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: Егорова Анна
(староста)
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).

Фигуры: восьмиугольник, шестиугольник, пятиугольник.

2. Описание программы

Программа состоит из главного файла main.cpp, CmakeLists.txt – утилиты, которая собирает все файлы и библиотеки Figure.hpp, в которой находятся методы, связанные с построением и вычислением всех величин фигур из моего варианта.
На вход подаются координаты точки (x, y) и радиус.

3. Набор testcases

Тест 1:

Пятиугольник: 2 2 4 class

Шестиугольник: 3 4 5 tuple

Восьмиугольник: 4 5 4 class

Тест 2:

Пятиугольник: 5 5 7 tuple

Шестиугольник: 7 6 5 tuple

Восьмиугольник: 3 10 4 class

4. Результаты выполнения тестов.

Тест 1:

```
anyaegorovaa@LAPTOP-9PIFT2VJ:~/2kurs/oop$ ./main
Press pe to show information about pentagon
Press h to show information about hexagon
Press o to show information about octagon
Then enter tuple to use std::tuple<> or class to use class
pe
Coordinates of center point :
2 2 4
Radius :
tuple or class
class
Pentagon :
Center point (2.000,2.000)
Area : 76.085
Coordinates :
(6.000,2.000)
(3.236,5.804)
(-1.236,4.351)
(-1.236,-0.351)
(3.236,-1.804)
h
Coordinates of center point :
3 4 5
```

Radius :

tuple or class

tuple

Hexagon :

Center point (3.000,4.000)

Area : 64.952

Coordinates :

(8.000,4.000)

(5.500,8.330)

(0.500,8.330)

(-2.000,4.000)

(0.500,-0.330)

(5.500,-0.330)

o

Coordinates of center point :

4 5 4

Radius :

tuple or class

class

Octagon :

Center point (4.000,5.000)

Area : 45.255

Coordinates :

(8.000,5.000)

(6.828,7.828)

(4.000,9.000)

(1.172,7.828)

(0.000,5.000)

(1.172,2.172)

(4.000,1.000)

(6.828,2.172)

Тест 2:

anyaegorovaa@LAPTOP-9PIFT2VJ:~/2kurs/oop$ ./main

Press pe to show information about pentagon

Press h to show information about hexagon

Press o to show information about octagon

Then enter tuple to use std::tuple<> or class to use class

pe

Coordinates of center point :

5 5 7

Radius :

tuple or class

tuple

Pentagon :

Center point (5.000,5.000)

Area : 233.009

Coordinates :

(12.000,5.000)

(7.163,11.657)

(-0.663,9.114)

(-0.663,0.886)

(7.163,-1.657)

h

Coordinates of center point :

7 6 5

Radius :

tuple or class

tuple

Hexagon :

Center point (7.000,6.000)

Area : 64.952

Coordinates :

(12.000,6.000)

(9.500,10.330)

(4.500,10.330)

(2.000,6.000)

(4.500,1.670)

(9.500,1.670)

o

Coordinates of center point :

3 10 4

Radius :

tuple or class

class

Octagon :

Center point (3.000,10.000)

Area : 45.255

Coordinates :

(7.000,10.000)

(5.828,12.828)

(3.000,14.000)

(0.172,12.828)

(-1.000,10.000)

(0.172,7.172)

(3.000,6.000)

(5.828,7.172)

