**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: Хренникова Ангелина Сергеевна

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

**Содержание**

1. Постановка задачи
2. Описание программы
3. Набор тестов
4. Результаты выполнения тестов
5. Листинг программы
6. Вывод
7. Список литературы

**Постановка задачи**

Цель:

* Изучить основы работы с шаблонами (template) в С++;
* Изучить шаблоны std::pair, std::tuple;
* Получить навыки работы со специализацией шаблонов и идиомой SFINAE.

Задача: Разработать шаблоны классов: ромб, пятиугольник, шестиугольник. 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. 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.
2. Функция square вычисления суммарной площади фигур. Функция должна принимать на вход std::tuple с фигурами, согласно варианту задания.

**Описание программы**

В программе описаны шаблоны классов для ромба, пятиугольника и шестиугольника. Ромб задается координатами четырех точек, которые проверяются функцией на то, составляют они ромб или нет. Пятиугольник задается координатами двух точек: центром фигуры и одной из вершин, остальные координаты вычисляются программой. Шестиугольник задается точно так же.

Написана рекурсивная функция для печати кортежа, которая вызывает функции печати для каждой фигуры. Также написана рекурсивная функция для вычисления суммарной площади всех фигур, которая вызывает функции для вычисления площади каждой фигуры и суммирует их результаты.

Площадь ромба вычисляется как полупроизведение диагоналей, площади пятиугольника и шестиугольника вычисляются через радиус описанной окружности по формуле , где — количество углов, а — радиус описанной окружности(расстояние между центром и точкой фигуры).

В самой программе есть меню, по которому пользователь может либо ввести фигуры самостоятельно, либо использовать уже заданные.

**Набор тестов**

Тест 1: 2 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Индекс | Фигура | Вершины | Геометрический центр | Площадь |
| 1 | Ромб | а1(1, 0), а2(0, -1),  а3(-1, 0), а4(0, 1). | с(0, 0) | 2 |
| 2 | Пятиугольник | а1(0, 6), а2(-5.71, 1.85),  а3(-3.53, -4.85), а4(3.53, -4.85), а5(5.71, 1.85). | c(0, 0) | 85.59 |
| 3 | Шестиугольник | a1(1, 1),  а2(-5.46, -0.4),  а3(-4.25, -6),  а4(0.6, -4.94),  а5(-0.31, -0.75),  а6(-3.95, -1.54). | c(-2, -3) | 64.95 |
| 4 | Ромб | а1(1.725, 0),  а2(0, 1.725), а3(-1.725, 0), а4(0, -1.725). | c(0, 0) | 5.95 |
| 5 | Пятиугольник | а1(0.15, 0.15), а2(-0.08, 0.19),  а3(-0.18, -0.02), а4(-0.02, -0.18), а5(0.19, -0.08). | c(0.01, 0.01) | 0.09 |
| 6 | Шестиугольник | a1(14, -3.12), а2(-10.45, -3.86),  а3(11.09, -6.94),  а4(13.76, -6.37),  а5(13.27, -4.07),  а6(11.28, -4.49). | c(12.34, -5.3) | 19.51 |

Общая площадь: 178,098

Тест 2: 1 1 2 -2 -2 -3 -1 -6 -5 1 1 0 0 0 0 1.732 -1 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Индекс | Фигура | Вершины | Геометрический центр | Площадь |
| 1 | Ромб | а1(1, 2),  а2(-2, -2),  а3(-3, -1),  а4(-6, -5). | с(-2.5, -1.5) | 7 |
| 2 | Пятиугольник | а1(0, 0),  а2(1.64, -0.26),  а3(2.39, 1.22), а4(1.22, 2.39), а5(-0.26, 1.64). | c(1, 1) | 4.75 |
| 3 | Шестиугольник | a1(1.732, -1),  а2(0.87, 1.5),  а3(-1.3, 0.75),  а4(-0.65, -1.13),  а5(0.97, -0.56),  а6(0.49, 0.84). | c(0, 0) | 10.39 |

