**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. |  | Юрин А.А. |
| 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. Программа так же должна содержать паттерн поиска подстроки Кнута-Морриса-Пратта (далее КМП).

**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.

Для удобной работы с строками существует высокоуровневый класс string в пространстве имен std. Этот класс предоставляет готовый набор методов для работы со строками, что значительно упрощает разработку и уменьшает объем исходного кода, делая его более читабельным.

**Постановка задачи.**

Разработать алгоритм и написать программу, которая:

1. Считывает строку с файла или с клавиатуры. Способ ввода выбирается пользователем.
2. Редактирует входной текст
   1. Удаляет лишние пробелы
   2. Удаляет лишние знаки препинания
   3. Исправляет регистр букв
3. Выводит на экран все слова, содержащие
   1. Только буквы
   2. Только цифры
   3. И буквы, и цифры
4. Выводит на экран ту же строку, заменив во всех словах цифры на буквы латинского алфавита, номера которых соответствуют заменяемой цифре (0 – a, 1 – b и т.д.)
5. Осуществляет поиск подстроки, введённой с клавиатуры.
   1. Производит поиск с помощью алгоритма линейного поиска
   2. Производит поиск с помощью алгоритма КМП

**Выполнение работы.**

Код программы представлен в приложении А.

1. При запуске программы пользователю предлагается ввести строку. Если пользователь оставит ввод пустым, то строка будет прочитана из файла (рис. 1).

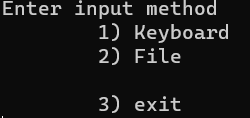


Рисунок . Запуск программы и ввод строки

1. После ввода пользователю выводится подсказка команды, которая выводит меню доступных функций и ожидается ввод команды с клавиатуры
2. Следующий шаг зависит от введенной команды, если пользователь ввёл:
   1. “0”, то выполнение программы завершается.
   2. “1”, то текст редактируется: удаляются лишние пробелы и знаки препинания, исправляется регистр букв (рис. 3)

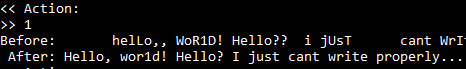


Рисунок 3. Задача 1

* 1. “2”, то поочередно выводятся: все слова, содержащие только буквы, только цифры и слова, содержащие и буквы, и слова (рис. 4).

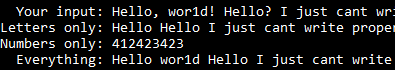


Рисунок 4. Задача 2

* 1. “3”, то выполняется замена всех встречаемых цифр в строке на соответствующие им буквы («соответствующие» - имеющие номер в латинском алфавите, равный заменяемой цифру) (рис. 5).

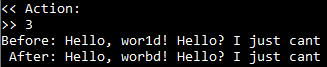


Рисунок 5. Задача 3

* 1. “4”, то выполняется поиск всех вхождений подстроки линейным алгоритмом и паттерном КМП (рис. 6).

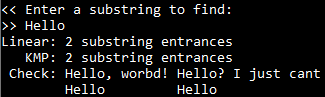


Рисунок 6. Задача 4

* 1. Любой символ или набор символов, отличный от существующий команд, то будет выведено сообщение об ошибке (рис. 8).



Рисунок 7. Неизвестная команда

**Выводы.**

В ходе работы были изучены текстовые строки как массивы, класс string и методы обработки строк. Изучен паттерн поиска подстроки в строке Кнута-Морриса-Пратта.

Приложение А

рабочий код

#include <iostream>  
#include <fstream>  
#include <string>  
  
using namespace std;  
//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.  
unsigned short input(string &str) {  
 cout << "Напишите предложение (или просто нажмите Enter, чтобы применить чтение из input.txt):" << endl;  
 cin.sync();  
 getline(cin, str);  
 if (str.empty()) {  
 const char \*file = "C:/Users/super/CLionProjects/prac4/input.txt";  
 ifstream File(file, ios::in);  
 if (!File.is\_open()) {  
 cout << "Открыть файл не удалось! \n";  
 return 1;  
 }  
 while (!File.eof()) getline(File, str);  
 File.close();  
 }  
 return 0;  
}  
  
bool letter(char x) {  
 return (x >= 'A' && x <= 'Z') || (x >= 'a' && x <= 'z');  
}  
  
