ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ

За курсом «Інформатика і Програмування»

Студента групи МС-19-1

Політіки Владислава Дмитровича

Лабораторна робота №9.1

**Тема**: «Обробка текстової інформації»

Варіант 11

**Завдання:** виконати задані дії із введенним текстом.

Індивідуальне завдання за варіантами:

**Варіант 11.** Вивести всі слова, які повторюються та число їх повторень.

**Про програму**

1. Опис розв’язку:

**Загальна ідея:** для розв’язку поставленої задачі можна кожне введене слово так записувати у двовимірний масив символів, щоб кожен його індекс, починаючи з нульового, вказував на масив символів, що містить деяке введене слово, яке не є повторенням попередніх слів, які були вже добавлені. У відповідність цьому двовимірному масиву ставиться одновимірний, що рахує кількість відповідних слів, що зустрілися у введеному тексті. Кількість повторень збільшується на одиницю у тому індексі, котрий у двовимірному масиві є однаковим зі словом, яке розглядається. До того ж, коли записується нове слово, що не повторювалось, у відповідність йому ставиться початкова кількість повторень, що є рівним одиниці.

**Опис коду:** після вводу тексту до масиву символів Put, розглядаємо всі індекси цього масиву, що відповідають введеним літерам та символу з кодом нуль (‘\0’), що є ознакою кінця строки (у програмі дію розгляду всіх цих індексів виконує цикл while (i <= strlen(Put))). Під час розгляду літер, вони переписуються в масив NextW допоки не стане розглядатися деякий розділовий знак (у програмі діє так звана перевірка чи є символ розділювачем чи ні). Тож, за виконанням умови “символ - літера”, відбувається продовження перезапису в масив NextW. Для можливості перезапису збільшується на 1 змінна symcnt, що вказує на чарунку, куди має бути вписана наступна літера. Значення виконання умови (true) повертає функція is\_let(). У ній розглянуто тільки умову, коли символ є розділювачем, при виконанні якоїї фуккція повертає false, а при невиконанні - true. Після того як слово закінчується, то розглядається передача записаного слова функції add\_new\_word(). Вона спочатку перевіряє чи зустрічалося таке саме слово, переглядаючи подібність слів, що були записані до двовимірного мвсиву qWords раніше, з переданим словом. Тут розглядаються всі індекси двовимірного масиву, що послужили записом до їхніх відповідних масивів, переданих до функції слів. Кількість таких слів зберігається у змінній wordCount. Тож спочатку розглядається цикл while (i < wordCount), де у спочатку цієї функції i = 0. Перевірку на ідентичність двох масивів здійснює функція similar(). Вона повертає false, якщо довжина до символу ‘\0’ масивів є різною або хоча б один з їхніх елементів не співпадає, тобто коли ці масиви до ‘\0’ включно не є еквівалентними, у іншому випадку функція повертає true. У випадку, коли були переглянуті усі індекси двовимірного масиву та функція повертала завжди false, тобто переданий функції масив до ‘\0’ включно не співпадав з тими, які були в двовимірному, функція дописує цей масив у раніше підготовлений для масиву масив, збільшуючи wordCount (кількість доданих слів до двовимірного масиву) на одиницю. Також у відповідному масиві cWords, що рахує кількість повторів слів, елементу з індексом wordCount, що має значення до розглянутого його збільшення, присвоюємо початкове значення кількості повторів слова - одиницю. Якщо ж, у деякому переглянутому індексі двовимірного масиву функція similar повернула true, то до елемента, що має переглянутий індекс, масиву cWords, що зберігає кількість повторів слів, додамо один (це означає, що кількість повторень слова збільшилось на одиницю). Далі, повертаючись до основної функції main(), змінній, що відповідає за розташування наступної літери в масиві NextW, надаємо значення нуль для можливості перезапису масиву. Після закінчення цього циклу, виводимо кожне слово масиву qWords до останнього покладеного у нього слова з відповідною кількістю їх повторів, що зберігається у масиві cWords. Тобто створюємо цикл від 0 до wordCount-1 включно (у моїй реалізації наступне: while (i < wordCount) ... i++). Також для виводу тількі тих елементів, що повторювались, можна додати умову таку, що cWords[i] (кількість повторів певного слова у тексті) повинно бути більшим за одиницю.

