**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: Трофимов Максим

Группа: 80-201

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. Постановка задачи

Разработать шаблоны классов квадратов, прямоугольников и трапеци.  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.

1. Описание программы

Программа сначала предлагает выбрать тип координатной сетки, который можно в последствии сменить.  
Затем выдаётся меню, которое предлагает проверить работу библиотеки с шаблонами квадрата, прямоугольника, трапеции

и шаблонных функций с использованием тьюпла.

Проверка работы заключается в том, что пользователь вводит 2 вершины для квадрата, три для прямоугольника и четыре для трапеции, а в ответ ему выдаётся информация, что это за фигура, координаты всех четырех её вершин, центра и площадь.

ссылка на github: https://github.com/student31415/oop\_exercsise\_04

1. Набор testcases

|  |  |
| --- | --- |
| Тест | Примичание |
| 1  2  0 3 7 4  3  0 0 5 0 5 2  4  0 5 0 3 3 0 5 0  5  2  0 3 7 4  5  3  0 0 5 0 5 2  5  4  0 5 0 3 3 0 5 0  0 0 | Работа со всеми фигурами и тьюплами на целой сетке |
| 2  2  0 3 7 4  3  0 0 5 0 5 2  4  0 5 0 3 3 0 5 0  5  2  0 3 7 4  5  3  0 0 5 0 5 2  5  4  0 5 0 3 3 0 5 0  0 0 | Всё то же самое, только на вещественной сетке. |

1. Результаты выполнения тестов.

**test\_01.txt:**

Program for work with figures in different coordinates grids.

Accepted figures: square, rectangle, trapeze.

Choose type of coordinate grid.

1. integer.

2. real.

0. exit

>>1

1.Print menu.

2.Work with square.

3.Work with rectangle.

4.Work with trapeze.

5.Work with tuple.

0. Exit or change coordinate grid type.

>>2

Input coordinates of two opposite square vertexes.

>>0 3 7 4

It's Square.

vertexes:

( 0 , 3 ) ; ( 3 , 7 ) ; ( 7 , 4 ) ; ( 4 , 0 ) .

square: 25 .

center: ( 3 , 3 ) .

>>3

Input coordinates of three rectangle vertexes, coming one after another.

>>0 0 5 0 5 2

It's Rectangle.

vertexes:

( 0 , 0 ) ; ( 5 , 0 ) ; ( 5 , 2 ) ; ( 0 , 2 ) .

square: 10 .

center: ( 2 , 1 ) .

>>4

Input coordinates of all trapeze vertexes, coming one after another.

>>0 5 0 3 3 0 5 0

trapeze

It's Trapeze.

vertexes:

( 0 , 5 ) ; ( 0 , 3 ) ; ( 3 , 0 ) ; ( 5 , 0 ) .

square: 17.8885 .

center: ( 2 , 2 ) .

>>5

Input number of vertexes (2-4).

>>2

Input vertexes of current figure.

(for squre: 2 opposite;

for rectangle 3, coming one after another;

for trapeze 4, coming one after another)

>>0 3 7 4

It's Square.

vertexes:

( 0 , 3 ) ; ( 3 , 7 ) ; ( 7 , 4 ) ; ( 4 , 0 ) .

square: 25 .

center: ( 3 , 3 ) .

>>5

Input number of vertexes (2-4).

>>3

Input vertexes of current figure.

(for squre: 2 opposite;

for rectangle 3, coming one after another;

for trapeze 4, coming one after another)

>>0 0 5 0 5 2

It's Rectangle.

vertexes:

( 0 , 0 ) ; ( 5 , 0 ) ; ( 5 , 2 ) ; ( 0 , 2 ) .

square: 10 .

center: ( 2 , 1 ) .

>>5

Input number of vertexes (2-4).

>>4

Input vertexes of current figure.

(for squre: 2 opposite;

for rectangle 3, coming one after another;

for trapeze 4, coming one after another)

>>0 5 0 3 3 0 5 0

It's Trapeze.

vertexes:

( 0 , 5 ) ; ( 0 , 3 ) ; ( 3 , 0 ) ; ( 5 , 0 ) .

square: 17.8885 .

center: ( 2 , 2 ) .

>>0

Choose type of coordinate grid.

1. integer.

2. real.

0. exit

>>0

exit.

**test\_02.txt**

Program for work with figures in different coordinates grids.

Accepted figures: square, rectangle, trapeze.

