**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: Мариничев Иван Александрович

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата: 09.11.20

Оценка:

Москва, 2020

1. Постановка задачи. Вариант 22

Разработать шаблоны классов **5-угольник**, **6-угольник** и **8-угольник**. 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 (выпуклые, равносторонние), 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.

При реализации шаблонных функций допускается использование вспомогательных шаблонов std::enable\_if, std::tuple\_size, std::is\_same.

1. Описание программы

Программа предназначена для работы с тремя типами равносторонних фигур: пятиугольник, шестиугольник и восьмиугольник. Программа может вычислить площадь фигуры, а также вывести координаты центра и всех вершин. Все фигуры имеют разные способы задания:

* пятиугольник через координаты центра и радиус описанной окружности,
* шестиугольник через сторону и координаты центра,
* восьмиугольник через координаты центра и координаты любой точки.

В программе предусмотрена возможность введения этих начальных данных пользователем, так как же как и использование предустановленных данных для демонстрации всех возможностей программы. Для этого был реализован пользовательский интерфейс:

1. Ввести фигуры вручную;
2. Использовать готовые фигуры;
3. Завершить работу программы;

Пользователь вводит цифру соответствующую пункту меню.

* При добавлении фигуры нужно ввести указанные выше начальные данные

Когда в программе в программе произошло создание определенного набора фигур (пользовательских или предустановленных), из этого набора фигур создается кортеж (tuple). Именно его содержимое выводиться на экран. Также программа подсчитывает сумму всех площадей в кортеже и выводит ее в качестве суммарной площади всех фигур.

1. Руководство по использованию программы

Таблица 1 – Функции файла main.cpp

|  |  |  |
| --- | --- | --- |
| Название | Аргументы | Описание |
| struct Pentagon | std::pair<T, T> center;  double radius; | Шаблонная структура для задания правильного пятиугольника |
| struct Hexagon | std::pair<T, T> center;  double side; | Шаблонная структура для задания правильного шестиугольника |
| struct Octagon | std::pair<T, T> center;  std::pair<T, T> point; | Шаблонная структура для задания правильного восьмиугольника |
| void print\_tuple() | T& tuple  int index;  std::tuple\_size<T>::value  auto figure | Шаблонная функция для печати кортежа, предусматривающая проверку через std::enable\_if |
| void print() | T &p  T::radius  double pi  double angle | Шаблонная функция для печати пятиугольника, предусматривающая проверку через std::enable\_if |
| void print() | T &h  T::side  double pi  double angle | Шаблонная функция для печати шестиугольника, предусматривающая проверку через std::enable\_if |
| void print() | T &o  T::point  double c;  double s;  double x2;  double y2;  double x3;  double y3;  double x4;  double y4;  double x5;  double y5;  double x6;  double y6;  double x7;  double y7;  double x8;  double y8; | Шаблонная функция для печати восьмиугольника, предусматривающая проверку через std::enable\_if |
| double TotalArea() | T& tuple  int index  std::tuple\_size<T>::value  auto figure  double cur\_area | Шаблонная функция для вычисления площади всех фигур в кортеже, предусматривающая проверку через std::enable\_if |
| double area() | T &p  T::radius  double pi  double side | Шаблонная функция для вычисления площади пятиугольника, предусматривающая проверку через std::enable\_if |
| double area() | T &h  T::side | Шаблонная функция для вычисления площади шестиугольника, предусматривающая проверку через std::enable\_if |
| double area() | T &o  T::point | Шаблонная функция для вычисления площади восьмиугольника, предусматривающая проверку через std::enable\_if |
| void Options() |  | Функция для вывода пунктов меню |
| int main() | int option;  Pentagon<double> p1;  Pentagon<double> p2;  Pentagon<double> p3;  Hexagon<double> h1;  Hexagon<double> h2;  Hexagon<double> h3;  Octagon<double> o1;  Octagon<double> o2;  Octagon<double> o3;  tuple{ p1, h1, o1, p2, h2, o2 };  tuple{ p3, h3, o3 }; | Основная функция программа, в которой происходит демонстрация всех возможностей программы |

