**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: Тимофеев Алексей Владимирович

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи.

ВАРИАНТ 14

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.

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).

Фигуры по варианту: 5-угольник, 6-угольник, 8-угольник.

1. Описание программы

В программе реализованы три шаблонных класса, в каждом из которых есть одно публичное поле - представленное вектором, элементами которого координаты вершин фигур. Для хранения координат фигур в векторе используется шаблон std::pair.В программе std::tuple содержит фигуры либо double, либо int на выбор.На вход функции печати - print подается std::tuple с фигурами, далее рекурсивно печатается каждое поле, перед печатью координат, используется typeid из библиотеки <typeidinfo>, для печати на экран типа элемента из tuple. Для печати координат перегружен оператор << . Функция square принимает std::tuple с фигурами, далее с помощью вспомогательной функции SFig, которая считает площадь отдельной фигуры, считается рекурсивно суммарная площадь фигур.

1. Набор тестов

double

Octagon

Coordinates of center point :1 1 Radius :4

Hexagon

Coordinates of center point :2 1 Radius :2

Pentagon

Coordinates of center point :4 1 Radius :1

int

Octagon

Coordinates of center point : 3 2 Radius : 5

Hexagon

Coordinates of center point : -2 3 Radius : 4

Pentagon

Coordinates of center point : 4 -3 Radius : 3

1. Результаты выполнения тестов

double or int

double

Octagon

Coordinates of center point :1 1

Radius :4

Hexagon

Coordinates of center point :2 1

Radius :2

Pentagon

Coordinates of center point :4 1

Radius :1

Pentagon: (1,4.87115) (0.508978,4.5178) (0.144496,4.03101) (0.000480776,3.53622)

(0.114047,0.00195304)

Hexagon: (1,3.62836) (2.16122,2.65156) (2.89089,1.43262) (2.91783,0.424534)

(2.23203,0.00195304) (1.08836,4.75206)

Octagon: (1,-0.792294) (4.57599,-1.39384) (-2.20461,4.93753) (0.295816,-0.134764)

(4.83566,-1.92061) (-1.73313,4.75206) (-0.386372,0.558211) (4.97553,1)

Square: 71.9924

dude@DESKTOP-9IO9OQQ:/mnt/d/Ycheba2kurs/OOp/OOPlab/nelaba4$ ./a.out

double or int

int

Octagon

Coordinates of center point : 3 2

Radius : 5

Hexagon

Coordinates of center point : -2 3

Radius : 4

Pentagon

Coordinates of center point : 4 -3

Radius : 3

Pentagon: (-3,6) (-4,5) (-5,4) (-5,2)

(-5,-5)

Hexagon: (3,1) (5,0) (6,-3) (6,-5)

(5,-5) (3,7)

Octagon: (2,0) (6,0) (-2,7) (1,1)

(6,0) (-1,7) (0,2) (6,998768500)

Square: 104.009

1. Листинг программы

#include <iostream>

#include <typeinfo>// Тимофеев Алексей М8О-207Б-19

#include <tuple>

#include <string>

#include <cmath>

#include <vector>

template<class T>

struct Pentagon {

std::vector<std::pair<T, T>> points;

};

template<class T>

struct Hexagon {

std::vector<std::pair<T, T>> points;

};

template<class T>

struct Octagon {

std::vector<std::pair<T, T>> points;

};

template<class T>

std::ostream &operator<<(std::ostream &os, const std::pair<T, T> &t) {

os << " (" << t.first << ',' << t.second << ") ";

return os;

}

template<size\_t ind = 0, class T>

void print(const T &t) {

if constexpr(ind < std::tuple\_size<T>::value) {

std::string s = typeid(std::get<ind>(t)).name();

s.erase(0, 1);

s.erase(s.length() - 3, 3);

std::cout << s << ": ";

for (int i = 0; i < std::get<ind>(t).points.size(); ++i){

std::cout << std::get<ind>(t).points[i];

if ((i + 1) % 4 == 0 && i != std::get<ind>(t).points.size() - 1) {

std::cout << std::endl << "\t ";

}

}

std::cout << std::endl << std::endl;

print<ind + 1, T>(t);

}

return;

}

template<class T>

long double SFig(const std::vector<std::pair<T, T>> &fig, std::string nametype) {

long double S = 0;

long double side = sqrt(pow(fig[0].first - fig[1].first, 2) + pow(fig[0].second - fig[1].second, 2));

if(nametype == "Pentagon") {

S = (side \* side \* sqrt(5) \* sqrt(5 + sqrt(20)));

}

if(nametype == "Hexagon") {

S = 3 \* sqrt(3) \* side \* side / 2;

}

if(nametype == "Octagon") {

S = 2 \* side \* side \* (1 + sqrt(2));

}

return S;

}

template<size\_t ind = 0, class T>

long double square(const T &t) {

long double S = 0;

if constexpr(ind < std::tuple\_size<T>::value) {

std::string name = typeid(std::get<ind>(t)).name();

name.erase(0, 1);

name.erase(name.length() - 3, 3);

S = SFig(std::get<ind>(t).points, name);

S += square<ind + 1, T>(t);

}

return S;

}

int main() {

std::cout << "double or int" << std::endl;

std::string data;

std::cin >> data;

if (data == "double"){

using pair\_double = std::pair<double, double>;

double points[16];// массив точек

double x1, y1, R, angle; // коорд ц х, коорд центра у, радиус, угол

std::cout << "Octagon" << std::endl;

std::cout << "Coordinates of center point :" << std::endl;

std::cin >> x1 >> y1;

std::cout << "Radius :" << std::endl;

std::cin >> R;

if (R <= 0.0) {

std::cout << "The entered values are not correct" << std::endl;

return 0;

