**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 3**

Тема: Механизмы наследования в С++

Студент: Ляшун Дмитрий Сергеевич

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. **Постановка задачи**

Разработать программу на языке C++ согласно варианту задания. Программа на C++ должна собираться с помощью системы сборки CMake. Программа должна получать данные из стандартного ввода и выводить данные в стандартный вывод.

Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure. Фигуры являются фигурами вращения. Все классы должны поддерживать набор общих методов:

1. Вычисление геометрического центра фигуры;

2. Вывод в стандартный поток вывода std::cout координат вершин фигуры;

3. Вычисление площади фигуры;

Создать программу, которая позволяет:

• Вводить из стандартного ввода std::cin фигуры, согласно варианту задания.

• Сохранять созданные фигуры в динамический массив std::vector<Figure\*>

• Вызывать для всего массива общие функции (1-3 см. выше).Т.е. распечатывать для каждой фигуры в массиве геометрический центр, координаты вершин и площадь.

• Необходимо уметь вычислять общую площадь фигур в массиве.

• Удалять из массива фигуру по индексу.

**Вариант №24:**

|  |  |  |
| --- | --- | --- |
| **Фигура №1** | **Фигура №2** | **Фигура №3** |
| 8-угольник | Треугольник | Квадрат |

Все фигуры выпуклые и равносторонние.

1. **Описание программы**

Для представления координат вершин фигур, а также геометрического центра опишем структуру **Point** с вещественными полями x и y.

Для реализации классов заданных фигур сперва опишем базовый абстрактный класс **Figure** со следующими полями:

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Назначение** |
| points | std::vector<Point> | Хранит координаты всех вершин фигуры. |

Класс **Figure** будет иметь следующие методы:

|  |  |
| --- | --- |
| **Метод** | **Назначение** |
| Figure(int count) | Конструктор, создающий фигуру с count вершинами. |
| Point Get\_center() | Возвращает геометрический центр фигуры. |
| virtual void Output() | Функция вывода координат вершин фигуры, может быть переопределена в классах-наследниках. |
| virtual double Get\_square() = 0 | Возвращает значение площади фигуры, определяется только в классах-наследниках. |
| void Input() | Процедура ввода координат вершин фигуры. |
| double Get\_side() | Возвращает значение длины стороны фигуры. |

Классы **Triangle**, **Square** и **Octagon**, которые представляют треугольник, квадрат и 8-угольник соответственно, будут являться классами-наследниками от класса **Figure**. В них не будут добавлены новые поля и методы, а только переопределены виртуальные Get\_square() и Output() (перед выводом координат вершин будет выводиться также тип фигуры) из класса-предка. Также в каждом из этих классов будет определен свой конструктор по умолчанию, который вызывает конструктор базового класса Figure(int count) со значением count = 3 для треугольника, 4 для квадрата и 8 для 8-угольника.

1. **Руководство по использованию программы**

**oop\_exercise\_03** - исполняемый файл программы.

**tests1.txt** - первый набор тестов для программы.

**tests2.txt** - второй набор тестов для программы.

**Входные данные**

Взаимодействие с программой реализовано посредством интерактивного меню, для вызова требуемой операции необходимо ввести ее номер. Программа поддерживает следующие операции:

1 - Добавить новую фигуру в массив.

2 - Вывести координаты фигур.

3 - Вывести геометрические центры фигур.

4 - Вывести площади фигур.

5 - Вывести общую площадь фигур.

6 - Удалить фигуру из массива.

0 - Выход из программы.

При вызове операции добавления фигуры в массив необходимо ввести номер фигуры (1 - квадрат, 2 - треугольник, 3 - восьмиугольник) и координаты ее вершин. Для удаления фигуры из массива необходимо также вводить ее позицию.

**Выходные данные**

Для каждого запроса на выполнение операции выводится результат в стандартный поток вывода.

**Тестовые пример**

> 1

Введите тип фигуры (1 - квадрат, 2 - треугольник, 3 - восьмиугольник) и её координаты: 1 0 0 2 2 0 2 2 0

> 1

Введите тип фигуры (1 - квадрат, 2 - треугольник, 3 - восьмиугольник) и её координаты: 2 0 0 3.464 0 1.732 3

> 1

Введите тип фигуры (1 - квадрат, 2 - треугольник, 3 - восьмиугольник) и её координаты: 3 0 0 2 0 3 2 3 4 2 6 0 6 -1 4 -1 2

> 2

[1] Квадрат: ( 0 , 0 ), ( 2 , 2 ), ( 0 , 2 ), ( 2 , 0 ).

