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: Ивенкова Любовь Васильевна
Группа: 80-208
Преподаватель: Чернышов Л.Н.
Дата:
Оценка:

Москва, 2020

Содержание

1. Постановка задачи

Вариант: 3
Задача:

Разработать шаблоны классов трех фигур: прямоугольника, равнобедренной трапеции и ромба. 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. Реализовать две шаблонные функции **print** для вывода координат всех фигур из std::tuple и **square** для нахождения общей площади фигур std::tuple.

Программа должна:

- создавать фигуры;
- сохранять фигуры в std::tuple;
- выводить координаты фигур из std::tuple с помощью **print**;
- выводить общую площадь фигур из std::tuple, найденную через **square**.

2. Описание программы

Программа принимает в себя данные из консоли и из файла, при перенаправлении потока ввода вывода, и выполняет заданные действия. При запуске появляется меню выбора операций, после выбора которой вводятся данные. В список операций входит *добавление фигуры* (трапеции, прямоугольника или ромба) *в std::tuple*, *вывод общей площади фигур*, *вывод всех координат фигур в std::tuple*.

Программа состоит из трех файлов: main.cpp, functions.h и figures.h:

- figures.h - описание классов фигур. Они представлены шаблонными классами, в которых присутствуют лишь поля классов;
- functions.h - реализация основных функций print (вывод координат фигуры) и square(вывод общей площади фигур в tuple). Также реализованы функции get_coords_rectanle, get_coords_rhomb, get_coord_trapeze, которые передают в соответствующие объекты координаты фигур, исходя из введенных пользователем сторон. Для нахождения площади прямоугольника используется формула a*b, где a, b - стороны прямоугольника, для ромба - a*b/2, где a, b - его диагонали, для трапеции - ((a+b)/2)*c, где a, b - ее основания, c - высота.
- main.cpp - главный файл. В нем создается std::tuple, куда в дальнейшем помещаются новые фигуры. Также там реализована функция menu(), которая выводит список возможных действий. При создании фигуры предлагается ввести ее стороны.

Переменные классов

Классы фигур по структуре одинаковы, поэтому рассмотрим лишь один класс:

Rectangle

```
using coords = std::pair<T, T>;
```

coords a, b, c, d; - координаты фигуры, представленные парой координат x, y;

char name[10] = "rectangle"; - название фигуры.

3. Набор тестов

Таблица 1. Тест 1

1 1 4 6 2	-добавить новую трапецию в std::tuple с типом данных координат int с основаниями 4 и 6 и высотой 2;
1 2 4 5 4	-добавить новый трапецию в std::tuple с типом данных координат double с основаниями 4 и 5 и высотой 4;
2 1 3 5	-добавить новый ромб в std::tuple с типом данных координат int с диагоналями 3 и 5 (выдаст ошибку);
2 2 4 6	-добавить новый ромб в std::tuple с типом данных координат double с диагоналями 4 и 6;
5	-найти общую площадь (двух трапеций и ромба);
4	-вывести все координаты фигур в std::tuple;
0	-выход из программы.

Таблица 2. Тест 2

2 2 3 5	-добавить новый ромб в std::tuple с типом данных координат double с диагоналями 3 и 5;
3 1 4 6	-добавить новый прямоугольник в std::tuple с типом данных координат int со сторонами 4 и 6;
1 2 4 6 2	-добавить новую трапецию в std::tuple с типом данных координат double со основаниями 4 и 6 и высотой 2;
4	-вывести все координаты фигур в std::tuple;
5	-найти общую площадь (ромб, прямоугольник, трапеция);
0	-выход и программы.

4. Результат выполнения тестов

Тест 1:
```
Enter 0-5 to:
1 - add a trapeze
2 - add a rhombus
3 - add a rectangle
4 - print all coordinates of figures
5 - find the total square
0 - exit
Your choice: 1
What datatype of shape coordinates do you want? int(1) or double(2)? 1
Enter the two bases and the height of the trapezoid: 4 6 2
What's next?
>>1
What datatype of shape coordinates do you want? int(1) or double(2)? 2
Enter the two bases and the height of the trapezoid: 4 5 4
What's next?
>>2
What datatype of shape coordinates do you want? int(1) or double(2)? 1
Enter the two diagonals of the rhombus: 3 5
Incorrect value!
What's next?
>>2
What datatype of shape coordinates do you want? int(1) or double(2)? 2
Enter the two diagonals of the rhombus: 4 6
What's next?
>>5
Total square of figures is 40
What's next?
>>4
Name of shape is trapeze
(0, 0), (5, 0), (4.5, 4), (0.5, 4)
Name of shape is trapeze
(0, 0), (6, 0), (5, 2), (1, 2)
Name of shape is rhombus
(0, 0), (2, 3), (4, 0), (2, -3)
Name of shape is rhombus
(0, 0), (0, 0), (0, 0), (0, 0)
Name of shape is rectangle
(0, 0), (0, 0), (0, 0), (0, 0)
Name of shape is rectangle
(0, 0), (0, 0), (0, 0), (0, 0)

What's next?
>>0
The program is closed, goodbye!
```

