**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 4**

Тема: Основы метапрограммирования

Студент: Воронов Кирилл Михайлович

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

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 только публичные поля. В классах не должно быть методов, только поля. Фигуры являются фигурами вращения (равнобедренными), за исключением трапеции и прямоугольника. Для хранения координат фигур необходимо использовать шаблон std::pair.

Необходимо реализовать две шаблонных функции:

1. Функция **print** печати фигур на экран std::cout (печататься должны координаты вершин фигур). Функция должна принимать на вход std::tuple с фигурами, согласно варианту задания (минимум по одной каждого класса).

2. Функция **square** вычисления суммарной площади фигур. Функция должна принимать на вход std::tuple с фигурами, согласно варианту задания (минимум по одной каждого класса).

Создать программу, которая:

1. Создает набор фигур согласно варианту задания (как минимум по одной фигуре каждого типа с координатами типа int и координатами типа double).

2.Сохраняет фигуры в std::tuple.

3. Печатает на экран содержимое std::tuple с помощью шаблонной функции print.

4.Вычисляет суммарную площадь фигур в std::tuple и выводит значение на экран.

*Вариант 27*

Прямоугольник Ромб Трапеция

1. **Описание программы**

Создадим три шаблонных класса для каждой из фигуры:

1. Класс Pramougol

Этот класс предназначен для прямоугольника. Будем хранить левую нижнюю координату t (в виде пары) и длины двух сторон а, b.

2.Класс Romb

Этот класс предназначен для ромба. Будем хранить нижнюю координату и длины двух диагоналей.

3. Класс Trapecia

Этот класс предназначен для трапеции. Будем хранить её нижнюю левую координату, длины оснований и боковой стороны.

Создадим tuple, в который поместим каждую фигуру двух типов: int и double. Значения соответствующих параметров вводятся пользователем с клавиатуры. Далее запускаем функцию print1, которая рекурсивно проходит по tuple и отправляет каждый элемент(класс) в функцию print2, выводящую координаты фигур на экран (для каждого класса реализована отдельно). Аналогично работают функции square1 и square2 для нахождения площади фигур.

1. **Набор тестов и результаты их выполнения**

**Тест 1**

Введите координату нижней левой точки прямоугольника и длины сторон, прилегающих к ней, начиная с горизонтальной (тип double)

jkk

Введены неверные данные

**Тест 2**

Введите координату нижней левой точки прямоугольника и длины сторон, прилегающих к ней, начиная с горизонтальной (тип double)

1.54 3.2 7.1 23.1

Введите координату нижней левой точки прямоугольника и длины сторон, прилегающих к ней, начиная с горизонтальной (тип int)

0 1 3 4

Введите координату нижней точки ромба и длину его диагоналей, начиная с горизонтальной (тип double)

7.43 4.34 1.43 54.3

Введите координату нижней точки ромба и длину его диагоналей, начиная с горизонтальной (тип int)

5 5 2 3

Введите координату нижней левой точки трапеции, наибольшее основание, наименьшее основание и длину боковой стороны (тип double)

1.8 2.3 10 9 5

Введите координату нижней левой точки трапеции, наибольшее основание, наименьшее основание и длину боковой стороны (тип int)

4 2 16 12 56

(1.54,26.3) (8.64,26.3)

(1.54,3.2) (8.64,3.2)

-----------------------

(0,5) (3,5)

(0,1) (3,1)

-----------------------

(7.43,58.64)

(6.715,31.49) (8.145,31.49)

(7.43,4.34)

-----------------------

(5,8)

(4,6) (6,6)

(5,5)

-----------------------

(2.3,7.27494) (1.3,7.27494)

(1.8,2.3) (1.8,11.3)

-----------------------

(6,57) (2,57)

(4,2) (4,14)

-----------------------

Общая площадь фигур - 1048.14

**Тест 3**

Введите координату нижней левой точки прямоугольника и длины сторон, прилегающих к ней, начиная с горизонтальной (тип double)

90.3 76.2134 8.34 88.31

Введите координату нижней левой точки прямоугольника и длины сторон, прилегающих к ней, начиная с горизонтальной (тип int)

45 32 90 48

Введите координату нижней точки ромба и длину его диагоналей, начиная с горизонтальной (тип double)