5. Листинг программы

MAIN.CPP

```
anyaegorovaa@LAPTOP-9PIFT2VJ:~/2kurs/oop$ cat main.cpp
/* Егорова Анна Владимировна М8о-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;
Вариант 6:
пятиугольник, шестиугольник, восьмиугольник.
*/

#include "Figures.hpp"


void help() {
   std::cout << "Press pe to show information about pentagon" << std::endl;
   std::cout << "Press h to show information about hexagon" << std::endl;
   std::cout << "Press o to show information about octagon" << std::endl;
   std::cout << "Then enter tuple to use std::tuple<> or class to use class" << std::endl;
}


int main() {
   help();
   using point = std::pair<double, double>;
      using std::make_pair;
   std::cout << std::fixed;
   std::cout.precision(3);
   std::string data, s;
   while(std::cin >> data) {
      if (data == "o") {
         double x1, y1, R;
         std::cout << "Coordinates of center point :" << std::endl;
```

```
    std::cin >> x1 >> y1;
    std::cout << "Radius :" << std::endl;
    std::cin >> R;
    if (check(R) == true) {
        std::cout << "tuple or class" << std::endl;
        std::cin >> s;
        if (s == "tuple") {
            std::tuple<point>  octagon(make_pair(x1, y1));
            information2(octagon, R);
        } else if (s == "class") {
            Octagon <double> octagon(x1, y1, R);
            information2(octagon);
        }
    }
} else if(data == "h") {
    double x1, y1, R;
    std::cout << "Coordinates of center point :" << std::endl;
    std::cin >> x1 >> y1;
    std::cout << "Radius :" << std::endl;
    std::cin >> R;
    if (check(R) == true) {
        std::cout << "tuple or class" << std::endl;
        std::cin >> s;
        if (s == "tuple") {
            std::tuple<point>  hexagon(make_pair(x1, y1));
            information(hexagon, R);
        } else if (s == "class") {
            Hexagon <double> hexagon(x1, y1, R);
            information(hexagon);
        }
    }
} else if (data == "pe") {
    double x1, y1, R;
    std::cout << "Coordinates of center point :" << std::endl;
    std::cin >> x1 >> y1;
    std::cout << "Radius :" << std::endl;
    std::cin >> R;
    if (check(R) == true) {
        std::cout << "tuple or class" << std::endl;
        std::cin >> s;
        if (s == "tuple") {
            std::tuple<point>  pentagon(make_pair(x1, y1));
```

```
                information3(pentagon, R);
            } else if (s == "class") {
                Pentagon <double> pentagon(x1, y1, R);
                information3(pentagon);
            }
        }
    } else if(data == "exit") {
        return 0;
    }
  }
  return 0;
}
```

CmakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
project(oop)
add_executable(main main.cpp)

set_target_properties(
  main PROPERTIES
    COMPILE_OPTIONS "-g;-Wall;-Wextra;-Wpedantic;"
)
```

Figures.hpp

```
anyaegorovaa@LAPTOP-9PIFT2VJ:~/2kurs/oop$ cat Figures.hpp
#ifndef FIGURES_H
#define FIGURES_H
#include <iostream>
#include <tuple>
#include <cmath>


template <class T>
struct Pentagon {
  using type = T;
  using vertex_t = std::pair<T,T>;
  vertex_t a;T R;
  Pentagon(T x1, T y1, T R) : a(x1, y1), R(R)
  {}
```

```
};

template <class T>
struct Hexagon {
   using type = T;
   using vertex_t = std::pair<T,T>;
   vertex_t a; T R;
   Hexagon(T x1, T y1, T R) : a(x1, y1), R(R)
   {}
};

template <class T>
struct Octagon {
   using type = T;
   using vertex_t = std::pair<T,T>;
   vertex_t a;T R;
   Octagon(T x1, T y1, T R) : a(x1, y1), R(R)
   {}
};

template <class T>
bool check(T R) {
   if(R > 0) {
      return true;
   } else {
      std::cout << "Error" << std::endl;
      return false;
   }
}

template <template <class> class F, class T>
typename std::enable_if< std::is_same< F<T>, Hexagon<T> >::value, F<T> >::type
information(F<T> h) {
   std::cout << "Hexagon :" << std::endl;
   std::cout << "Center point (" << h.a.first << "," << h.a.second << ")" << std::endl;
   T sq, x, y, angle;
   sq = 3 * sin(M_PI / 3) * h.R * h.R;
   std::cout << "Area : " << sq << std::endl;
   std::cout << "Coordinates :" << std::endl;
   for(int i = 0; i < 6; i++) {
      angle =  M_PI * i / 3;
      x = h.R * cos(angle) + h.a.first;
```

```
        y = h.R * sin(angle) + h.a.second;
        std::cout << "(" << x << "," << y << ")" << std::endl;
    }
    return h;
}

template <template <class> class F, class T>
typename std::enable_if< std::is_same< F<T>, Octagon<T> >::value, F<T> >::type
information2(F<T> o) {
    std::cout << "Octagon :" << std::endl;
    std::cout << "Center point (" << o.a.first << "," << o.a.second << ")" << std::endl;
    T sq,x,y,angle;
    sq = 4 * sin(M_PI / 4) * o.R * o.R;
    std::cout << "Area : " << sq << std::endl;
    std::cout << "Coordinates :" << std::endl;
    for(int i = 0; i < 8; i++) {
        angle =  M_PI * i / 4;
        x = o.R * cos(angle) + o.a.first;
        y = o.R * sin(angle) + o.a.second;
        std::cout << "(" << x << "," << y << ")" << std::endl;
    }
    return o;
}