Choose type of coordinate grid.

1. integer.

2. real.

0. exit

>>2

1.Print menu.

2.Work with square.

3.Work with rectangle.

4.Work with trapeze.

5.Work with tuple.

0. Exit or change coordinate grid type.

>>2

Input coordinates of two opposite square vertexes.

>>0 3 7 4

It's Square.

vertexes:

( 0 , 3 ) ; ( 3 , 7 ) ; ( 7 , 4 ) ; ( 4 , 0 ) .

square: 25 .

center: ( 3.5 , 3.5 ) .

>>3

Input coordinates of three rectangle vertexes, coming one after another.

>>0 0 5 0 5 2

It's Rectangle.

vertexes:

( 0 , 0 ) ; ( 5 , 0 ) ; ( 5 , 2 ) ; ( 0 , 2 ) .

square: 10 .

center: ( 2.5 , 1 ) .

>>4

Input coordinates of all trapeze vertexes, coming one after another.

>>0 5 0 3 3 0 5 0

trapeze

It's Trapeze.

vertexes:

( 0 , 5 ) ; ( 0 , 3 ) ; ( 3 , 0 ) ; ( 5 , 0 ) .

square: 17.8885 .

center: ( 2 , 2 ) .

>>5

Input number of vertexes (2-4).

>>2

Input vertexes of current figure.

(for squre: 2 opposite;

for rectangle 3, coming one after another;

for trapeze 4, coming one after another)

>>0 3 7 4

It's Square.

vertexes:

( 0 , 3 ) ; ( 3 , 7 ) ; ( 7 , 4 ) ; ( 4 , 0 ) .

square: 25 .

center: ( 3.5 , 3.5 ) .

>>5

Input number of vertexes (2-4).

>>3

Input vertexes of current figure.

(for squre: 2 opposite;

for rectangle 3, coming one after another;

for trapeze 4, coming one after another)

>>0 0 5 0 5 2

It's Rectangle.

vertexes:

( 0 , 0 ) ; ( 5 , 0 ) ; ( 5 , 2 ) ; ( 0 , 2 ) .

square: 10 .

center: ( 2.5 , 1 ) .

>>5

Input number of vertexes (2-4).

>>4

Input vertexes of current figure.

(for squre: 2 opposite;

for rectangle 3, coming one after another;

for trapeze 4, coming one after another)

>>0 5 0 3 3 0 5 0

It's Trapeze.

vertexes:

( 0 , 5 ) ; ( 0 , 3 ) ; ( 3 , 0 ) ; ( 5 , 0 ) .

square: 17.8885 .

center: ( 2 , 2 ) .

>>0

Choose type of coordinate grid.

1. integer.

2. real.

0. exit

>>0

exit.

1. fgЛистинг программы

**Figures\_templates.cpp**

#ifndef \_\_FIG\_TEMPL\_\_

#define \_\_FIG\_TEMPL\_\_

#include<iostream>

#include<cmath>

//#include<string>

//#include<vector>

#include<tuple>

template <class T>

double length(std::pair<T, T> p1, std::pair<T, T> p2){

return sqrt((p2.first-p1.first)\*(p2.first-p1.first) + (p2.second-p1.second)\*(p2.second-p1.second));

}

template <class T>

double cos(std::pair<T, T> p1, std::pair<T, T> p2, std::pair<T, T> p3, std::pair<T, T> p4){

return ( (p2.first-p1.first) \* (p4.first-p3.first)

+ (p2.second-p1.second) \* (p4.second-p3.second) )

/ ( length(p1, p2) \* length( p3, p4) );

}

template <class T>

class Square{

public:

using type = T;

using vertex = std::pair<T, T>;

vertex p1={}, p2={};

Square(vertex a, vertex b){

if(a.first!=b.first || a.second!=b.second){

p1 = a;

p2 = b;

}

}

};

template <class T>

class Rectangle{

public:

using type = T;

using vertex = std::pair<T, T>;

vertex p1 ={}, p2 ={} , p3 = {};

Rectangle(vertex a, vertex b, vertex c){

if( (a.first!=b.first || a.second!=b.second)

&& (c.first!=b.first || c.second!=b.second)

&& (a.first!=c.first || a.second!=c.second)

&& (cos(a, b, b, c)<=1e-9 && cos(a, b, b, c)>=-1e-9) ) {

p1=a;

p2=b;

p3=c;

