1 ПОСТАНОВКА ЗАДАЧИ

Создать класс для работы со строками. Память под строки выделять динамически. Определить конструктор без параметров, конструктор с параметрами. Реализовать методы: ввод данных в строку, вывод строки на экран, объединение строк. Проверить работу методов этого класса.

2 ЛИСТИНГ КОДА

Файл main.cpp

```
#include "program.h"
#include <iostream>

int main() {
    Program program;

    program.run();
    return 0;
}
```

Файл my_string.h

```
#pragma once

class String {
    char *data;
    int len;
    int cap;

  public:
    String();
    explicit String(char *&str);
    String(const String &other);
    String(String &&move) noexcept;
    ~String();
    String &operator=(const String &other);
    String &operator=(String &&move) noexcept;

    void input(const char *msg);
    void show(const char *msg) const;
    void concatenate(String str);
    bool isEmpty() const;
};
```

Файл my_string.cpp

```
#include "my_string.h"
#include "utils.h"
#include <iostream>

String::String() : data(nullptr), len(0), cap(0) {
}

String::String(char *&str) : len(myStrlen(str)), cap(len + 1) {
    data = new char[cap];

    for (int i = 0; i < len; i++) {
        data[i] = str[i];
    }

    data[len] = '\0';
    delete[] str;
    str = nullptr;
}

String::~String() {
```

```
    delete[] data;
    data = nullptr;
    len = 0;
    cap = 0;
}

String::String(const String &other) : len(other.len), cap(other.cap) {
    data = new char[cap];

    for (int i = 0; i < len; i++) {
        data[i] = other.data[i];
    }
    data[len] = '\0';
}

String::String(String &&move) noexcept : data(move.data), len(move.cap),
cap(move.cap) {
    move.data = nullptr;
    move.len = 0;
    move.cap = 0;
}

String &String::operator=(const String &other) {
    if (this != &other) {

        cap = other.cap;
        len = other.len;

        delete[] data;
        data = nullptr;

        if (cap > 0) {
            data = new char[cap];

            for (int i = 0; i < len; i++) {
                data[i] = other.data[i];
            }

            data[len] = '\0';
        }
    }
    return *this;
}

String &String::operator=(String &&move) noexcept {
    if (this != &move) {
        cap = move.cap;
        len = move.len;

        delete[] data;
        data = move.data;

        move.data = nullptr;
        move.len = 0;
        move.cap = 0;
    }

    return *this;
}

void String::input(const char *msg) {
    if (data != nullptr) {
        delete[] data;
        data = nullptr;
```

```
    }
    data = getString(msg);
    len = myStrlen(data);
    cap = len + 1;
}

void String::show(const char *msg) const {
    std::cout << msg;
    std::cout << data << std::endl;
}

void String::concatenate(const String str) {
    if (str.data == nullptr || str.len == 0) {
        return;
    }
    int new_len = len + str.len;
    cap = new_len + 1;

    data = resizeString(data, cap);

    for (int ind = 0; ind < str.len; ind++) {
        data[len + ind] = str.data[ind];
    }

    len = new_len;
    data[len] = '\0';
}

bool String::isEmpty() const {
    return (data == nullptr && len == 0 && cap == 0);
}
```

Файл utils.h

```
#pragma once

int myStrlen(const char *str);
char *resizeString(char *&str, int len);
char *getString(const char *msg);
int getDigit();
```