}

for(int i = 0; i < 16; i = i = i+2) {

angle = 360 / 8;

points[i] = R \* cos(i\*angle) + x1;

points[i+1] = R \* sin(i\*angle) + y1;;

}

Octagon<double> oct1 = {{pair\_double(points[1],points[2]), pair\_double(points[3],points[4]), pair\_double(points[5],points[6]), pair\_double(points[7],points[8]), pair\_double(points[9],points[10]), pair\_double(points[11],points[12]), pair\_double(points[13],points[14]), pair\_double(points[15],points[16])}};

std::cout << "Hexagon" << std::endl;

std::cout << "Coordinates of center point :" << std::endl;

std::cin >> x1 >> y1;

std::cout << "Radius :" << std::endl;

std::cin >> R;

if (R <= 0.0) {

std::cout << "The entered values are not correct" << std::endl;

return 0;

}

for(int i = 0; i < 12; i = i+2) {

angle = 360 / 6;

points[i] = R \* cos(i\*angle) + x1;

points[i+1] = R \* sin(i\*angle) + y1;;

}

Hexagon<double> hex1 = {{pair\_double(points[1],points[2]), pair\_double(points[3],points[4]), pair\_double(points[5],points[6]), pair\_double(points[7],points[8]), pair\_double(points[9],points[10]), pair\_double(points[11],points[12])}};

std::cout << "Pentagon" << std::endl;

std::cout << "Coordinates of center point :" << std::endl;

std::cin >> x1 >> y1;

std::cout << "Radius :" << std::endl;

std::cin >> R;

if (R <= 0.0) {

std::cout << "The entered values are not correct" << std::endl;

return 0;

}

for(int i = 0; i < 10; i = i+2) {

angle = 360 / 5;

points[i] = R \* cos(i\*angle) + x1;

points[i+1] = R \* sin(i\*angle) + y1;;

}

Pentagon<double> pen1 = {{pair\_double(points[1],points[2]), pair\_double(points[3],points[4]), pair\_double(points[5],points[6]), pair\_double(points[7],points[8]), pair\_double(points[9],points[10])}};

std::tuple<Pentagon<double>, Hexagon<double>, Octagon<double>> t1(pen1, hex1, oct1);

print(t1);

std:: cout << "Square: " << square(t1) << std::endl;;

}else if (data == "int"){

using pair\_int = std::pair<int, int>;

int x1, y1, R, angle;// коорд ц х, коорд центра у, радиус, угол

int points[16];// массив точек

std::cout << "Octagon" << std::endl;

std::cout << "Coordinates of center point :" << std::endl;

std::cin >> x1 >> y1;

std::cout << "Radius :" << std::endl;

std::cin >> R;

if (R <= 0) {

std::cout << "The entered values are not correct" << std::endl;

return 0;

}

for(int i = 0; i < 16; i = i+2) {

angle = 360 / 8;

points[i] = R \* cos(i\*angle) + x1;

points[i+1] = R \* sin(i\*angle) + y1;

}

Octagon<int> oct2 = {{pair\_int(points[1],points[2]), pair\_int(points[3],points[4]), pair\_int(points[5],points[6]), pair\_int(points[7],points[8]), pair\_int(points[9],points[10]), pair\_int(points[11],points[12]), pair\_int(points[13],points[14]), pair\_int(points[15],points[16])}};

std::cout << "Hexagon" << std::endl;

std::cout << "Coordinates of center point :" << std::endl;

std::cin >> x1 >> y1;

std::cout << "Radius :" << std::endl;

std::cin >> R;

if (R <= 0) {

std::cout << "The entered values are not correct" << std::endl;

return 0;

}

for(int i = 0; i < 12; i = i+2) {

angle = 360 / 6;

points[i] = R \* cos(i\*angle) + x1;

points[i+1] = R \* sin(i\*angle) + y1;

}

Hexagon<int> hex2 = {{pair\_int(points[1],points[2]), pair\_int(points[3],points[4]), pair\_int(points[5],points[6]), pair\_int(points[7],points[8]), pair\_int(points[9],points[10]), pair\_int(points[11],points[12])}};

std::cout << "Pentagon" << std::endl;

std::cout << "Coordinates of center point :" << std::endl;

std::cin >> x1 >> y1;

std::cout << "Radius :" << std::endl;

std::cin >> R;

if (R <= 0) {

std::cout << "The entered values are not correct" << std::endl;

return 0;

}

for(int i = 0; i < 10; i = i+2) {

angle = 360 / 5;

points[i] = R \* cos(i\*angle) + x1;

points[i+1] = R \* sin(i\*angle) + y1;

}

Pentagon<int> pen2 = {{pair\_int(points[1],points[2]), pair\_int(points[3],points[4]), pair\_int(points[5],points[6]), pair\_int(points[7],points[8]), pair\_int(points[9],points[10])}};

std::tuple<Pentagon<int>, Hexagon<int>, Octagon<int>> t1(pen2, hex2, oct2);

print(t1);

std:: cout << "Square: " << square(t1) << std::endl;

}else{

std:: cout << "Wrong type " << std::endl;

return 0;

}

Список литературы

1. Презентация — “Введение в метапрограммирование ЛЕКЦИЯ №6” Д. В. Дзюба (дата обращения: 11.11.20)
2. Основы шаблонов С++: шаблоны функций [Электронный ресурс]. URL: <https://habr.com/ru/post/436880/> (дата обращения: 11.11.20)
3. Работа с кортежами С++ (std::tuple). Функции foreach, map и call. [Электронный ресурс]. URL: <https://habr.com/ru/post/318236/> (дата обращения: 11.11.20)