**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(ка) гр. |  | Коврижных А.Н. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цели работы:** разработать алгоритм и написать программу, которая позволяет работать с текстом.

**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.

При выводе строк можно использовать форматирование (манипуляторы или функции потока вывода). Вывод текстовых строк на экран крайне простая задача:

char Str[21] = “Это пример текста”;

cout  <<  Str << endl;

cout  <<  “Это текстовый литерал.” << endl;

Ввод текста с клавиатуры можно осуществлять разными способами, каждый из которых имеет определенные особенности.

Непосредственное чтение текстовых строк из потока вывода осуществляется до первого знака пробела.

Такой способ чтения обеспечивает ввод символов до первого пробельного символа (не до конца строки). Остальные символы введенного с клавиатуры остаются в потоке ввода и могут быть прочитаны из него следующими операторами >>.

Для того чтобы прочесть всю строку полностью, можно воспользоваться одной из функций gets или gets\_s (для этого в программу должен быть включен заголовочный файл <stdio.h>).

Функция gets имеет один параметр, соответствующий массиву символов, в который осуществляется чтение. Вторая функция (gets\_s) имеет второй параметр, задающий максимальную длину массива символов .

Ввод текста, длина которого (вместе с нулевым символом) превышает значение второго параметра (то есть длины символьного массива ), приводит к возникновению ошибки при выполнении программы

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”

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:

string S1; // Переменная с именем s1 типа string

string S2 = “Пример”; // объявление с инициализацией

Создание нового типа string было обусловлено недостатками работы с строками символов, который показывал тип char. В сравнении с этим типом string имеет ряд основных преимуществ:

·        возможность использования для обработки строк стандартные операторы С++(=,+,<,==,>,+=,!=,<=,>=,[])(=,+,<,==,>,+=,!=,<=,>=,[]). Использование типа char приводило требовало написание чрезмерного программного кода;

·        обеспечение лучшей надежности программного кода;

·        обеспечение строки, как самостоятельного типа данных.

### Прямой поиск

При работе со строками часто будет возникать потребность в поиске набора символа или слов (поиска подстроки в строке). При условии, что текст может быть крайне большим, хочется, чтобы алгоритм поиска подстроки работал быстро.

Самый простой способ подстроки в строке – Линейный поиск – циклическое сравнение всех символов строки с подстрокой. Действительно, этот способ первый приходит в голову, но очевидно, что он будет самым долгим.

Изображение выглядит как текст, Шрифт, число, линия

Автоматически созданное описание

На первых двух итерациях цикла сравниваемые буквы не будут совпадать. На третьей же итерации, совпал символ ‘L’, это означает, что теперь нужно сравнивать следующий символ подстроки со следующим символом строки. Видно, что символы отличаются, поэтому алгоритм продолжает свою работу. На четвертой же итерации подстрока была найдена.

Если представить, что исходная строка непорядок больше и подстрока находится в конце строки (или вовсе отсутствует), то сразу видны минусы данного алгоритма.

Одной из вариаций алгоритма полного перебора является реализация таблицы включений. Суть алгоритма: дана подстрока S и строка T. Требуется определить индекс, начиная с которого образец S содержится в строке T. Если S не содержится в T, необходимо вернуть индекс, который не может быть интерпретирован как позиция в строке.

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Хоть алгоритм и работает быстрее, по-прежнему необходимо сначала пройти всю строку, чтобы определить префиксы или суффиксы (вхождение (индексы) символов).

**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 обратном порядке.

4) Вывести все слова исходной последовательности на экран вертикально.

**Выполнение работы:**

В функции keyboardInput() и fileInput() создается переменная типа string.

В keyboardInput() строку заполняет пользователь.

void keyboardInput() {

system("cls");

cout << "Введи текст до 50 символов.\n";

string text;

cin.ignore(); // Игнорируем символ новой строки после выбора способа ввода

getline(cin, text);

editText(text);

}

cin.ignore() убирает новую строку. А само заполнение происходит с помощью getline().

В fileInput() строка заполняется из файла.

ifstream inputFile;

inputFile.open("LabWork4.txt");

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

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

return;

}

getline(inputFile, text);

editText(text);

Данный код открывает файл "LabWork4.txt" в режиме чтения с помощью объекта ifstream с именем inputFile. Затем он проверяет, успешно ли удалось открыть файл, используя метод is\_open(). Если файл не удалось открыть, код выводит сообщение об ошибке с помощью объекта cerr и завершает функцию с помощью оператора return.

Если файл успешно открыт, код считывает строку из файла с помощью функции getline, записывает ее в переменную text и затем передает эту строку в функцию editText для дальнейшей обработки. Однако в данном коде не указано, что происходит с переменной text после вызова функции editText. Возможно, после обработки текста выполняются другие операции, но эта часть кода не представлена.

