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: Суханов Егор Алексеевич
Группа: 80-206
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:

- Создавать 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.

Вариант моего задания 21: Ромб, 5-угольник и 6-угольник.

Разобьем задачу на подзадачи:

- Ознакомиться с теорией: шаблоны; стандартные шаблоны std::pair и std::tuple; Идиома SFINAE;
- Создать проект
- Изучить пример кода
- Классы для фигур
- Функция print
- Функция Square
- Пользовательский ввод-вывод
- Создать репозиторий GitHub и опубликовать ЛР

2. Описание программы

GitHub репозиторий можно найти по ссылке: https://github.com/Reterer/oop_exercise_04
Проект состоит из трех файлов:

CMakeLists.txt -- собирает проект
figure.hpp -- заголовочный файл, который содержит классы фигур, и шаблонные функции для вывода координат вершин и расчета площади.

- Класс Rhombus реализует ромб, Класс Pentagon реализует пятиугольник, а класс Hexagon -- шестиугольник. Эти классы имеют одни и те же атрибуты: центр и первую вершину фигуры.
- Шаблонная функция print_cords выводит вершины VERTEX_COUNT-угольной фигуры. Шаблонные функции print_cords, перегруженная для классов фигур, являются оболочками для print_cords.
- Шаблонная функция calc_area рассчитывает площадь VERTEX_COUNT-угольной фигуры, а функции calc_area, перегруженные для классов фигур, являются оболочками над calc_area.

main.cpp -- отвечает за обработку пользовательского ввода. Содержит следующие функции:

- clear -- вспомогательная функция для очистки потока ввода и сброса состояния ошибки после неправильного ввода.
- square -- вычисляет сумму площадей фигур из кортежа.
- set -- устанавливает вершины фигуры (считывает центр и первую вершину из потока ввода).

Последние две функции имеют схожий принцип работы: они рекурсивно вызывают себя же. Рекурсия создается во время компиляции. с помощью конструкции constexpr if.

Программа работает в интерактивном режиме: ожидает от пользователя ввода команд, которые она обрабатывает и выдает ответы.
Пример работы с программой:

```
>help
help   -- выводит этот текст
exit   -- выход из программы
set <id> <center> <vertex> -- устанавливает координаты для фигуры под номером id
print -- выводит координаты фигур
square -- вычисляет общую площадь

>set 0 0 0 1 1
>set 1 1 2 2 1
>print
Координаты ромба: (1 , 1) (0 , 1) (-1 , 0) (0 , -1)
Координаты пятиугольника: (2 , 1) (2.26007 , 2.64204) (0.778768 , 3.3968) (-0.396802 , 2.22123) (0.35796 , 0.739926)
Координаты шестиугольника: (0 , 0) (0 , 0) (0 , 0) (0 , 0) (0 , 0) (0 , 0)

>exit
```

3. Набор тестов и результаты их выполнения

test01.txt

```
set             0             0             0             1             1
set             1             0             0             1             1
set             2             0             0             1             1
print
square
exit
help
```

Вывод:

```
Координаты ромба: (1 , 1) (-1 , 1) (-1 , -1) (1 , -1)
Координаты пятиугольника: (1 , 1) (-0.64204 , 1.26007) (-1.3968 , -
0.221232) (-0.221232 , -1.3968) (1.26007 , -0.64204)
Координаты шестиугольника: (1 , 1) (-0.366025 , 1.36603) (-1.36603 ,
0.366025) (-1 , -1) (0.366025 , -1.36603) (1.36603 , -0.366025)

Сумма площадей фигур: 13.9514
```

test02.txt

```
set             0             0             0             1             1
square
set             0             0             0             1             0
square
set             0             1             1             2             3
square
```

Вывод:

```
Сумма                  площадей                фигур:                 4
Сумма                  площадей                фигур:                 2
Сумма площадей фигур: 10
```

test03.txt

```
kfekj
set ekfj 1 2 1 2
set 0 0 0 0 0
set 0 1ikewjfw2 4iefj3 4edifj 0
set 0 1 1 0 0
set -1 1 1 0 0
set 234 1 1 0 0
```