Файл utils.cpp

```
#include "utils.h"
#include "consts.h"
#include <iostream>

int myStrlen(const char *str) {
    if (str == nullptr) {
        return 0;
    }
    int len = 0;

    for (int ind = 0; str[ind] != '\0'; ind++) {
        len++;
    }

    return len;
}

char *resizeString(char *&str, const int len) {
```

```
    auto *new_str = new char[len];

    for (int ind = 0; ind < len; ind++) {
        new_str[ind] = '\0';
    }

    for (int ind = 0; str[ind] != '\0'; ind++) {
        new_str[ind] = str[ind];
    }

    delete[] str;
    str = nullptr;

    return new_str;
}

char *getString(const char *msg) {
    std ::cout << msg;

    int cap = 1;
    auto *str = new char[cap];
    int check = 0;
    int len = 0;
    str[len] = '\0';

    while (true) {
        check = std::cin.get();
        if ((char)check == '\n') {
            break;
        }
        if (len + 1 == cap) {
            cap *= 2;
            str = resizeString(str, cap);
        }

        str[len] = char(check);
        len++;

        if (str[0] == '\0') {
            std::cout << kRedColor << "\nError, the string cannot be
empty.Please try again: " << kWhiteColor;
            len = 0;
            cap = 1;
        }
    }

    str[len] = '\0';
    return str;
}

int getDigit() {
    int check = 0;

    while (true) {
        check = std::cin.get();

        if (check != '\n' && check != EOF) {
            if (isdigit(check) == 1 && std::cin.get() == '\n') {
                return check - '0';
            }

            while (std::cin.get() != '\n' && !std::cin.eof())
                ;
        }
```

```
        std::cout << kRedColor << "\nError, invalid input. Please try again:
" << kWhiteColor;
    }
}
```

Файл program.h

```
#pragma once

#include "my_string.h"

class Program {
    String str;

    void useDefaultStrConstructor();
    void useParameterizedStrConstructor();
    void inputString();
    void showString() const;
    void concatenateStrings();

  public:
    Program();

    void run();
};
```

Файл program.cpp

```
#include "program.h"
#include "consts.h"
#include "menus.h"
#include "my_string.h"
#include "utils.h"
#include <iostream>

void Program::useDefaultStrConstructor() {
    String tmp_str;
    str = tmp_str;
    std::cout << kGreenColor << "The string object was successfully created
using the default constructor!"
              << kWhiteColor << std ::endl;
}

void Program::useParameterizedStrConstructor() {
    char *input = getString("Please enter the string: ");

    String tmp_str(input);
    str = tmp_str;

    std::cout << kGreenColor << "The string object was successfully created
using the constructor with parameters!"
              << kWhiteColor << std ::endl;
}

Program::Program() {
    int opt = 0;

    system("clear");
    showConstructorsMenu();
```

```
    while (true) {
        std::cout << "\nPlease select a constructor menu option: ";
        opt = getDigit();

        switch (opt) {
        case 1:
            useDefaultStrConstructor();
            return;
        case 2:
            useParameterizedStrConstructor();
            return;
        default:
            std::cout << kRedColor << "\nError, you picked is an incorrect
menu option. Please try again."
                      << kWhiteColor << std::endl;
        }
    }
}

void Program::inputString() {
    if (str.isEmpty()) {
        str.input("\nPlease enter the string: ");
        std::cout << kGreenColor << "String successfully entered using String
method(input)!" << kWhiteColor
                  << std::endl;
        return;
    }

    std::cout << kRedColor << "The string has already been entered!" <<
kWhiteColor << std::endl;
}

void Program::showString() const {
    if (str.isEmpty()) {
        std::cout << kRedColor << "\nError, string has not been entered.
Please use the first option and try again!"
                  << kWhiteColor << std::endl;
        return;
    }

    str.show("\nString: ");

    std::cout << kGreenColor << "The string was successfully displayed on the
screen using the String method(show)!"
              << kWhiteColor << std::endl;
}

void Program::concatenateStrings() {
    if (str.isEmpty()) {
        std::cout << kRedColor << "\nError, string has not been entered.
Please use the first option and try again!"
                  << kWhiteColor << std::endl;
        return;
    }

    String src;

    src.input("Please enter a string to append to the original string: ");
    str.concatenate(src);

    std::cout << kGreenColor << "Strings were successfully concatenated!" <<
kWhiteColor << std::endl;
}
```

```
void Program::run() {
    int opt = 0;

    showTaskMenu();

    while (true) {
        std::cout << "\nPlease select a task menu option: ";

        opt = getDigit();

        switch (opt) {
        case 1:
            inputString();
            break;
        case 2:
            showString();
            break;
        case 3:
            concatenateStrings();
            break;
        case 4:
            std::cout << kGreenColor << "\nYou have successfully exited the
program." << kWhiteColor << std::endl;
            return;
        default:
            std::cout << kRedColor << "\nError, you picked is an incorrect
menu option. Please try again."
                      << kWhiteColor << std::endl;
        }
    }
}
```

Файл menus.h

```
#pragma once

void showConstructorsMenu();
void showTaskMenu();
```

