**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Информационных Систем**

отчет

**по практической работе №3**

**по дисциплине «Программирование»**

Тема: Работа с текстовыми строками.

|  |  |  |
| --- | --- | --- |
| Студент гр. 3372 |  | Кипень В. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цель работы.**

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

· исправить регистр букв, если это требуется

3) Вывести на экран только те слова последовательности, в которых встречаются одинаковые буквы.

4) Вывести на экран ту же последовательность, заменив во всех словах цифры на буквы латинского алфавита, номера которых в алфавите равны заменяемой цифре.

5) Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке. (Линейный поиск)

**Основные теоретические положения.**

Текстовая строка представляет собой массив элементов типа char (символ), конец строки обозначается символом <\0>.

Массив представляет собой индексированную последовательность однотипных элементов с заранее определенным количеством элементов. Наглядно одномерный массив можно представить, как набор пронумерованных ячеек, в каждой из которых содержится определенное значение.

Все массивы можно разделить на две группы: одномерные и многомерные. Описание массива в программе отличается от объявления обычной переменной наличием размерности массива, которая задается в квадратных скобках после имени.

Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы (класс памяти, const и инициализатор), что и для простых переменных.

**Постановка задачи.**

Необходимо написать программу, которая реализует поставленную задачу:

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) Необходимо отредактировать входной текст:

· удалить лишние пробелы;

· удалить лишние знаки препинания

· исправить регистр букв, если это требуется

3) Вывести на экран только те слова последовательности, в которых встречаются одинаковые буквы.

4) Вывести на экран ту же последовательность, заменив во всех словах цифры на буквы латинского алфавита, номера которых в алфавите равны заменяемой цифре.

5) Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке. (Линейный поиск)

**Выполнение работы.**

|  |  |
| --- | --- |
| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| Меню | |
| При запуске программы перед пользователем появляется окно с меню, где он может выбрать пункт, который требуется выполнить | Меню:    Проверка на ввод символов, которые не входят в диапазон выбора: |

|  |  |
| --- | --- |
| Считывание строки | |
| При вводе пользователем корректного значения пункта меню (1) изначально происходит считывание строки и её вывод на экран | Как только все данные заданы, считывается строка и выводится на экран в исходном виде. |
| Обработка строки | |
| При вводе пользователем корректного значения пункта меню (2) происходит обработка и вывод на экран полученной из пункта 1 строки (убираются лишние пробелы, символы пунктуации и корректируется регистр) | Выводится обработанная строка (без лишних пробелов, символов пунктуации и корректным регистром) |
| Анализ на повторяющиеся символы | |
| При вводе пользователем корректного значения пункта меню (3) происходит вывод только тех слов строки, в которых есть повторяющиеся символы |  |
| Замена чисел на символы | |
| При вводе пользователем корректного значения пункта меню (4) все числа в строке заменяются символами (по их номеру из таблицы ASCII) и вывод на экран | Было:    Стало: |
| Поиск подстроки | |
| При вводе пользователем корректного значения пункта меню (5) происходит поиск (линейный) подстроки в исходной строке и вывод индексов совпадения | Исходная строка:    Подстрока: |

**Выводы.**

Необходимо написать программу, которая реализует поставленную задачу:

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) Необходимо отредактировать входной текст:

· удалить лишние пробелы;

· удалить лишние знаки препинания

· исправить регистр букв, если это требуется

3) Вывести на экран только те слова последовательности, в которых встречаются одинаковые буквы.

4) Вывести на экран ту же последовательность, заменив во всех словах цифры на буквы латинского алфавита, номера которых в алфавите равны заменяемой цифре.

5) Необходимо найти все подстроки, которую введёт пользователь в имеющейся строке. (Линейный поиск)

Приложение А

рабочий код

#include <iostream>

#include <Windows.h>

#include <clocale>

using namespace std;

int getWordLength(char arr[]) {

    unsigned L = 0;

     while (arr[L]) ++L;

     return L;

}

bool checkSymbolInWord(const char\* arr, char symbol) {

    const char\* ptr = arr;

    for (size\_t i = 0; \*ptr != '\0'; ++i) {

        if (\*ptr == symbol) {

            return true;

        }

        ++ptr;

    }

    return false;

}

void clearArray(char arr[], const int size) {

    memset(arr, 0, size);

}

int countOccurrences(const char\* arr, char el) {

    int count = 0;

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

        if (arr[i] == el) {

            count++;

        }

    }

    return count;

}

void findSubstring(const char\* arr, const char\* needle, const int arrSize, const int needleSize) {

    for (int i = 0; i <= arrSize - needleSize; ++i) {

        bool found = true;

        for (int j = 0; j < needleSize; ++j) {

            if (arr[i + j] != needle[j]) {

                found = false;

                break;

