**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

|  |  |  |
| --- | --- | --- |
| Студент гр. |  |  |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение текстовых строк как массивов символов и их обработке, а также структуры «строка» из области имен std. Изучение потоков, используемых для работы с файлами и непосредственно чтения текстовой строки из файла. Написание программы, способной отредактировать входной текст, исключив из него наиболее очевидные ошибки.

**Основные теоретические положения.**

Текстовые строки представляются с помощью одномерных массивов символов. В изучаемом языке C++ текстовая строка представляет собой набор символов, заканчивающийся нулевым символом ‘\n’. Поэтому для текстовой строки необходимо выделить память под N+1 символов, где N – количество символов в строке.

Чтение текстовых строк из потока ввода осуществляется до первого пробельного символа. Таким образом, чтобы прочесть всю введённую строку полностью, следует воспользоваться функцией получения строки из потока ввода.

Основными задачами при обработке текстовых строк являются следующие: определение фактической длины текста, копирование текста из одной строки в другую, объединение строк, сравнение (больше, меньше, равно) строк и т.п.

**Постановка задачи.**

Вся работа должна быть выполнена с использованием массивов символов.

1. С клавиатуры или с файла (по выбору) 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:

- удалив лишние пробелы.

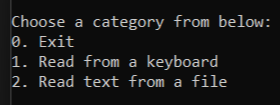
- удалить лишние знаки препинания.

- исправить регистр букв, если это требуется.

1. 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.
2. Вывести все слова исходной последовательности на экран вертикально.
3. Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке, используя алгоритм линейного поиска и Бойера-Мура.

**Выполнение работы.**

1. При запуске программы пользователю предлагается ввести строку. Если пользователь нажмет “1”, то он сможет ввести ее с клавиатуры.
2. Если нажмет “2”, то строка будет прочитана из файла.

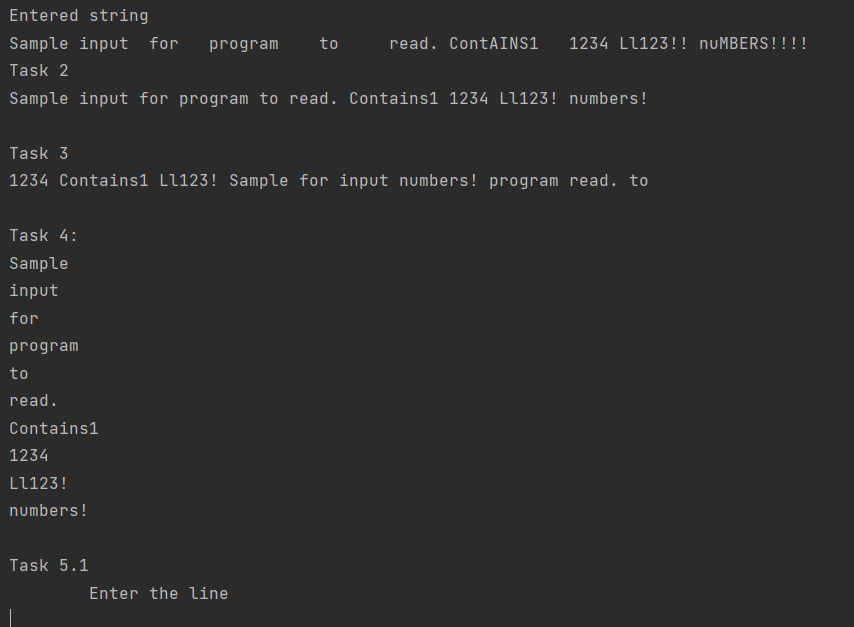


1. После ввода пользователю выводится меню доступных функций и ожидается ввод команды с клавиатуры
2. Следующий шаг зависит от введенной команды, если пользователь ввёл:

“0”, то выполнение программы завершается.

“1”, то перед пользователем выведется отформатированный текст и появится меню с выбором алгоритма поиска линейного или Бойера-мура.

“2”, то перед пользователем выведется отформатированный текст и появится меню с выбором алгоритма поиска линейного или Бойера-мура.