Вывод:

```
Такой                  команды                 не                  существует.
Введен                 неверный                индекс                 кортежа.
```

```
Центр и вершина совпадают.
Координаты введены неверно.
Введен неверный индекс кортежа.
Введен неверный индекс кортежа.
```

4. Листинг программы

main.cpp:

```
/*
  Лабораторная работа: 4
  Вариант: 21
  Группа: М8О-206Б-19
  Автор: Суханов Егор Алексеевич

  Задание:
      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.
      Реализовать шаблонную функцию print и square.
      Реализовать ввод фигур.
*/
#include <iostream>
#include <string>
#include <utility>
#include <tuple>
#include "figure.hpp"

void clear()
{
      std::cin.clear();
      std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}

// Вычисляет сумму площадей фигур
template <class T, size_t index = 0>
double square(T value) {
      if constexpr (index < std::tuple_size<T>::value) {
            return calc_area(std::get<index>(value)) + square<T, index + 1>(value);
      }
      else {
            return 0;
      }
```

```
}

// Выводит кортеж фигур
template <class T, size_t index = 0>
void print_tuple(T value) {
	if constexpr (index < std::tuple_size<T>::value) {
		print_figure(std::get<index>(value));
		print_tuple<T, index + 1>(value);
	}
	else {
		std::cout << std::endl;
	}
}

// Устанавливает координаты данной фигуры
template <class T>
void set_figure(T& figure) {
	if (! (std::cin >> figure.center.first >> figure.center.second
		>> figure.vertex.first >> figure.vertex.second)) {
		std::cout << "Координаты введены неверно." << std::endl;
		clear();
	}
	if (figure.center.first == figure.vertex.first
		&& figure.center.second == figure.vertex.second) {
		std::cout << "Центр и вершина совпадают." << std::endl;
		figure.center = {};
		figure.vertex = {};
		clear();
	}
}
// Устанавливает координаты i-й фигуры в кортеже value
template <class T, size_t index = 0>
void set(T& value, size_t i) {
	if constexpr (index < std::tuple_size<T>::value) {
		if (i == index) {
			set_figure(std::get<index>(value));
		}
		else {
			set<T, index + 1>(value, i);
		}
	}
	else {
		return;
	}
}

// Обрабатывает команду "set" -- установить координаты какой то фигуры
template <class T>
void set(T& value){
	size_t index;
	if (std::cin >> index && index < std::tuple_size<T>::value) {
		set(value, index);
	}
```

```
	else {
		std::cout << "Введен неверный индекс кортежа." << std::endl;
		clear();
	}
}

// Выводит справку об использовании программы
void help() {
	std::cout <<
		"help   -- выводит этот текст\n"
		"exit   -- выход из программы\n"
		"set <id> <center> <vertex> -- устанавливает координаты для
фигуры под номером id\n"
		"print -- выводит координаты фигур\n"
		"square -- вычисляет общую площадь\n"
		<< std::endl;
}

int main() {
	setlocale(LC_ALL, "russian");
	std::tuple<   Rhombus<double>,   Pentagon<double>,   Hexagon<double>>
tuple;

	std::string cmd;
	while (std::cout << '>', std::cin >> cmd) {
		if (cmd == "exit")
			break;
		else if (cmd == "help")
			help();
		else if (cmd == "set")
			set(tuple);
		else if (cmd == "print")
			print_tuple(tuple);
		else if (cmd == "square")
			std::cout << "Сумма площадей фигур: " << square(tuple) <<
std::endl;
		else {
			std::cout << "Такой команды не существует." << std::endl;
			clear();
		}
	}

	return 0;
}
```