Общая площадь: 22.147

**Результаты выполнения тестов**

Тест 1:

Menu:

1. Enter figures.

2. Use a ready-made set.

3. Exit.

Select menu item: 2

Rhombus:

Coordinates:(1; 0), (0; -1), (-1; 0), (0; 1)

Center: (0, 0)

Area figure: 2

Pentagon:

Coordinates:(0; 6), (-5.70634; 1.8541), (-3.52671; -4.8541), (3.52671; -4.8541), (5.70634; 1.8541)

Center: (0, 0)

Area figure: 85.5951

Hexagon:

Coordinates: (1; 1), (-5.4641; -0.401924), (-4.25; -6), (0.598076; -4.94856), (-0.3125; -0.75), (-3.94856; -1.53858)

Center: (-2, -3)

Area figure: 64.9519

Rhombus:

Coordinates:(1.725; 0), (0; 1.725), (-1.725; 0), (0; -1.725)

Center: (0, 0)

Area figure: 5.95125

Pentagon:

Coordinates:(0.15; 0.15), (-0.0798855; 0.18641), (-0.185552; -0.0209724), (-0.0209724; -0.185552), (0.18641; -0.0798855)

Center: (0.01, 0.01)

Area figure: 0.0932035

Hexagon:

Coordinates: (14; -3.12), (10.4521; -3.8624), (11.095; -6.935), (13.756; -6.3782), (13.2737; -4.07375), (11.278; -4.49135)

Center: (12.34, -5.3)

Area figure: 19.5064

Total square: 178.097801

Select menu item: 3

Тест 2:

Menu:

1. Enter figures.

2. Use a ready-made set.

3. Exit.

Select menu item: 1

Enter the points of the rhombus: 1 2 -2 -2 -3 -1 -6 -5

Enter the points of the pentagon: 1 1 0 0

Enter the points of the hexagon: 0 0 1.732 -1

Rhombus:

Coordinates:(1; 2), (-2; -2), (-6; -5), (-3; -1)

Center: (-2.5, -1.5)

Area figure: 7

Pentagon:

Coordinates:(0; 0), (1.64204; -0.260074), (2.3968; 1.22123), (1.22123; 2.3968), (-0.260074; 1.64204)

Center: (1, 1)

Area figure: 4.75528

Hexagon:

Coordinates: (1.732; -1), (0.866025; 1.49996), (-1.299; 0.75), (-0.649519; -1.12497), (0.97425; -0.5625), (0.487139; 0.843725)

Center: (0, 0)

Area figure: 10.3918

Total square: 22.147130

Select menu item: 3

**Листинг программы**

#include <iostream>

#include <string>

#include <utility>

#include <tuple>

#include <algorithm>

template<class T>

struct Rhombus {

using type = T;

std::pair<T, T> a[4]; //координаты четырех точек

};

template<class T>

struct Pentagon {

using type = T;

std::pair<T, T> center; //координаты центра

std::pair<T, T> b; //координаты одной точки

};

template<class T>

struct Hexagon {

using type = T;

std::pair<T, T> center; //координаты центра

std::pair<T, T> c; //координаты одной точки

};

//печать кортежа

template<class T, int index>

typename std::enable\_if<index >= std::tuple\_size<T>::value, void>::type print\_tuple(T& tuple) {

std::cout << std::endl;

}

template<class T, int index>

typename std::enable\_if < index < std::tuple\_size<T>::value, void>::type print\_tuple(T& tuple) {

auto figure = std::get<index>(tuple);

print(figure);

std::cout << "Area figure: " << area(figure) << "\n";

std::cout << "\n";

print\_tuple<T, index + 1>(tuple);

}

//печать ромба

template<class T>

typename std::enable\_if<(sizeof(T::a) > 0), void>::type print(T& r) {

double xc = (r.a[0].first + r.a[1].first + r.a[2].first + r.a[3].first) / 4;

double yc = (r.a[0].second + r.a[1].second + r.a[2].second + r.a[3].second) / 4;

std::cout << "Rhombus:\n";

std::cout << "Coordinates:";

std::cout<< "(" << r.a[0].first << "; " << r.a[0].second << "), ("

