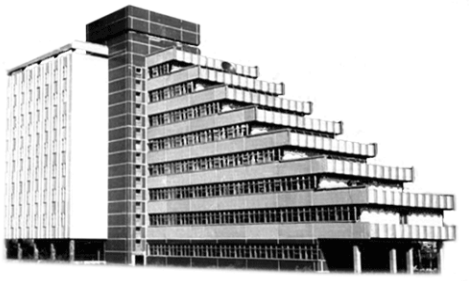
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Учебный Центр Информационных Технологий «Информатика»



Лабораторная работа №4

по дисциплине «Информатика и программирование I часть»

Направление подготовки: 230105 - «Программное обеспечение вычислительной техники и автоматизированных систем»

Выполнил слушатель: Аветисян Арташес Робертович
Вариант: 13
Дата сдачи: 23.09.2020
Преподаватель: Прытков Д. В.

Новосибирск, 2020г.

1 Задание

"Перевернуть" в строке все слова. (Например: "Жили были дед и баба" - "илиЖ илиб дед и абаб").

2 Теоретический материал

Си-строки — это массивы символов. Наряду со строками, есть также строковые литералы, такие как этот `"literal"`. В действительности, как строки, так и литералы — это просто наборы символов, расположенных рядом в памяти компьютера. Но между массивами и литералами все таки есть разница, литералы нельзя изменять и строки — можно.

Любая функция, которая принимает строку в стиле С, также может принимать в качестве параметра — литерал. В си также есть некоторые сущности, которые могут выглядеть как строки, хотя, на самом деле, они таковыми не являются. Речь идет о символах, они заключены в одинарные кавычки, например — `'a'`. Символ можно, в определенном месте, присвоить строке, но символы не могут быть обработаны в виде строки. Массивы работают как указатели, поэтому, если передать один символ в строку, это будет считаться ошибкой.

Си-строки всегда должны завершаться нулевым символом `'\0'`. Поэтому, чтобы объявить строку, состоящую из 49 букв, необходимо зарезервировать дополнительную ячейку под нулевой символ.

Важно помнить, что в конце си-строк всегда должен быть нуль-символ, точно так же как и в конце каждого предложения есть точка. Хотя нуль символ не отображается при выводе строки, он занимает место в памяти. Поэтому, технически, в массиве из пятидесяти элементов вы смогли бы сохранить только 49 букв, потому что, последний символ нужен для завершения строки. Кроме того, указатели также могут быть использованы в качестве строки.

Строки полезно использовать тогда, когда необходимо выполнять различные операции с текстовой информацией. Использование функции `scanf()` для ввода строки — работает, но это может привести к переполнению буфера. Ведь входная строка может оказаться больше, чем размер строки-буфера. Есть несколько способов для решения этой проблемы, но самый простой способ — это использовать `fgets()` функцию, которая объявлена в заголовочном файле `<stdio.h>`.

Когда `fgets()` считывает входные данные от пользователя, она будет читать все символы, кроме последнего. После этого в конец считанной строки, `fgets()` поместит нулевой терминатор. Функция `fgets()` будет считывать символы до тех пор, пока пользователь не нажмет `Enter`.

Операции со строками

Большинство операций языка Си, имеющих дело со строками, работает с указателями. Для размещения в оперативной памяти строки символов необходимо:

- выделить блок оперативной памяти под массив;
- проинициализировать строку.

Для выделения памяти под хранение строки могут использоваться функции динамического выделения памяти. При этом необходимо учитывать требуемый размер строки.

Функции ввода строк

Для ввода строки может использоваться функция scanf(). Однако функция scanf() предназначена скорее для получения слова, а не строки. Если применять формат "%s" для ввода, строка вводится до (но не включая) следующего пустого символа, которым может быть пробел, табуляция или перевод строки.

Для ввода строки, включая пробелы, используется функция

char * gets(char *);

или её эквивалент

char * gets_s(char *);

В качестве аргумента функции передается указатель на строку, в которую осуществляется ввод. Функция просит пользователя ввести строку, которую она помещает в массив, пока пользователь не нажмет *Enter*.

Функции вывода строк

Для вывода строк можно воспользоваться рассмотренной ранее функцией

printf("%s", str); // str — указатель на строку

или в сокращенном формате

printf(str);

Для вывода строк также может использоваться функция

int puts (char *s);

которая печатает строку s и переводит курсор на новую строку (в отличие от printf()). Функция puts() также может использоваться для вывода строковых констант, заключенных в кавычки.

Функция ввода символов

Для ввода символов может использоваться функция

char getchar();

которая возвращает значение символа, введенного с клавиатуры. Указанная функция использовалась в рассмотренных ранее примерах для задержки окна консоли после выполнения программы до нажатия клавиши.

Функция вывода символов

Для вывода символов может использоваться функция

char putchar(char);

