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 № 4**

**по учебной дисциплине «Конструкторы и деструкторы»**

**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 механизм перегрузки операций.

Задание:

Практическое задание (100%)

Все классы следует наделить конструкторами, деструктором. Необходимо явно

реализовать конструктор копирования и перегрузить оператор присваивания. Необходимо

подготовить демонстрацию по работе перегруженных для класса операторов.  
Вариант 8.

**Ход работы:**

На рисунке 1 изображена вся работы программы

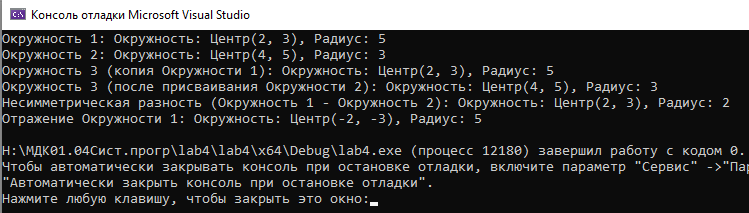


Рисунок 1 – Вся работы программы

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

private:

double x; // x-координата центра

double y; // y-координата центра

double radius; // Радиус окружности

};

#endif // CIRCLE\_H

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

#include <cstring>

#include <fstream>

#include <utility>

// Конструктор по умолчанию

MyString::MyString() : str(nullptr), length(0) {

std::cout << "Конструктор по умолчанию вызван" << std::endl;

}

// Конструктор с параметрами

MyString::MyString(const char\* s) {

std::cout << "Конструктор с параметрами вызван" << std::endl;

length = strlen(s);

str = new char[length + 1];

strcpy\_s(str, length + 1, s); // Используем безопасную функцию strcpy\_s

}

// Конструктор копирования

MyString::MyString(const MyString& other) {

std::cout << "Конструктор копирования вызван" << std::endl;

length = other.length;

str = new char[length + 1];

strcpy\_s(str, length + 1, other.str); // Используем безопасную функцию strcpy\_s

}

// Деструктор

MyString::~MyString() {

std::cout << "Деструктор вызван" << std::endl;

delete[] str;

}

// Метод для ввода строки

void MyString::set(const char\* s) {

std::cout << "Метод set() вызван" << std::endl;

delete[] str; // Освобождаем старую строку, если она была

length = strlen(s);

str = new char[length + 1];

strcpy\_s(str, length + 1, s); // Используем безопасную функцию strcpy\_s

}

// Остальная часть кода остаётся без изменений...

// Метод для изменения строки

void MyString::update() {

std::cout << "Метод update() вызван" << std::endl;

// Проверка, что длина строки кратна 4

if (length % 4 == 0 && length > 0) {

// Сохраняем исходную строку для записи в файл

std::string originalStr(str);

// Меняем первую половину строки с второй

size\_t half\_length = length / 2;

for (size\_t i = 0; i < half\_length; ++i) {

std::swap(str[i], str[half\_length + i]);

}

// Открываем файл для записи

std::ofstream outFile("output.txt");

if (outFile.is\_open()) {

outFile << "Исходная строка: " << originalStr << std::endl;

outFile << "Измененная строка: " << str << std::endl;

outFile.close();

std::cout << "Результат сохранен в файл output.txt" << std::endl;

}

else {

std::cerr << "Ошибка при открытии файла!" << std::endl;

}

}

else {

std::cout << "Длина строки не кратна 4, изменения не произведены." << std::endl;

}

}

// Метод для вывода строки на экран

void MyString::print() const {

std::cout << "Метод print() вызван" << std::endl;

if (str != nullptr) {

std::cout << "Строка: " << str << std::endl;

}

else {

std::cout << "Строка пуста" << std::endl;

}

}

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

Circle c2(4.0, 5.0, 3.0); // Создаем 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 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

return 0;

}

**Circle**

| Метод | Описание |
| --- | --- |
| Circle(double x, double y, double radius) | Конструктор, который инициализирует окружность с заданными координатами центра и радиусом. Если радиус отрицательный, он устанавливается в ноль. |
| Circle(const Circle& other) | Конструктор копирования, который создает новый объект Circle, копируя данные из другого объекта Circle. |
| Circle& operator=(const Circle& other) | Перегрузка оператора присваивания. Позволяет копировать данные из одного объекта Circle в другой, включая проверку на самоприсваивание. |
| ~Circle() | Деструктор, который освобождает ресурсы, занимаемые объектом. В этом случае он пустой, так как нет динамической памяти. |
| void display() const | Метод, который выводит на экран координаты центра окружности и её радиус. |
| Circle operator-(const Circle& other) const | Перегрузка бинарного оператора несимметрической разности, который вычисляет разность радиусов двух окружностей. Если результат отрицательный, возвращает окружность с радиусом ноль. |
| Circle operator-() const | Перегрузка унарного оператора, который отражает окружность относительно начала координат, меняя знак координат центра. |