}

}

};

template <class T>

class Trapeze{

public:

using type =T;

using vertex= std::pair<T, T>;

vertex p1={}, p2={}, p3={}, p4={};

Trapeze(vertex a, vertex b, vertex c, vertex d){

if( ( ( length(a, b) - length(c, d)<=1e-9) && ( length(a, b) - length(c, d)>=-1e-9 ) )

&&!( (length(a, b) <= 1e-9) && (length(a, b) >= -1e-9) )

&& ( ( (cos(a, d, c, b) <= 1+1e-9) && (cos(a, d, c, b) >= 1-1e-9) )

|| ( (cos(a, d, c, b)<=-1+1e-9) && (cos(a, d, c, b)>=-1-1e-9) ) )

) {

std::cout<<"trapeze"<<std::endl;

p1=a;

p2=b;

p3=c;

p4=d;

}

}

};

template <class T>

double square( Square<T> a){

std::pair<T, T> c {

-a.p2.second + (a.p2.second + a.p1.second + a.p2.first + a.p1.first)/2,

a.p2.first + (a.p2.second + a.p1.second - a.p2.first - a.p1.first)/2

};

return (double) length( a.p1 , c )

\* length( a.p2 , c );

}

template <class T>

double square(Rectangle<T> a){

return (double) length( a.p1, a.p2 )

\* length( a.p2, a.p3 );

}

template <class T>

double square(Trapeze<T> a){

return (double)

(length(a.p1, a.p4)+length(a.p2, a.p3))/2

\* length(a.p2, std::pair<T, T> {

a.p1.first + ( length(a.p1, a.p4) - length(a.p2, a.p3) ) / 2

\* ( length(a.p1, std::pair{ a.p4.first, a.p1.second} ) / length(a.p1, a.p4) ),

a.p1.second + ( length(a.p1, a.p4)-length(a.p2, a.p3) )/2

\* ( length(a.p4, std::pair{a.p4.first, a.p1.second}) / length(a.p1, a.p4) )

}

);

}

template <class T>

std::pair<T, T> center(Square<T> a){

return std::pair{

( a.p1.first + a.p2.first ) / 2,

( a.p1.second + a.p2.second ) / 2

};

}

template <class T>

std::pair<T, T> center(Rectangle<T> a){

return std::pair{

( a.p1.first + a.p3.first ) / 2,

( a.p1.second + a.p3.second ) / 2

};

}

template <class T>

std::pair<T, T> center(Trapeze<T> a){

return std::pair{

( a.p1.first + a.p2.first + a.p3.first + a.p4.first ) / 4,

( a.p1.second + a.p2.second + a.p3.second + a.p4.second ) / 4

};

}

template <class T>

std::ostream& operator <<(std::ostream &o, std::pair<T, T>& a){

o<<" ( "<<a.first<<" , "<<a.second<<" ) ";

return o;

}

template <class T>

std::ostream& operator <<(std::ostream &o, std::pair<T, T>&& a){

o<<" ( "<<a.first<<" , "<<a.second<<" ) ";

return o;

}

template <class T>

void print\_info(Square<T> a){

std::cout<<"It's Square."<<std::endl

<<"vertexes:"<<std::endl

<<a.p1<<';'<<std::pair<T, T>{

-a.p2.second + (a.p2.second + a.p1.second + a.p2.first + a.p1.first)/2,

a.p2.first + (a.p2.second + a.p1.second - a.p2.first - a.p1.first)/2

}<<';'<<a.p2<<';'<<std::pair<T, T>{

a.p2.second + (a.p2.first + a.p1.first - a.p2.second - a.p1.second)/2,

-a.p2.first + (a.p2.second + a.p1.second + a.p2.first + a.p1.first)/2

}<<'.'<<std::endl

<<"square: "<<square(a)<<" ."<<std::endl

<<"center:"<<center(a)<<'.'<<std::endl;

}

template <class T>

void print\_info(Rectangle<T> a){

std::cout<<"It's Rectangle."<<std::endl

<<"vertexes:"<<std::endl

<<a.p1<<';'<<a.p2<<';'<<a.p3<<';'<<std::pair<T, T>{

a.p3.first-a.p2.first+a.p1.first,

a.p3.second-a.p2.second+a.p1.second

}<<'.'<<std::endl

<<"square: "<<square(a)<<" ."<<std::endl

<<"center:"<<center(a)<<'.'<<std::endl;