bool number(char x) {  
 return (x >= '0' && x <= '9');  
}  
  
//2. Необходимо отредактировать входной текст:  
//·удалить лишние пробелы;  
//·удалить лишние знаки препинания (под ?лишними? подразумевается несколько подряд идущих знаков;  
//·исправить регистр букв, если это требуется (пример некорректного использования регистра букв: пРиМЕр);  
void task2(string &str) {  
 for (int i = 0; i < str.length(); i++) {  
 if ((str[i] == ' ') && (str[i + 1] == ' ')) {  
 while (!((str[i] == ' ') && (str[i + 1] != ' '))) {  
 str.erase(str.begin() + i);  
 }  
 }  
 }  
 for (int i = 0; i < str.length(); i++) {  
 if ((str[i] == '?') && (str[i + 1] == '?')) {  
 while (!((str[i] == '?') && (str[i + 1] != '?'))) {  
 str.erase(str.begin() + i);  
 }  
 }  
 }  
 for (int i = 0; i < str.length(); i++) {  
 if ((str[i] == '!') && (str[i + 1] == '!')) {  
 while (!((str[i] == '!') && (str[i + 1] != '!'))) {  
 str.erase(str.begin() + i);  
 }  
 }  
 }  
 for (int i = 0; i < str.length(); i++) {  
 if ((str[i] == '"') && (str[i + 1] == '"')) {  
 while (!((str[i] == '"') && (str[i + 1] != '"'))) {  
 str.erase(str.begin() + i);  
 }  
 }  
 }  
 for (int i = 0; i < str.length(); i++) {  
 if ((str[i] == ':') && (str[i + 1] == ':')) {  
 while (!((str[i] == ':') && (str[i + 1] != ':'))) {  
 str.erase(str.begin() + i);  
 }  
 }  
 }  
 for (int i = 0; i < str.length(); i++) {  
 if ((str[i] == ';') && (str[i + 1] == ';')) {  
 while (!((str[i] == ';') && (str[i + 1] != ';'))) {  
 str.erase(str.begin() + i);  
 }  
 }  
 }  
 for (int i = 0; i < str.length(); i++) {  
 if ((str[i] == ',') && (str[i + 1] == ',')) {  
 while (!((str[i] == ',') && (str[i + 1] != ','))) {  
 str.erase(str.begin() + i);  
 }  
 }  
 }  
 for (int i = 0; i < str.length(); i++) {  
 if (str[i] == '.') {  
 string buff;  
 buff = str[i];  
 int j = i;  
 j++;  
 while (str[j] == '.') {  
 buff += str[j];  
 j++;  
 }  
 if (buff.length() < 3) {  
 while (!((str[i] == '.') && (str[i + 1] != '.'))) {  
 str.erase(str.begin() + i);  
 }  
 } else {  
 while (str[i + 3] == '.') {  
 str.erase(str.begin() + i);  
 }  
 i += 3;  
 }  
 }  
 }  
}  
  
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++;  
 }  
 }  
 }  
}  
  
//3.5. Вывести на экран только те слова последовательности, в которых первая буква слова встречается в этом слове еще раз.  
void task3(string &str) {  
 cout << "Task 3\n";  
 bool flagSpace = false, flag = false;  
 char first;  
 string str1 = "";  
 for (int i = 0; i < str.length(); i++) {  
 if (str[i] == ' ') {  
 if (flag) {  
 cout << str1 << " ";  
 }  
 flagSpace = true;  
 flag = false;  
 str1 = "";  
 } else {  
 if (flagSpace) {  
 flagSpace = false;  
 str1 += str[i];  
 first = str[i];  
 continue;  
 }  
 if (first == str[i]) {  
 flag = true;  
 }  
 str1 += str[i];  
 }  
 }  
}  
  
void task4(string &str) {  
 for (int i = 0; i < str.length(); i++) {  
 if (letter(str[i])) {  
 string buffA, buff1;  
 int j = i;  
 while (letter(str[j]) || number(str[j])) {  
 if (letter(str[j])) {  
 buffA += str[j];  
 } else {  
 buff1 += str[j];  
 }  
 j++;  
 }  
 for (int p = 0; p < buffA.length(); p++) {  
 str[i] = buffA[p];  
 i++;  
 }  
 for (int p = 0; p < buff1.length(); p++) {  
 str[i] = buff1[p];  
 i++;  
 }  
 }  
 }  
}  
  
void linear(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 myrr(string strM, string str1) {  
 int len = str1.length(), mass[len], index = 0; // Заполняем таблицу отступов  
 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) { // Находим подстроку  
 //cout << index << '\t' << strM.length() - len;  
 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 << "Enter input method" << '\n' << '\t' << "1) Keyboard" << '\n' << '\t' << "2) File" << '\n' << '\n'  
 << '\t' << "3) exit" << '\n';  
 cin.clear();  
 cin >> key;  
 if (key == '3') break;  
 switch (key) {  
 case '1':  
 cin.clear();  
 while (cin.get() != '\n');  
 getline(cin, strM);  
 break;  
 case '2':  
 input(strM);  
 break;  
 default:  
 cout << "Error";  
 goto repeat1;  
 }  
 task2(strM);  
 cout << "Task 2.1" << '\n' << strM << '\n' << '\n';  
 task2\_1(strM);  
 cout << "Task 2.2" << '\n' << strM << '\n' << '\n';  
 strCopy = strM;  
 task3(strM);  
 task4(strM);  
 cout << "\n\nTask 4" << '\n' << strM << '\n' << '\n' << "Task 5.1" << '\n' << '\t' << "Enter the line" << '\n';  
 cin >> str1;  
 repeat2:  
 cout << "Enter search method" << '\n' << '\t' << "1) Linear search" << '\n' << '\t'  
 << "2) Fast search (Boyer Moore)" << '\n';  
 cin >> key;  
 switch (key) {  
 case '1':  
 linear(strCopy, str1);  
 break;  
 case '2':  
 myrr(strCopy, str1);  
 break;  
 default:  
 goto repeat2;  
 }  
 }  
 return 0;  
}