**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: Калугин Кирилл Алексеевич

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 позволяет:

* Создает набор фигур согласно варианту задания (как минимум по одной фигуре каждого типа с координатами типа int и координатами типа double).
* Сохраняет фигуры в std::tuple
* Печатает на экран содержимое std::tuple с помощью шаблонной функции print.
* Вычисляет суммарную площадь фигур в std::tuple и выводит значение на экран.

Используемые фигуры: пяти-, шести- и восьмиугольник.

1. Описание программы

Программа создает три шаблонных класса фигур: пяти-, шести- и восьмиугольник, а также функции для их обработки, после чего создает tuple, в который вносит по две фигуры каждого вида (с целыми и дробными координатами). Взаимодействие с пользователем не требуется, все данные заданы заранее.

1. Набор тестов

Тестовые данные:

Пятиугольник (int) - координаты центра(0, 0), радиус 6.

Пятиугольник (double) - координаты центра(1, 1), радиус 2.12. Шестиугольник (int) - координаты центра(0, 0), радиус 8.

Шестиугольник (double) - координаты центра(1, 1), радиус 9.41.

Восьмиугольник (int) - координаты центра(0, 0), радиус 3.

Восьмиугольник (double) - координаты центра(1, 1), радиус 5.05.

A = (0,6)

B = (5.29769,2.81683)

C = (4.31604,-4.16795)

D = (-2.63023,-5.39276)

E = (-5.94161,0.835039)

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

A = (1,3.12)

B = (2.87185,1.99528)

C = (2.525,-0.472676)

D = (0.0706532,-0.905443)

E = (-1.09937,1.29505)

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

A = (0,8)

B = (6.9282,4)

C = (6.9282,-4)

D = (0,-8)

E = (-6.9282,-4)

F = (-6.9282,4)

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

A = (1,10.41)

B = (9.1493,5.705)

C = (9.1493,-3.705)

D = (1,-8.41)

E = (-7.1493,-3.705)

F = (-7.1493,5.705)

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

A = (0,3)

B = (2.12132,2.12132)

C = (3,0)

D = (2.12132,-2.12132)

E = (0,-3)

F = (-2.12132,-2.12132)

G = (-3,0)

H = (-2.12132,2.12132)

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

A = (1,6.05)

B = (4.57089,4.57089)

C = (6.05,1)

D = (4.57089,-2.57089)

E = (1,-4.05)

F = (-2.57089,-2.57089)

G = (-4.05,1)

H = (-2.57089,4.57089)

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

Общая площадь фигур - 588.873

1. Листинг программы

//Калугин Кирилл М8О-207Б-19

//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 позволяет:

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

//• Сохраняет фигуры в std::tuple

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

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

//Используемые фигуры: пяти-, шести- и восьмиугольник.

#include <iostream>

#include <vector>

#include <cmath>

#include <tuple>

using namespace std;

const double PI = 3.141592653589793238462643383279502884197169399375105820974944;

template <typename T>//шаблон класса пятиугольника

class five {

public:

using type = T;

pair <T, T> O;

T R;

five (T x, T y, T r): O (x, y), R (r) {};

};

template <typename T>//шаблон класса шестиугольника

class six {

public:

using type = T;

pair <T, T> O;

T R;

six (T x, T y, T r): O (x, y), R (r) {};

};

template <typename T>//шаблон класса восьмиугольника

class eight {

public:

using type = T;

pair <T, T> O;

T R;

eight (T x, T y, T r): O (x, y), R (r) {};

};

template <int i = 0, class... Ts>//шаблон функции вывода вершин фигур

void points1 (tuple <Ts ...> t){

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

points2(get <i> (t));

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

points1 < i + 1, Ts ... > (t);

}

}

template <int i = 0, class... Ts>//шаблон функции вывода общей площади

void s1 (tuple <Ts ...> t, double S){

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

S += s2 (get <i> (t));

s1 < i + 1, Ts ... > (t, S);

}

else {

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

}

}

template <typename T>//шаблон функции поиска площади пятиугольника

T s2 (five <T> x) {

return (2.5 \* pow (x.R, 2) \* sin (72.0 / 180.0 \* PI));

}

template <typename T>//шаблон функции поиска площади шестиугольника

T s2 (six <T> x) {

return (1.5 \* sqrt (3) \* pow (x.R, 2));

}

template <typename T>//шаблон функции поиска площади восьмиугольника

T s2 (eight <T> x) {

return (4 \* pow (x.R, 2) \* sin (PI / 4));