}

template <class T>

void print\_info(Trapeze<T> a){

std::cout<<"It's Trapeze."<<std::endl

<<"vertexes:"<<std::endl

<<a.p1<<';'<<a.p2<<';'<<a.p3<<';'<<a.p4<<'.'<<std::endl

<<"square: "<<square(a)<<" ."<<std::endl

<<"center:"<<center(a)<<'.'<<std::endl;

}

//classic classes ^^^

//=============================================================

//tuples vvv

template <class T>

double square( std::tuple<std::pair<T, T>, std::pair<T, T>> a){

if(std::get<0>(a).first!=std::get<1>(a).first || std::get<0>(a).second!=std::get<1>(a).second){

std::pair<T, T> c {

-std::get<1>(a).second + (std::get<1>(a).second + std::get<0>(a).second + std::get<1>(a).first + std::get<0>(a).first)/2,

std::get<1>(a).first + (std::get<1>(a).second + std::get<0>(a).second - std::get<1>(a).first - std::get<0>(a).first)/2

};

return (double) length( std::get<0>(a) , c )

\* length( std::get<1>(a) , c );

}

return 0.0;

}

template <class T>

double square(std::tuple<std::pair<T, T>, std::pair<T, T>, std::pair<T, T>> a){

if( (std::get<0>(a).first!=std::get<1>(a).first || std::get<0>(a).second!=std::get<1>(a).second)

&& (std::get<2>(a).first!=std::get<1>(a).first || std::get<2>(a).second!=std::get<1>(a).second)

&& (std::get<0>(a).first!=std::get<2>(a).first || std::get<0>(a).second!=std::get<2>(a).second)

&& (cos(std::get<0>(a), std::get<1>(a), std::get<1>(a), std::get<2>(a))<=1e-9

&& cos(std::get<0>(a), std::get<1>(a), std::get<1>(a), std::get<2>(a))>=-1e-9) ) {

return (double) length( std::get<0>(a), std::get<1>(a) )

\* length( std::get<1>(a), std::get<2>(a) );

}

return 0.0;

}

template <class T>

double square( std::tuple<std::pair<T, T>, std::pair<T, T>, std::pair<T, T>, std::pair<T, T>> a){

if( ( ( length(std::get<0>(a), std::get<1>(a)) - length(std::get<2>(a), std::get<3>(a)) <= 1e-9 )

&& ( length(std::get<0>(a), std::get<1>(a)) - length(std::get<2>(a), std::get<3>(a)) ) >=-1e-9 )

&&!( (length(std::get<0>(a), std::get<1>(a)) <= 1e-9) && (length(std::get<0>(a), std::get<1>(a)) >= -1e-9) )

&& ( ( (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) <= 1+1e-9)

&& (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) >= 1-1e-9) )

|| ( (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) <= -1+1e-9)

&& (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) >= -1-1e-9) ) ) ) {

return (double)

(length(std::get<0>(a), std::get<3>(a))+length(std::get<1>(a), std::get<2>(a)))/2

\* length(std::get<1>(a), std::pair<T, T> {

std::get<0>(a).first + ( length(std::get<0>(a), std::get<3>(a)) - length(std::get<1>(a), std::get<2>(a)) ) / 2

\* ( length(std::get<0>(a), std::pair{ std::get<3>(a).first, std::get<0>(a).second} ) / length(std::get<0>(a), std::get<3>(a)) ),

std::get<0>(a).second + ( length(std::get<0>(a), std::get<3>(a))-length(std::get<1>(a), std::get<2>(a)) )/2

\* ( length(std::get<3>(a), std::pair{std::get<3>(a).first, std::get<0>(a).second}) / length(std::get<0>(a), std::get<3>(a)) )

}

);

}

return 0.0;

}

template <class T>

std::pair<T, T> center(std::tuple<std::pair<T, T>, std::pair<T, T>> a){

if(std::get<0>(a).first!=std::get<1>(a).first || std::get<0>(a).second!=std::get<1>(a).second){

return std::pair{

( std::get<0>(a).first + std::get<1>(a).first ) / 2,

( std::get<0>(a).second + std::get<1>(a).second ) / 2

};

}

return std::pair<T, T> {0.0, 0.0};

}

template <class T>

std::pair<T, T> center(std::tuple<std::pair<T, T>, std::pair<T, T>, std::pair<T, T>> a){

if( (std::get<0>(a).first!=std::get<1>(a).first || std::get<0>(a).second!=std::get<1>(a).second)

