**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: Обработка текста и чтение данных из файла

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4372 |  | Кубрина А.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Разработать программу для обработки текстовых данных, которая позволяет пользователю вводить текст (с клавиатуры или из файла), анализировать его и исправлять опечатки; выполнять поиск подстроки в строке.

**Основные теоретические положения.**

Библиотека <fstream> предоставляет классы для работы с файловыми потоками.

1. Классы для работы с файлами

* std::ifstream: Класс для чтения данных из файлов. Он позволяет открывать файлы для чтения.
* std::ofstream: Класс для записи данных в файлы. Он используется для создания и записи в файлы.
* std::fstream: Класс, который объединяет функциональность ifstream и ofstream, позволяя как читать, так и записывать данные в файл.

1. Открытие файла

Файл открывается с помощью метода open(), который принимает имя файла и режим открытия (например, std::ios::in для чтения, std::ios::out для записи). Также можно указать режимы при создании объекта, например: std::ifstream file("example.txt");

1. Проверка успешности открытия файла

После открытия файла важно проверить, был ли файл успешно открыт, используя метод is\_open() или проверяя состояние потока:

if (!file.is\_open()) {

std::cerr << "Ошибка открытия файла!" << std::endl;

}

1. Чтение данных

Данные из файла можно читать с помощью метода getline() для чтения строк целиком, включая пробелы:

std::string line;

std::getline(file, line);

1. Режимы открытия файла

При открытии файла можно использовать различные режимы, такие как:

* std::ios::in: Открытие для чтения.
* std::ios::out: Открытие для записи (файл будет создан, если он не существует).
* std::ios::app: Открытие для добавления данных в конец файла.
* std::ios::binary: Открытие в бинарном режиме (для работы с бинарными файлами).

Класс std::string в C++ представляет собой удобный и мощный инструмент для работы с текстовыми строками. Он является частью стандартной библиотеки и предоставляет множество функций для манипуляции строками. Класс std::string позволяет динамически управлять памятью, что делает его более гибким по сравнению с обычными массивами символов. Строки могут изменять свою длину в процессе выполнения программы, что упрощает работу с текстовыми данными.

Основные операции, которые можно выполнять с объектами класса std::string, включают создание строк, их копирование, конкатенацию (объединение), сравнение и поиск подстрок. Для создания строки можно использовать конструкторы, которые принимают различные параметры, такие как литералы строк, другие строки или количество повторений символа.

Класс std::string поддерживает оператор + для конкатенации строк, а также оператор +=, который позволяет добавлять к существующей строке. Сравнение строк можно выполнять с помощью операторов ==, !=, <, >, и других, что позволяет легко проверять равенство и порядок строк.

Методы класса std::string предоставляют возможность получения длины строки с помощью метода length() или size(), доступа к отдельным символам через оператор индексирования [], а также извлечения подстрок с помощью метода substr(). Для поиска подстрок можно использовать метод find(), который возвращает индекс первого вхождения подстроки или std::string::npos, если подстрока не найдена.

**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. Вывести на экран слова последовательности в алфавитном порядке;
4. Вывести на экран количество символов в каждом слове исходной последовательности;
5. Реализовать два алгоритма поиска подстроки в строке: линейный алгоритм и КМП.

**Выполнение работы.**

Код программы представлен в приложении А.

| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| --- | --- |
| Загрузка текста в программу | |
| При запуске программы пользователь вводит текст | Ввод текста: |
| Обработка текста | |
| Пользователь видит исправленный текст, отсортированные слова, подсчёт количества букв в каждом слове. Ему предлагается возможность начать поиск подстроки в строке. | В тексте убираются лишние пробелы, знаки препинания, исправляется регистр букв в словах. Выводится список без цифр и без последовательности, которую ввел пользователь. |
| Поиск подстроки в строке | |
| Пользователь вводит подстроку для поиска | Программа обрабатывает ввод пользователя и выводит количество найденных в строке подстрок при помощи алгоритмов линейного поиска: |

**Выводы.**

Разработанная программа предназначена для обработки текстовых данных, позволяя вводить текст с клавиатуры, выделять слова и знаки препинания, сортировать слова в алфавитном порядке и выводить количество символов в каждом слове. Были изучены два метода поиска подстрок: линейный поиск.

