МИНИСТЕРСТВО НАУКИ и высшего образования

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**Отчёт по лабораторной работе № 5**

**по учебной дисциплине «Статические, константные и дружественные члены класса»**

**Тема: «Структура консольного приложения в C#»**

Выполнил(а) студент(ка)

специальности 09.02.03

Программирование в компьютерных

системах

IV курса группы 419/7

Косовский Семен

Андреевич

Преподаватель

Молькова Лолита Юрьевна

Санкт-Петербург,

2024

Цель работы:

изучить работу со статическими и константными членами класса.

Задание:

Практическое задание (100%)

Для задания практической работы 6 «Перегрузка операций» выполнить доработку

программы в соответствии с заданиями:

ЗАДАНИЕ 1. Статические члены-данные класса

1. Добавить статическое поле int сount, выступающее в роли счетчика объектов

класса.

2. Деструктор класса должен уменьшать на единицу значение счетчика.

3. Добавить статический метод int getCount() возвращающий значение счетчика.

4. Продемонстрировать изменение значения статического поля.

ЗАДАНИЕ 2. Константные методы и объекты

1. Определить какие методы являются константными, определить константные

параметры и константные возвращаемые значения методов.

2. Добавить константное поле, хранящее идентификатор объекта (номер созданного

объекта), предусмотреть методы вывода информации о идентификаторе.

3. Описать и инициализировать обычные и константные объекты.

4. Выполнить вызовы обычных и константных методов для каждого вида объектов.

5. Провести тестирование программы: Откомпилировать программу. Имеются ли

ошибки компиляции и какие? Если имеются, то закомментировать

соответствующие строки кода и вновь провести компиляцию. Какие

предупреждения выдает компилятор и в чем их смысл? Как их можно объяснить с

позиции обеспечения надежности программы?  
Вариант 8.

**Ход работы:**

На рисунке 1 изображена вся работы программы

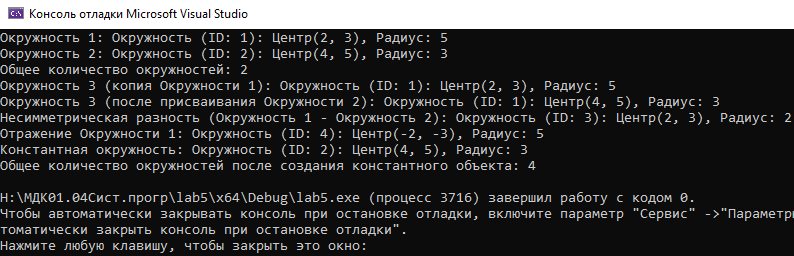


Рисунок 1 – Вся работы программы

Код:

Заголовочный файл Circle.h  
#ifndef CIRCLE\_H

#define CIRCLE\_H

#include <iostream>

#include <cmath>

class Circle {

public:

Circle(double x, double y, double radius);

Circle(const Circle& other); // Конструктор копирования

Circle& operator=(const Circle& other); // Перегрузка оператора присваивания

~Circle(); // Деструктор

void display() const; // Метод для отображения информации об окружности

Circle operator-(const Circle& other) const; // Несимметрическая разность

Circle operator-() const; // Унарное отражение

static int getCount(); // Статический метод для получения значения счетчика

private:

double x; // x-координата центра

double y; // y-координата центра

double radius; // Радиус окружности

static int count; // Статический член-данные для подсчета объектов

const int id; // Константное поле для идентификатора объекта

};

#endif // CIRCLE\_H

Файл реализации методов класса Circle.cpp  
#include "Circle.h"

// Инициализация статического поля count

int Circle::count = 0;

// Конструктор

Circle::Circle(double x, double y, double radius)

: x(x), y(y), radius(radius), id(++count) { // Увеличиваем счетчик и присваиваем идентификатор

if (radius < 0) {

this->radius = 0; // Убедиться, что радиус не отрицательный

}

}

// Конструктор копирования

Circle::Circle(const Circle& other)