&& (std::get<2>(a).first!=std::get<1>(a).first || std::get<2>(a).second!=std::get<1>(a).second)

&& (std::get<0>(a).first!=std::get<2>(a).first || std::get<0>(a).second!=std::get<2>(a).second)

&& (cos(std::get<0>(a), std::get<1>(a), std::get<1>(a), std::get<2>(a))<=1e-9

&& cos(std::get<0>(a), std::get<1>(a), std::get<1>(a), std::get<2>(a))>=-1e-9) ) {

return std::pair{

( std::get<0>(a).first + std::get<2>(a).first ) / 2,

( std::get<0>(a).second + std::get<2>(a).second ) / 2

};

}

return std::pair<T, T>{0.0, 0.0};

}

template <class T>

std::pair<T, T> center( std::tuple<std::pair<T, T>, std::pair<T, T>, std::pair<T, T>, std::pair<T, T>> a ){

if( ( ( length(std::get<0>(a), std::get<1>(a)) - length(std::get<2>(a), std::get<3>(a)) <= 1e-9 )

&& ( length(std::get<0>(a), std::get<1>(a)) - length(std::get<2>(a), std::get<3>(a)) ) >=-1e-9 )

&&!( (length(std::get<0>(a), std::get<1>(a)) <= 1e-9) && (length(std::get<0>(a), std::get<1>(a)) >= -1e-9) )

&& ( ( (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) <= 1+1e-9)

&& (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) >= 1-1e-9) )

|| ( (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) <= -1+1e-9)

&& (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) >= -1-1e-9) ) ) ) {

return std::pair{

(std::get<0>(a).first + std::get<1>(a).first + std::get<2>(a).first + std::get<3>(a).first)/4,

(std::get<0>(a).second+ std::get<1>(a).second+ std::get<2>(a).second+ std::get<3>(a).second)/4

};

}

return std::pair<T, T>{0.0, 0.0};

}

template <class T>

void print\_info(std::tuple<std::pair<T, T>, std::pair<T, T>> a){

if(std::get<0>(a).first!=std::get<1>(a).first || std::get<0>(a).second!=std::get<1>(a).second){

std::cout<<"It's Square."<<std::endl

<<"vertexes:"<<std::endl

<<std::get<0>(a)<<';'<<std::pair<T, T>{

-std::get<1>(a).second + (std::get<1>(a).second + std::get<0>(a).second + std::get<1>(a).first + std::get<0>(a).first)/2,

std::get<1>(a).first + (std::get<1>(a).second + std::get<0>(a).second - std::get<1>(a).first - std::get<0>(a).first)/2

}<<';'

<<std::get<1>(a)<<';'<<std::pair<T, T>{

std::get<1>(a).second + (std::get<1>(a).first + std::get<0>(a).first - std::get<1>(a).second - std::get<0>(a).second)/2,

-std::get<1>(a).first + (std::get<1>(a).second + std::get<0>(a).second + std::get<1>(a).first + std::get<0>(a).first)/2

}<<'.'<<std::endl

<<"square: "<<square(a)<<" ."<<std::endl

<<"center:"<<center(a)<<'.'<<std::endl;

}else{

std::cout<<"It's Square."<<std::endl;

}

}

template <class T>

void print\_info(std::tuple<std::pair<T, T>, std::pair<T, T>, std::pair<T, T>> a){

if( (std::get<0>(a).first!=std::get<1>(a).first || std::get<0>(a).second!=std::get<1>(a).second)

&& (std::get<2>(a).first!=std::get<1>(a).first || std::get<2>(a).second!=std::get<1>(a).second)

&& (std::get<0>(a).first!=std::get<2>(a).first || std::get<0>(a).second!=std::get<2>(a).second)

&& (cos(std::get<0>(a), std::get<1>(a), std::get<1>(a), std::get<2>(a))<=1e-9

&& cos(std::get<0>(a), std::get<1>(a), std::get<1>(a), std::get<2>(a))>=-1e-9) ) {

std::cout<<"It's Rectangle."<<std::endl

<<"vertexes:"<<std::endl

<<std::get<0>(a)<<';'<<std::get<1>(a)<<';'<<std::get<2>(a)<<';'<<std::pair<T, T>{

std::get<2>(a).first-std::get<1>(a).first+std::get<0>(a).first,

std::get<2>(a).second-std::get<1>(a).second+std::get<0>(a).second

}<<'.'<<std::endl

<<"square: "<<square(a)<<" ."<<std::endl

<<"center:"<<center(a)<<'.'<<std::endl;

