**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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 2373 |  | Шавлохова А. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение представления текстовых строк в программах. Работа со строками как с массивами символов. Операции над строками. Знакомство с классом **std::string**.

**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.

Объявленный таким образом массив может использоваться для хранения текстовых строк, содержащих не более 10 символов. Нулевой символ позволяет определить границу между содержащимся в строке текстом и неиспользованной частью строки.

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:

**Постановка задачи.**

Необходимо написать программу, которая:

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) Выполнить задание по варианту: вывести на экран слова последовательности в обратном порядке.

4) Выполнить задание по варианту: вывести на экран ту же последовательность, заменив во всех словах первую букву соответствующей прописной буквой.

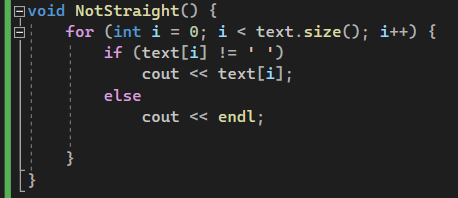
5) Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке. Реализуйте два алгоритма: первый алгоритма – Линейный поиск, а второй алгоритм согласно вашему номеру в списке. Четные номера должны реализовать алгоритм КНМ, а нечетные – Бойера-Мура.

**Выполнение работы.**

Код программы представлен в приложении А.

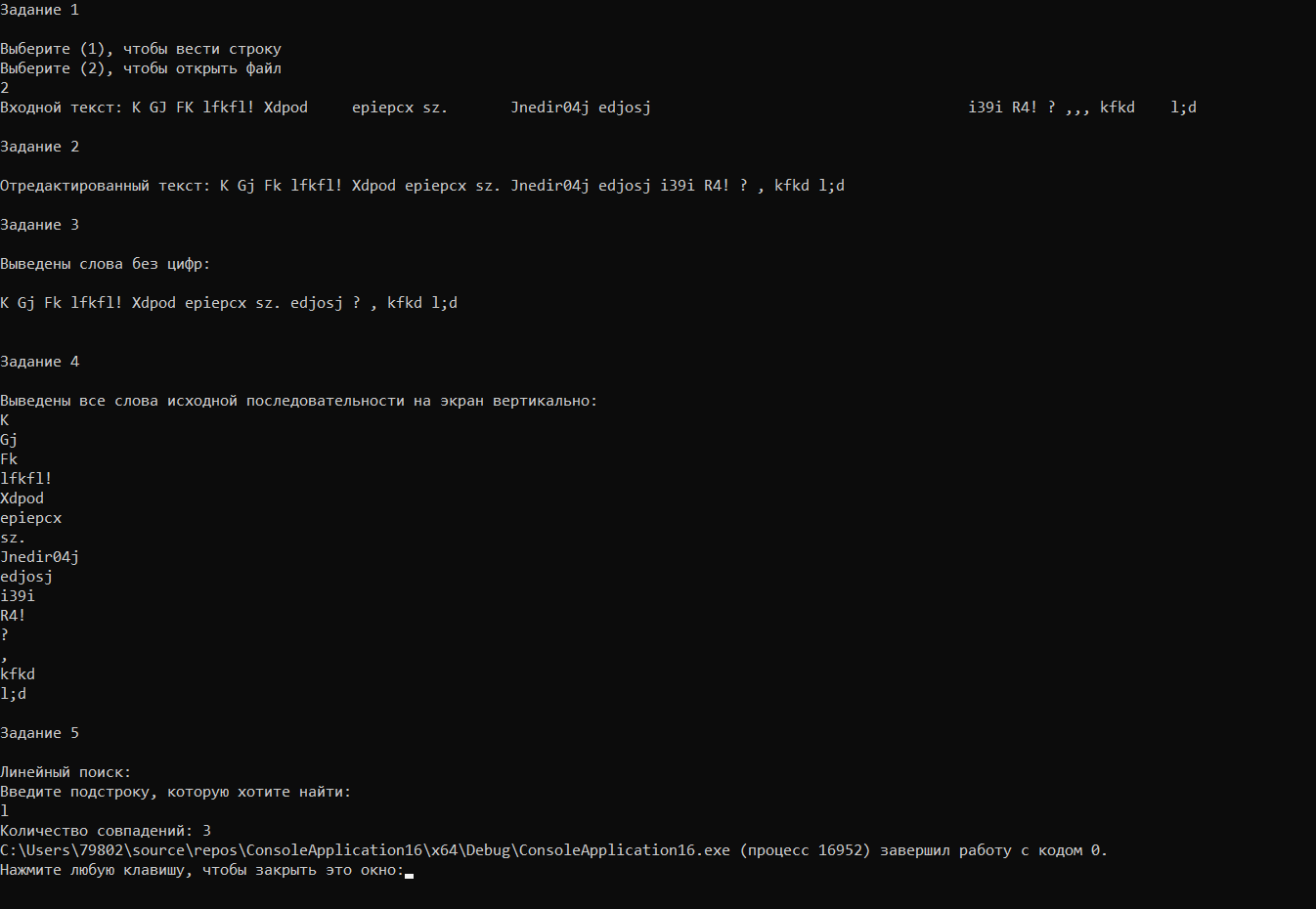
**Блок описания кода и использованных алгоритмов:**

