МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

“МИРЭА - Российский технологический университет”

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра математического обеспечения и стандартизации
информационных технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5
по дисциплине
«Структуры и алгоритмы обработки данных»

Тема «Строки»

Выполнил студент группы ИКБО-50-23 Русаков М.Ю.

Принял старший преподаватель Скворцова Л.А.

Москва 2024

Оглавление

1. Условие задачи 3
2. Описание алгоритма решения задачи 4
3. Реализация задания 1 4
3.1. Описание применяемых в реализации методов строки функций из библиотеки `<string.h>` 4
3.2. Код реализации задания 1 4
3.3. Время выполнения задачи варианта 9
3.4. Тестирование 10
3.4.1. Таблица с тестами 10
3.4.2. Результаты тестирования 11
4. Реализация задания 2 12
4.1. Описание применяемых в реализации методов строки функций из библиотеки `<string>` 12
4.2. Код реализации задания 2 12
4.3. Время выполнения задачи варианта 15
4.4. Тестирование 15
4.4.1. Таблица с тестами 15
4.4.2. Результаты тестирования 16
5. Выводы 16

1. Условие задачи

Требуется выполнить два задания по обработке текста.

В первом задании для представления текста использовать нольтерминированную строку языка C. Для управления строкой использовать функции файла заголовка (библиотека функций управления нольтерминированной строкой) `<string.h>`.

Во втором задании использовать для представления текста строку класса `<string>`. Для управления строкой использовать методы класса `string`.

Задание моего персонального варианта (№22):

Даны два предложения. Вывести слова, общие для этих двух предложений. Если таких слов нет, то вывести сообщение об этом.

2. Описание алгоритма решения задачи

Первым делом для решения задачи нужно разбить оба предложения на отдельные слова (строки), а затем попарно их сравнить. Для этого можно воспользоваться алгоритмом перебора с вложенным циклом. Допустим, у нас есть 2 массива со словами (строками) длинами n и m соответственно. Запустим цикл от 0 до $n-1$ для первого массива, а внутри данного цикла запустим еще один от 0 до $m-1$ для второго массива (таким образом мы гарантируем однократное сравнение каждого слова первого массива с каждым словом второго массива). Далее просто сравним элементы каждой пары. Если элементы равны, то выведем любой из этих элементов на экран. Если среди элементов пар ни найдется ни одних равных, то выведем сообщение об этом.

3. Реализация задания 1

3.1. Описание применяемых в реализации методов строки функций из библиотеки <string.h>

1. `strtok` – поиск лексем строки по разделителям.
2. `strlen` – возврат длины строки.
3. `strcpy` – копирование содержимого строки.
4. `strncpy` – копирование n первых символов строки.
5. `strcat` – конкатенация двух строк.
6. `memcpy` – копирование блока данных из строки.

3.2. Код реализации задания 1

1. Код файла `myString.h`

```
#include <iostream>
#include <string.h>

struct myString {
private:
    size_t size{};
    char* charArray = nullptr;

public:
    // Конструктор структуры
    myString(const size_t size);

    ~myString();
```

```
    // Ввод содержимого строки
    void input();

    // Вывод содержимого строки в консольы
    void print();

    // Возврат указателя на срез строки с индекса index длины size
    char* slice(size_t index, size_t size);

     // Возврат указателя на новую строку со вставленной строкой source
после индекса index
    void insert(size_t index, char* source);

    // Возврат указателя на новую строку с удалением подстроки начиная с
index длиной size
    void erase(size_t index, size_t size);

    std::pair<char**, size_t> split(const char* sep);
};
```