}else{

std::cout<<"It's not Rectangle."<<std::endl;

}

}

template <class T>

void print\_info(std::tuple<std::pair<T, T>, std::pair<T, T>, std::pair<T, T>, std::pair<T, T>> a){

if( ( ( length(std::get<0>(a), std::get<1>(a)) - length(std::get<2>(a), std::get<3>(a)) <= 1e-9 )

&& ( length(std::get<0>(a), std::get<1>(a)) - length(std::get<2>(a), std::get<3>(a)) ) >=-1e-9 )

&&!( (length(std::get<0>(a), std::get<1>(a)) <= 1e-9) && (length(std::get<0>(a), std::get<1>(a)) >= -1e-9) )

&& ( ( (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) <= 1+1e-9)

&& (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) >= 1-1e-9) )

|| ( (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) <= -1+1e-9)

&& (cos(std::get<0>(a), std::get<3>(a), std::get<2>(a), std::get<1>(a)) >= -1-1e-9) ) ) ) {

std::cout<<"It's Trapeze."<<std::endl

<<"vertexes:"<<std::endl

<<std::get<0>(a)<<';'<<std::get<1>(a)<<';'<<std::get<2>(a)<<';'<<std::get<3>(a)<<'.'<<std::endl

<<"square: "<<square(a)<<" ."<<std::endl

<<"center:"<<center(a)<<'.'<<std::endl;

}else{

std::cout<<"It's not Trapeze."<<std::endl;

}

}

#endif

**main.cpp**

#include<tuple>

#include<iostream>

#include<string>

#include"Figures\_templates.cpp"

/\*

Трофимов Максим м8о-201б

вариант 15

Разработать шаблоны классов фигур квадрата, прямоугольника и трапеции.

Также сделать так, чтобы шаблонные функции для классов так же работа ли для std::tuple .

\*/

void print\_menu(){

std::cout

<<"1.Print menu."<<std::endl

<<"2.Work with square."<<std::endl

<<"3.Work with rectangle."<<std::endl

<<"4.Work with trapeze."<<std::endl

<<"5.Work with tuple."<<std::endl

<<"0. Exit or change coordinate grid type."<<std::endl;

return;

}

bool is\_num(std::string a){

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

if( !((a[i]>='0' && a[i]<='9') || (a[i]=='.') || (a[i]=='-')) )

return false;

return true;

}

template <class T>

void work\_with\_squard(){

std::cout<<"Input coordinates of two opposite square vertexes."<<std::endl<<">>";

std::pair<T, T> a[2];

std::string buf;

for(int i=0; i<2; ++i){

std::cin>>buf;

if(is\_num(buf) ){

a[i].first=std::atof(buf.c\_str());

}else{

std::cout<<"Wrong input."<<std::endl;

return;

}

std::cin>>buf;

if(is\_num(buf) ){

a[i].second=std::atof(buf.c\_str());

}else{

std::cout<<"Wrong input."<<std::endl;

return;

}

}

Square f {a[0],a[1]};

if(f.p1.first==f.p2.first && f.p1.second==f.p2.second){

std::cout<<"Vertexes don't belong square."<<std::endl;

return;

}

print\_info(f);

return;

}

template <class T>

void work\_with\_rectangle(){

std::cout<<"Input coordinates of three rectangle vertexes, coming one after another."<<std::endl<<">>";

std::pair<T, T> a[3];

std::string buf;

for(int i=0; i<3; ++i){

std::cin>>buf;

if(is\_num(buf) ){

a[i].first=std::atof(buf.c\_str());

}else{

std::cout<<"Wrong input."<<std::endl;

return;

}

std::cin>>buf;

if(is\_num(buf) ){

a[i].second=std::atof(buf.c\_str());

}else{

std::cout<<"Wrong input."<<std::endl;

return;

}

}

Rectangle f {a[0],a[1],a[2]};

if(f.p1.first==f.p2.first && f.p1.second==f.p2.second){

std::cout<<"Vertexes don't belong rectangle."<<std::endl;

return;

}

print\_info(f);

return;

}

template <class T>

void work\_with\_trapeze(){

std::cout<<"Input coordinates of all trapeze vertexes, coming one after another."<<std::endl<<">>";

std::pair<T, T> a[4];

std::string buf;

