МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСТИЕТ

Кафедра вычислительной техники

Лабораторная работа №1

по дисциплине «Программирование»

на тему: «Разработка классов, создание конструкторов и деструкторов. Использование статических членов класса»

Группа: АВТ-441

Студент: Тютрин Влад

Преподаватель: Орлов Илья Викторович

Новосибирск 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc208001212)

[1 Описание разрабатываемого класса 4](#_Toc208001213)

[2 Программная реализация 5](#_Toc208001214)

[3 Тестирование 6](#_Toc208001215)

[ЗАКЛЮЧЕНИЕ 7](#_Toc208001216)

[ПРИЛОЖЕНИЕ 8](#_Toc208001217)

ВВЕДЕНИЕ

**Цель работы:**

Тема: Разработка классов, создание конструкторов и деструкторов. Использование статических членов класса Цель работы: Изучить структуру класса, механизм создания и использования, описание членов-данных класса и методов доступа к ним, возможность инициализации объектов класса с помощью конструкторов и уничтожение их с помощью деструкторов. Задания на лабораторную работу Реализовать класс в соответствии с вариантом. Класс должен обеспечивать набор методов для работы с данными. Создать пеpегpуженные констpуктоpы: констpуктоp копирования, констpуктоp по умолчанию, конструктор с параметрами. Реализовать в классах поля с динамическим выделением памяти для хранения некоторых данных. Создать деструктор для освобождения памяти. Изучить порядок вызова конструкторов и деструктора. Обязательно добавить в класс статические члены. Составить демонстpационную пpогpамму, реализующую вызов всех методов класса. Для демонстpационной пpогpаммы использовать отдельный модуль. Пpогpамму постpоить с использованием пpоекта. Посмотpеть pаботу пpогpаммы в отладчике, обpатить внимание на пpедставление данных. Постpоить пpогpамму без отладочной инфоpмации. Обpатить внимание на pазмеp пpогpаммы.

Постpоить класс для pаботы со cтpоками. Класс должен включать следующие поля: указатель на стpоку, ее длину. Класс должен обеспечивать пpостейшие функции для pаботы со строками: изменение стpоки, вывод стpоки на экран, нахождение подстpоки в cтpоке, объединение двух строк и т.д

**Вариант:** 10

1. Описание разрабатываемого класса

Разработанный класс String предназначен для работы со строками символов и предоставляет функциональность для управления строковыми данными с использованием динамического выделения памяти. Класс инкапсулирует указатель на массив символов и длину строки, обеспечивая безопасные операции над текстовыми данными.

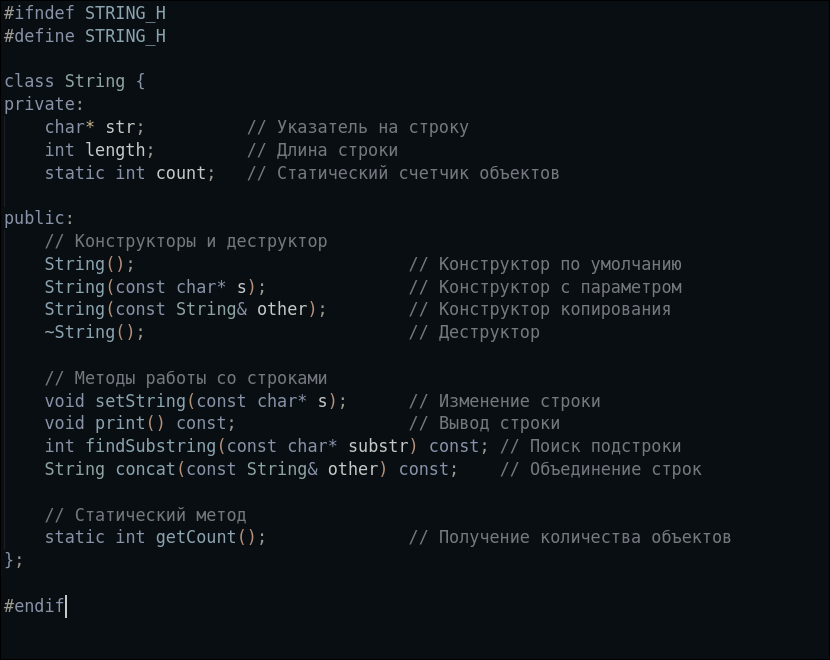
Основной особенностью класса является реализация механизмов управления памятью через конструкторы и деструктор. Конструктор по умолчанию создает пустую строку, конструктор с параметром инициализирует объект переданной строкой, а конструктор копирования обеспечивает глубокое копирование данных. Деструктор автоматически освобождает выделенную память при уничтожении объекта.

