**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 метопрограммирования.

Студент: Курносов Максим Юрьевич

Группа: 80-208

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

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:

·         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.

Вариант задания №8:

|  |  |  |  |
| --- | --- | --- | --- |
| 8. | 8-угольник | Треугольник | Квадрат |

1. **Описание программы**

Программа считывает из файла test\_xx.txt данные о фигурах, а именно, количество вершин, их координаты (абсциссу и ординату первой, второй и т. д.).

затем программа ожидает ввода цифры (справка о вводе появится на экране), соответствующей своей команде. После чего используя функцию выводит фигуры или считает их площадь.

1. **Набор тестов**

Демонстрирует как вводить значения для создания фигур.

Первое значение – количество вершин, затем их координаты как было описано ранее в отчёте.

For file test\_01.txt:

4

13 8

7 9

5 7

9 6

8

13 8

7 9

9 6

5 7

9 6

13 8

7 9

11 0

3

45 8

5 44

-1 -39

For file test\_02.txt:

4

13 8

7 9

5 7

9 6

8

13 8

7 9

3 6

5 7

9 6

13 8

7 9

11 0

3

45 8

5 44

-1 -39

For file test\_03.txt:

4

13 8

7 9

5 7

9 6

8

13 8

7 9

9 6

5 7

9 6

13 8

7 59

11 0

3

45 8

5 44

-1 -39

1. **Результаты выполнения тестов**

root@g8-RM:/mnt/c/Users/g7/Desktop/коды/2sem/oop/4laboop# ./oop\_exercise\_04

Для выполнения операции введите соответствующую цифру:

1 - Вывести Координаты всех вершин для всех фигур

2 - Вычислить общую площадь всех фигур

0 - Выйти из программы

1

Вершины Квадрата:

x = 13 y = 8

x = 7 y = 9

x = 5 y = 7

x = 9 y = 6

Вершины Восьмиугольника:

x = 13 y = 8

x = 7 y = 9

x = 9 y = 6

x = 5 y = 7

x = 9 y = 6

x = 13 y = 8

x = 7 y = 9

x = 11 y = 0

Вершины Треугольника:

x = 45 y = 8

x = 5 y = 44

x = -1 y = -39

2

Общая площадь фигур = 504.775804

3

Некоректный ввод

Для выполнения операции введите соответствующую цифру:

1 - Вывести Координаты всех вершин для всех фигур

2 - Вычислить общую площадь всех фигур

0 - Выйти из программы

0

Выход из программы...

1. **Листинг программы**

For file main.cpp:

#include <iostream>

#include <tuple>

#include <cmath>

template <class T>

struct Eight

{

int count = 8;

std::pair<T, T> a[8];

};

template <class T>

struct Square

{

int count = 4;

std::pair<T, T> a[4];

};

template <class T>

struct Triangle

{

int count = 3;

std::pair<T, T> a[3];

};

template <class Tf>

void PrintFig(Tf figure)

{

//std::cout<< figure.count << std::endl;

switch (figure.count)

{

case 3:

{

std::cout << "Вершины Треугольника:" << std::endl;

PrintT(figure);

break;

}

case 4:

{

std::cout << "Вершины Квадрата:" << std::endl;

PrintSq(figure);

break;

}

case 8:

{

std::cout << "Вершины Восьмиугольника:" << std::endl;

PrintEight(figure);

break;

}

default:

{

std::cout << "Введена непредусмотренная фигура или что-то пошло не так" << std::endl;

break;

}

}

}

template <class T>

void PrintT(T figure)

{

for (int i = 0; i < 3; i++)

{

std::cout << "x = " << figure.a[i].first << "\ty = " << figure.a[i].second << std::endl;

}

}

template <class T>

void PrintSq(T figure)

{

for (int i = 0; i < 4; i++)

{

std::cout << "x = " << figure.a[i].first << "\ty = " << figure.a[i].second << std::endl;

}

}

template <class T>

void PrintEight(T figure)

{

for (int i = 0; i < 8; i++)

{

std::cout << "x = " << figure.a[i].first << "\ty = " << figure.a[i].second << std::endl;

}

}

template <class Tf>

double ShowS(Tf figure)

{

double Sqr = 0;

//std::cout<< figure.count << std::endl;

switch (figure.count)

{

case 3:

{

double s, a, h;

a = sqrt((double)((figure.a[0].first - figure.a[1].first) \* (figure.a[0].first - figure.a[1].first) - (figure.a[0].second - figure.a[1].second) \* (figure.a[0].second - figure.a[1].second)));

h = a \* sin(3.14 / 3);

s = (a \* h) / 2;

return s;

break;

