**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №4**

**по дисциплине «Программирование»**

Тема: ТЕКСТОВЫЕ СТРОКИ КАК МАССИВЫ СИМВОЛОВ

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4373 | Некрасова А.В. |  |
| Преподаватель | Глущенко А. Г. |  |

Санкт-Петербург

2024

**Цель работы.**

Изучение работы строк и их внутреннего представления в памяти; сортировки; получение практических навыков работы со строками.

**Основные теоретические положения.**

Текстовые строки представляются с помощью одномерных массивов символов. В языке C++ текстовая строка представляет собой набор символов, обязательно заканчивающийся нулевым символом (‘\0’). Поэтому, если вы хотите создать текстовый массив для хранения 10 (N) символов, нужно выделить память под 11(N+1) символов.

Объявленный таким образом массив может использоваться для хранения текстовых строк, содержащих не более 10 символов. Нулевой символ позволяет определить границу между содержащимся в строке текстом и неиспользованной частью строки.

При определении строковых переменных их можно инициализировать конкретными значениями с помощью строковых литералов:

char S1[15] = “This is text”;

char S2[] = “Пример текста”;

Последние два элемента переменной  просто не используются, а строка  автоматически подстраивается под длину инициализирующего текста.

При работе со строками можно обращаться к отдельным символам строки как в обычном одномерном массиве с помощью индексов:

cout << S1[0]; // На экране будет выведен символ ‘T’

Если строка формируется при помощи цикла (или иного способа), то необходимо в ее конец обязательно записать нулевой символ '\0'.

**Постановка задачи.**

Необходимо написать программу, которая реализует поставленную задачу:

1)    С клавиатуры или с файла (\*) (пользователь сам может выбрать способ ввода) вводится последовательность, содержащая от 1 до 50 слов, в каждом из которых от 1 до 10 строчных латинских букв и цифр. Между соседними словами произвольное количество пробелов. За последним символом стоит точка.

2)    Необходимо отредактировать входной текст:

·        удалить лишние пробелы;

·        удалить лишние знаки препинания (под «лишними» подразумевается несколько подряд идущих знаков (обратите внимание, что «…» - корректное использование знака) в тексте);

·        исправить регистр букв, если это требуется (пример некорректного использования регистра букв: пРиМЕр);

3) Выполнить задание по варианту:

|  |  |
| --- | --- |
| 1 | Вывести на экран слова последовательности в обратном порядке. |
| 2 | Вывести на экран слова последовательности в алфавитном порядке. |
| 3 | Вывести на экран слова последовательности, не содержащие цифр. |
| 4 | Вывести на экран только те слова последовательности, в которых встречаются одинаковые буквы. |
| 5 | Вывести на экран только те слова последовательности, в которых первая буква слова встречается в этом слове еще раз. |
| 6 | После окончания ввода последовательности вывести на экран сначала все слова, содержащие только буквы, затем слова, содержащие только цифры, а потом слова, содержащие и буквы, и цифры. |

4) Выполнить задание по варианту:

|  |  |
| --- | --- |
| 1 | Вывести на экран ту же последовательность, заменив во всех словах первую букву соответствующей прописной буквой. |
| 2 | Вывести на экран количество символов в каждом слове исходной последовательности. |
| 3 | Вывести на экран ту же последовательность, удалив из всех слов заданный набор букв и (или) цифр. |
| 4 | Вывести на экран ту же последовательность, заменив во всех словах цифры на буквы латинского алфавита, номера которых в алфавите равны заменяемой цифре. |
| 5 | Вывести на экран ту же последовательность, переместив все цифры, содержащиеся в словах, в конец соответствующих слов. |
| 6 | Вывести все слова исходной последовательности на экран вертикально. |
| 7 | Вывести на экран  все слова последовательности в две или три колонки (в зависимости от количества слов) с выравниванием слов по правой границе колонки. |

5)  Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке. Реализуйте два алгоритма: первый алгоритма – Линейный поиск, а второй алгоритм согласно вашему номеру в списке. Четные номера должны реализовать алгоритм КМП, а нечетные – Бойера-Мура. (\*)

**Выполнение работы**

#include <iostream>

#include <stdio.h>

#include <vector>

using namespace std;

void DeleteProbels(char\* str) {

int l = strlen(str);

if (l == 0) return;

char\* result = new char[l + 1];

int j = 0;

bool probel = false;

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

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

if (!probel) {

result[j++] = ' '; // Добавляем один пробел

probel = true; // Устанавливаем флаг, что мы в пробеле

}

}

else {

result[j++] = str[i]; // Добавляем текущий символ

probel = false; // Сбрасываем флаг

}

}

if (j > 0 && result[0] == ' ') {

std::memmove(result, result + 1, j - 1); // Удаляем пробел в начале

--j; // Уменьшаем длину результирующей строки

}

if (j > 0 && result[j - 1] == ' ') {

--j; // Удаляем пробел в конце

}

result[j] = '\0';

strcpy\_s(str, l+1, result);

delete[] result;

}

bool Sign(char c) {

return (c == '!' || c == '?' || c == '.' || c == ',' || c == ';' || c == ':');

}

void RemoveSigns(const char\* input, char\* output) {

bool last\_was\_punct = false;

bool last\_was\_ellipsis = false;

int output\_index = 0;

int length = strlen(input);

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

char current = input[i];

// Проверяем, является ли текущий символ знаком препинания

if (Sign(current)) {

// Проверяем, не является ли это многоточием

if (current == '.' && i + 1 < length && input[i + 1] == '.' && i + 2 < length && input[i + 2] == '.') {