template <template <class> class F, class T>
typename std::enable_if< std::is_same< F<T>, Pentagon<T> >::value, F<T> >::type
information3(F<T> pe) {
    std::cout << "Pentagon :" << std::endl;
    std::cout << "Center point (" << pe.a.first << "," << pe.a.second << ")" << std::endl;
    T sq, x, y, angle;
    sq = 5 * sin(2*M_PI / 5) * pe.R * pe.R;
    std::cout << "Area : " << sq << std::endl;
    std::cout << "Coordinates :" << std::endl;
    for(int i = 0; i < 5; i++) {
        angle =  2*M_PI * i / 5;
        x = pe.R * cos(angle) + pe.a.first;
        y = pe.R * sin(angle) + pe.a.second;
        std::cout << "(" << x << "," << y << ")" << std::endl;
    }
    return pe;
}
```

```
template <class T>
void information(std::tuple<std::pair<T,T>> Hexagon,T R) {
   std::cout << "Hexagon :" << std::endl;
   T sq, hx, hy;
         std::cout << "Center point (" << std::get<0>(Hexagon).first << "," << std::get<0>(Hexagon).second << ")" << std::endl;
   sq = 3 * sin(M_PI / 3) * R * R;
   std::cout << "Area : " << sq << std::endl;
   T angle;
   std::cout << "Coordinates :" << std::endl;
   for(int i = 0; i < 6; i++) {
      angle =  M_PI * i / 3;
      hx = R * cos(angle) + std::get<0>(Hexagon).first;
      hy = R * sin(angle) + std::get<0>(Hexagon).second;
      std::cout << "(" << hx << "," << hy << ")" << std::endl;
   }
}

template <class T>
void information2(std::tuple<std::pair<T,T>> Octagon,T R) {
   std::cout << "Octagon :" << std::endl;
   T sq, ox, oy;
         std::cout << "Center point (" << std::get<0>(Octagon).first << "," << std::get<0>(Octagon).second << ")" << std::endl;
   sq = 4 * sin(M_PI / 4) * R * R;
   std::cout << "Area : " << sq << std::endl;
   T angle;
   std::cout << "Coordinates :" << std::endl;
   for(int i = 0; i < 8; i++) {
      angle =  M_PI * i / 4;
      ox = R * cos(angle) + std::get<0>(Octagon).first;
      oy = R * sin(angle) + std::get<0>(Octagon).second;
      std::cout << "(" << ox << "," << oy << ")" << std::endl;
   }
}

template <class T>
void information3(std::tuple<std::pair<T,T>> Pentagon,T R) {
   std::cout << "Pentagon :" << std::endl;
   T sq, pex, pey;
         std::cout << "Center point (" << std::get<0>(Pentagon).first << "," << std::get<0>(Pentagon).second << ")" << std::endl;
```

```
    sq = 5 * sin(2*M_PI / 5) * R * R;
    std::cout << "Area : " << sq << std::endl;
    T angle;
    std::cout << "Coordinates :" << std::endl;
    for(int i = 0; i < 5; i++) {
        angle = 2*M_PI * i / 5;
        pex = R * cos(angle) + std::get<0>(Pentagon).first;
        pey = R * sin(angle) + std::get<0>(Pentagon).second;
        std::cout << "(" << pex << "," << pey << ")" << std::endl;
    }
}

#endif
```

6. Вывод

Шаблоны — средство языка C++, предназначенное для кодирования обобщенных алгоритмов, без привязки к некоторым параметрам (например, типам данных, размерам буферов, значениям по умолчанию). В C++ возможно создание шаблонов функций и классов.
Шаблоны позволяют создавать параметризованные классы и функции. Параметром может быть любой тип или значение одного из допустимых типов (целое число, enum, указатель на любой объект с глобально доступным именем, ссылка).
Шаблон функции начинается с ключевого слова template, за которым в угловых скобках следует список параметров. Затем следует объявление функции:
Хотя шаблоны предоставляют краткую форму записи участка кода, на самом деле их использование не сокращает исполняемый код, так как для каждого набора параметров компилятор создаёт отдельный экземпляр функции или класса. Как следствие, исчезает возможность совместного использования скомпилированного кода в рамках разделяемых библиотек.

7. Список литературы.

1. Шаблоны в C++ [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D1%8B_C%2B%2B

2. Основы шаблонов в C++ [Электронный ресурс]. URL: https://habr.com/ru/post/436880/