Приложение А

рабочий код

#include <iostream>

#include <stdio.h>

using namespace std;

unsigned lengthStroki(char\* Arr) {

int length = 0;

while (Arr[length]) {

length++;

}

return length;

}

int deleteZnak(char\* arr, char znak, char\* newarr) {

int count = 0;

int index = 0;

for (int i = 0; arr[i] != '\0'; i++) {

if (arr[i] == znak) {

count++;

if (count <= 1) {

newarr[index] = arr[i];

index++;

}

}

else {

newarr[index] = arr[i];

index++;

count = 0;

}

newarr[index] = '\0';

}

return 1;

}

int deleteTochka(char\* arr, char znak, char\* newarr, int newlength) {

int count = 0;

int index = 0;

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

if (arr[i] == znak) {

count++;

if (count <= 3) {

newarr[index] = arr[i];

index++;

}

}

else {

newarr[index] = arr[i];

index++;

count = 0;

}

newarr[index + 1] = '\0';

}

return 1;

}

int BookvaMal(char\* arr, int newlength) {

const int bookva = 32;

if (arr[0] >= 97 and arr[0] <= 122)

arr[0] -= bookva;

for (int i = 1; i < newlength - 2; i++) {

if (arr[i] == ' ' and arr[i - 1] == '.' and arr[i + 1] >= 97 and arr[i + 1] <= 122) {

arr[i + 1] -= bookva;

}

if (arr[i] == ' ' and arr[i - 1] != '.' and arr[i + 1] >= 65 and arr[i + 1] <= 90) {

arr[i + 1] += bookva;

}

if (arr[i] != ' ' and (arr[i - 2] != '.' and arr[i - 1] != '.') and arr[i] >= 65 and arr[i] <= 90)

arr[i] += bookva;

}

return 1;

}

int Number(char\* arr, char\* newarr, int newlength) {

int index = 0;

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

if (arr[i] < 48 or arr[i]>57) {

newarr[index] = arr[i];

index++;

}

newarr[index + 1] = '\0';

}

return 1;

}

int Number2(char\* arr, char\* podstroka, char\* newarr) {

int length1 = lengthStroki(arr);

int length2 = lengthStroki(podstroka);

int count = 0;

int index = 0;

for (int i = 0; i <= length1 - length2; i++) {

int j;

for (j = 0; j < length2; j++) {

if (arr[i + j] != podstroka[j])

break;

}

if (j != length2) {

newarr[index] = arr[i];

index++;

}

else {

i = i + length2-1;

}

newarr[index + 1] = '\0';

}

return 1;

}

int linearSearch(char\* arr, char\* podstroka) {

int length1 = lengthStroki(arr);

int length2 = lengthStroki(podstroka);

int count = 0;

for (int i = 0; i <= length1 - length2; i++) {

int j;

for (j = 0; j < length2; j++) {

if (arr[i + j] != podstroka[j])

break;

}

if (j == length2)

count++;

}

return count;

}

int main() {

setlocale(0,"");

int const length = 100;

char stroka[length];

char newstroka[length];

int newlength;

cout << "Введите вашу строку: " << endl;

cin.getline(stroka, length);

cin.sync();

char Znak1[length];

deleteZnak(stroka, ' ', Znak1);

char Znak2[length];

deleteZnak(Znak1, ',', Znak2);

char Znak3[length];

deleteZnak(Znak2, '!', Znak3);

char Znak4[length];

deleteZnak(Znak3, '?', Znak4);

char Znak5[length];

deleteZnak(Znak4, '/', Znak5);

char Znak6[length];

deleteZnak(Znak5, ':', Znak6);

char Znak7[length];

deleteZnak(Znak6, ';', Znak7);

char Znak8[length];

deleteZnak(Znak7, '(', Znak8);

char Znak9[length];

deleteZnak(Znak8, ')', Znak9);

char Znak10[length];

deleteZnak(Znak9, '[', Znak10);

char Znak11[length];

deleteZnak(Znak10, ']', Znak11);

char Znak12[length];

deleteZnak(Znak11, '-', Znak12);

char Znak13[length];

deleteZnak(Znak12, '\"', Znak13);

int newlength2 = lengthStroki(Znak5);

deleteTochka(Znak5, '.', newstroka, newlength2);

BookvaMal(newstroka, newlength2);

cout << "Ваша строка: " << endl << newstroka << endl;

char newstroka2[length];

Number(newstroka, newstroka2, newlength2);

cout << "Ваша строка без чисел: " << endl << newstroka2 << endl;

cout << "Введите последовательность которую нужно удалить: " << endl;

char posled[length];

cin.getline(posled, length);

cin.sync();

int posledlength = lengthStroki(posled);

char newstroka3[length];

Number2(newstroka, posled, newstroka3);

cout << "Ваша строка: " << endl << newstroka3 << endl;

cout << "Линейный поиск: " << endl;

char podstroka[length];

cin.getline(podstroka, length);

cin.sync();

int count = linearSearch(newstroka, podstroka);

cout << count << " подстрок было найдено " << endl;

}