**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

|  |  |  |
| --- | --- | --- |
| Студент гр. 2373 |  | Чесноков М. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение представления текстовых строк в программах. Работа со строками как с массивами символов. Операции над строками. Знакомство с классом **std::string**.

**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.

Класс string предназначен для работы со строками типа char, которые представляют собой строчку с завершающим нулем (символ ‘\0’). Класс string был введен как альтернативный вариант для работы со строками типа char.

Чтобы использовать возможности класса **string**, нужно подключить библиотеку <string> и пространство имен **std**. Объявление же переменной типа **string** осуществляется схоже с обычной переменной:

**Постановка задачи.**

Необходимо написать программу, которая:

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: вывести на экран слова последовательности в обратном порядке.

4) Выполнить задание по варианту: вывести на экран ту же последовательность, заменив во всех словах первую букву соответствующей прописной буквой.

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 КНМ, а нечетные – Бойера-Мура.

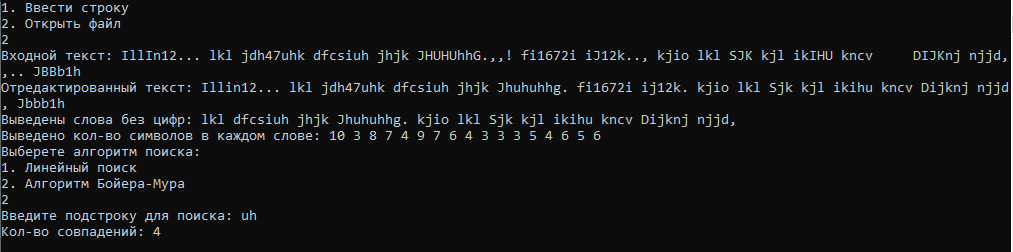
**Выполнение работы.**

Код программы представлен в приложении А.

**Блок описания кода и использованных алгоритмов:**

1. Пользователь вводит строку или считывает её с файла.
2. Эта строка редактируется с помощью прописанных функций: 1 - удаляет лишние пробелы, 2 - удаляет лишние знаки препинания. 3 – исправляет регистры букв (если первая буква заглавная, то считается, что она и должна такой быть).
3. Выводятся слова, не содержащие цифр.
4. Выводится кол-во символов в каждом слове отредактированной строки.
5. Линейным поиском или по алгоритму Бойера-Мура находится подстрока, введённая пользователем.

**Блок скриншотов работы программы**



**Вывод.**

Во время выполнения данной работы были изучены строки, массивы символов и принципы работы с ними. Также был получен навык работы со строками.

Приложение А

рабочий код

#include <iostream>

#include <fstream>

#include <string>

#include <map>

using namespace std;

string text;

void SpaceDel() {

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

if (text[i] == ' ' && text[i + 1] == ' ')

while (text[i + 1] == ' ')

text.erase((i + 1), 1);

}

}

void SignDel() {

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

if ((text[i] == '.' || text[i] == ',' || text[i] == '?' || text[i] == '!' || text[i] == ';' || text[i] == ':') && (text[i + 1] == '.' || text[i + 1] == ',' || text[i + 1] == '?' || text[i + 1] == '!' || text[i + 1] == ';' || text[i + 1] == ':')) {

if (text[i] == '.' && text[i + 1] == '.' && text[i + 2] == '.' && text[i + 3] == ' ')

i += 3;

else

while (text[i + 1] == '.' || text[i + 1] == ',' || text[i + 1] == '?' || text[i + 1] == '!' || text[i + 1] == ';' || text[i + 1] == ':')

text.erase((i + 1), 1);

}

}

void Reg() {

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

if (text[i] == ' ')

i += 2;

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

}

}

void NumOut(string txt) {

txt += ' ';

for (int i = 0, k = 0, a = 0; i < txt.size(); i++) {

if (txt[i] == '0' || txt[i] == '1' || txt[i] == '2' || txt[i] == '3' || txt[i] == '4' || txt[i] == '5' || txt[i] == '6' || txt[i] == '7' || txt[i] == '8' || txt[i] == '9')

k++;

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

if (k != 0) {

while (txt[a] != ' ')

txt.erase(a, 1);

txt.erase(a, 1);

i = a;

}

else

a = i + 1;

k = 0;

}

}

cout << txt << "\n";

}

void NumOfChar() {

text += ' ';

for (int i = 0, k = 0; i < text.size(); i++) {

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

cout << k << " ";

k = 0;

}

else

k++;

}

}

void Search() {

string key;

cin >> key;

int count = 0;

for (int i = 0; i < (text.size() - key.size() + 1); i++) {

if (text[i] == key[0]) {

if (key.size() == 1)

count++;

else{

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

if (key[j] != text[i + j])

break;

else if (j == (key.size() - 1))

count++;

}

}

}

}

cout << "Кол-во совпадений: " << count << "\n";

}

void BoierMur() {

string key;

map <char, int> dict;

int count = 0;

cin >> key;

int size = key.size();

if (size == 1) {

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

if (text[i] == key[0])

count++;

cout << "Кол-во совпадений: " << count << "\n";

return;

}

dict.emplace('\*', size);

for (int i = size - 1; i >= 0; i--)

if (dict[key[i]] == 0)

dict[key[i]] = size - i - 1;

if (dict[key[size - 1]] == 0)

dict[key[size - 1]] = dict['\*'];

int i = size - 1;

while (i < text.size()) {

if (text[i] == key[size - 1])

for (int j = i - 1, k = size - 2; j > i - size; j--, k--) {

if (key[k] != text[j]) {

if (dict[text[j]] == 0)

i += dict[text[i]];

else

i += dict[text[j]];

break;

}

else if (j == 1 + i - size) {

count++;

i += dict[key[size - 1]];

}

}

else if (i < text.size() - size)

for (int k = size - 2; k >= 0; k--) {

if (key[k] == text[i]) {

i += dict[key[k]];

break;

}

else if (k == 0)

i += dict['\*'];

}

else

break;

}

if (i < text.size() - 1) {

i = text.size() - 1;

if (text[i] == key[size - 1])

for (int j = i - 1, k = size - 2; j > i - size; j--, k--) {

if (key[k] != text[j]) {

break;

}

else if (j == 1 + i - size) {

count++;

}

}

}

cout << "Кол-во совпадений: " << count << "\n";

}

int main()

{

setlocale(0, "");

//Задание 1

int inp;

cout << "1. Ввести строку" << "\n" << "2. Открыть файл" << "\n";

cin >> inp;

switch (inp) {

case 1:

cin.ignore();

getline(cin, text);

break;

case 2:

ifstream file("C:\\Users\\MAX\\Desktop\\Prog\\text.txt");

if (file.is\_open())

getline(file, text);

file.close();

break;

}

cout << "Входной текст: " << text << "\n";

//Задание 2

SpaceDel();

SignDel();

Reg();

cout << "Отредактированный текст: " << text << "\n";

//Задание 3

cout << "Выведены слова без цифр: ";

NumOut(text);

//Задание 4

cout << "Выведено кол-во символов в каждом слове: ";

NumOfChar();

cout << "\n";

//Задание 5

cout << "Выберете алгоритм поиска:\n1. Линейный поиск \n2. Алгоритм Бойера-Мура \n";

cin >> inp;

switch (inp) {

case 1:

cout << "Введите подстроку для поиска: ";

Search();

break;

case 2:

cout << "Введите подстроку для поиска: ";

BoierMur();

break;

}

}