<< r.a[1].first << "; " << r.a[1].second << "), ("

<< r.a[2].first << "; " << r.a[2].second << "), ("

<< r.a[3].first << "; " << r.a[3].second << ")\n";

std::cout << "Center: (" << xc << ", " << yc << ")\n";

}

//проверка ромба

template<class T>

typename std::enable\_if<(sizeof(T::a) > 0), T>::type check(T& r) {

double x1 = r.a[0].first;

double y1 = r.a[0].second;

double x2 = r.a[1].first;

double y2 = r.a[1].second;

double x3 = r.a[2].first;

double y3 = r.a[2].second;

double x4 = r.a[3].first;

double y4 = r.a[3].second;

if (sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2)) == sqrt(pow(x1 - x4, 2) + pow(y1 - y4, 2))

&& sqrt(pow(x3 - x4, 2) + pow(y3 - y4, 2)) == sqrt(pow(x2 - x3, 2) + pow(y2 - y3, 2))

&& sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2)) == sqrt(pow(x2 - x3, 2) + pow(y2 - y3, 2))) {

}

else if (sqrt(pow(x1 - x4, 2) + pow(y1 - y4, 2)) == sqrt(pow(x1 - x3, 2) + pow(y1 - y3, 2))

&& sqrt(pow(x2 - x3, 2) + pow(y2 - y3, 2)) == sqrt(pow(x2 - x4, 2) + pow(y2 - y4, 2))

&& sqrt(pow(x1 - x4, 2) + pow(y1 - y4, 2)) == sqrt(pow(x2 - x4, 2) + pow(y2 - y4, 2))) {

std::swap(r.a[1].first, r.a[2].first);

std::swap(r.a[1].second, r.a[2].second);

}

else if (sqrt(pow(x1 - x3, 2) + pow(y1 - y3, 2)) == sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2))

&& sqrt(pow(x2 - x4, 2) + pow(y2 - y4, 2)) == sqrt(pow(x3 - x4, 2) + pow(y3 - y4, 2))

&& sqrt(pow(x1 - x2, 2) + pow(y1 - y2, 2)) == sqrt(pow(x2 - x4, 2) + pow(y2 - y4, 2))) {

std::swap(r.a[2].first, r.a[3].first);

std::swap(r.a[2].second, r.a[3].second);

}

else {

std::cout << "Error!\n" << std::endl;

r.a[0].first = 0;

r.a[0].second = 0;

r.a[1].first = 0;

r.a[1].second = 0;

r.a[2].first = 0;

r.a[2].second = 0;

r.a[3].first = 0;

r.a[3].second = 0;

}

return r;

}

//печать пятиугольника

template<class T>

typename std::enable\_if<(sizeof(T::b) > 0), void>::type print(T& p) {

double c = (sqrt(5) - 1) / 4;

double s = (sqrt(10 + 2 \* sqrt(5))) / 4;

double x2 = p.center.first + (p.b.first - p.center.first) \* c - (p.b.second - p.center.second) \* s;

double y2 = p.center.second + (p.b.first - p.center.first) \* s + (p.b.second - p.center.second) \* c;

double x3 = p.center.first + (x2 - p.center.first) \* c - (y2 - p.center.second) \* s;

double y3 = p.center.second + (x2 - p.center.first) \* s + (y2 - p.center.second) \* c;

double x4 = p.center.first + (x3 - p.center.first) \* c - (y3 - p.center.second) \* s;

double y4 = p.center.second + (x3 - p.center.first) \* s + (y3 - p.center.second) \* c;

double x5 = p.center.first + (x4 - p.center.first) \* c - (y4 - p.center.second) \* s;

double y5 = p.center.second + (x4 - p.center.first) \* s + (y4 - p.center.second) \* c;

double xc = p.center.first;

double yc = p.center.second;

std::cout << "Pentagon:\n";

std::cout << "Coordinates:";

std::cout << "(" << p.b.first << "; " << p.b.second << "), ("