1. Пользователь вводит строку или считывает её с файла.
2. Эта строка редактируется с помощью прописанных функций: 1 - удаляет лишние пробелы, 2 - удаляет лишние знаки препинания. 3 – исправляет регистры букв (если первая буква заглавная, то считается, что она и должна такой быть).
3. Слова выводятся на экран вертикально согласно программе:



1. В уже отредактированной строке во всех словах первая буква заменяется на заглавную. Делается это с помощью соответствующей функции.
2. Линейным поиском находится подстрока введённая пользователем.

**Блок скриншотов работы программы**



**Вывод.**

Во время выполнения данной работы были изучены строки, массивы символов и принципы работы с ними. Также был получен навык работы со строками.

Приложение А

рабочий код

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

string text;

void SpaceAnnihilator() {

for (int i = 0; i < text.size(); i++) {

if (text[i] == ' ' && text[i + 1] == ' ')

while (text[i + 1] == ' ')

text.erase((i + 1), 1);

}

}

void SignAnnihilator() {

for (int i = 0; i < text.size(); i++)

if ((text[i] == '.' || text[i] == ',' || text[i] == '?' || text[i] == '!' || text[i] == ';' || text[i] == ':') && (text[i + 1] == '.' || text[i + 1] == ',' || text[i + 1] == '?' || text[i + 1] == '!' || text[i + 1] == ';' || text[i + 1] == ':')) {

if (text[i] == '.' && text[i + 1] == '.' && text[i + 2] == '.' && text[i + 3] == ' ')

i += 3;

else

while (text[i + 1] == '.' || text[i + 1] == ',' || text[i + 1] == '?' || text[i + 1] == '!' || text[i + 1] == ';' || text[i + 1] == ':')

text.erase((i + 1), 1);

}

}

void Big2() {

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

if (text[i] == ' ')

i += 2;

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

}

}

void DigitalRacist(string txt) {

txt += ' ';

for (int i = 0, k = 0, a = 0; i < txt.size(); i++) {

if (txt[i] == '0' || txt[i] == '1' || txt[i] == '2' || txt[i] == '3' || txt[i] == '4' || txt[i] == '5' || txt[i] == '6' || txt[i] == '7' || txt[i] == '8' || txt[i] == '9')

k++;

if (txt[i] == ' ') {

if (k != 0) {

while (txt[a] != ' ')

txt.erase(a, 1);

txt.erase(a, 1);

i = a;

}

else

a = i + 1;

k = 0;

}

}

cout << txt << "\n";

}

void NotStraight() {

for (int i = 0; i < text.size(); i++) {

if (text[i] != ' ')

cout << text[i];

else

cout << endl;

}

}

void Search(string txt) {

string under;

int k = 0;

int count = 0;

cin >> under;

for (int i = 0; i < txt.size(); i++) {

if (txt[i] == under[0]) {

k = 1;

for (int j = 1; j < under.size(); j++) {

if (txt[i + j] != under[j]) {

k = 0;

break;

}

}

if (k == 1) {

count++;

}

}

}

cout << "Количество совпадений: " << count;

}

int main()

{

setlocale(0, "");

int choice;

cout << "Задание 1\n\n";

cout << "Выберите (1), чтобы вести строку" << "\n" << "Выберите (2), чтобы открыть файл" << "\n";

cin >> choice;

switch (choice) {

case 1:

cin.ignore();

getline(cin, text);

break;

case 2:

ifstream file("C:\\Users\\79802\\Desktop\\sus.txt");

if (file.is\_open())

getline(file, text);

file.close();

break;

}

cout << "Входной текст: " << text << "\n\n";

cout << "Задание 2\n\n";

SpaceAnnihilator();

SignAnnihilator();

Big2();

cout << "Отредактированный текст: " << text << "\n\n";

cout << "Задание 3\n\n";

cout << "Выведены слова без цифр: \n\n";

DigitalRacist(text);

cout << "\n\n" << "Задание 4\n\n";

cout << "Выведены все слова исходной последовательности на экран вертикально: \n";

NotStraight();

cout << "\n\n" << "Задание 5";

cout << "\n\nЛинейный поиск: \nВведите подстроку, которую хотите найти: \n";

Search(text);

}