: x(other.x), y(other.y), radius(other.radius), id(other.id) {}

// Перегрузка оператора присваивания

Circle& Circle::operator=(const Circle& other) {

// Проверка на самоприсваивание

if (this != &other) {

x = other.x;

y = other.y;

radius = other.radius;

}

return \*this; // Возврат текущего объекта

}

// Деструктор

Circle::~Circle() {

--count; // Уменьшаем счетчик при уничтожении объекта

}

// Метод для отображения информации об окружности

void Circle::display() const {

std::cout << "Окружность (ID: " << id << "): Центр(" << x << ", " << y << "), Радиус: " << radius << std::endl;

}

// Несимметрическая разность двух окружностей

Circle Circle::operator-(const Circle& other) const {

double newRadius = radius - other.radius; // Разность радиусов

if (newRadius < 0) {

return Circle(0, 0, 0); // Вернуть окружность с радиусом ноль, если результат отрицательный

}

return Circle(x, y, newRadius); // Сохранить центр таким же

}

// Унарное отражение окружности

Circle Circle::operator-() const {

return Circle(-x, -y, radius); // Отразить окружность относительно начала координат

}

// Статический метод для получения значения счетчика

int Circle::getCount() {

return count; // Возвращает текущее значение счетчика

}

3. Основной файл main.cpp  
#include <iostream>

#include "Circle.h"

int main() {

setlocale(LC\_ALL, "rus");

Circle c1(2.0, 3.0, 5.0); // Создаем окружность с центром (2, 3) и радиусом 5

const Circle c2(4.0, 5.0, 3.0); // Создаем константную окружность с центром (4, 5) и радиусом 3

std::cout << "Окружность 1: ";

c1.display(); // Отображаем первую окружность

std::cout << "Окружность 2: ";

c2.display(); // Отображаем вторую окружность

// Использование статического метода getCount

std::cout << "Общее количество окружностей: " << Circle::getCount() << std::endl;

// Использование оператора присваивания

Circle c3 = c1; // Копируем c1 в c3

std::cout << "Окружность 3 (копия Окружности 1): ";

c3.display(); // Отображаем третью окружность

c3 = c2; // Присваиваем c2 переменной c3

std::cout << "Окружность 3 (после присваивания Окружности 2): ";

c3.display(); // Отображаем обновлённую третью окружность

// Несимметрическая разность

Circle c4 = c1 - c2;

std::cout << "Несимметрическая разность (Окружность 1 - Окружность 2): ";

c4.display(); // Отображаем результат разности

// Унарное отражение

Circle c5 = -c1;

std::cout << "Отражение Окружности 1: ";

c5.display(); // Отображаем отражённую окружность

// Вызов константного метода

std::cout << "Константная окружность: ";

c2.display(); // Отображаем константную окружность

std::cout << "Общее количество окружностей после создания константного объекта: " << Circle::getCount() << std::endl;

return 0;

}**Circle**

| Метод | Описание |
| --- | --- |
| Circle(double x, double y, double radius) | Конструктор класса, который инициализирует центр окружности координатами (x, y) и радиусом radius. Увеличивает статический счетчик объектов. |
| ~Circle() | Деструктор класса, который уменьшает статический счетчик объектов на единицу при уничтожении объекта. |
| Circle(const Circle &other) | Конструктор копирования, создающий новый объект Circle, копируя данные из существующего объекта other. |
| Circle& operator=(const Circle &other) | Перегрузка оператора присваивания, позволяющая присваивать один объект Circle другому, корректно обрабатывая копирование данных и предотвращая утечки памяти. |
| void display() const | Метод для вывода информации об окружности (координаты центра и радиус). Является константным методом, так как не изменяет состояние объекта. |
| double area() const | Метод, который возвращает площадь окружности. Также является константным методом. |
| double circumference() const | Метод, который возвращает длину окружности. Является константным методом. |