<< x2 << "; " << y2 << "), (" << x3 << "; " << y3 << "), ("

<< x4 << "; " << y4 << "), (" << x5 << "; " << y5 << ")\n";

std::cout << "Center: (" << xc << ", " << yc << ")\n";

}

//печать шестиугольника

template<class T>

typename std::enable\_if<(sizeof(T::c) > 0), void>::type print(T& h) {

double c = 1 / 2;

double s = sqrt(3) / 2;

double x2 = h.center.first + (h.c.first - h.center.first) \* c - (h.c.second - h.center.second) \* s;

double y2 = h.center.second + (h.c.first - h.center.first) \* s + (h.c.second - h.center.second) \* c;

double x3 = h.center.first + (x2 - h.center.first) \* c - (y2 - h.center.second) \* s;

double y3 = h.center.second + (x2 - h.center.first) \* s + (y2 - h.center.second) \* c;

double x4 = h.center.first + (x3 - h.center.first) \* c - (y3 - h.center.second) \* s;

double y4 = h.center.second + (x3 - h.center.first) \* s + (y3 - h.center.second) \* c;

double x5 = h.center.first + (x4 - h.center.first) \* c - (y4 - h.center.second) \* s;

double y5 = h.center.second + (x4 - h.center.first) \* s + (y4 - h.center.second) \* c;

double x6 = h.center.first + (x5 - h.center.first) \* c - (y5 - h.center.second) \* s;

double y6 = h.center.second + (x5 - h.center.first) \* s + (y5 - h.center.second) \* c;

double xc = h.center.first;

double yc = h.center.second;

std::cout << "Hexagon:\n";

std::cout << "Coordinates: ";

std::cout << "(" << h.c.first << "; " << h.c.second << "), ("

<< x2 << "; " << y2 << "), (" << x3 << "; " << y3 << "), ("

<< x4 << "; " << y4 << "), (" << x5 << "; " << y5 << "), ("

<< x6 << "; " << y6 << ")\n";

std::cout << "Center: (" << xc << ", " << yc << ")\n";

}

//вычисление площади всех фигур в кортеже

template<class T, int index>

typename std::enable\_if<index >= std::tuple\_size<T>::value, double>::type square(T& tuple) {

return 0;

}

template<class T, int index>

typename std::enable\_if < index < std::tuple\_size<T>::value, double>::type square(T& tuple) {

auto figure = std::get<index>(tuple);

double cur\_square = area(figure);

return cur\_square + square<T, index + 1>(tuple);

}

//вычисление площади ромба

template<class T>

typename std::enable\_if<(sizeof(T::a) > 0), typename T::type>::type area(T& r) {

return sqrt(pow(r.a[0].first - r.a[2].first, 2) + pow(r.a[0].second - r.a[2].second, 2)) \*

sqrt(pow(r.a[1].first - r.a[3].first, 2) + pow(r.a[1].second - r.a[3].second, 2)) / 2;

}

//вычисление площади пятиугольника

template<class T>

typename std::enable\_if<(sizeof(T::b) > 0), double>::type area(T& p) {

double s = (sqrt(10 + 2 \* sqrt(5))) / 4;

double pi = acos(-1);

double r = sqrt(pow(p.b.first - p.center.first, 2) + pow(p.b.second - p.center.second, 2));

//double side = r \* cos(13 \* pi / 10) - r \* cos(17 \* pi / 10);

//return sqrt(25 + 10 \* sqrt(5)) \* pow(side, 2) \* 0.25;

return 0.5 \* r \* r \* 5 \* s;

}

//вычисление площади шестиугольника

template<class T>

typename std::enable\_if<(sizeof(T::c) > 0), double>::type area(T& h) {

return pow(sqrt(pow(h.c.first - h.center.first, 2) + pow(h.c.second - h.center.second, 2)), 2) \* 3 \* sqrt(3) \* 0.5;