1. Вихідний текст програми розв’язку задачі:

#include <iostream>

#include <Windows.h>

using namespace std;

const int maxLenPut = 300;

const int maxWords = 300;

char Put[maxLenPut]; // массив, содержащий ввод

char NextW[maxLenPut]; // массив, содержащий на каждой итерации определённого цыкла соотвецтвующее слово из массива Put

char qWords[maxWords][maxLenPut]; // двумерный массив, содержащий уникальный (не повторяющейся) набор слов из массива Put

int cWords[maxWords]; // массив, индексы которого в соотвецтвии с индексами массива qWords, показывают количество повторений слов

int wordCount = 0; // переменная хранит количество добавленных слов в массив qWords

bool similar(char \*A1, char \*A2)

{

if (strlen(A1) == strlen(A2))

{

for (int i = 0; A1[i] != '\0'; i++)

if (A1[i] != A2[i]) return false;

}

else return false;

return true;

}

void str\_copy(char \*A1, char \*A2)

{

int i;

for (i = 0; A2[i] != '\0'; i++)

A1[i] = A2[i];

A1[i] = '\0';

}

bool is\_let(char a)

{

if (a == ' ' || a == '.' || a == '!' || a == '/' || a == '?' || a == '\0'|| a == ',' || a == ';'|| a=='(' || a == ')')

return false;

else return true;

}

int add\_new\_word(char NewWord[maxLenPut]) //передаем адрес первого элемента массива NextW(массив, содержащий на каждой итерации определённого цыкла соотвецтвующее слово из массива Put)

{

int i = 0;

while (i < wordCount) // int wordCount = 0;

{

if (similar(qWords[i], NewWord)) // проверка на соотвецтвие // if (strcmp(qWords[i], NewWord) == 0)

{

// слово нашли в нашем массиве уникальных слов qWords

cWords[i] = cWords[i] + 1;

return 0;

}

i++;

}

// Если слово в NewWord не нашлось в массиве qWords

// Добавляем слово в наш масив уникальных слов qWords

str\_copy(qWords[wordCount], NewWord); //strcpy\_s(qWords[wordCount], NewWord);

cWords[wordCount] = 1; //слово NewWord повторилось 1 раз

wordCount++; //готовимся записывать новое слово в следующую строку двумерного массива qWords

return 0;

}

int main()

{

int i = 0;

int symcnt = 0;

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

cout << " Лабораторная работа №9.1\n";

cout << " Введите текст:\n ";

cin.getline(Put, maxLenPut);

NextW[0] = '\0';

while (i <= strlen(Put)) // цикл просматривает все введённые буквы

{

if (is\_let(Put[i])) // если россматриваемый символ - буква слова

{

NextW[symcnt] = Put[i]; // добавляем в массив NextW

symcnt++;

}

else // иначе

{

if (symcnt > 0)

{

NextW[symcnt] = '\0'; // добавляем признак конца

add\_new\_word(NextW); // полученный массив добавляем в двумерный, если такой же массив не был добавлен раньше

NextW[0] = '\0';

symcnt = 0; // в последующей итерации вводим следующее слово массива Put в начало NextW

}

}

i++;

}

// -------------------------------

i = 0;

while (i < wordCount)

{

if (cWords[i] > 1) cout << " Слово [" << qWords[i] << "] встретилось " << cWords[i] << " раз" << '\n';

i++;

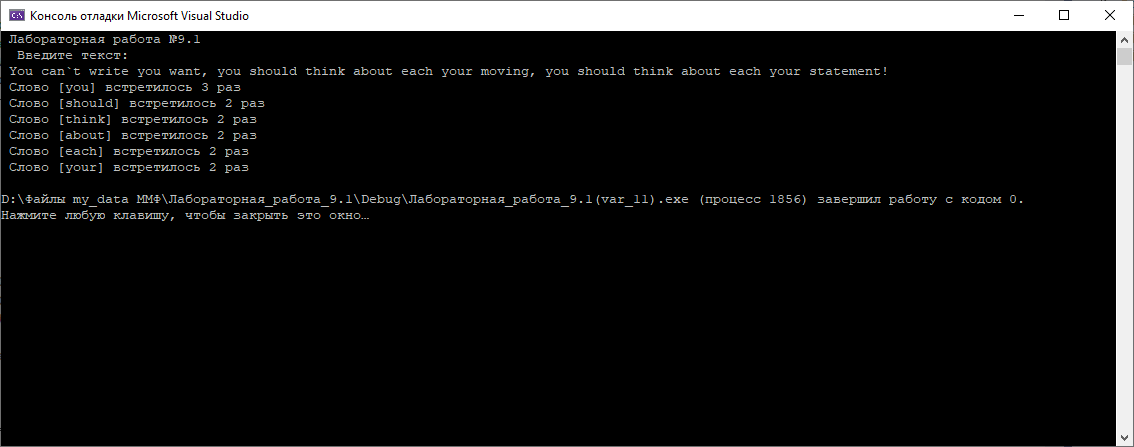
}

return 0;

}

1. Опис інтерфейсу:

Програма матиме наступний інтерфейс:



1. Опис тестових прикладів:

При всіх моїх тестових прикладах, програма видає правильну кількість повторень слів. Проте програма сприймає слова, що мають одкакові літери, але мають таку різницю, що деякі літери в першому слові великі, а другому такі самі - малі та навпаки, як різні.

1. Аналіз помилок:

Помилок немає.