}

template <typename T>//шаблон функции поиска вершин пятиугольника

void points2 (five <T> x) {

cout << "\nA = (" << (x.O).first << "," << (x.O).second + x.R << ")\n" << "B = ("

<< (x.O).first + x.R \* cos (28.0 / 180.0 \* PI) << "," << (x.O).second + x.R \* sin (28.0 / 180.0 \* PI)

<< ")\n" << "C = (" << (x.O).first + x.R \* cos (- 44.0 / 180.0 \* PI) << ","

<< (x.O).second + x.R \* sin (- 44.0 / 180.0 \* PI) << ")\n" << "D = ("

<< (x.O).first + x.R \* cos (- 116.0 / 180.0 \* PI) << "," << (x.O).second + x.R \* sin (- 116.0 / 180.0 \* PI)

<< ")\n" << "E = (" << (x.O).first + x.R \* cos (- 188.0 / 180.0 \* PI)

<< "," << (x.O).second + x.R \* sin (- 188.0 / 180.0 \* PI) << ")\n";

}

template <typename T>//шаблон функции поиска вершин шестиугольника

void points2 (six <T> x) {

cout << "\nA = (" << (x.O).first << "," << (x.O).second + x.R << ")\n" << "B = ("

<< (x.O).first + x.R \* cos (PI / 6.0) << "," << (x.O).second + x.R \* sin (PI / 6.0)

<< ")\n" << "C = (" << (x.O).first + x.R \* cos (- PI / 6.0) << ","

<< (x.O).second + x.R \* sin (- PI / 6.0) << ")\n" << "D = (" << (x.O).first << ","

<< (x.O).second - x.R << ")\n" << "E = (" << (x.O).first + x.R \* cos (- 5.0 \* PI / 6.0)

<< "," << (x.O).second + x.R \* sin (- 5.0 \* PI / 6.0) << ")\n" << "F = ("

<< (x.O).first + x.R \* cos (- 7.0 \* PI / 6.0) << ","

<< (x.O).second + x.R \* sin (- 7.0 \* PI / 6.0) << ")\n";

}

template <typename T>//шаблон функции поиска вершин восьмиугольника

void points2 (eight <T> x) {

cout << "\nA = (" << (x.O).first << "," << (x.O).second + x.R << ")\n" << "B = ("

<< (x.O).first + x.R \* cos (PI / 4) << "," << (x.O).second + x.R \* sin (PI / 4)

<< ")\n" << "C = (" << (x.O).first + x.R << "," << (x.O).second << ")\n"

<< "D = (" << (x.O).first + x.R \* cos (- PI / 4) << "," << (x.O).second + x.R \* sin (- PI / 4)

<< ")\n" << "E = (" << (x.O).first << "," << (x.O).second - x.R << ")\n"

<< "F = (" << (x.O).first + x.R \* cos (- 3 \* PI / 4) << "," << (x.O).second + x.R \* sin (- 3 \* PI / 4)

<< ")\n" << "G = (" << (x.O).first - x.R << "," << (x.O).second << ")\n" << "H = ("

<< (x.O).first + x.R \* cos (- 5 \* PI / 4) << "," << (x.O).second + x.R \* sin (- 5 \* PI / 4) << ") \n";

}

int main()

{

five <int> f1 (0, 0, 6);//создание пятиугольника с заданными параметрами

five <double> f2 (1.0, 1.0, 2.12);//создание пятиугольника с заданными параметрами

six <int> si1 (0, 0, 8);//создание шестиугольника с заданными параметрами

six <double> si2 (1.0, 1.0, 9.41);//создание шестиугольника с заданными параметрами

eight <int> e1 (0, 0, 3);//создание восьмиугольника с заданными параметрами

eight <double> e2 (1.0, 1.0, 5.05);//создание восьмиугольника с заданными параметрами

tuple <five <int>, five <double>, six <int>, six <double>, eight <int>, eight <double> > t {f1, f2, si1, si2, e1, e2};//создание tuple'а фигур

points1 (t);//вызов функции вывода вершин

s1 (t, 0);///вызов функции подсчета общей площади

return 0;

}

1. Вывод

Шаблоны - это мощный инструмент языка C++, позволяющий создавать универсальные структуры, например функции или классы, работающие сразу с большим количеством разнотипных данных.

1. Литература

# Основы шаблонов С++: шаблоны функций[Электронный ресурс]

URL:https://habr.com/ru/post/436880/ дата обращения (30.10.2020)

# Шаблоны C++: шаблоны функций[Электронный ресурс]

URL:https://ru.wikipedia.org/wiki/Шаблоны\_C%2B%2B дата обращения (30.10.2020)