}

void menu() {

std::cout << "Menu: \n"; //Меню

std::cout << "1. Enter figures.\n"; //Ввести фигуры

std::cout << "2. Use a ready-made set.\n"; //Использовать готовые фигуры

std::cout << "3. Exit.\n"; //Завешить программу

}

int main() {

int k;

menu();

std::cout << "Select menu item: ";

while (std::cin >> k) {

if (k == 1) {

std::cout << "Enter the points of the rhombus: "; //ввод координат ромба

Rhombus<double> r3;

std::cin >> r3.a[0].first >> r3.a[0].second >> r3.a[1].first >> r3.a[1].second >>

r3.a[2].first >> r3.a[2].second >> r3.a[3].first >> r3.a[3].second;

check(r3); //проверка, является ли введенная фигура ромбом

std::cout << "\n";

std::cout << "Enter the points of the pentagon: "; //ввод координат пятиугольника

Pentagon<double> p3;

std::cin >> p3.center.first >> p3.center.second >> p3.b.first >> p3.b.second;

std::cout << "\n";

std::cout << "Enter the points of the hexagon: "; //ввод координат шестиугольника

Hexagon<double> h3;

std::cin >> h3.center.first >> h3.center.second >> h3.c.first >> h3.c.second;

std::cout << "\n";

std::tuple<decltype(r3), decltype(p3), decltype(h3)> //создание кортежа

tuple{ r3, p3, h3 };

print\_tuple<decltype(tuple), 0>(tuple); //печать кортежа

std::cout << std::fixed << "Total square: " << square<decltype(tuple), 0>(tuple); //вычисление площади всех фигур кортежа

}

else if (k == 2) { //использование готовых фигур

Rhombus<int> r1;

r1.a[0] = { 1, 0 };

r1.a[1] = { 0, -1 };

r1.a[2] = { 0, 1 };

r1.a[3] = { -1, 0 };

check(r1);

Rhombus<double> r2;

r2.a[0] = { 1.725, 0 };

r2.a[1] = { 0, 1.725 };

r2.a[2] = { -1.725, 0 };

r2.a[3] = { 0, -1.725 };

check(r2);

Pentagon<int> p1;

p1.center = { 0, 0 };

p1.b = { 0, 6 };

Pentagon<double> p2;

p2.center = { 0.01, 0.01 };

p2.b = { 0.15, 0.15 };

Hexagon<int> h1;

h1.center = { -2, -3 };

h1.c = { 1, 1 };

Hexagon<double> h2;

h2.center = { 12.34, -5.3 };

h2.c = { 14, -3.12 };

std::tuple<decltype(r1), decltype(p1), decltype(h1), decltype(r2), decltype(p2), decltype(h2)> //создание кортежа

tuple{ r1, p1, h1, r2, p2, h2 };

print\_tuple<decltype(tuple), 0>(tuple); //печать всех фигур кортежа

std::cout << std::fixed << "Total square: " << square<decltype(tuple), 0>(tuple); //вычисление площади всех фигур кортежа

}

else if (k == 3) {

break; //завершение работы

}

else std::cout << "Error!\n";

std::cout << "\nSelect menu item: ";

}

return 0;

}

**Вывод**

В результате данной лабораторной работы я научилась работать с шаблонами в С++, изучила такие шаблоны, как std::pair, std::tuple, получила навыки работы со специализацией шаблонов и идиомой SFINAE на основе работы с шаблонными классами для фигур: ромба, пятиугольника и шестиугольника.

**Список литературы**

1. CppStudio — Шаблоны классов в С++/Электронный ресурс/Режим доступа: <http://cppstudio.com/post/5188/> (дата обращения: 04.11.2020).
2. Хабр — Особенности реализации кортежей на С++/Электронный ресурс/Режим доступа: <https://habr.com/ru/post/179153/> (дата обращения: 04.11.2020).
3. Qosys — Кортежи std::tuple в C++/Электронный ресурс/Режим доступа: <https://qosys.info/tuple-cpp> (дата обращения: 04.11.2020).