Класс включает методы для изменения содержимого строки, вывода на экран, поиска подстроки и объединения строк. Поиск подстроки реализован с возвратом позиции первого вхождения, а объединение строк создает новый объект, содержащий результирующую строку.

В качестве статического члена класса реализован счетчик объектов, который отслеживает текущее количество экземпляров класса в программе. Это позволяет мониторить использование памяти и контролировать жизненный цикл объектов. Все методы класса обеспечивают целостность данных и корректную работу с динамической памятью.

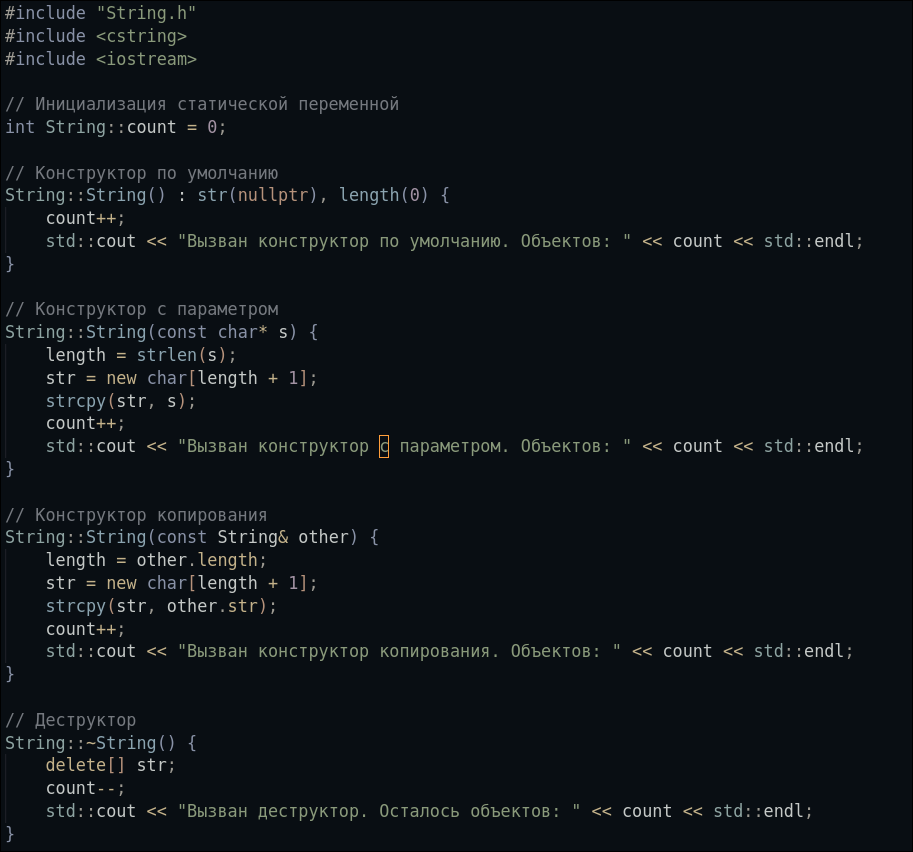
1. Программная реализация

Программная реализация проекта выполнена в соответствии с принципами объектно-ориентированного программирования и состоит из трех основных модулей: заголовочного файла, файла реализации и демонстрационной программы.