2. Код файла myString.cpp

```
#include "../headers/my_string.h"
#include <cstdio>

myString::myString(const size_t size) {
    charArray  = new char[size + 1];
    this->size = size;
}

myString::~myString() {
    delete[] charArray;
}

// Максимальный размер вводимого текста - 1000 символов
void myString::input() {
    char* buffer = new char[size + 1];

    printf("Введите предложение: ");
    fgets(buffer, size, stdin);

    strcpy(charArray, buffer);
    delete[] buffer;
}

void myString::print() {
    printf("%s\n", charArray);
```

```
}

char* myString::slice(size_t index, size_t size) {
    if (index > strlen(charArray)) {
        std::cout << "slice: Ошибка: неверное значение индекса!\n";
        return nullptr;
    }

    char* slice = new char[size + 1];
    if (slice == nullptr) {
        std::cout << "slice: Ошибка выделения памяти!\n";
        delete[] slice;

        return nullptr;
    }

    memcpy(slice, charArray + index, size);
    if (slice == nullptr) {
       std::cout << "slice: Ошибка: неверные входные данные\n";
        return nullptr;
    }

    return slice;
}

void myString::insert(size_t index, char* source) {
    if (index > strlen(charArray)) {
        std::cout << "insert: Ошибка: неверное значение индекса!\n";
        return;
    }

    const size_t sourceSize = strlen(source);
    const size_t newSize    = strlen(charArray) + sourceSize;

    char* newCharArray = new char[newSize + 1];
    strncpy(newCharArray, charArray, index);
    strncpy(newCharArray + index, source, sourceSize);
    strcat(newCharArray + index + sourceSize, charArray + index);

    delete charArray;
    charArray = newCharArray;
}

void myString::erase(size_t index, size_t size) {
    const size_t charArraySize = strlen(charArray);
    if (index > charArraySize) {
```

```
        std::cout << "erase: Ошибка: неверное значение индекса!\n";
        return;
    }

    char* newCharArray = new char[charArraySize - size];
    this->size -= size;
    strncpy(newCharArray, charArray, index);
    if (index + size > charArraySize) {
        delete charArray;
        charArray = newCharArray;
        return;
    }

    strcat(newCharArray, charArray + index + size);
    delete charArray;
    charArray = newCharArray;
}

// Возврат указателя на динамический массив и его длины в качестве полей структуры std::pair
std::pair<char**, size_t> myString::split(const char* sep) {
    size_t arraySize = 0;
    char** array = new char*[this->size];
    if (array == nullptr) {
        printf("split: Ошибка выделения памяти!");
    }

    for (size_t i{}; i < size; i++) {
        array[i] = nullptr;
        if (array[i] != nullptr) {
            std::cout << "split: Ошибка!\n";
        }
    }

    char* charArrayCopy = new char[this->size + 1];
    if (charArrayCopy == nullptr) {
        printf("split: Ошибка выделения памяти!");
        delete[] array;
        delete[] charArrayCopy;
        return std::make_pair(nullptr, 0);
    }

    strcpy(charArrayCopy, charArray);
    if (charArrayCopy == nullptr) {
        printf("split: Ошибка выделения памяти!");
    }
```

```
    char* word = strtok(charArrayCopy, sep);
    if (word == nullptr) {
        printf("split: Ошибка выделения памяти!");
    }

    while (word != nullptr) {
        if (strlen(word) == 0 || *word == char(10) || *word == char(32))
{
            word = strtok(nullptr, sep);
            continue;
        }

        array[arraySize] = new char[strlen(word) + 1];
        strcpy(array[arraySize], word);
        arraySize++;

        word = strtok(nullptr, sep);
    }

    if (array == nullptr) {
        printf("split: Ошибка при разделении строки!\n");
    }

    return std::make_pair(array, arraySize);
}
```

3. Код файла main.cpp

```
#include "myString.h"
#include <chrono>

void findSameWords(char** _sentence_1, const size_t _sentence_1_size,
char** _sentence_2, const size_t _sentence_2_size);

int main()
{
    myString str1(1000);
    myString str2(1000);

    str1.input();
    str2.input();

    size_t str1_size, str2_size;
    char** str1_split;
    char** str2_split;
```

```
    std::tie(str1_split, str1_size) = str1.split(" .,?!");
    std::tie(str2_split, str2_size) = str2.split(" .,?!");

    auto start = std::chrono::high_resolution_clock::now();
    findSameWords(str1_split, str1_size, str2_split, str2_size);
    auto end = std::chrono::high_resolution_clock::now();
    int time = std::chrono::duration_cast<std::chrono::milliseconds>(end
- start).count();
    printf("Время выполнения: %d ms", time);
}

void findSameWords(char** _sentence_1, const size_t _sentence_1_size,
char** _sentence_2, const size_t _sentence_2_size)
{
    bool common = false;
    for (size_t i{}; i < _sentence_1_size; i++)
    {
        for (size_t j{}; j < _sentence_2_size; j++)
        {
            if (strcmp(_sentence_1[i], _sentence_2[j]) == 0)
            {
                if (!common)
                {
                    printf("Общие слова: ");
                    common = true;
                }
                printf("%s ", _sentence_1[i]);
            }
        }
    }

    if (!common)
    {
        printf("Общих слов у данных предложений нет");
    }
}
```

