**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: Работы с текстовыми строками.

|  |  |  |
| --- | --- | --- |
| Студент гр. 4372 |  | Федюшкин Л.В. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

 Вводится последовательность; Необходимо отредактировать входной текст; Вывести на экран слова последовательности в обратном порядке; Вывести на экран количество символов в каждом слове исходной последовательности;

**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'.

Предпочтительно использование функции потока ввода cin.getline:

const int N = 21;

char Str [N];

cin.getline (Str, N);      // Пусть введена строка “Это пример текста”

cout  <<  Str << endl;  // На экран будет выведено “ Это пример текста”

Если длина введенного с клавиатуры текста превышает максимальную длину массива , в него будет записано (в нашем примере) 20 символов вводимого текста и нулевой символ. Остальные символы введенного текста остаются во входном потоке и могут быть взяты из него следующими инструкциями ввода.

Функция cin.getline может иметь третий параметр, задающий символ, при встрече которого чтение строки из потока прекращается:

cin.getline (Str, N,  ‘.’);

Иногда чтение из потока невозможно (например, попытка считать слишком длинный текст). Для того чтобы продолжить чтение из потока, необходимо восстановить его нормальное состояние. Этого можно достигнуть с помощью функции потока cin.clear(), которая сбрасывает состояние потока в нормальное. Если забирать остатки данных из потока ввода не надо, то следует очистить его с помощью функции cin.sync().

Широкое применение в программировании получила специальная форма двумерных массивов символов, которые можно рассматривать как одномерный массив строк, заканчивающихся нулевым символом. Определяются такие массивы следующим образом:

char  arr\_str [30] [21];

Этот массив содержит 30 строк. Каждая строка может содержать текст из максимум 20 символов + нулевой символ.

Для того чтобы обратиться к некоторой строке этого массива достаточно указать только первый индекс. Например:

strcpy (arr\_str [10] , “Иванов”);

cout <<  arr\_str [10] << endl;    // На экран выведено “Иванов”

Инициализация такого массива:

char  arr\_str [3] [21] = {“Иванов”,  “Петров”,  “Сидоров”};

cout <<  arr\_str [1] << endl;  // На экране “Петров”

или так:

char  arr\_str [ ] [21] = {“Иванов”,  “Петров”,  “Сидоров”};

cout <<  arr\_str [2] << endl;  // На экране “Сидоров”

Для получения доступа к отдельным символам строк необходимо указывать два индекса, как в обычном двумерном массиве:

​​​​​​​cout  <<  arr\_str [2] [2];   //  На экране буква ‘д’ из строки “Сидоров”

**Постановка задачи.**

 Вводится последовательность; Необходимо отредактировать входной текст; Вывести на экран слова последовательности в обратном порядке; Вывести на экран количество символов в каждом слове исходной последовательности;

**Выполнение работы.**

|  |
| --- |
| Меню выбора задания: |
| При выборе 1 пункта открывается меню выбора передачи данных: |
| Чтение файла или потока ввода |
| В 3 пункте выводится последовательность в обратном порядке: |
| В 4 пункте выводится количество символов в каждом слове исходной последовательности: |
| В 5 пункте применение алгоритма Бойера - Мура: |

**Выводы.**

Научились работать с вводом, выводом строк, а также с их обработкой.

Приложение А

рабочий код

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include <vector>

using namespace std;

namespace Color {

// ANSI escape codes для изменения цвета текста

const string RESET = "\033[0m"; //сброс

const string RED = "\033[1m\033[31m"; //красный

const string GREEN = "\033[1m\033[32m";//зеленый

const string YELLOW = "\033[1m\033[33m"; //желтый

const string BLUE = "\033[1m\033[34m";//синий

const string MAGENTA = "\033[1m\033[35m";// пурпурный

const string CYAN = "\033[1m\033[36m";// голубой

const string BLACK = "\033[1m\033[30m";// черный

const string WHITE = "\033[1m\033[37m"; // белый

void Red(const string& text) {

cout << RED << text << RESET << endl;

}

void Green(const string& text) {

cout << GREEN << text << RESET << endl;

}

void Yellow(const string& text) {

cout << YELLOW << text << RESET << endl;

}

void Blue(const string& text) {

cout << BLUE << text << RESET << endl;

}

void Magenta(const string& text) {

cout << MAGENTA << text << RESET << endl;

}

void Cian(const string& text) {

cout << CYAN << text << RESET << endl;

}

void Black(const string& text) {

cout << BLACK << text << RESET << endl;

}

void White(const string& text) {

cout << WHITE << text << RESET << endl;

}

}

using namespace Color;

//boyerMoore

char to\_lower(char c) {

if (c >= 'A' && c <= 'Z') {

return c + 32;

}

return c;

}

void boyerMoore(string A, string B) {

int n = 0;

while (A[n] != '\0') n++;

int m = 0;

while (B[m] != '\0') m++;

bool found = false;

for (int i = 0; i <= n - m; i++)

{

int j;

for (j = 0; j < m; j++)

{

if (to\_lower(A[i + j]) != to\_lower(B[j]))

{

break;

}

}

if (j == m)

{

cout << "Подстрока найдена в позиции: " << i << endl;

found = true;

}

}

if (!found) {

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

}

}

//Удалить все ненужное

string editText(string inputText) {

string result;

bool lastWasSpace = false;

char lastChar = '0';

for (char current : inputText) {

if (current == ' ') {

if (!lastWasSpace) {

result += current;

lastWasSpace = true;

}

continue;

}

if ((current >= 'a' && current <= 'z') || (current >= 'A' && current <= 'Z') || (current >= '0' && current <= '9'))

{

if (lastChar == current) {

continue;