Тест 2:
```
Enter 0-5 to:
1 - add a trapeze
2 - add a rhombus
3 - add a rectangle
4 - print all coordinates of figures
5 - find the total square
```

```
0 - exit
Your choice: 2
What datatype of shape coordinates do you want? int(1) or double(2)? 2
Enter the two diagonals of the rhombus: 3 5
What's next?
>>3
What datatype of shape coordinates do you want? int(1) or double(2)? 1
Enter the two sides of the rectangle: 4 6
What's next?
>>1
What datatype of shape coordinates do you want? int(1) or double(2)? 2
Enter the two bases and the height of the trapezoid: 4 6 2
What's next?
>>4
Name of shape is trapeze
(0, 0), (6, 0), (5, 2), (1, 2)
Name of shape is trapeze
(0, 0), (0, 0), (0, 0), (0, 0)
Name of shape is rhombus
(0, 0), (1.5, 2.5), (3, 0), (1.5, -2.5)
Name of shape is rhombus
(0, 0), (0, 0), (0, 0), (0, 0)
Name of shape is rectangle
(0, 0), (0, 0), (0, 0), (0, 0)
Name of shape is rectangle
(0, 0), (4, 0), (4, 6), (0, 6)

What's next?
>>5
Total square of figures is 41.5
What's next?
>>0
The program is closed, goodbye!
```

5. Листинг программы

/*https://github.com/Li-Iven/OOP/tree/main/oop_exercise_04
Ивенкова Любовь, 208 группа*/
main.cpp

```
#include "figures.h"
#include "functions.h"
#include <iostream>
#include <tuple>

void menu() {
    std::cout << "Enter 0-5 to:" << std::endl;
    std::cout << "1 - add a trapeze" << std::endl;
    std::cout << "2 - add a rhombus" << std::endl;
    std::cout << "3 - add a rectangle" << std::endl;
    std::cout << "4 - print all coordinates of figures" << std::endl;
    std::cout << "5 - find the total square" << std::endl;
    std::cout << "0 - exit" << std::endl;
}

void error() {
    std::cout << "Incorrect value!" << std::endl;
}

int main(){
    int choice, type;
    double a, b, c;
    menu();
    std::tuple<Trapeze<double>, Trapeze<int>, Rhombus<double>, Rhombus<int>, Rectangle<double>,
Rectangle<int>> all;
    std::cout << "Your choice: ";

    do {
        std::cin >> choice;
        switch (choice)
        {
        case 1:
            std::cout << "What datatype of shape coordinates do you want? int(1) or double(2)? ";
            std::cin >> type;
            if (type > 2) {
                error();
            }
            else {
                std::cout << "Enter the two bases and the height of the trapezoid: ";
                std::cin >> a >> b >> c;
                if (type == 2) {
                    get_coord_trapeze(std::get<0>(all), a, b, c);
                }
                else if (type == 1) {
```

```
                double correct = abs(a-b)/2;
                if (correct == (long long)correct && a==(long long)a && b == (long long)b && c == (long
long)c) {
                    get_coord_trapeze(std::get<1>(all), a, b, c);
                }
                else {
                    error();
                }
            }
        }
        break;
    case 2:
        std::cout << "What datatype of shape coordinates do you want? int(1) or double(2)? ";
        std::cin >> type;
        if (type > 2) {
            error();
        }
        else {
            std::cout << "Enter the two diagonals of the rhombus: ";
            std::cin >> a >> b;
            if (type == 2) {
                get_coords_rhomb(std::get<2>(all), a, b);
            }
            else if (type == 1) {
                double correct1 = a/2, correct2=b/2;
                if (correct1 == (long long)correct1 && correct2 == (long long)correct2) {
                    get_coords_rhomb(std::get<3>(all), a, b);
                }
                else {
                    error();
                }
            }
        }
        break;
    case 3:
        std::cout << "What datatype of shape coordinates do you want? int(1) or double(2)? ";
        std::cin >> type;
        if (type > 2) {
            error();
        }
        else {
            std::cout << "Enter the two sides of the rectangle: ";
            std::cin >> a >> b;
            if (a != b) {
                if (type == 2) {
                    get_coords_rectanle(std::get<4>(all), a, b);
                }
                else if (type == 1) {
                    if (a == (long long)a && b == (long long)b) {
                        get_coords_rectanle(std::get<5>(all), a, b);
                    }
                    else {
```

```
                    error();
                }
            }
        }
        else {
            error();
        }
    }
    break;

case 4:
    print_tuple<decltype(all), 0>(all);
    break;
case 5:
    std::cout <<"Total square of figures is "<< square(all)<<std::endl;
    break;
case 0:
    std::cout << "The program is closed, goodbye!\n";
    return false;
    break;
default:
    std::cout << "Incorrect values! Try again.\n";
    break;
}
std::cout << "What's next?\n";
std::cout << ">>";
} while (choice);

}
```