[2] Треугольник: ( 0 , 0 ), ( 3.464 , 0 ), ( 1.732 , 3 ).

[3] Восьмиугольник: ( 0 , 0 ), ( 2 , 0 ), ( 3 , 2 ), ( 3 , 4 ), ( 2 , 6 ), ( 0 , 6 ), ( -1 , 4 ), ( -1 , 2 ).

> 3

[1]: ( 1 , 1 ) .

[2]: ( 1.732 , 1 ) .

[3]: ( 1 , 3 ) .

> 4

[1] S = 8

[2] S = 5.19585

[3] S = 19.3137

> 5

Общая площадь фигур S = 32.5096

> 6

Введите номер удаляемой фигуры: 5

Ошибка! Указан некорректный номер фигуры!

> 6

Введите номер удаляемой фигуры: 2

Фигура №2 удалена.

> 1

Введите тип фигуры (1 - квадрат, 2 - треугольник, 3 - восьмиугольник) и координаты ее вершин: 3 5 5 8 5 9 8 9 11 8 14 5 14 4 11 4 8

> 1

Введите тип фигуры (1 - квадрат, 2 - треугольник, 3 - восьмиугольник) и координаты ее вершин: 1 -5 -5 -1 -5 -1 -1 -5 -1

> 3

[1]: ( 1 , 1 ) .

[2]: ( 1 , 3 ) .

[3]: ( 6.5 , 9.5 ) .

[4]: ( -3 , -3 ) .

> 4

[1] S = 8

[2] S = 19.3137

[3] S = 43.4558

[4] S = 16

> 5

Общая площадь фигур S = 86.7696

> 0

1. **Листинг программы**

Исходный код программы состоит из трех файлов: **figures.hpp** - содержит объявление классов **Figure**, **Triangle**, **Square** и **Octagon**; **figures.cpp** - содержит реализацию всех методов классов **Figure**, **Triangle**, **Square** и **Octagon**; **main.cpp** это основная программа. **CMakeLists.txt** - содержит код для утилиты cmake, которая проводит сборку программы.

**Содержимое файла figures.hpp**

**/\***Ляшун Дмитрий Сергеевич, группа М8О-207Б-19\*/

#include <vector>

#pragma once

struct Point {

double x;

double y;

};

class Figure {

public :

Figure(int count) { points.resize(count);}

Point Get\_center();

virtual double Get\_square() = 0;

virtual void Output();

void Input();

double Get\_side();

private :

std::vector<Point> points;

};

class Square : public Figure {

public :

Square() : Figure(4) {}

virtual double Get\_square() override;

virtual void Output() override;

};

class Triangle : public Figure {

public :

Triangle() : Figure(3) {}

virtual double Get\_square() override;

virtual void Output() override;

};

class Octagon : public Figure {

public :

Octagon() : Figure(8) {}

virtual double Get\_square() override;

virtual void Output() override;

};

**Содержимое файла figures.cpp**

**/\***Ляшун Дмитрий Сергеевич, группа М8О-207Б-19\*/

#include "figures.hpp"

#include <iostream>

#include <cmath>

Point Figure::Get\_center() {

Point answer{0, 0};

for (int i = 0; i < points.size(); ++i)

{

answer.x += points[i].x;

answer.y += points[i].y;

}

answer.x /= points.size();

answer.y /= points.size();

return answer;

}

double Figure::Get\_side() {

double v\_x = points[1].x - points[0].x;

double v\_y = points[1].y - points[0].y;

return sqrt(v\_x\*v\_x+v\_y\*v\_y);

}

void Figure::Input() {

for (int i = 0; i < points.size(); ++i) {

std::cin >> points[i].x >> points[i].y;

}

}

void Figure::Output() {

for (int i = 0; i < points.size(); ++i) {

std::cout << "( " << points[i].x << " , " << points[i].y << " )";

if (i != points.size()-1) {

std::cout << ", ";

}

}

std::cout << ".";

}

void Square::Output() {

std::cout << "Квадрат: ";

Figure::Output();