3.3. Время выполнения задачи варианта

Таблица 1 – Таблица зависимости времени выполнения программы от объема входных данных

Номер	Объем входных данных, символы	Время выполнения, мс
1	10	2
2	100	29
3	500	54
4	1000	103

3.4. Тестирование

3.4.1. Таблица с тестами

Таблица 2 – Таблица тестов для задания 1

Номер	Входные данные	Ожидаемый вывод
1	Escape the society We live in society	society
2	The quick brown fox jumps over the lazy dog A quick brown dog jumps over a lazy fox	quick brown jumps over lazy

3.4.2. Результаты тестирования

Рисунок 1 – Тест №1

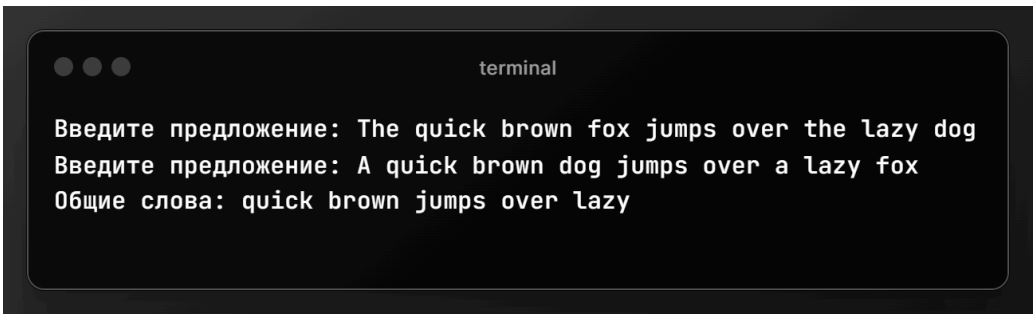

Рисунок 2 – Тест №2

4. Реализация задания 2

4.1. Описание применяемых в реализации методов строки функций из библиотеки <string>

1. `resize` – изменение текущего размера строки
2. `substr` – возврат подстроки
3. `insert` – вставка подстроки
4. `erase` – удаление подстроки
5. `find_first_of` – возврат первого индекса любого из набора символов
6. `empty` – проверка строки на пустоту

4.2. Код реализации задания 2

1. Код файла `myString.h`

```
#include <iostream>
#include <string>
#include <vector>

struct myString {
private:
    std::string charVector;

public:
    myString(size_t size);

    // Ввод содержимого строки
    void input();

    // Вывод содержимого строки в консоль
    void print();

     // Возврат вектора строк, являющихся разбиением исходной строки по
разделителям sep
    std::vector<std::string> split(const std::string& sep);

    // Возврат среза строки с индекса index длины size
    std::string slice(size_t index, size_t size);

    // Вставка в исходную строку строки source после индекса index
    void insert(const std::string& source, size_t index);

     // Удаление из исходной строки подстроки, начиная с index длиной
size
```

```
    void erase(size_t index, size_t size);
};
```

2. Код файла myString.cpp

```
#include <iostream>
#include "../headers/my_string.h"

myString::myString(size_t size) {
    charVector.resize(size);
}

void myString::input() {
    std::cout << "Введите ваше предложение: ";
    std::getline(std::cin, charVector);
}

void myString::print() {
    std::cout << charVector << '\n';
}

std::string myString::slice(size_t index, size_t size) {
    return charVector.substr(index, size);
}

void myString::insert(const std::string& source, size_t index) {
    charVector.insert(index, source);
}

void myString::erase(size_t index, size_t size) {
    charVector.erase(index, size);
}

std::vector<std::string> myString::split(const std::string& sep) {
    std::vector<std::string> result;
    size_t start = 0, end = charVector.find_first_of(sep);
    while (end != std::string::npos) {
        if (!charVector.substr(start, end - start).empty()) {
            result.push_back(charVector.substr(start, end - start));
        }
        start = end + 1;
        end = charVector.find_first_of(sep, start);
    }
    if (!charVector.substr(start).empty()) {
        result.push_back(charVector.substr(start));
    }
```

```
    return result;
}
```