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

std::cin>>buf;

if(is\_num(buf) ){

a[i].first=std::atof(buf.c\_str());

}else{

std::cout<<"Wrong input."<<std::endl;

return;

}

std::cin>>buf;

if(is\_num(buf) ){

a[i].second=std::atof(buf.c\_str());

}else{

std::cout<<"Wrong input."<<std::endl;

return;

}

}

Trapeze f {a[0],a[1],a[2],a[3]};

if(f.p1.first==f.p2.first && f.p1.second==f.p2.second){

std::cout<<"Vertexes don't belong trapeze."<<std::endl;

return;

}

print\_info(f);

return;

}

template<class T>

void work\_with\_tuple(){

std::string buf;

std::cout<<"Input number of vertexes (2-4)."<<std::endl<<">>";

int size;

std::cin>>buf;

if(is\_num(buf) ){

size=std::atoi(buf.c\_str());

}else{

std::cout<<"Wrong input."<<std::endl;

return;

}

if(size<2 || size>4){

std::cout<<"Wrong number of vertexes."<<std::endl;

return;

}

std::pair<T,T> a[4];

std::cout<<"Input vertexes of current figure."

<<std::endl<<"(for squre: 2 opposite;"

<<std::endl<<"for rectangle 3, coming one after another;"

<<std::endl<<"for trapeze 4, coming one after another)"

<<std::endl<<">>";

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

std::cin>>buf;

if(is\_num(buf) ){

a[i].first=std::atof(buf.c\_str());

}else{

std::cout<<"Wrong input."<<std::endl;

return;

}

std::cin>>buf;

if(is\_num(buf) ){

a[i].second=std::atof(buf.c\_str());

}else{

std::cout<<"Wrong input."<<std::endl;

return;

}

}

switch(size){

case 2:

print\_info(std::make\_tuple(a[0], a[1]));

break;

case 3:

print\_info(std::make\_tuple(a[0], a[1], a[2]));

break;

case 4:

print\_info(std::make\_tuple(a[0], a[1], a[2], a[3]));

break;

}

return;

}

template <class T>

void menu(){

std::string buffer;

int status=1;

while(status){

switch(status){

case 0:

return;

case 1:

print\_menu();

break;

case 2:

work\_with\_squard<T>();

break;

case 3:

work\_with\_rectangle<T>();

break;

case 4:

work\_with\_trapeze<T>();

break;

case 5:

work\_with\_tuple<T>();

break;

default:

std::cout<<"Wrong input."<<std::endl;

print\_menu();

}

std::cout<<">>";

std::cin>>buffer;

if(is\_num(buffer) ){

status=std::atoi(buffer.c\_str());

}else{

status=0;

return;

}

}

}

int main(){

std::string buffer;

int axises;

std::cout<<"Program for work with figures in different coordinates grids."<<std::endl

<<"Accepted figures: square, rectangle, trapeze."<<std::endl

<<"Choose type of coordinate grid."<<std::endl

<<"1. integer."<<std::endl

<<"2. real."<<std::endl

<<"0. exit"<<std::endl

<<">>";

std::cin>>buffer;

if(is\_num(buffer) ){

axises=std::atoi(buffer.c\_str());

}else{

axises=-1;

}

while(true){

switch(axises){

case 0:

std::cout<<"exit."<<std::endl;

return 0;

case 1 :{

//code for int

menu<int>();

break;

}

case 2 :{

//code for double

menu<double>();

break;

}

default:{

std::cout<<"Неправильный ввод."<<std::endl;

}

}

std::cout<<"Choose type of coordinate grid."<<std::endl

<<"1. integer."<<std::endl

<<"2. real."<<std::endl

<<"0. exit"<<std::endl

<<">>";

std::cin>>buffer;

if(is\_num(buffer) ){

axises=std::atoi(buffer.c\_str());

}else{

axises=-1;

}

}

}

1. Вывод

Научился основам метапрограммирования, работе с шаблонами и тьюплами.

**Список литературы**

1. Шилдт, Герберт. С++: базовый курс, 3-е изд. : Пер. с англ. - М. : ООО “И.Д. Вильямс”, 2018. - 624 с. : ил. - Парал. тит. англ.

2. Справочник по языку С++ [Электронный ресурс]. URL: [http://www.cplusplus.com/reference/](http://www.cplusplus.com/reference/deque/) (дата обращения: 20.09.2019).