|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 3** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Строки»** | |
|  | |
| Выполнил студент группы ИКБО-72-23 | Шатохин Б.А. |
| Принял преподаватель | Филатов А.С. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2024

1. Цель работы

Получение навыков в разработке алгоритмов обработки текста – извлечение отдельных элементов. Получение навыков использования средств языка Си и С++ для реализации алгоритмов обработки текстовых данных.

1. Постановка задачи
2. Разработать программу согласно задаче варианта, используя для представления обрабатываемого в программе текста нуль терминальную строку и средства языка С для выполнения операций над этой строкой.
3. Разработать программу согласно задаче варианта, используя для представления обрабатываемого в программе текста строку string стандартной библиотеки шаблонов и возможности класса для выполнения действий со строкой.
4. Тестирование программы должно производиться на введенных вручную данных.
5. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Индивидуальное задание:

Дан текст, содержащий слова, которые разделены пробелами. Сформировать массив слов. Найти пары слов (анаграммы)- при прочтении каждого из которых в обратном порядке образуется другое слово этой пары, например, ТОК,КОТ; ПОЛК, КЛОП; БАР, РАБ. Сформировать двумерный массив этих пар.

1. Решение

Нуль-терминированная строка или C-строка или ASCIIZ-строка — способ представления строк в языках программирования, при котором вместо введения специального строкового типа используется массив символов, а концом строки считается первый встретившийся специальный нуль-символ.

Нуль-символ или терминальный ноль - символ конца строки в стиле C “\0”.

string - класс с методами и переменными для организации работы со строками в языке программирования C++. Он включён в стандартную библиотеку C++. Название образовано от имени строчного типа данных. В языке C++ и его предшественнике, языке программирования Си, нет встроенной поддержки строкового типа данных, вместо этого используется массив символов. string управляет строками, как и string.h в Си. string использует единственный объект string для организации работы со строками. Являясь частью стандартной библиотеки C++, эти объекты также являются частью стандартного пространства имён — std.

Для решения программы через массив char:

Для разбиения текста на отдельные слова была разработана функция SplitText, принимающая в качестве аргументов указатель на массив элементов char, в котором храниться предложение, двумерный массив, в котором будут храниться наши будущие слова, и переменную, которая служит счётчиком для количества слов.

void SplitText(char\* sentence, char\*\* words, int& count) { //Формируем массив слов

int word\_index = 0;

while (\*sentence) {

// Пропускаем пробелы

while (\*sentence && \*sentence == ' ') {

++sentence;

}

if (!\*sentence) {

break;

}

// Начало нового слова

words[word\_index] = new char[100]; // Предполагаем максимальную длину слова 100

int char\_index = 0;

// Чтение слова до пробела или конца строки

while (\*sentence && \*sentence != ' ') {

words[word\_index][char\_index] = \*sentence;

++char\_index;

++sentence;

}

words[word\_index][char\_index] = '\0'; // Нуль-терминатор

++count;

++word\_index;

}

}

Для выполнения индивидуального задания была разработана функция FindAnagrams, которая принимает в качестве аргументов два массива, массив со всеми словами, а также переменную счётчик.

void FindAnagramms(char\* words[], char\*\* anagrams, int count) { // Находим анаграммы

int anagrams\_count = 0;

for (int i = 0; i < count; ++i) {

if (words[i] == nullptr) {

continue;

}

anagrams[anagrams\_count] = new char[strlen(words[i]) + 1];

strcpy(anagrams[anagrams\_count], words[i]);

++anagrams\_count;

for (int j = ++i; j < count; ++j) {

if (words[i] != nullptr && strlen(words[i]) == strlen(words[j])) {

int freq[256] = { 0 };

bool isAnagram = true;

for (int k = 0; words[i][k] != '\0'; ++k) {

++freq[words[i][k]];

--freq[words[j][k]];

}

for (int b = 0; b < 255; b++)

if (freq[b] != 0)

isAnagram = false;

if (isAnagram) {

anagrams[anagrams\_count] = new char[strlen(words[j]) + 1]; // Выделяем память для анаграммы

strcpy(anagrams[anagrams\_count], words[j]); // Копируем слово

++anagrams\_count;

words[j] = nullptr; // Помечаем слово как обработанное

}

}

}

}

}

Для вывода результата используется функция ShowResult, принимающая в качестве аргумента двумерный массив и количество элементов этого массива.

void ShowResult(char\*\* anagrams, int count) {

for (int i = 0; i < count; ++i) {

std::cout << anagrams[i] << "\n";

}

}

Для решения через string:

Для решения через этот класс была изменена функция SplitText. Теперь она принимает в качестве аргумента только одну строку.

std::vector<std::string> SplitText(std::string sentence) { //Разбиваем текст на слова

std::vector<std::string> words;

std::string word = "";

for (int i = 0; i < sentence.size(); i++) {

if (sentence[i] == ' ') {

words.push\_back(word);

word = "";

}

else word += sentence[i];

}

words.push\_back(word);

return words;

}

Так же была изменена функция FindAnagrams. Она в качестве аргумента принимает вектор со словами, а возвращает двумерный вектор с парными словами.

FindAnagrams(std::vector<std::string> words) { //Ищем слова палиндромы

std::vector<std::vector<std::string>> anagrams;

std::vector<std::string> temp\_vector;

std::string temp = "";

for (int i = 0; i < words.size(); i++) {

temp = "";

for (int j = i + 1; j < words.size(); j++){

if (words[i].size() == words[j].size()) {

for (int k = 0; k < words[i].size(); k++) {

temp = words[j][k] + temp;

}

if (temp == words[i]) {

temp\_vector.push\_back(words[i]);

temp\_vector.push\_back(words[j]);

anagrams.push\_back(temp\_vector);

temp\_vector.clear();

}

}

}

}

return anagrams;

}

1. Тестирование

Для демонстрации работы всей программы введём слова из поставленной задачи и добавим два слова, которые явным образом не являются словами палиндромами. Программа, возвращает двумерный массив со словами палиндромами. Как видим слова “ббб” и “ввв” не были добавлены в массив, так как они не имеют пары.

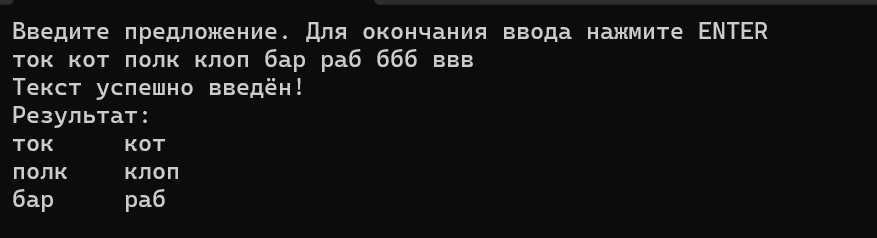


Рисунок 1. Результат тестирования программы

1. Вывод

Получены навыки по работе с нуль-терминальной строкой. Так же был изучен класс string и методы, встроенные в этот класс. Были разработаны алгоритмы для работы с текстом.

1. Исходный код

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Ru");

char sentence[255]; //Создаем 256-1 т.к. последний символ \0

char\*\* anagrams = new char\* [255]; // Массив для хранения анаграмм

char\* words[255];

int word\_count = 0;

std::cout << "Введите предложение. Для окончания ввода нажмите ENTER\n";

gets\_s(sentence);

std::cout << "Текст успешно введён!\n";

SplitText(sentence, words, word\_count);

FindAnagramms(words, anagrams, word\_count);

std::cout << "Результат:\n";

ShowResult(anagrams, word\_count);

return 0;

}