3. Код файла `main.cpp`

```
#include "headers/my_string.h"
#include <vector>
#include <chrono>

void findSameWords(std::vector<std::string>& _sentence_1, const size_t
_sentence_1_size, std::vector<std::string>& _sentence_2, const size_t
_sentence_2_size);

int main() {
    myString str1(1500);
    myString str2(1500);

    str1.input();
    str2.input();

    std::vector<std::string> str1_split = str1.split(" .,?!");
    std::vector<std::string> str2_split = str2.split(" .,?!");

    auto start = std::chrono::high_resolution_clock::now();
            findSameWords(str1_split,  str1_split.size(),  str2_split,
str2_split.size());
    auto end = std::chrono::high_resolution_clock::now();
                    std::cout    <<    "Время    выполнения:    "
<<     std::chrono::duration_cast<std::chrono::milliseconds>(end     -
start).count() << " ms";
}

void findSameWords(std::vector<std::string>& _sentence_1, const size_t
_sentence_1_size, std::vector<std::string>& _sentence_2, const size_t
_sentence_2_size) {
    bool common = false;
    for (size_t i{}; i < _sentence_1_size; i++) {
        for (size_t j{}; j < _sentence_2_size; j++) {
            if (_sentence_1[i] == _sentence_2[j]) {
                if (!common) {
                    std::cout << "Общие слова: ";
                    common = true;
                }
                std::cout <<  _sentence_1[i] << ' ';
            }
        }
    }
```

```
    std::cout << "\n";

    if (!common) {
        std::cout << "Общих слов у данных предложений нет\n";
    }
}
```

4.3. Время выполнения задачи варианта

Номер	Объем входных данных, символы	Время выполнения, мс
1	10	4
2	100	32
3	500	57
4	1000	122

4.4. Тестирование

4.4.1. Таблица с тестами

Таблица 3 – Таблица тестов для задания 2

Номер	Входные данные	Ожидаемый вывод
1	Don’t take my snacks, please Can you borrow me some cheaps?	Общих слов нет
2	He has a decent problem to solve I am stuck on this problem	problem

4.4.2. Результаты тестирования

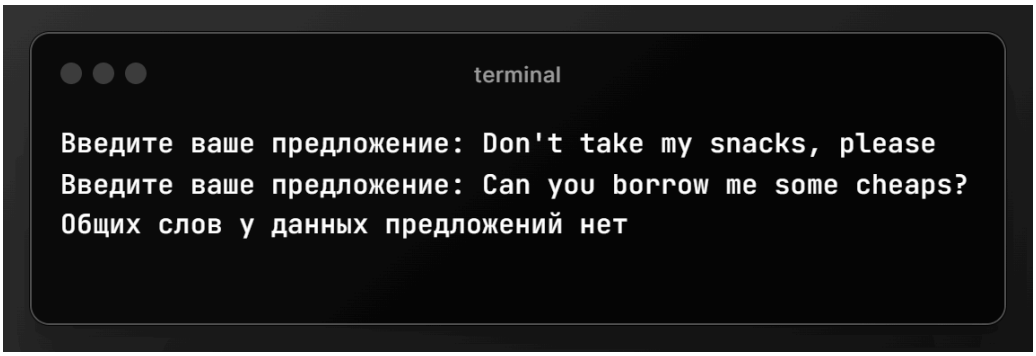

Рисунок 3 – Тест №1

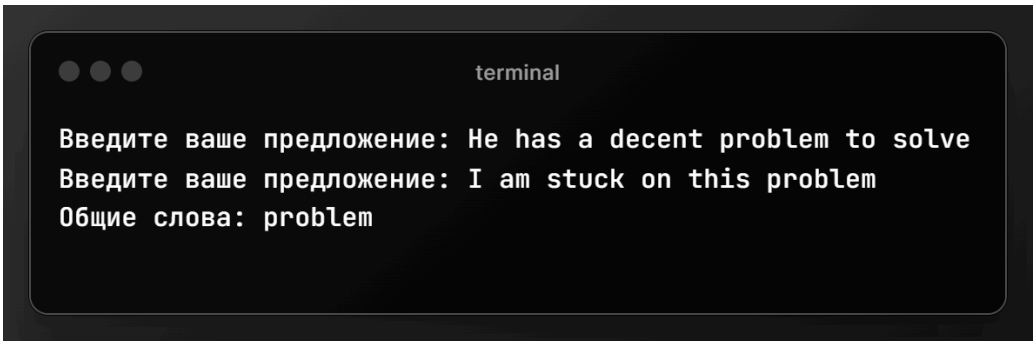

Рисунок 4 – Тест №2

5. Выводы

В результате выполнения работы были получены знания по выполнению операций над ноль-терминированной строкой, организации хранения строковых данных класса `string` и использованию средств языка C++ для реализации алгоритмов обработки текстовых данных.