**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 как массивы символов.

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. 4372 |  | Вердян А.Р. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Изучение текстовых строк как массивов символов и способов работы с ними.

**Основные теоретические положения.**

**Массив** представляет собой индексированную последовательность однотипных элементов с заранее определенным количеством элементов. Все массивы можно разделить на две группы: одномерные и многомерные. Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных. Объявление в программах одномерных массивов выполняется в соответствии со следующим правилом:

**<Базовый тип элементов> <Идентификатор массива>[<Количество элементов>]**

Так, например:

int ArrInt [10], A1 [20];

Обращение к определенному элементу массива осуществляется с помощью указания значения индекса этого элемента:

A1 [4] = -1200;

cout << A1 [4];

При обращении к конкретному элементу массива этот элемент можно рассматривать как обычную переменную, тип которой соответствует базовому типу элементов массива, и осуществлять со значением этого элемента любые операции, которые характерны для базового типа.

При объявлении массива его можно инициализировать определенными значениями:

short S [5] = {1, 4, 9, 16, 25};

Или:

short S [] = {1, 4, 9, 16, 25};

**Текстовые строки** представляются с помощью одномерных массивов символов. В языке C++ текстовая строка представляет собой набор символов, обязательно заканчивающийся нулевым символом (‘\0’). 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 текстовых литералов:

char S1[15] = “This is text”;

char S2[] = “Пример текста”;

При выводе строк можно использовать форматирование (манипуляторы или функции потока вывода). Ввод текста с клавиатуры можно осуществлять разными способами. Непосредственное чтение текстовых строк из потока вывода осуществляется до первого знака пробела. Для того, чтобы прочесть всю строку полностью, можно воспользоваться одной из функций gets или get\_s.

Существует несколько способов поиска подстроки в строке:

1. Прямой поиск (линейный поиск) – это циклическое сравнение всех символов строки с подстрокой. Он является самым долгим. Одной из вариаций данного алгоритма является реализация таблицы включений.
2. Алгоритм Кнута-Мориса-Пратта – это эффективный алгоритм поиска подстроки в строке. Он основан на принципе, что при несовпадении символов между искомой подстрокой и текстом, можно использовать уже полученные результаты для пропуска частей текста, которые точно не совпадут.
3. Алгоритм Бойера – Мура считается наиболее быстрым среди алгоритмов общего назначения, предназначенных для поиска подстроки в строке. Его преимущество заключается в том, что ценной некоторого количества предварительных вычислений над подстрокой, подстрока сравнивается с исходным текстом не во всех позициях (пропускаются позиции, которые точно не дадут положительный результат). Поиск подстроки ускоряется благодаря созданию таблиц сдвигов. Сравнение подстроки со строкой начинается с последнего символа подстроки, а затем происходит прыжок, длина которого определяется по таблице сдвигов. Таблица сдвигов строится по подстроке так, чтобы перепрыгнуть максимальное количество символов строки и не пропустить вхождение строки в подстроку.

**Постановка задачи.**

Необходимо разработать алгоритм и написать программу, которая реализует следующие задачи:

1. С клавиатуры 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. Отредактировать входной текст: удалить лишние пробелы; удалить лишние знаки препинания (под «лишними» подразумевается несколько подряд идущих знаков); исправить регистр букв, если это требуется.
3. Вывести на экран последовательность без цифр.
4. Вывести на экран ту же последовательность, удалив из всех слов заданный набор букв и (или) цифр.
5. Необходимо найти все подстроки, которые введет пользователь в имеющейся строке. Реализовать алгоритм линейного поиска.

**Выполнение работы.**

Код программы представлен в приложении А.

Описание кода и использованных алгоритмов:

При запуске программы перед пользователем появляется окно, в котором на экран выводится

1. Дается возможность ввести строку текста.
2. Исправленная строка.
3. Выводит последовательность без цифр.
4. Вывод на экран той же последовательности, удалив из всех слов заданный набор букв и (или) цифр.
5. Интерактивное поле, где пользователь может ввести подстроку, которую он хочет найти в строке.
6. Результат поиска с использованием алгоритма линейного поиска.

Работа алгоритма и вывод на экран:



Рис. 1. Поле для ввода строки.

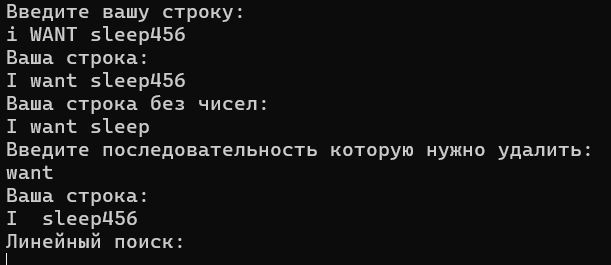


Рис.2 Вывод вашей строки.