**Результаты выполнения тестов**

Таблица 2 – Тесты и результаты работы с ними

|  |  |  |
| --- | --- | --- |
| Название тестового файла | Входные данные | Результат |
| test\_01.txt | 2  3 | Options:  1. Enter figures.  2. Preset figures.  3. Exit.  Select option: 2  Pentagon:  Coordinates: (14; -4.8), (7; 0), (0.34; -4.8), (2.9; -13), (11; -13)  Center: (7, -7)  Area figure: 1.2e+02  Hexagon:  Coordinates: (2.5; -10), (-1.2; -3.5), (-8.7; -3.5), (-12; -10), (-8.8; -16), (-1.2; -16)  Center: (-5, -10)  Area figure: 1.5e+02  Octagon:  Coordinates: (1; 1), (-2.7; 1.9), (-6; 8.9e-16), (-6.9; -3.7), (-5; -7), (-1.3; -7.9), (2; -6), (2.9; -2.3)  Center: (-2, -3)  Area figure: 71  Pentagon:  Coordinates: (10; 3.5), (0.25; 11), (-9.7; 3.5), (-5.9; -8.2), (6.4; -8.2)  Center: (0.25, 0.25)  Area figure: 2.6e+02  Hexagon:  Coordinates: (1.5; 0.87), (1; 1.7), (2.2e-16; 1.7), (-0.5; 0.87), (-4.4e-16; 0.004), (1; 0.004)  Center: (0.5, 0.87)  Area figure: 2.6  Octagon:  Coordinates: (14; -3.1), (12; -2.6), (10; -3.6), (9.6; -5.7), (11; -7.5), (13; -8), (15; -7), (15; -4.9)  Center: (12, -5.3)  Area figure: 21  Total area: 619.33  Select option: 3 |
| test\_02.txt | 1  0 0 2  0.5 0.87 1  0 0 0 10  3 | Options:  1. Enter figures.  2. Preset figures.  3. Exit.  Select option: 1  Enter (coordinates of center) and radius of pentagon accordingly: 0 0 2  Enter (coordinates of center) and side of hexagon accordingly: 0.5 0.87 1  Enter coordinates of center and one point of octagon accordingly: 0 0 0 10  Pentagon:  Coordinates: (1.9; 0.62), (1.2e-16; 2), (-1.9; 0.62), (-1.2; -1.6), (1.2; -1.6)  Enter coordinates of center and one point of octagon accordingly: 0 0 0 10  Pentagon:  Coordinates: (1.9; 0.62), (1.2e-16; 2), (-1.9; 0.62), (-1.2; -1.6), (1.2; -1.6)  Center: (0, 0)  Area figure: 9.5  Hexagon:  Coordinates: (1.5; 0.87), (1; 1.7), (2.2e-16; 1.7), (-0.5; 0.87), (-4.4e-16; 0.004), (1; 0.004)  Center: (0.5, 0.87)  Area figure: 2.6  Octagon:  Coordinates: (0; 10), (-7.1; 7.1), (-10; 0), (-7.1; -7.1), (0; -10), (7.1; -7.1), (10; 0), (7.1; 7.1)  Center: (0, 0)  Area figure: 2.8e+02  Total area: 294.95  Select option: 3 |

1. Листинг программы

**main.cpp**

/\* Мариничев Иван М8О-208Б-19

\*

\* github: IvaMarin/oop\_exercise\_04

\*

\* Вариант 22:

\* 1. Пятиугольник;

\* 2. Шестиугольник;

\* 3. Восьмиугольник;