            }

        }

        if (found) {

            cout << "\nfind one on " << i + 1 << "-" << i + 1 + needleSize << " position\n";

            found = false;

        }

    }

}

char getCharForDigit(int digit) {

    return 'a' + (digit - 1);

}

void replaceDigitsWithLetters(char arr[]) {

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

        if (isdigit(arr[i])) {

            arr[i] = getCharForDigit(arr[i] - '0');

        } else if (ispunct(arr[i])) {

            break;

        }

    }

}

void equateArrays(char convertibleArr[], char subArr[]) {

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

        convertibleArr[i] = subArr[i];

    }

}

void getSubstring(const char arr[], char substring[], int start, int end) {

    int subIdx = 0;

    for (int i = start; i < end; i++, subIdx++) {

        substring[subIdx] = arr[i];

    }

}

int main() {

*// If when entering RU characters there are hieroglyphs, then write the command in the console: chcp 1251*

    SetConsoleCP(1251); *// Setting the input encoding*

    SetConsoleOutputCP(1251); *// Setting the output encoding*

    setlocale(LC\_ALL, "Russian");

    cout << "Navigation:" << "\n"

    << "1) A sequence of characters is entered from the keyboard (maximum 50 words of 10 characters each)" << "\n"

    << "2) Editing of entered text" << "\n"

    << "3) Display only those words that contain the same letters" << "\n"

    << "4) Replace numbers in all words with letters" << "\n"

    << "5) Find substring" << "\n";

    const int MAX\_SIZE = 500;

    char sequence[MAX\_SIZE];

    const char punctuation[] = { '.', ';', ':', '(', ')', '!', '?', ',', '&', '\*', '/', '<', '>', '?', '\"', '\'',  '#', '^' };

    while(true) {

        cin.clear(); *// Clearing the input stream from possible errors*

        cin.sync();

        short int workPoint;

        cout << "Select point of work (number 1 to 5): ";

        cin >> workPoint;

        switch (workPoint)

        {

            case 1: {

                cout << "Enter a string:\n";

                cin.get();

                cin.getline(sequence, MAX\_SIZE, '.');

                cout << "Raw text:\n" << sequence << endl;

                break;

            }

            case 2: {

                cout << "Edited text:\n";

                char editedSequence[MAX\_SIZE] = {};

                if (!checkSymbolInWord(punctuation, sequence[0])) {

                    editedSequence[0] = sequence[0];

                }

                int curIndex = 1;

                for (int i = 1; i < getWordLength(sequence) + 1; i++) {

                    if (sequence[i] != ' ') {

                        if (!checkSymbolInWord(punctuation, sequence[i]) ||

                        (checkSymbolInWord(punctuation, sequence[i]) &&

                        (sequence[i - 1] == ' ' ||

                        sequence[i + 1] == ' '))) {

                            editedSequence[curIndex] = tolower(sequence[i]);

                            curIndex++;

                        }

                    } else {

                        if (editedSequence[curIndex - 1] != ' ') {

                            editedSequence[curIndex] = ' ';

                            curIndex++;

                        }

                    }

                }

                cout << editedSequence << ".\n";

                equateArrays(sequence, editedSequence);

                break;

            }

            case 3: {

                int startIndex = 0;

                char currentWord[MAX\_SIZE] = {};

                for (int i = 0; i < getWordLength(sequence) + 1; i++) {

                    if (sequence[i] == ' ' || sequence[i] == '\0') {

                        clearArray(currentWord, MAX\_SIZE);

                        getSubstring(sequence, currentWord, startIndex, i);

                        startIndex = i + 1;

                        for (int j = 0; j < getWordLength(currentWord); j++) {

                            if (countOccurrences(currentWord, currentWord[j]) > 1) {

                                cout << currentWord << " ";

                                break;

                            }

                        }

                    }

                }

                break;

            }

            case 4: {

                replaceDigitsWithLetters(sequence);

                cout << "Replaced text:\n";

                cout << sequence << "\n";

                break;

            }

            case 5: {

                char substring[MAX\_SIZE];

                cout << "Enter a substring:\n";

                cin.get();

                cin.getline(substring, MAX\_SIZE, '.');

                findSubstring(sequence, substring, getWordLength(sequence), getWordLength(substring));

                break;

            }

            default: {

                cout << "\n" << "You did not enter a number in the range from 1 to 5";

                break;

            }

        }

        cin.clear(); *// Clearing the input stream from possible errors*

        cin.sync();

        char stopFlag;

        cout << "\n" << "Continue the program? (Y/N) ";

        cin >> stopFlag;

        if (stopFlag != 'Y' && stopFlag != 'y') {

            break;

        }

    }

    return 0;

}