54.01 88.3 18.22 32.32

Введите координату нижней точки ромба и длину его диагоналей, начиная с горизонтальной (тип int)

5 87 93 34

Введите координату нижней левой точки трапеции, наибольшее основание, наименьшее основание и длину боковой стороны (тип double)

3.54 666.3 78.2 60.323 100.1

Введите координату нижней левой точки трапеции, наибольшее основание, наименьшее основание и длину боковой стороны (тип int)

4 6 54 48 50

(90.3,164.523) (98.64,164.523)

(90.3,76.2134) (98.64,76.2134)

-----------------------

(45,80) (135,80)

(45,32) (135,32)

-----------------------

(54.01,120.62)

(44.9,104.46) (63.12,104.46)

(54.01,88.3)

-----------------------

(5,121)

(-41,104) (51,104)

(5,87)

-----------------------

(12.4785,766) (-5.3985,766)

(3.54,666.3) (3.54,726.623)

-----------------------

(7,55) (1,55)

(4,6) (4,54)

-----------------------

Общая площадь фигур - 16382.3

1. **Листинг программы**

/\*Воронов К.М.

Разработать шаблоны классов согласно варианту задания.

Параметром шаблона должен являться скалярный тип данных задающий тип данных для оси координат.

Классы должны иметь только публичные поля. В классах не должно быть методов, только поля.

Фигуры являются фигурами вращения (равнобедренными), за исключением трапеции и прямоугольника.

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

Вариант 27

Прямоугольник ромб трапеция

\*/

#include "iostream"

#include <vector>

#include <valarray>

#include <tuple>

using namespace std;

template <typename T>

class Pramougol{

public:

using type = T;

pair<T,T> t;

T a;

T b;

Pramougol(){};

Pramougol(T x,T y,T a1, T b1): t(x,y), a(a1), b(b1){};

};

template <typename T>

class Romb{

public:

using type = T;

pair<T,T> t;

T a;

T b;

Romb(T x,T y,T a1, T b1): t(x,y), a(a1), b(b1){};

};

template <typename T>

class Trapecia{

public:

using type = T;

pair<T,T> t;

T a;

T b;

T c;

Trapecia(T x,T y,T a1, T b1, T s1): t(x,y), a(a1), b(b1), c(s1){};

};

template <int i = 0, class... Ts>

void print1(tuple<Ts ...> t){

if constexpr (i < tuple\_size<decltype(t)>::value){

print2(get<i>(t));

cout << "-----------------------"<< endl;

print1< i + 1, Ts ... >(t);

}

else {

return;

}

}

template <int i = 0, class... Ts>

void square1(tuple<Ts ...> t,double pl){

if constexpr (i < tuple\_size<decltype(t)>::value){

pl+=square2(get<i>(t));

square1< i + 1, Ts ... >(t, pl);

}

else {

cout << "Общая площадь фигур - " << pl << endl;

return;

}

}

template <typename T>

T square2(Pramougol<T> p) {

return p.a\*p.b;

}

template <typename T>

T square2(Romb<T> p) {

return p.a\*p.b/2;

}

template <typename T>

T square2(Trapecia<T> p) {

double g = (p.a - p.b)/2;

g=sqrt(p.c\*p.c-g\*g);

return((p.a+p.b)\*g/2);

}

template <typename T>

void print2(Pramougol<T> p) {

pair<T,T> t2((p.t).first + p.a, (p.t).second );

pair<T,T> t3((p.t).first + p.a, (p.t).second + p.b);

pair<T,T> t4((p.t).first, (p.t).second + p.b);

cout << "(" << t4.first << "," << t4.second << ") "

<< "(" << t3.first << "," << t3.second << ")"

<< endl << endl << "(" << (p.t).first << ","

<< (p.t).second << ") " << "(" << t2.first << ","

<< t2.second << ")" << endl;

}

template <typename T>

void print2(Romb<T> p) {

pair<T,T> t2((p.t).first,(p.t).second+p.b);

pair<T,T> t3((p.t).first + p.a/2,(p.t).second+p.b/2);

pair<T,T> t4((p.t).first - p.a/2,(p.t).second+p.b/2);

cout << " (" << t2.first << "," << t2.second << ") " << endl

<< endl << "(" << t4.first << "," << t4.second << ") " << "("