Файл menus.cpp

```
#include "menus.h"
#include <iostream>

void showConstructorsMenu() {
    std::cout << "\t\t\tCONSTRUCTORS MENU" << std::endl;
    std::cout << "1.Use default constructor." << std::endl;
    std::cout << "2.Use constructor with parameters." << std::endl;
}

void showTaskMenu() {
    std::cout << "\n\t\t\t\tTASK" << std::endl;
    std::cout << "Create a class for working with strings." << std::endl;
    std::cout << "Allocate memory for strings dynamically." << std::endl;
    std::cout << "Define a default constructor and a parameterized
constructor." << std::endl;
    std::cout << "Implement the following methods: input data into a string,"
<< std::endl;
    std::cout << "display the string, and concatenate strings." << std::endl;
    std::cout << "\n\t\t\t\tMENU" << std::endl;
    std::cout << "1.Enter the string." << std::endl;
    std::cout << "2.Display the string." << std::endl;
```

```
    std::cout << "3.Concatenate the strings." << std::endl;
    std::cout << "4.Exit the program." << std ::endl;
}
```

Файл consts.h

```
#pragma once

inline constexpr const char *kWhiteColor = "\o{33}[0m";
inline constexpr const char *kRedColor = "\o{33}[31m";
inline constexpr const char *kGreenColor = "\o{33}[32m";
```

3 РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ

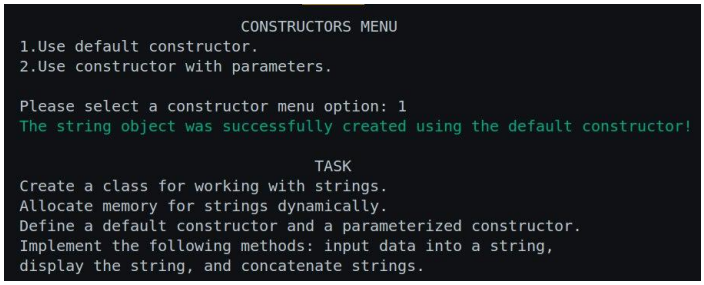

Рисунок 4.1 – Меню конструкторов класса String и описание задания

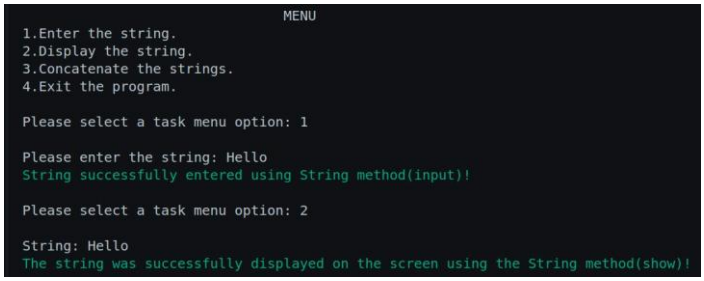

Рисунок 4.2 – Главное меню программы и ввод строки

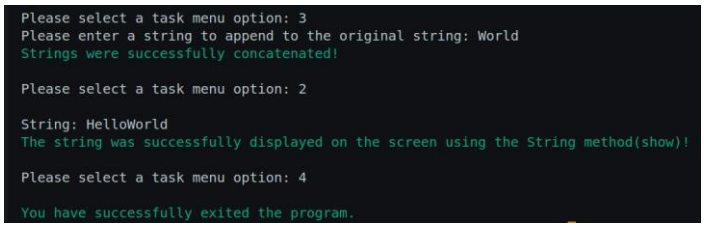

Рисунок 4.3 – Вывод строки на экран, объединение строк и завершение программы

4 ЗАКЛЮЧЕНИЕ

В результате выполнения данной лабораторной работы я закрепил навыки работы с динамическим выделением памяти в C++. Был создан класс для работы со строками, реализованы конструкторы по умолчанию и с параметрами, а также методы ввода, вывода и объединения строк. Кроме того, был реализован класс Program, который является главным связующим звеном программы: через него организовано меню, управление выбором действий пользователя и проверка корректности выполнения методов класса String. В ходе работы программы удалось проверить правильность реализации всех функций и убедиться в корректном взаимодействии классов.