figures.h

```
#pragma once
#include <iostream>

template<typename T>
struct Rectangle {
        using coords = std::pair<T, T>;
        coords a, b, c, d;
        char name[10] = "rectangle";
};


template<typename T>
struct Rhombus {
        using coords = std::pair<T, T>;
        coords a, b, c, d;
        char name[8] = "rhombus";
};


template<typename T>
struct Trapeze {
        using coords = std::pair<T, T>;
        coords a, b, c, d;
```

```
        char name[8] = "trapeze";
};
```

functions.h

```
#pragma once
#include<iostream>
#include<tuple>

template<typename T, typename S>
void get_coord_trapeze(T& tuple, S base1, S base2, S height) {
    double tmp;
    if (base1 < base2) {
        tmp = base2;
        base2 = base1;
        base1 = tmp;
    }
    tuple.a.first = tuple.a.second = tuple.b.second = 0;
    tuple.b.first = base1;
    tuple.c.first = base2 + (base1 - base2) / 2;
    tuple.c.second = tuple.d.second = height;
    tuple.d.first = (base1 - base2) / 2;
}

template<typename T, typename S>
void get_coords_rectanle(T& tuple_elem, S horiz, S vert) {
    tuple_elem.a.first = tuple_elem.a.second = tuple_elem.d.first = tuple_elem.b.second = 0;
    tuple_elem.b.first = tuple_elem.c.first = horiz;
    tuple_elem.c.second = tuple_elem.d.second = vert;
}

template<typename T, typename S>
void get_coords_rhomb(T& tuple_elem, S horiz, S vert) {
    tuple_elem.a.first = tuple_elem.a.second = tuple_elem.c.second = 0;
    tuple_elem.b.first = tuple_elem.d.first = horiz / 2;
    tuple_elem.b.second = vert / 2;
    tuple_elem.c.first = horiz;
    tuple_elem.d.second = -vert / 2;
}

template <class T, size_t index> typename std::enable_if<index >= std::tuple_size<T>::value, void>::type
print_tuple(T&) {
    std::cout << std::endl;
}
template <class T, size_t index> typename std::enable_if < index<std::tuple_size<T>::value, void>::type
print_tuple(T& tuple) {
    auto vertex = std::get<index>(tuple);
    print(vertex);
    print_tuple<T, index + 1>(tuple);
}

template<typename T>
void print(T& obj) {
    std::cout <<"Name of shape is "<< obj.name <<std::endl;
    std::cout << "(" << obj.a.first << ", " << obj.a.second << "), (" << obj.b.first << ", " << obj.b.second << "),
(" << obj.c.first << ", " << obj.c.second << "), (" << obj.d.first << ", " << obj.d.second << ")" << std::endl;
}

template<typename T>
```

```
double square_of_rectangle(T& elem) {
   return elem.c.first* elem.c.second;
}

template<typename T>
double square_of_rhombus(T& elem) {
   return (elem.b.first * elem.b.second * 2);
}

template<typename T>
double square_of_trapeze(T& elem) {
   return ((elem.b.first + elem.c.first - elem.d.first) / 2) * elem.c.second;
}

template<typename T>
double square(T& tuple) {
   double t = 0;
   t = t + square_of_trapeze(std::get<0>(tuple)) + square_of_trapeze(std::get<1>(tuple));
   t = t + square_of_rhombus(std::get<2>(tuple)) + square_of_rhombus(std::get<3>(tuple));
   t = t + square_of_rectangle(std::get<4>(tuple)) + square_of_rectangle(std::get<5>(tuple));
   return t;
}
```

6. Вывод

В данной лабораторной работе были изучены шаблоны и метафункции в языке C++. При помощи шаблонов и метапрограммирования можно добиться того, чтобы компилятор сам генерировал код, необходимый для некоторых функций и классов. В стандартной библиотеке имеется много шаблонных структур, которые упрощают метапрограммирование.

В данной лабораторной работе была применена техника SFINAE - Substitution Failure Is Not An Error. Она основывается на том, что компилятор, пытаясь вывести тип параметра для параметра шаблона, встречая ошибку в конкретной специализации, не выдает ошибку пользователю, а анализирует все возможные варианты.

7. Список используемых источников

1. Enable_if [Электронный ресурс]. URL: https://riptutorial.com/ru/cplusplus/example/3777/enable-if (Дата обращения: 10.04.2021)
2. std::tuple [Электронный ресурс]. URL: https://en.cppreference.com/w/cpp/utility/tuple (Дата обращения: 10.04.2021)
3. C++11: Обход элементов в кортеже (std::tuple) [Электронный ресурс]. URL: http://artlang.net/post/c++11-obkhod-elementov-kortezhe-std-tuple/ (Дата обращения: 10.04.2021)
4. Шаблонные классы и функции [Электронный ресурс]. https://code-live.ru/post/cpp-template-functions/ (Дата обращения: 10.04.2021)