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 № 3**

**по учебной дисциплине «Системное программирование»**

**Тема: «Конструкторы и деструкторы»**

Выполнил студент

специальности 09.02.07

Информационные системы и

программирование

IV курса группы 42919/4

Марков Данил

Петрович

Преподаватель

Молькова Лолита Юрьевна

Санкт-Петербург

2023

# **Цели работы:**

Изучение возможности инициализации объектов класса с помощью конструкторов и уничтожение их с помощью деструкторов.

# **Задание:**

* Пользовательский класс MyString должен содержать необходимые элементы данные, которые создаются в динамической области памяти.
* Конструкторы (без параметров, с параметрами, копирования) для создания строк: MyString (…);
* Деструктор: ~MyString();
* Метод ввода исходной строки: set();
* Метод изменения исходной строки согласно варианту (исходная и измененная строка должны сохраняться в файле): update() ;
* Метод вывода на экран: print(…);
* Каждый вызов методов (в том числе конструкторов и деструктора) сопровождается выдачей соответствующего сообщения; Код методов – вне пространства определения класса.
* Написать демонстрационную программу, в которой показать использование объектов созданного класса.
* Длина L кратна 3-м, то удаляются все цифры, делящиеся на 3;

# **Ход работы**:

В ходе работы был разработан класс MyString для управления строками, включая конструктор по умолчанию, конструктор с параметром, конструктор копирования и деструктор. Класс также обеспечивает методы для установки и получения значений строки. Кроме того, в классе реализован функционал обработки строки, согласно указаниям лабораторной работы.

Код файла MyString.h:

#pragma once

#include <iostream>

#include <cstring>

#include <fstream>

class MyString

{

private:

char\* data;

public:

MyString();

MyString(const char\* str);

MyString(const MyString& other);

~MyString();

void set(const char\* str);

void update();

void print();

};

Код файла MyString.cpp:

#include "MyString.h"

#include <iostream>

#include <cstring>

#include <fstream>

#define \_CRT\_SECURE\_NO\_WARNINGS

using namespace std;

MyString::MyString()

{

data = nullptr;

cout << "Default constructor called." << endl;

}

MyString::MyString(const char\* str)

{

data = new char[strlen(str) + 1];

strcpy\_s(data, strlen(str) + 1, str);

cout << "Parameterized constructor called." << endl;

}

MyString::MyString(const MyString& other)

{

data = new char[strlen(other.data) + 1];

strcpy\_s(data, strlen(other.data) + 1, other.data);

cout << "Copy constructor called." << endl;

}

MyString::~MyString()

{

if (data) {

delete[] data;

}

cout << "Destructor called." << endl;

}

void MyString::set(const char\* str)

{

cout << "Set string";

data = new char[strlen(str) + 1];

strcpy\_s(data, strlen(str) + 1, str);

}

void MyString::update()

{

int length = strlen(data);

if (length % 3 == 0) {

char\* newData = nullptr;

int letterCounter = 0;

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

if (!isdigit(data[i]) || (isdigit(data[i]) && data[i] % 3 != 0)) {

letterCounter++;

}

}

newData = new char[letterCounter + 1];

newData[letterCounter] = '\0';

for (size\_t i = 0, j = 0; i < length; i++) {

if (!isdigit(data[i]) || (isdigit(data[i]) && data[i] % 3 != 0)) {

newData[j] = data[i];

j++;

}

}

delete[] data;

data = newData;

}

}

void MyString::print()

{

if (data) {

cout << "String: " << data << endl;

}

else {

cout << "String is empty." << endl;

}

}

Код файла Main.cpp:

#include <iostream>

#include "MyString.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "ru");

MyString str1;

str1.set("Строка номер один");

str1.print();

MyString str2("1Строка333 666номер999 2");

str2.print();

MyString str3 = str2;

cout << "\nНиже не строка 2, а строка 3 в которую скопировали строку 2" << endl;

str3.print();

cout << "\nОбновляем по заданию строку 2:" << endl;

str2.update();

str2.print();

// Демонстрация деструктора

{

MyString str4("Temporary String");

} // str4 выходит из области видимости и вызывается деструктор

return 0;

}

На рисунке 1 представлен, результат работы вызова дефолтного конструктора, сеттора, конструктора с параметром, вывод начальной строки, конструктора копирования, после чего вызывается метод из задания, и выводится строка после обработки.

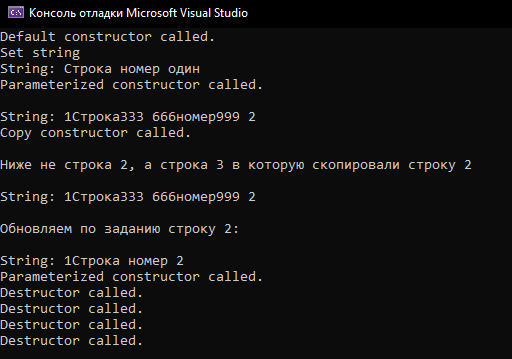


Рисунок 1 – Результат работы программы

# **Контрольные вопросы**:

1. Что такое конструктор?

Конструктор - это метод класса, который автоматически вызывается при создании объекта этого класса. Он инициализирует объект, устанавливая начальные значения его членов.

1. Как задается имя конструктора?

Имя конструктора совпадает с именем класса.

1. Может ли класс иметь более одного конструктора?

Да, класс может иметь несколько конструкторов, каждый из которых может иметь разные параметры (перегрузка конструкторов).

1. Что такое деструктор?

Деструктор - это метод класса, который автоматически вызывается при уничтожении объекта. Он используется для освобождения ресурсов, выделенных объекту.

1. В чем состоит преимущество определения конструктора со списком инициализации элементов?

Список инициализации позволяет инициализировать члены класса непосредственно при создании объекта, что может быть более эффективным и предотвращает дополнительные присваивания после создания объекта.

1. Какие виды конструкторов создаются по умолчанию?

По умолчанию создаются конструктор по умолчанию (без параметров), конструктор копирования и деструктор. Если вы явно не предоставите эти методы, компилятор сгенерирует их автоматически.

1. В каком порядке инициализируются поля в классе?

Поля в классе инициализируются в порядке их объявления в классе, не в порядке, указанном в списке инициализации конструктора.

1. Какая ошибка в следующей реализации конструктора?
2. Может ли деструктор иметь аргументы?

Нет, деструктор не принимает аргументов.

1. Какая ошибка в следующей реализации деструктора?