}

case 4:

{

double s, a;

a = sqrt((double)((figure.a[0].first - figure.a[1].first) \* (figure.a[0].first - figure.a[1].first) - (figure.a[0].second - figure.a[1].second) \* (figure.a[0].second - figure.a[1].second)));

s = a \* a;

return s;

break;

}

case 8:

{

double s, a, h, ctg;

a = sqrt((double)((figure.a[0].first - figure.a[1].first) \* (figure.a[0].first - figure.a[1].first) - (figure.a[0].second - figure.a[1].second) \* (figure.a[0].second - figure.a[1].second)));

ctg = cos(3.14 / 8) / sin(3.14 / 8);

h = ctg \* a;

s = a \* h \* 4;

return s;

break;

}

default:

{

std::cout << "Введена непредусмотренная фигура или что-то пошло не так" << std::endl;

break;

}

}

}

/\* template<class T>

void ST(T figure){

}

template<class T>

void SSq(T figure){

return

}

template<class T>

void SE(T figure){

} \*/

void help()

{

printf("Для выполнения операции введите соответствующую цифру:\n ");

printf("\t 1 - Вывести Координаты всех вершин для всех фигур\n ");

printf("\t 2 - Вычислить общую площадь всех фигур\n ");

printf("\t 0 - Выйти из программы\n ");

}

int main()

{

int enter, number, c;

FILE \*fl = fopen("test\_01.txt", "r");

//считывание фигур

Triangle<int> tr;

Square<int> s;

Eight<int> e;

//считывание фигур из файла

while (!feof(fl))

{

fscanf(fl, "%d", &c);

//printf("%d\n", c);

number++;

switch (c)

{

case 3:

{

for (int i = 0; i < 3; i++)

{

fscanf(fl, "%d %d", &tr.a[i].first, &tr.a[i].second);

//printf("%d %d\n", tr.a[i].first, tr.a[i].second);

}

//printf("\n");

break;

}

case 4:

{

for (int i = 0; i < 4; i++)

{

fscanf(fl, "%d %d", &s.a[i].first, &s.a[i].second);

//printf("%d %d\n", s.a[i].first, s.a[i].second);

}

//printf("\n");

break;

}

case 8:

{

for (int i = 0; i < 8; i++)

{

fscanf(fl, "%d %d", &e.a[i].first, &e.a[i].second);

//printf("%d %d\n", e.a[i].first, e.a[i].second);

}

//printf("\n");

break;

}

default:

{

std::cout << "Некоректный ввод фигуры " << std::endl;

std::cout << "Введено значение ";

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

break;

}

}

}

std::tuple<Square<int>, Eight<int>, Triangle<int>> t(s, e, tr);

//auto t = std::make\_tuple(s, e, tr);

fclose(fl);

//std::cin >> enter;

//взаимодействие

help();

while (1)

{

std::cin >> enter;

switch (enter)

{

case 1:

{

PrintFig(std::get<0>(t));

PrintFig(std::get<1>(t));

PrintFig(std::get<2>(t));

break;

}

case 2:

{

double S;

S += ShowS(std::get<0>(t));

S += ShowS(std::get<1>(t));

S += ShowS(std::get<2>(t));

printf("Общая площадь фигур = %lf\n", S);

break;

}

case 0:

{

std::cout << "Выход из программы...\n " << std::endl;

return 0;

}

default:

{

std::cout << "Некоректный ввод\n " << std::endl;

help();

break;

}

}

}

std::cout << "Похоже введена буква, ну в другой раз не промахнётесь по клавиатуре \n\n"

<< std::endl;

return 0;

}

For file CMakeLists.txt:

all: oop\_exercise\_04

oop\_exercise\_04: main.o

g++ main.o -o oop\_exercise\_04

main.o: main.cpp

g++ -c main.cpp

clean:

rm -rf \*.o oop\_exercise\_04

1. **Вывод**

В процессе выполнения лабораторной работы были закреплены навыки создания структур, работы с ними, а также приобретены навыки в области работы с шаблонами и стандартной библиотекой картежей std::tuple С++ .

**Список литературы**

1.Бьярне Страуструп. Программирование: принципы и практика использования C++, исправленное издание = Programming: Principles and Practice Using C++. — М.: «Вильямс», 2011. — С. 1248. — ISBN 978-5-8459-1705-8.

2.Канал #SimpleCode YouTube https://www.youtube.com/watch?v=O7ruEWCa7zc

3. Сайт cppstudio.com

<http://cppstudio.com/post/10103/>

4. Сайт cppreference.com

<https://en.cppreference.com/w/cpp/container/vector>