}

}

if ((current >= 'a' && current <= 'z') || (current >= 'A' && current <= 'Z'))

{

if (current >= 'A' && current <= 'Z')

{

if(current )

if (result[result.length() - 1] >= 'a' && result[result.length() - 1] <= 'z')

{

current = current + 32;

}

}

else if (current >= 'a' && current <= 'z')

{

if (result.length() == 0)

{

current = current - 32;

}

}

}

result += current;

lastWasSpace = false;

lastChar = current;

}

return result;

}

//Слова в обратном порядке

void reverse(string editedText)

{

const int maxWords = 500; // Максимальное количество слов

string words[maxWords];

int wordCount = 0;

// Разделение строки на слова

stringstream ss(editedText);

string word;

while (ss >> word && wordCount < maxWords) {

words[wordCount] = word;

wordCount++;

}

// Вывод слов в обратном порядке

for (int i = wordCount - 1; i >= 0; i--) {

cout << words[i] << " ";

}

cout << endl;

}

//Кол символов в каждом слове исходной посл.

void countChar(string inputText)

{

const int maxWords = 500; // Максимальное количество слов

string words[maxWords];

int wordCount = 0;

// Разделение строки на слова

stringstream ss(inputText);

string word;

while (ss >> word && wordCount < maxWords)

{

words[wordCount] = word;

wordCount++;

}

for (int i = 0; i <= wordCount - 1; i++) {

cout << words[i].length() << " ";

}

cout << endl;

}

void print\_reverse(string editedText) {

int length = 0;

while (editedText[length] != '\0') {

length++;

}

for (int i = length - 1; i >= 0; i--) {

cout << editedText[i];

}

cout << endl;

}

int main()

{

bool first = false;

string editedText;

string inputText1 ="";

string inputText2 = "";

setlocale(LC\_ALL, "RU");

while (true)

{

//Меню

Red("Выберите пункт (1-4) или 0 для выхода\n");

cout << "1. Пункт 1. Ввод последовательности слов\n";

cout << "2. Пункт 2. Отредактировать входной текст \n";

cout << "3. Пункт 3. Вывести на экран слова последовательности в обратном порядке.\n";

cout << "4. Пункт 4. Вывести на экран количество символов в каждом слове исходной последовательности.\n";

cout << "5. Пункт 5. Алгоритм Бойера-Мура\n";

cout << "0. Выход\n";

int choice;

cin >> choice;

switch (choice)

{

case 1:

{

Green("1 пункт.");

Red("\nВвыберите как вы хотите передавать данные");

cout << "1. Ввод с клавиатуры\n";

cout << "2. Ввод с файла(input.txt)\n";

int file\_or\_text;

cin >> file\_or\_text;

inputText1.clear();

inputText2.clear();

if (file\_or\_text == 2)

{

ifstream inputFile("input.txt");

if (inputFile.is\_open()) {

string line;

while (getline(inputFile, line)) {

inputText2 += line + "\n";

}

inputFile.close();

}

else

{

cout << "Не удалось открыть файл." << endl;

return 1;

}

break;

}

if (file\_or\_text == 1)

{

system("cls");

string line;

first = true;

cout << "Введите текст: " << endl;

getline(cin, line, '.');

inputText1 += line;

}

break;

}

case 2:

{

while (true)

{

system("cls");

Green("2 пункт.");

if (first)

{

string result;

bool lastWasSpace = false;

char lastChar = '0';

for (char current : inputText1) {

if (current == ' ') {

if (!lastWasSpace) {

result += current;

lastWasSpace = true;

}

continue;

}

if ((current >= 'a' && current <= 'z') || (current >= 'A' && current <= 'Z') || (current >= '0' && current <= '9'))

{

if (lastChar == current) {

continue;

}

}

if ((current >= 'a' && current <= 'z') || (current >= 'A' && current <= 'Z'))

{

if (current >= 'A' && current <= 'Z')

{

if (current)

if (result[result.length() - 1] >= 'a' && result[result.length() - 1] <= 'z')

{

current = current + 32;

}

}

else if (current >= 'a' && current <= 'z')

{

if (result.length() == 0)

{

current = current - 32;

}

}

}

result += current;

lastWasSpace = false;

lastChar = current;

}

//##########################

string editedText = result; /\*editText(inputText1);\*/

cout << "Отредактированный текст: " << result << endl;

break;

}

else

{

string editedText = editText(inputText2);

cout << "Отредактированный текст: " << editedText << endl;

break;

}

}

}

case 3:

{

while (true)

{

Green("3 пункт.");

if (first)

{

editedText = editText(inputText1);

}

else

editedText = editText(inputText2);

Cian("Последовательность в обратном порядке: ");

/\*print\_reverse(editedText);\*/

//cout << endl;

reverse(editedText);

int n;

Red("Если хотите перейти в следующий пункт, введите 0");

cout << "Введите число: ";

cin >> n;

cout << endl;

if (n == 0)

{

break;

}

}

}

case 4:

{

Green("4 пункт.");

system("cls");

Yellow("Исходная последовательность: ");

cout << inputText1 << inputText2 << endl;

Cian("Количество символов в каждом слове исходной последовательности: ");

countChar(inputText1);

countChar(inputText2);

break;

}

case 5:

{

Green("5 пункт.");

if (first)

{

editedText = editText(inputText1);

}

else

editedText = editText(inputText2);

Yellow("Алгоритм Бойера-Мура");

string text;

Blue("Введите подстроку, которую необходимо найти в строке: ");

cin >> text;

cout << endl;

Cian("В данной строке происходит поиск: ");

cout << editedText;

cout << endl;

boyerMoore(editedText, text);

break;

}

}

if (choice == 0)

{

cout << "\nПрограмма завершила свою работу" << endl;

break;

}

}

}