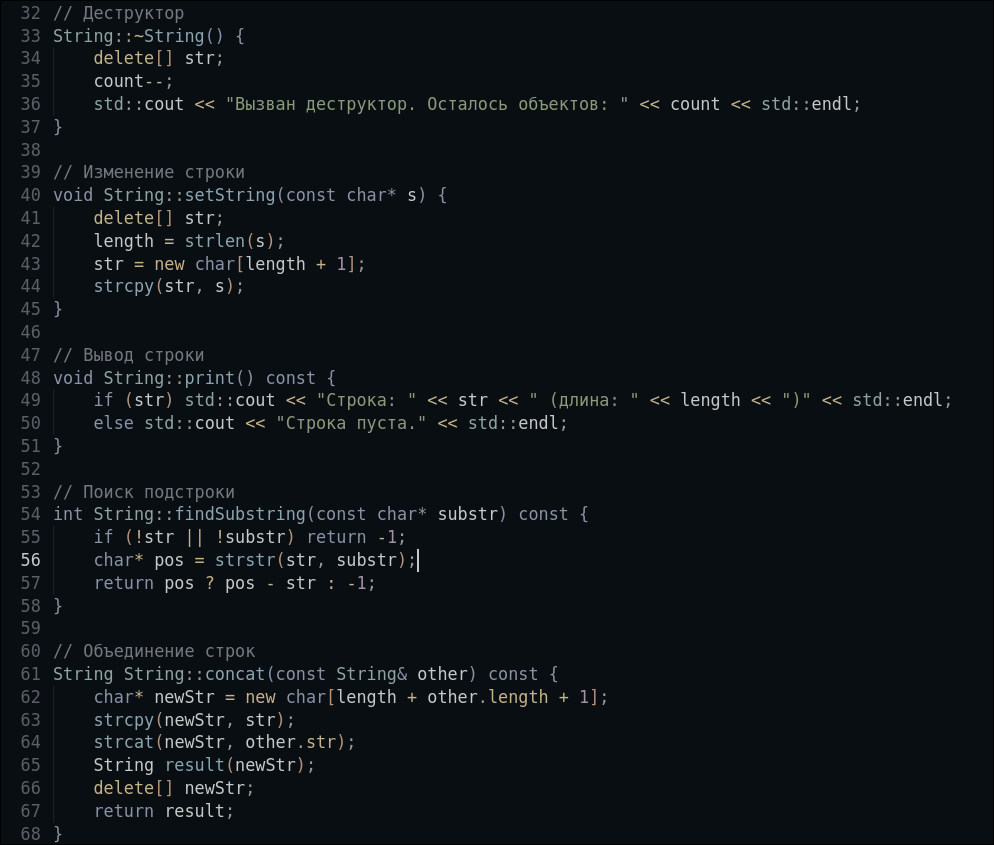


Заголовочный файл String.h содержит объявление класса String с полями для хранения указателя на строку и ее длины, а также статической переменной-счетчика объектов. В открытой секции класса определены конструкторы (по умолчанию, с параметром, копирования), деструктор и методы для работы со строками: изменения содержимого, вывода на экран, поиска подстроки и объединения строк. Также объявлен статический метод для получения текущего количества объектов.

Файл реализации String.cpp содержит детальную реализацию всех методов класса. В конструкторах выполняется инициализация полей класса и динамическое выделение памяти под строку, при этом статический счетчик увеличивается при создании каждого нового объекта.



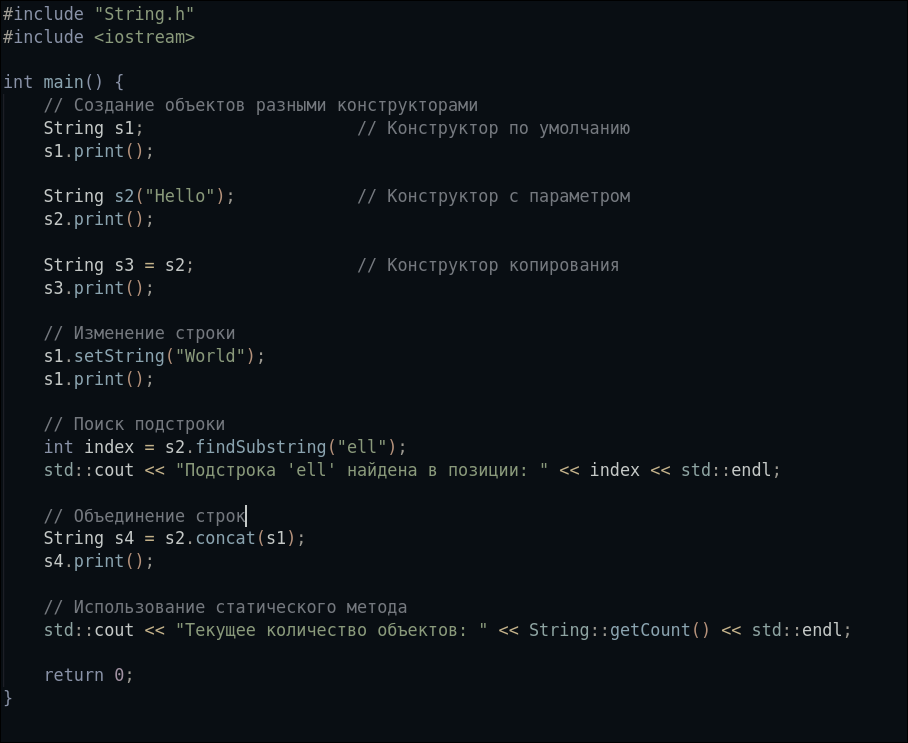
Деструктор обеспечивает освобождение выделенной памяти и уменьшение значения счетчика. Метод setString() позволяет безопасно изменять содержимое строки с предварительным освобождением предыдущей памяти.



