int y3 = get<3>(shape).second;

int y4 = y0;

double r1 = 0,r2 = 0;

r1 = x0\*y1 + x1\*y2 + x2\*y3 + x3\*y4;

r2 = y0\*x1 + y1\*x2 + y2\*x3 + y3\*x4;

return (r1 - r2) / 2;

}

template<class T>

typename enable\_if <!is\_same<T, tuple<pair <int, int>, pair <int, int>, pair <int, int>, pair <int, int>>>::value,double>::type Square(T shape){

typename T::type x0 = shape.a.first;

typename T::type x1 = shape.b.first;

typename T::type x2 = shape.c.first;

typename T::type x3 = shape.d.first;

typename T::type x4 = x0;

typename T::type y0 = shape.a.second;

typename T::type y1 = shape.b.second;

typename T::type y2 = shape.c.second;

typename T::type y3 = shape.d.second;

typename T::type y4 = y0;

double r1 = 0,r2 = 0;

r1 = x0\*y1 + x1\*y2 + x2\*y3 + x3\*y4;

r2 = y0\*x1 + y1\*x2 + y2\*x3 + y3\*x4;

return (r1 - r2) / 2;

}

template<class T>

typename enable\_if <is\_same<T, tuple<pair <int, int>, pair <int, int>, pair <int, int>, pair <int, int>>>::value,pair <double, double>>::type Center(T shape){

int x0 = get<0>(shape).first;

int x1 = get<1>(shape).first;

int x2 = get<2>(shape).first;

int x3 = get<3>(shape).first;

int x4 = x0;

int y0 = get<0>(shape).second;

int y1 = get<1>(shape).second;

int y2 = get<2>(shape).second;

int y3 = get<3>(shape).second;

int y4 = y0;

double r1 = 0,r2 = 0,s = 0;

s = 0.5 \* (x0\*y1 - x1\*y0 + x1\*y2 - x2\*y1 + x2\*y3 - x3\*y2 + x3\*y4 - x4\*y3);

r1 = (x0 + x1)\*(x0\*y1 - x1\*y0) + (x1 + x2)\*(x1\*y2 - x2\*y1) + (x2 + x3)\*(x2\*y3 - x3\*y2) + (x3 + x4)\*(x3\*y4 - x4\*y3);

r2 = (y0 + y1)\*(x0\*y1 - x1\*y0) + (y1 + y2)\*(x1\*y2 - x2\*y1) + (y2 + y3)\*(x2\*y3 - x3\*y2) + (y3 + y4)\*(x3\*y4 - x4\*y3);

return make\_pair(r1 \* (1 / (6 \* s)), r2 \* (1 / (6 \* s)));

}

template<class T>

typename enable\_if <!is\_same<T, tuple<pair <int, int>, pair <int, int>, pair <int, int>, pair <int, int>>>::value,pair <double, double>>::type Center(T shape){

typename T::type x0 = shape.a.first;

typename T::type x1 = shape.b.first;

typename T::type x2 = shape.c.first;

typename T::type x3 = shape.d.first;

typename T::type x4 = x0;

typename T::type y0 = shape.a.second;

typename T::type y1 = shape.b.second;

typename T::type y2 = shape.c.second;

typename T::type y3 = shape.d.second;

typename T::type y4 = y0;

double r1 = 0,r2 = 0, s = 0;

s = 0.5 \* (x0\*y1 - x1\*y0 + x1\*y2 - x2\*y1 + x2\*y3 - x3\*y2 + x3\*y4 - x4\*y3);

r1 = (x0 + x1)\*(x0\*y1 - x1\*y0) + (x1 + x2)\*(x1\*y2 - x2\*y1) + (x2 + x3)\*(x2\*y3 - x3\*y2) + (x3 + x4)\*(x3\*y4 - x4\*y3);

r2 = (y0 + y1)\*(x0\*y1 - x1\*y0) + (y1 + y2)\*(x1\*y2 - x2\*y1) + (y2 + y3)\*(x2\*y3 - x3\*y2) + (y3 + y4)\*(x3\*y4 - x4\*y3);

return make\_pair(r1 \* (1 / (6 \* s)), r2 \* (1 / (6 \* s)));

}

int main(){

tuple<pair <int, int>, pair <int, int>, pair <int, int>, pair <int, int>> t;

Rectangle<double> tmp\_rec;

Rhomb<double> tmp\_rh;

Trapeze<double> tmp\_t;

pair <double, double> p;

p.first = 0;

p.second = 0;

int check = 0, checkp = 0;

char menu = '1';

double s = 0;

cout << "program for work with figures\n";

while (menu != '0'){

cout << "1 - enter rectangle\n2 - enter rhomb\n3 - enter trapeze\n4 - square for figure\n5 - geomcenter for figure\n6 - print figure\n7 - enter tuple\n8 - square tuple\n9 - center tuple\na - print tuple\n0 - exit\n";

cin >> menu;

switch (menu)

{

case '1':

cout << "input cordinats(2 vertex)\n";

cin >> tmp\_rec;

check = 1;

break;

case '2':

cout << "input rhomb(angle and side)\n";

cin >> tmp\_rh;

check = 2;

break;

case '3':

cout << "input cordinats(4 vertex)\n";

cin >> tmp\_t;

check = 3;

break;

case '4':

if (check == 1){

s = Square(tmp\_rec);

} else if (check == 2){

s = Square(tmp\_rh);

} else if (check == 3){

s = Square(tmp\_t);

}

if (s == 0){

cout << "wrong figure\n";

} else {

cout << "square for figure = " << abs(s) << '\n';

}

break;

case '5':

if (check == 1){

p = Center(tmp\_rec);

} else if (check == 2){

p = Center(tmp\_rh);

} else if (check == 3){

p = Center(tmp\_t);

}

if (p.first == 0 && p.second == 0){

cout << "wrong figure\n";

} else{

cout << "center = " << p.first << " " << p.second << '\n';

}

break;

case '6':

if (check == 1){

cout << tmp\_rec;

} else if (check == 2){

cout << tmp\_rh;

} else if (check == 3){

cout << tmp\_t;

} else {

"wrong figure\n";

}

break;

case '7':

cout << "enter coordinats (4 vertex)\n";

cin >> t;

checkp = 1;

break;

case '8':

s = Square(t);

cout << s;

break;

case '9':

p = Center(t);

cout << p.first << " " << p.second << '\n';

break;

case 'a':

if (checkp == 1)

cout << t;

else

cout << "wrong tuple\n";

default:

break;

}

}

}

https://github.com/trol53/oop\_exercise\_01/tree/master/oop\_exercise\_04

6.Вывод

Данная программа может быть полезна для работы с геометрическими фигурами. Так-же ее можно улучшить добавив дополнительные функции вроде нахождения периметра фигуры и прочие. Благодаря парадигме ООП мы можем представлять пользовательские типы данных в виде объектов что значительно упрощает программирование. Так-же эта работа показывает практическое применение шаблонных классов.

Литература

1.Онлайн библиотека c++ cppreference[электронный ресурс] URL:https://ru.cppreference.com/w/cpp/utility/tuple

(дата обращения 9.11.2019)

2.Информационный портал хабр [электронный ресурс] URL: https://habr.com/ru/post/260899/

(дата обращения 9.11.2019)