\*/

#include <iostream>

#include <tuple>

#include <cmath>

template<class T>

struct Pentagon { //Структура Пятиугольник

using type = T;

std::pair<T, T> center; //Координаты центра пятиугольника

double radius; //Радиус пятиугольника

};

template<class T>

struct Hexagon { //Структура Шестиугольник

using type = T;

std::pair<T, T> center; //Координаты центра шестиугольника

double side; //Сторона шестиугольника

};

template<class T>

struct Octagon { //Структура Восьмиугольник

using type = T;

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

std::pair<T, T> point; //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) << std::endl;

std::cout << std::endl;

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

}

//Печать Пятиугольника

template<class T>

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

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

std::cout << "Coordinates: ";

double pi = acos(-1);

for (int i = 0; i < 5; ++i) {

double angle = 2 \* pi \* i / 5;

std::cout.precision(2);

std::cout << "(" << p.center.first + p.radius \* cos(angle + pi / 10) << "; "

<< p.center.second + p.radius \* sin(angle + pi / 10) << ")";

if (i != 4) {

std::cout << ", ";

}

}

std::cout << std::endl;

std::cout << "Center: (" << p.center.first << ", " << p.center.second << ")" << std::endl;

}

//Печать Шестиугольника

template<class T>

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

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

std::cout << "Coordinates: ";

double pi = acos(-1);

for (int i = 0; i < 6; ++i) {

double angle = pi \* i / 3;

std::cout.precision(2);

std::cout << "(" << h.center.first + h.side \* cos(angle) << "; "

<< h.center.second + h.side \* sin(angle) << ")";

if (i != 5) {

std::cout << ", ";

}

}

std::cout << std::endl;

std::cout << "Center: (" << h.center.first << ", " << h.center.second << ")" << std::endl;

}

//Печать Восьмиугольника

template<class T>

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

double c = sqrt(2) / 2; //Косинус 45 градусов

double s = sqrt(2) / 2; //Синус 45 градусов

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

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

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

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

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

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

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

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

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

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

double x7 = o.center.first + (x6 - o.center.first) \* c - (y6 - o.center.second) \* s;

double y7 = o.center.second + (x6 - o.center.first) \* s + (y6 - o.center.second) \* c;

double x8 = o.center.first + (x7 - o.center.first) \* c - (y7 - o.center.second) \* s;

double y8 = o.center.second + (x7 - o.center.first) \* s + (y7 - o.center.second) \* c;

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

std::cout << "Coordinates: ";

std::cout << "(" << o.point.first << "; " << o.point.second << "), ("

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

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

<< x6 << "; " << y6 << "), (" << x7 << "; " << y7 << "), ("

<< x8 << "; " << y8 << ")" << std::endl;

std::cout << "Center: (" << o.center.first << ", " << o.center.second << ")" << std::endl;

}

//Вычисление площади всех фигур в кортеже

template<class T, int index>

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

return 0;

}

template<class T, int index>

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

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

double cur\_area = area(figure);

return cur\_area + TotalArea<T, index + 1>(tuple);

}

//Вычисление площади пятиугольника

template<class T>

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

double pi = acos(-1);

double side = p.radius \* cos(13 \* pi / 10) - p.radius \* cos(17 \* pi / 10);

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

}

//Вычисление площади шестиугольника

template<class T>

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

return pow(h.side, 2) \* 3 \* sqrt(3) \* 0.5;

}

//Вычисление площади восьмиугольника

template<class T>

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

return pow(sqrt(pow(o.point.first - o.center.first, 2) + pow(o.point.second - o.center.second, 2)), 2) \* 2 \* sqrt(2); //Площадь восьмиугольника через радиус описанной окружности

}

void Options() {

std::cout << "Options: " << std::endl; //Параметры

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

std::cout << "2. Preset figures." << std::endl; //Использовать готовые фигуры

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

}

int main() {

int option;

Options();

std::cout << "Select option: ";

while (std::cin >> option) {

if (option == 1) { //Ввод фигур пользователем

std::cout << "Enter (coordinates of center) and radius of pentagon accordingly: "; //Ввод (координат центра) и радиуса пятиугольника