Поиск подстроки реализован с использованием стандартной функции strstr, а объединение строк создает временный буфер для формирования результирующей строки.



Демонстрационная программа main.cpp создает несколько объектов класса с использованием различных конструкторов, тестирует все реализованные методы и отслеживает изменение количества объектов через статический метод. Программа наглядно демонстрирует порядок вызова конструкторов и деструкторов, а также корректность работы всех операций со строками.



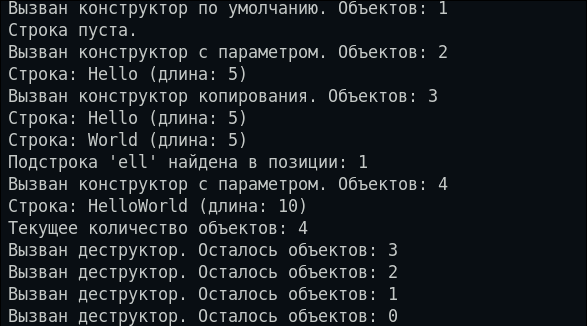
Для сборки проекта использовался компилятор g++ с параметрами по умолчанию. Программа успешно компилируется и выполняется без ошибок памяти, что подтверждает корректность реализации механизмов управления динамической памятью.

1. Тестирование

Ожидаемые результаты:

Мы ждем полную реализацию задания, с выводом всех промежуточных процессов, для отладки.

Результат:



ЗАКЛЮЧЕНИЕ

В ходе выполнения Лабораторной работы задание было выполненно без проблем, реализация полная.

ПРИЛОЖЕНИЕ

String.h

#ifndef STRING\_H

#define STRING\_H

class String {

private:

char\* str; // Указатель на строку

int length; // Длина строки

static int count; // Статический счетчик объектов

public:

// Конструкторы и деструктор

String(); // Конструктор по умолчанию

String(const char\* s); // Конструктор с параметром

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

~String(); // Деструктор

// Методы работы со строками

void setString(const char\* s); // Изменение строки

void print() const; // Вывод строки

int findSubstring(const char\* substr) const; // Поиск подстроки

String concat(const String& other) const; // Объединение строк

// Статический метод

static int getCount(); // Получение количества объектов

};

#endif

String.cpp

#include "String.h"

#include <cstring>

#include <iostream>

// Инициализация статической переменной

int String::count = 0;

// Конструктор по умолчанию

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

count++;

std::cout << "Вызван конструктор по умолчанию. Объектов: " << count << std::endl;

}

// Конструктор с параметром

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

length = strlen(s);

str = new char[length + 1];

strcpy(str, s);

count++;

std::cout << "Вызван конструктор с параметром. Объектов: " << count << std::endl;

}

// Конструктор копирования

String::String(const String& other) {

length = other.length;

str = new char[length + 1];

strcpy(str, other.str);

count++;

std::cout << "Вызван конструктор копирования. Объектов: " << count << std::endl;

}

// Деструктор

String::~String() {

delete[] str;

count--;

std::cout << "Вызван деструктор. Осталось объектов: " << count << std::endl;

}

// Изменение строки

void String::setString(const char\* s) {

delete[] str;

length = strlen(s);

str = new char[length + 1];

strcpy(str, s);

}

// Вывод строки

void String::print() const {

if (str) std::cout << "Строка: " << str << " (длина: " << length << ")" << std::endl;

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

}

// Поиск подстроки

int String::findSubstring(const char\* substr) const {

if (!str || !substr) return -1;

char\* pos = strstr(str, substr);

return pos ? pos - str : -1;

}

// Объединение строк

String String::concat(const String& other) const {

char\* newStr = new char[length + other.length + 1];

strcpy(newStr, str);

strcat(newStr, other.str);

String result(newStr);

delete[] newStr;

return result;

}

// Получение количества объектов

int String::getCount() {

return count;

}

main.cpp

#include "String.h"

#include <iostream>

int main() {

// Создание объектов разными конструкторами

String s1; // Конструктор по умолчанию

s1.print();

String s2("Hello"); // Конструктор с параметром

s2.print();

String s3 = s2; // Конструктор копирования

s3.print();

// Изменение строки

s1.setString("World");

s1.print();

// Поиск подстроки

int index = s2.findSubstring("ell");

std::cout << "Подстрока 'ell' найдена в позиции: " << index << std::endl;

// Объединение строк

String s4 = s2.concat(s1);

s4.print();

// Использование статического метода

std::cout << "Текущее количество объектов: " << String::getCount() << std::endl;

return 0;

}