// Если предыдущий символ не был многоточием, добавляем его

if (!last\_was\_ellipsis) {

output[output\_index++] = '.';

output[output\_index++] = '.';

output[output\_index++] = '.';

last\_was\_ellipsis = true;

i += 2; // Пропускаем следующие два символа

}

}

else {

// Если это не многоточие и предыдущий символ также был знаком препинания

if (!last\_was\_punct) {

output[output\_index++] = current; // Добавляем текущий знак

last\_was\_punct = true; // Устанавливаем флаг

}

}

}

else {

output[output\_index++] = current; // Добавляем обычный символ

last\_was\_punct = false; // Сбрасываем флаг

last\_was\_ellipsis = false; // Сбрасываем флаг для многоточия

}

}

output[output\_index] = '\0'; // Завершаем строку нулем

}

void Register(char\* input) {

if (input[0] == '\0')

return;

input[0] = toupper(input[0]);

for (size\_t i = 1; input[i] != '\0'; ++i) {

input[i] = tolower(input[i]);

}

}

void NoNumbers(const char\* str) {

const char\* start = str;

const char\* end;

while (\*start) {

// Пропускаем пробелы

while (\*start == ' ') {

start++;

}

// Если достигнут конец строки, выходим из цикла

if (\*start == '\0') {

break;

}

// Находим конец слова

end = start;

while (\*end != ' ' && \*end != '\0') {

end++;

}

// Проверяем слово на наличие цифр

bool yesNumbers= false;

for (const char\* символ = start; символ < end; символ++) {

if (isdigit(\*символ)) {

yesNumbers = true;

break;

}

}

// Если слово не содержит цифр, выводим его

if (!yesNumbers) {

for (const char\* символ = start; символ < end; символ++) {

cout << \*символ;

}

cout << ' ';

}

// Переходим к следующему слову

start = end;

}

}

void NumberOfLetters(const char\* str) {

const char\* start = str;

const char\* end;

while (\*start) {

// Пропускаем пробелы

while (\*start == ' ') {

start++;

}

// Если достигнут конец строки, выходим из цикла

if (\*start == '\0') {

break;

}

// Находим конец слова

end = start;

while (\*end != ' ' && \*end != '\0') {

end++;

}

// Подсчитываем количество букв в слове

int number = 0;

for (const char\* symbol = start; symbol < end; symbol++) {

if (isalpha(\*symbol)) { // Проверяем, является ли символ буквой

number++;

}

}

// Выводим количество букв в слове

cout << "Количество букв в слове \"";

for (const char\* symbol = start; symbol < end; symbol++) {

cout << \*symbol;

}

cout << "\": " << number << endl;

// Переходим к следующему слову

start = end;

}

}

void LinearSearch(const char\* str, const char\* substr) { // Ф У Н К Ц И Я Д Л Я Л И Н Е Й Н О Г О П О И С К А

int lengthStr = strlen(str);

int lengthSubstr = strlen(substr);

bool bober = false;

// Линейный поиск подстроки

for (int i = 0; i <= lengthStr - lengthSubstr; i++) {

int j;

for (j = 0; j < lengthSubstr; j++) {

if (str[i + j] != substr[j]) {

break;

}

}

if (j == lengthSubstr) { // Если мы нашли подстроку

cout << i << ' ';

bober = true;

}

}

if (!bober) {

cout << "Подстрока не найдена." << endl;

}

}

void PrefixFunction(const char\* substr, vector<int>& prefix) {

int lengthSubstr = strlen(substr);

prefix.resize(lengthSubstr);

prefix[0] = 0; // Начальное значение префикса

int j = 0; // Длина предыдущего префикса

for (int i = 1; i < lengthSubstr; ++i) {

while (j > 0 && substr[i] != substr[j]) {

j = prefix[j - 1]; // Переход к предыдущему префиксу

}

if (substr[i] == substr[j]) {

++j; // Увеличиваем длину префикса

}

prefix[i] = j; // Устанавливаем значение префикса

}

}

// Функция для поиска подстроки с использованием алгоритма КМП

void АлгоритмКМП(const char\* str, const char\* substr) {

int lengthStr = strlen(str);

int lengthSubstr = strlen(substr);

vector<int> prefix;

PrefixFunction(substr, prefix);

int j = 0; // Индекс для подстроки

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

while (j > 0 && str[i] != substr[j]) {

j = prefix[j - 1]; // Переход к предыдущему префиксу

}

if (str[i] == substr[j]) {

++j; // Увеличиваем длину совпадения

}

if (j == lengthSubstr) {

cout << i - j + 1 << ' ';

j = prefix[j - 1]; // Продолжаем поиск

}

}

}

int main()

{

setlocale(0, "");

const int n = 501;

char str0[n], str[n], substr[n];

cout <<"Введите строку: " ;

cin.getline(str0, 501);

cout << "Отформатированная строка: ";

DeleteProbels(str0);

RemoveSigns(str0, str);

Register(str);

cout << str << endl << "\nСлова последовательности, не содержащие цифр: ";

NoNumbers(str);

cout << endl << "Кол-во символов в каждом слове исходной последовательности" << endl;

NumberOfLetters(str);

cout << endl << "ПОИСК ПОДСТРОКИ В СТРОКЕ" << endl << "Введите подстроку: ";

cin.getline(substr, 501);

cout << "Подстрока найдена в позиции: ";

АлгоритмКМП(str, substr);

}

**Выводы.**

Я изучила строки и поняла, как они представлены в памяти компьютера, научилась работать с различными строками.