Pentagon<double> p3;

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

std::cout << std::endl;

std::cout << "Enter (coordinates of center) and side of hexagon accordingly: "; //Ввод (координат центра) и стороны шестиугольника

Hexagon<double> h3;

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

std::cout << std::endl;

std::cout << "Enter coordinates of center and one point of octagon accordingly: "; //Ввод координат центра и одной точки восьмиугольника

Octagon<double> o3;

std::cin >> o3.center.first >> o3.center.second >> o3.point.first >> o3.point.second;

std::cout << std::endl;

std::tuple<decltype(p3), decltype(h3), decltype(o3)> //Создание кортежа

tuple{ p3, h3, o3 };

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

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

}

else if (option == 2) { //Использование готовых фигур

Pentagon<int> p1;

p1.center = { 7, -7 };

p1.radius = 7;

Pentagon<double> p2;

p2.center = { 0.25, 0.25 };

p2.radius = 10.5;

Hexagon<int> h1;

h1.center = { -5, -10 };

h1.side = 7.5;

Hexagon<double> h2;

h2.center = { 0.5, 0.87 };

h2.side = 1;

Octagon<int> o1;

o1.center = { -2, -3 };

o1.point = { 1, 1 };

Octagon<double> o2;

o2.center = { 12.34, -5.3 };

o2.point = { 14, -3.12 };

std::tuple<decltype(p1), decltype(h1), decltype(o1), decltype(p2), decltype(h2), decltype(o2)> //Создание кортежа

tuple{ p1, h1, o1, p2, h2, o2 };

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

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

}

else if (option == 3) { //Выход из программы

break;

}

else //Неверный выбор параметра

std::cout << "There is no such option, please try again!" << std::endl;

std::cout << std::endl;

std::cout << "Select option: "; //Повторный ввод

}

return 0;

}

**CmakeLists.txt**

cmake\_minimum\_required(VERSION 3.16.3)

set(CMAKE\_CXX\_FLAGS "-pedantic -Wall -Wextra -Werror")

set(CMAKE\_CXX\_STANDART 17)

project(oop\_lab3)

add\_executable(oop\_exercise\_03 vertex.cpp hexagon.cpp octagon.cpp triangle.cpp main.cpp)

1. Вывод

В ходе лабораторной работы я изучил основы работы с шаблонами (template) в С++, ознакомился с шаблонами std::pair, std::tuple. Кроме того, я получил навыки работы со специализацией шаблонов и идиомой SFINAE.

Список литературы

1. Стефан К. Дьюхэрст Скользкие места С++. Как избежать проблем при проектировании и компиляции ваших программ — ISBN: 5-94074-083-9 — 266 с.
2. Tuples in C++ [Электронный ресурс]. URL: https://www.geeksforgeeks.org/tuples-in-c/ (дата обращения: 1.11.2020).
3. Pair in C++ Standard Template Library (STL) [Электронный ресурс]. URL: https://www.geeksforgeeks.org/pair-in-cpp-stl/ (дата обращения: 1.11.2020).
4. Template Specialization in C++ [Электронный ресурс]. URL: https://www.geeksforgeeks.org/template-specialization-c/ (дата обращения: 2.11.2020).
5. Площадь правильного пятиугольника [Электронный ресурс]. URL: https://www.rapidus.ru/area-5angle.html (дата обращения: 4.11.2020).
6. Площадь правильного шестиугольника [Электронный ресурс]. URL: https://www.fxyz.ru/формулы\_по\_геометрии/формулы\_площади/площадь\_правильного\_шестиугольника/ (дата обращения: 4.11.2020).
7. Площадь правильного восьмиугольника [Электронный ресурс]. URL: https://www.rapidus.ru/area-8angle.html (дата обращения: 5.11.2020).