**Выводы.**

Я научился работать с текстовыми строками, классом string, изучил метод поиска подстроки Бойера-Мура.

Приложение А

рабочий код

#include <iostream>  
#include <string>  
#include <fstream>  
#include <Windows.h>  
#include <set>  
#include <sstream>  
#include <iterator>  
  
using namespace std;  
bool TextInput(string &str1){  
 char buff;  
 ifstream openF("C:\\Users\\simak\\CLionProjects\\first\\inputtt.txt");  
 if(!openF.is\_open()){  
 cout << "Failed to open file";  
 return false;  
 }  
 while ( ! openF.eof () ){  
 openF.get(buff);  
 str1 += buff;  
 }  
  
 cout << str1 << '\n';  
 openF.close ();  
 return true;  
}  
void task2( string &str1) {  
 for (int i = 0; i < str1.length(); i++) {  
 if ((str1[i] == ' ') && (str1[i + 1] == ' ')) {  
 while (!((str1[i] == ' ') && (str1[i + 1] != ' '))) {  
 str1.erase(str1.begin() + i);  
 }  
 }  
 if ((str1[i] == '!') && (str1[i + 1] == '!')) {  
 while (!((str1[i] == '!') && (str1[i + 1] != '!'))) {  
 str1.erase(str1.begin() + i);  
 }  
 }  
 if ((str1[i] == '?') && (str1[i + 1] == '?')) {  
 while (!((str1[i] == '?') && (str1[i + 1] != '?'))) {  
 str1.erase(str1.begin() + i);  
 }  
 }  
 if ((str1[i] == ':') && (str1[i + 1] == ':')) {  
 while (!((str1[i] == ':') && (str1[i + 1] != ':'))) {  
 str1.erase(str1.begin() + i);  
 }  
 }  
 if ((str1[i] == ';') && (str1[i + 1] == ';')) {  
 while (!((str1[i] == ';') && (str1[i + 1] != ';'))) {  
 str1.erase(str1.begin() + i);  
 }  
 }  
 if ((str1[i] == '"') && (str1[i + 1] == '"')) {  
 while (!((str1[i] == '"') && (str1[i + 1] != '"'))) {  
 str1.erase(str1.begin() + i);  
 }  
 }  
 if ((str1[i] == ',') && (str1[i + 1] == ',')) {  
 while (!((str1[i] == ',') && (str1[i + 1] != ','))) {  
 str1.erase(str1.begin() + i);  
 }  
 }  
 if (str1[i] == '.') {  
 string buff;  
 buff = str1[i];  
 int j = i;  
 j++;  
 while (str1[j] == '.') {  
 buff += str1[j];  
 j++;  
 }  
 if (buff.length() < 3) {  
 while (!((str1[i] == '.') && (str1[i + 1] != '.'))) {  
 str1.erase(str1.begin() + i);  
 }  
 }  
 else {  
 while (str1[i + 3] == '.') {  
 str1.erase(str1.begin() + i);  
 }  
 i += 3;  
 }  
 }  
 }  
}  
bool letter(char x)  
{  
 return (x >= 'A' && x <= 'Z') || (x >= 'a' && x <= 'z');  
}  
void task2\_1(string &str){  
 for (int i = 0; i < str.length(); i++){  
 if (letter(str[i])){  
 i++;  
 while(letter(str[i])){  
 str[i] = tolower(str[i]);  
 i++;  
 }  
 }  
 }  
}  
  
void scopy(char\* str\_to, char\* str\_from) {  
 unsigned i = 0;  
 for (; str\_from[i]; ++i)  
 str\_to[i] = str\_from[i];  
 str\_to[i] = '\0';  
}  
  
void tok(string& str, char words[][11]) {  
 char word[11]{};  
 int wlx = 0;  
 int wx = 0;  
  
 for (int x = 0; str[x]; x++) {  
 if (wx != 0 && str[x] == ' ') {  
 word[wx] = '\0';  
 scopy(words[wlx++], word);  
 wx = 0;  
 }  
 if (str[x] != ' ')  
 word[wx++] = str[x];  
 }  
  
 word[wx] = '\0';  
 words[wlx][0] = '\0';  
 if (wx != 0)  
 scopy(words[wlx], word);  
}  
  