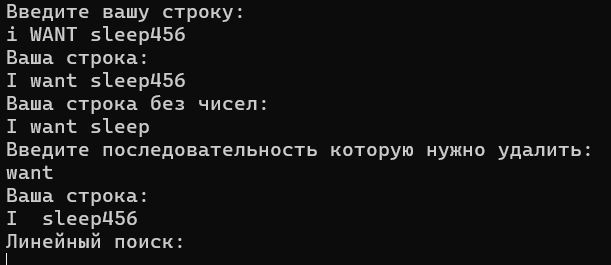


Рис. 3. Вывод строки без цифр.

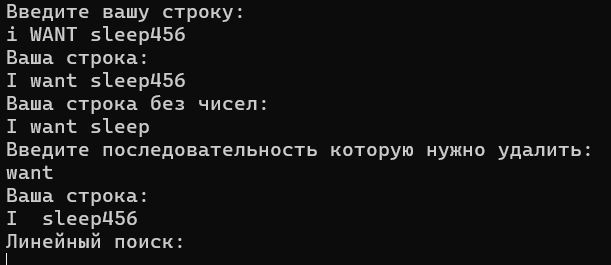


Рис. 4. Вывод на экран той же последовательности, удалив из всех слов заданный набор букв и (или) цифр.

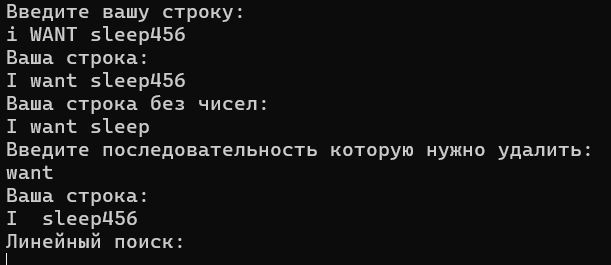


Рис. 5. Интерактивное поле для линейного поиска.

Тестовые данные:

|  |  |
| --- | --- |
| String | Substring |
| Hello, world | world |

**Выводы.**

В ходе практической работы было изучение представление текстовых строк в виде массивов символом, а также способы работы с такими строками.

Приложение А

рабочий код

#include <iostream>

#include <stdio.h>

using namespace std;

unsigned string\_len(char\* str) {

int len = 0;

while (str[len]) {

len++;

}

return len;

}

int deletePunctuation(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 new\_len) {

int count = 0;

int index = 0;

for (int i = 0; i < new\_len; 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 len1 = string\_len(arr);

int len2 = string\_len(podstroka);

int count = 0;

int index = 0;

for (int i = 0; i <= len1; i++) {

int j;

for (j = 0; j < len2; j++) {

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

break;

}

if (j != len2) {

newarr[index] = arr[i];

index++;

}

else {

i = i + len2 - 1;

}

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

}

return 1;

}

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

int length1 = string\_len(arr);

int length2 = string\_len(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 len = 100;

char stroka[len];

char newstroka[len];

int newlength;

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

cin.getline(stroka, len);

cin.sync();

char Znak1[len];

deletePunctuation(stroka, ' ', Znak1);

char Znak2[len];

deletePunctuation(Znak1, ',', Znak2);

char Znak3[len];

deletePunctuation(Znak2, '!', Znak3);

char Znak4[len];

deletePunctuation(Znak3, '?', Znak4);

char Znak5[len];

deletePunctuation(Znak4, '/', Znak5);

char Znak6[len];

deletePunctuation(Znak5, ':', Znak6);

char Znak7[len];

deletePunctuation(Znak6, ';', Znak7);

char Znak8[len];

deletePunctuation(Znak7, '(', Znak8);

char Znak9[len];

deletePunctuation(Znak8, ')', Znak9);

char Znak10[len];

deletePunctuation(Znak9, '[', Znak10);

char Znak11[len];

deletePunctuation(Znak10, ']', Znak11);

char Znak12[len];

deletePunctuation(Znak11, '-', Znak12);

char Znak13[len];

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

int newlength2 = string\_len(Znak5);

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

BookvaMal(newstroka, newlength2);

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

char newstroka2[len];

Number(newstroka, newstroka2, newlength2);

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

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

char posled[len];

cin.getline(posled, len);

cin.sync();

int posledlength = string\_len(posled);

char newstroka3[len];

Number2(newstroka, posled, newstroka3);

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

cout << "Линейный поиск: " << endl;

char podstroka[len];

cin.getline(podstroka, len);

cin.sync();

int count = linearSearch(newstroka, podstroka);

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

}