}

double Square::Get\_square() {

return pow(Get\_side(), 2);

}

double Triangle::Get\_square() {

return pow(Get\_side(), 2)\*sqrt(3)/4;

}

void Triangle::Output() {

std::cout << "Треугольник: ";

Figure::Output();

}

double Octagon::Get\_square() {

return 8\*pow(Get\_side(), 2) / (4\*tan(acos(-1)/8));

}

void Octagon::Output() {

std::cout << "Восьмиугольник: ";

Figure::Output();

}

**Содержимое файла main.cpp**

**/\***Ляшун Дмитрий Сергеевич, группа М8О-207Б-19\*/

#include "figures.hpp"

#include <iostream>

#include <vector>

using namespace std;

int main() {

cout << "Введите номер необходимой операции: " << endl;

cout << "\t1 - Добавить новую фигуру в массив. " << endl;

cout << "\t2 - Вывести координаты фигур. " << endl;

cout << "\t3 - Вывести геометрические центры фигур. " << endl;

cout << "\t4 - Вывести площади фигур. " << endl;

cout << "\t5 - Вывести общую площадь фигур. " << endl;

cout << "\t6 - Удалить фигуру из массива." << endl;

cout << "\t0 - Выход из программы." << endl;

vector<Figure\*> figures;

while (true) {

int oper;

cout << "> ";

cin >> oper;

switch (oper) {

case 0:

return 0;

case 1: {

cout << "Введите тип фигуры (1 - квадрат, 2 - треугольник, 3 - восьмиугольник) и координаты её вершин: ";

int type;

cin >> type;

switch (type) {

case 1: {

Square\* sq = new Square;

sq->Input();

figures.push\_back(sq);

break;

}

case 2: {

Triangle\* tr = new Triangle;

tr->Input();

figures.push\_back(tr);

break;

}

case 3: {

Octagon\* oc = new Octagon;

oc->Input();

figures.push\_back(oc);

break;

}

default:

cout << "Ошибка! Указан некорректный тип фигуры!" << endl;

break;

}

break;

}

case 2: {

for (int i = 0; i < figures.size(); ++i) {

cout << "[" << i+1 << "] ";

figures[i]->Output();

cout << endl;

}

break;

}

case 3: {

for (int i = 0; i < figures.size(); ++i) {

Point center = figures[i]->Get\_center();

cout << "[" << i+1 << "]: " << "( " << center.x << " , " << center.y << " ) ." << endl;

}

break;

}

case 4: {

for (int i = 0; i < figures.size(); ++i) {

cout << "[" << i+1 << "] S = " << figures[i]->Get\_square() << endl;

}

break;

}

case 5: {

double all\_square = 0;

for (int i = 0; i < figures.size(); ++i) {

all\_square += figures[i]->Get\_square();

}

cout << "Общая площадь фигур S = " << all\_square << endl;

break;

}

case 6: {

cout << "Введите номер удаляемой фигуры: ";

int number;

cin >> number;

--number;

if (0 <= number && number < figures.size()) {

figures.erase(figures.begin() + number);

cout << "Фигура №" << number+1 << " удалена. " << endl;

}

else {

cout << "Ошибка! Указан некорректный номер фигуры!" << endl;

}

break;

}

default:

cout << "Ошибка! Указан некорректный номер опреации!" << endl;

break;

}

}

}

**Содержимое файла CMakeLists.txt**

cmake\_minimum\_required(VERSION 2.8) # Проверка версии CMake.

project(oop\_exercise\_03) # Название проекта

set(SOURCE\_EXE main.cpp) # Установка переменной со списком исходников для исполняемого файла

set(SOURCE\_LIB figures.cpp) # Тоже самое, но для библиотеки

add\_library(figures STATIC ${SOURCE\_LIB}) # Создание статической библиотеки с именем foo

add\_executable(main ${SOURCE\_EXE}) # Создает исполняемый файл с именем main

target\_link\_libraries(main figures) # Линковка программы с библиотекой

**5. Выводы**

В результате выполнения данной лабораторной работы я познакомился с механизмом наследования классов, основными принципами работы абстрактных классов и виртуальных функций в них.

**Список литературы**

1. Справочник по языку C++ [Электронный ресурс]. URL: [http://www.cplusplus.com/](about:blank) (дата обращения: 13.10.2020).