  
  
void task3(string& str)  
{  
 istringstream ss(str); // поток из строки str  
 multiset<string> vv; // контейнер упорядоченных значений  
 string tmp;  
 while (ss >> tmp) // берем из потока по словам  
 vv.insert(tmp); // вставляем в упорядоченный контейнер  
 for (auto x : vv)  
 cout << x << ' '; // выводим содержимое контейнера  
}  
  
  
void task4(string& str)  
{  
 char tokenized[50][11]{};  
 tok(str, tokenized);  
  
 for (int i = 0; tokenized[i][0]; i++)  
 cout << tokenized[i] << '\n';  
}  
void linSearch(string strM, string str1){  
 int len = str1.length();  
 string buff;  
 bool flag = 0;  
 for (int i = 0; i < strM.length(); i++){  
 int j = i, count = 0;  
 while(count != len){  
 buff += strM[j];  
 j++;  
 count++;  
 }  
 if (buff == str1){  
 cout << i << ' ';  
 flag = 1;  
 }  
 buff = "";  
 }  
 if (!flag) cout << "Line not found";  
 cout << '\n';  
}  
void b\_mur(string strM, string str1){  
 int len = str1.length(), mass[len], index = 0;  
 string strBuff = str1.substr(0, len - 1);  
 bool flag = 0;  
 for (int i = len - 2; i >= 0; i--){  
 if (strBuff.rfind(str1[i]) != i){  
 mass[i] = mass[strBuff.rfind(str1[i])];  
 }  
 else{  
 mass[i] = len - (i + 1);  
 }  
 }  
 if(str1.find(str1[len - 1]) != len - 1){  
 mass[len - 1] = mass[str1.find(str1[len - 1])];  
 }  
 else{  
 mass[len - 1] = len;  
 }  
 while(true){  
 if(index <= strM.length() - len) strBuff = strM.substr(index, len);  
 else break;  
 if (strBuff == str1){  
 cout << index <<' ';  
 flag = true;  
 index++;  
 }  
 else{  
 if(str1.find(strBuff[len - 1]) != std :: string :: npos ){  
 index += mass[str1.find(strBuff[len - 1])];  
 }  
 else{  
 index += len;  
 }  
 }  
 }  
 cout << '\n';  
 if (!flag) cout << "Line not found" <<'\n';  
}  
int main() {  
 char key;  
 string strM, str1,strCopy;  
 while(true){  
 repeat1 :  
 cout << "\nChoose a category from below:\n";  
 cout << "0. Exit\n";  
 cout << "1. Read from a keyboard\n";  
 cout << "2. Read text from a file\n";  
 cin.clear();  
 cin >> key;  
 if (key == '0') break;  
 switch (key){  
 case '1':  
 cin.clear();  
 while(cin.get() != '\n');  
 getline(cin, strM);  
 break;  
 case '2':  
 TextInput(strM);  
 break;  
 default:  
 cout << "Error";  
 goto repeat1;  
 }  
 task2(strM);  
 task2\_1(strM);  
 cout << '\n' << "Task 2" <<'\n'<< strM << '\n'<< '\n';  
 strCopy = strM;  
 cout << "Task 3" << '\n';  
 task3(strM);  
 cout <<'\n'<< '\n';  
 cout << "Task 4:" <<'\n';  
 task4(strM);  
 cout <<'\n';  
 cout << "Task 5" <<'\n'<< "Enter the line:" << '\n';  
 cin >> str1;  
 repeat2:  
 cout << "Search method:"<< '\n'<< "1. Linear search"<<'\n'<<"2 Boyer Moore search"<<'\n';  
 cin >> key;  
 switch(key){  
 case '1':  
 linSearch(strCopy, str1);  
 break;  
 case '2':  
 b\_mur(strCopy, str1);  
 break;  
 default:  
 goto repeat2;  
 }  
 }  
 return 0;  
}