<< t3.first << "," << t3.second << ")" << endl << endl << " ("

<< (p.t).first << "," << (p.t).second << ")" << endl;

}

template <typename T>

void print2(Trapecia<T> p) {

T g1 = (p.a - p.b)/2;

T g2=sqrt(p.c\*p.c-g1\*g1);

pair<T,T> t2((p.t).first,(p.t).second+p.b);

pair<T,T> t3((p.t).first + g1, (p.t).second + g2);

pair<T,T> t4(t2.first-g1,(p.t).second + g2);

cout << " (" << t3.first << "," << t3.second << ") ("

<< t4.first << "," << t4.second << ")" << endl << endl

<< "(" << (p.t).first << "," << (p.t).second << ") ("

<< t2.first << "," << t2.second << ")" << endl;

}

int main(){

printf("Введите координату нижней левой точки прямоугольника и длины сторон, прилегающих к ней, начиная с горизонтальной (тип double)\n");

double x,y,a,b,c;

cin >> x >> y >> a >> b;

if((!cin)||(a<=0)||(b<=0)){

printf("Введены неверные данные\n");

return 1;

}

Pramougol<double> p1(x,y,a,b);

printf("Введите координату нижней левой точки прямоугольника и длины сторон, прилегающих к ней, начиная с горизонтальной (тип int)\n");

int x1,y1,a1,b1,c1;

cin >> x1 >> y1 >> a1 >> b1;

if((!cin)||(a1<=0)||(b1<=0)){

printf("Введены неверные данные\n");

return 1;

}

Pramougol<int> p2(x1,y1,a1,b1);

printf("Введите координату нижней точки ромба и длину его диагоналей, начиная с горизонтальной (тип double)\n");

cin >> x >> y >> a >> b;

if((!cin)||(a<=0)||(b<=0)){

printf("Введены неверные данные\n");

return 1;

}

Romb<double> r1(x,y,a,b);

printf("Введите координату нижней точки ромба и длину его диагоналей, начиная с горизонтальной (тип int)\n");

cin >> x1 >> y1 >> a1 >> b1;

if((!cin)||(a1<=0)||(b1<=0)){

printf("Введены неверные данные\n");

return 1;

}

Romb<int> r2(x1,y1,a1,b1);

printf("Введите координату нижней левой точки трапеции, наибольшее основание, наименьшее основание и длину боковой стороны (тип double)\n");

cin>>x>>y>>a>>b>>c;

if(!cin){

printf("Введены неверные данные\n");

return 1;

}

double g1 = (a - b)/2;

double g2=sqrt(c\*c-g1\*g1);

if((a<=0)||(b<=0)||(c<0)||(g1+c<g2)||(g2+g1<c)||(g2+c<g1)||(g1<=0)||(g2<=0)){

printf("Введены неверные данные\n");

return 1;

}

Trapecia<double> t1(x,y,a,b,c);

printf("Введите координату нижней левой точки трапеции, наибольшее основание, наименьшее основание и длину боковой стороны (тип int)\n");

cin>>x1>>y1>>a1>>b1>>c1;

if(!cin){

printf("Введены неверные данные\n");

return 1;

}

int g11 = (a1 - b1)/2;

int g22=sqrt(c1\*c1-g11\*g11);

if((a1<=0)||(b1<=0)||(c1<=0)||(g11+c1<g22)||(g22+g11<c1)||(g22+c1<g11)||(g11<=0)||(g22<=0)){

printf("Введены неверные данные\n");

return 1;

}

Trapecia<int> t2(x1,y1,a1,b1,c1);

tuple<Pramougol<double>,Pramougol<int>,Romb<double>,Romb<int>,Trapecia<double>,Trapecia<int>> t{p1,p2,r1,r2,t1,t2};

print1(t);

square1(t,0);

}

1. **Вывод**

Я изучил основы работы с шаблонами в С++, включая std::pair, std::tuple.

**Список литературы**

1. Мануал std::tuple\_size<std::tuple> [Электронный ресурс]

URL:<https://en.cppreference.com/w/cpp/utility/tuple/tuple_size> (дата обращения 30.10.2020).

1. Мануал std::pair [Электронный ресурс]

URL:<https://en.cppreference.com/w/cpp/utility/pair> (дата обращения 30.10.2020).