figure.hpp:

```
#pragma once

template <class T>
struct Rhombus {
	std::pair<T, T> center;
	std::pair<T, T> vertex;
};
```

```
template <class T>
struct Pentagon {
	std::pair<T, T> center;
	std::pair<T, T> vertex;
};

template <class T>
struct Hexagon {
	std::pair<T, T> center;
	std::pair<T, T> vertex;
};


// Векторное сложение
template <class T>
std::pair<T, T> operator + (const std::pair<T, T> a, const std::pair<T, T>
b) {
	return { a.first + b.first, a.second + b.second };
}
// Векторное вычитание
template <class T>
std::pair<T, T> operator - (const std::pair<T, T> a, const std::pair<T, T>
b) {
	return { a.first - b.first, a.second - b.second };
}
// Скалярное умножение векторов
template <class T>
T operator * (const std::pair<T, T> a, const std::pair<T, T> b) {
	return a.first * b.first + a.second * b.second;
}
// Умножение на скаляр
template <class T>
std::pair<T, T> operator * (const T a, const std::pair<T, T> b) {
	return { a * b.first, a * b.second };
}

const double PI = 3.141592653589793;
// Поворот вектора pair на угол angle
template <class T>
std::pair<T, T> rotate(std::pair<T, T> pair, const double angle) {
	std::pair<T, T> rotated;
	rotated.first = (T)(pair.first * std::cos(angle) - pair.second *
std::sin(angle));
	rotated.second = (T)(pair.first * std::sin(angle) + pair.second *
std::cos(angle));
	return rotated;
}

// Вывод координат точки
template <class T>
void print_pair(std::pair<T, T> p) {
	std::cout << '(' << p.first << " , " << p.second << ')';
```

```
}
// Вывод координат фигуры
template <int VERTEX_COUNT, class T>
void print_cords(const T center, const T vertex)
{
	const double ANGLE = 2 * PI / VERTEX_COUNT; // угол между двумя соседними вершинами и центром
	print_pair(vertex);

	T vec = vertex - center;
	for (int i = 2; i <= VERTEX_COUNT; ++i)
	{
		vec = rotate(vec, ANGLE);
		std::cout << ' ';
		print_pair(center + vec);
	}
}
template <class T>
void print_figure(Rhombus<T> value) {
	std::cout << "Координаты ромба: ";
	print_cords<4>(value.center, value.vertex);
	std::cout << std::endl;
}
template <class T>
void print_figure(Pentagon<T> value) {
	std::cout << "Координаты пятиугольника: ";
	print_cords<5>(value.center, value.vertex);
	std::cout << std::endl;
}
template <class T>
void print_figure(Hexagon<T> value) {
	std::cout << "Координаты шестиугольника: ";
	print_cords<6>(value.center, value.vertex);
	std::cout << std::endl;
}

// Вычисление площади фигуры
template <int VERTEX_COUNT, class T>
double calc_area(const T center, const T vertex)
{
	T vecRadius = vertex - center; // Вектор-радиус описанной окружности
	double sqRadius = vecRadius * vecRadius; // Квадрат радуиса описанной окружности
	return VERTEX_COUNT / 2. * sqRadius * std::sin(2 * PI / VERTEX_COUNT);
}
template <class T>
double calc_area(Rhombus<T> figure) {
	return calc_area<4>(figure.center, figure.vertex);
}
template <class T>
double calc_area(Pentagon<T> figure) {
	return calc_area<5>(figure.center, figure.vertex);
```

```
}
template <class T>
double calc_area(Hexagon<T> figure) {
	return calc_area<6>(figure.center, figure.vertex);
}
```

5. Выводы

Выполняя данную лабораторную я научился основам работы с шаблонами, узнал об стандартном шаблоне tuple и использовал идиому SFINAE. О шаблонах и метапрограммировании в C++ можно сказать следующее: иногда оно может очень сильно выручить, например можно сделать контейнеры для почти любого типа, описав одну “абстрактную” реализацию; Однако использование шаблонов делает код менее читабельным, а вероятность сделать ошибку повышается. Делаю вывод, что использовать метапрограммирование можно, но только при крайней необходимости.

6. Литература

1. Страуструп, Бьёрн. Язык программирования C++. Краткий курс, 2-е изд. : Пер. с англ. - СПб.: ООО "Диалектика", 2019. - 320 с.: ил. - Парал. тит. англ.