В функции editText идет редактирование текста.

В цикле while, первое условие убирает пробел. Второе условие сдвигает индекс, если есть троеточие. Третье условие убирает лишние знаки препинания.

В цикле for исправление регистра букв. Первая буква заглавная. Далее внутри цикла идет проверка. Если перед буквой стоит знак препинания на за ним пробел, значит буква заглавная. Это сделано, чтобы вначале предложении были заглавные буквы.

int ind = 0;

int size = text.size();

while (ind < size) {

if (ind < size - 1 && (text[ind] == text[ind + 1] && text[ind] == ' ')) {

text.replace(ind, 1, "");

size--;

}

else if (ind + 3 < size && text[ind] == '.' && text[ind + 1] == '.' && text[ind + 2] == '.' && text[ind + 3] == ' ') {

ind += 3;

}

else if (ind < size - 1 && ispunct(text[ind]) && ispunct(text[ind + 1])) {

text.replace(ind + 1, 1, "");

size--;

}

else {

ind++;

}

}

text[0] = toupper(text[0]);

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

if (ispunct(text[i]) && i + 1 < size) {

char nextChar = text[i + 1];

if (nextChar == ' ' && i + 2 < size) {

char charAfterSpace = text[i + 2];

if (charAfterSpace != ',' && charAfterSpace != ':' && charAfterSpace != ';') {

text[i + 2] = toupper(text[i + 2]);

i += 2;

}

}

}

else {

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

}

}

В функции reverseText() слова последовательности выводятся в обратном порядке. Строка text переворачивается с использованием функции reverse(), примененной к диапазону от начала до конца строки. Объявляются итераторы textbegin, указывающий на начало строки text, и textend, указывающий на конец строки text. Выполняется внешний цикл, который продолжается, пока итератор textbegin не достигнет конца строки text. Внутри внешнего цикла выполняется внутренний цикл, который продолжается, пока итератор textbegin не достигнет конца строки text или не встретит пробельный символ. Внутренний цикл предназначен для пропуска пробельных символов в начале слова. Итератор textend устанавливается равным textbegin. Выполняется второй внутренний цикл, который продолжается, пока итератор textend не достигнет конца строки text или не встретит пробельный символ. Внутренний цикл предназначен для нахождения конца текущего слова.Диапазон символов между итераторами textbegin и textend переворачивается с использованием функции reverse(). Таким образом, каждое слово в строке обращается.

Итератор textbegin устанавливается равным textend, чтобы перейти к началу следующего слова.

cout << "Текст в обратном порядке.\n";

reverse(text.begin(), text.end());

auto textbegin = text.begin();

auto textend = text.end();

while (textbegin != text.end()) {

while (textbegin != text.end() && isspace(\*textbegin)) {

textbegin++;

}

textend = textbegin;

while (textend != text.end() && ! isspace(\*textend)) {

textend++;

}

reverse(textbegin, textend);

textbegin = textend;

}

cout << text << endl;

system("pause");

vertText(oldText);

searchWords(oldText);

В функции vertText() строки кода разбивают строку **text** на отдельные слова и выводят их по одному слову в строке.

system("cls");

cout << "Вертикальный текст\n";

istringstream iss(text);

string token;

while (iss >> token) {

cout << token << endl;

}

В начале программы пользователь должен выбрать, способ заполнения строки.

Изображение выглядит как текст, Шрифт, снимок экрана, число

Автоматически созданное описание

Рис. 1. – Способы ввода.

Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание Изображение выглядит как текст, Шрифт, снимок экрана, черный

Автоматически созданное описание

Рис. 2. – Заполнения строки.

Если пользователь выбрал заполнения строки из файла. Он может ввести название своего файла для выполнения след действий.

Далее пользователю выводится изначальный текст и изменённый.

Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание

Рис. 3. – Изменённый текст.

Текст в обратном порядке.

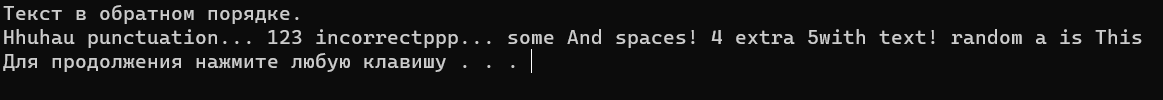


Рис. 4. - Текст в обратном порядке.

Далее выводится текст вертикально.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис. 5. – Вертикальный вывод.

**Вывод:** В ходе лабораторной работы, я узнал как работать с файлами и со строками.