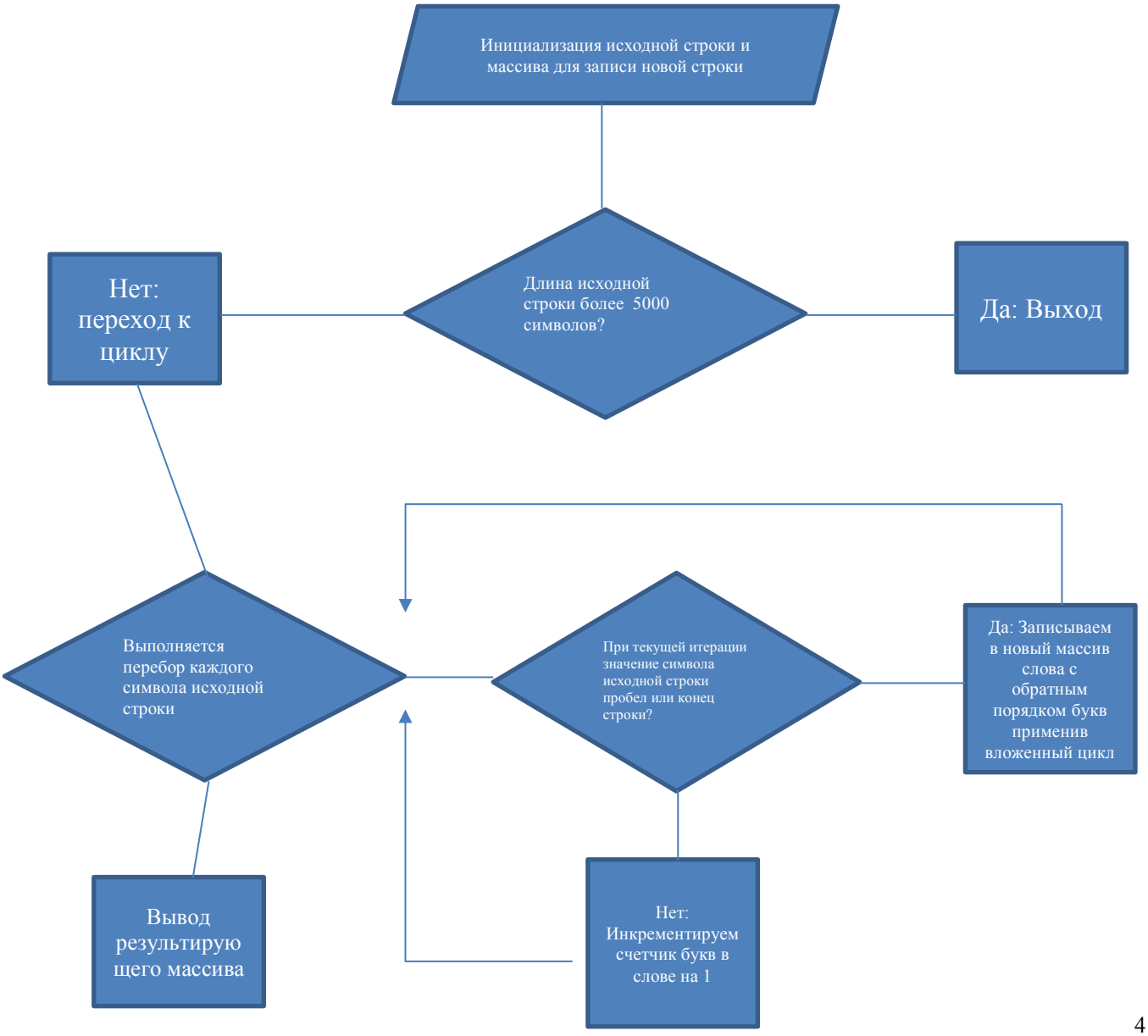
которая возвращает значение выводимого символа и выводит на экран символ, переданный в

качестве аргумента.

Основные функции стандартной библиотеки string.h

Функция	Описание
char *strcat(char *s1, char *s2)	присоединяет s2 к s1, возвращает s1
char *strncat(char *s1, char *s2, int n)	присоединяет не более n символов s2 к s1, завершает строку символом '\0', возвращает s1
char *strcpy(char *s1, char *s2)	копирует строку s2 в строку s1, включая '\0', возвращает s1
char *strncpy(char *s1, char *s2, int n)	копирует не более n символов строки s2 в строку s1, возвращает s1;
int strcmp(char *s1, char *s2)	сравнивает s1 и s2, возвращает значение 0, если строки эквивалентны
int strncmp(char *s1, char *s2, int n)	сравнивает не более n символов строк s1 и s2, возвращает значение 0, если начальные n символов строк эквивалентны
int strlen(char *s)	возвращает количество символов в строке s
char *strset(char *s, char c)	заполняет строку s символами, код которых равен значению c, возвращает указатель на строку s
char *strnset(char *s, char c, int n)	заменяет первые n символов строки s символами, код которых равен c, возвращает указатель на строку s

3 Описание алгоритма приложения



4 Описание реализации

Препроцессорные директивы:

#include - используется для включения в исходный код заголовочных файлов библиотек

Подключаемые заголовочные файлы библиотеки:

- stdio.h
- string.h

Используемые функции:

int main()

Указанная основная функция, являющаяся начальной точкой для выполнения программы. Она обычно управляет выполнением программы, вызывая другие ее функции. Как правило, выполнение программы завершается в конце функции **main**, но по разным причинам это может случиться и в других местах программы. Может быть ипользована для передачи аргументов в приложение.

int printf(const char *format, arg-list)

Функция printf() записывает в stdout аргументы из списка arg-list под управлением строки, на которую указывает аргумент format. Строка, на которую указывает format, состоит из объектов двух различных назначений. Во- первых, это символы, которые сами должны быть выведены на экран. Во-вторых, это спецификаторы формата, определяющие вид, в котором будут выведены аргументы из списка arg-list. Спецификаторы формата состоят из символа процент, за которым следует код формата.

int strlen(const char * str)

Функция strlen() возвращает длину строки, оканчивающейся нулевым символом, на которую указывает str. При определении длины строки нулевой символ не учитывается.

Используемые конструкции:

if

Оператор выбора **if** позволяет выполнять или опустить выполнять определенных участков кода, в зависимости от того является ли истинным или ложным условие этого оператора. Одно из самых важных назначений оператора выбора **if** так это то, что он позволяет программе совершить действие на выбор, в зависимости от того, например, какие данные ввел пользователь.

for

Параметрический цикл (цикл с заданным числом повторений), реализуемый при помощи операций инициализация (присваивание параметру цикла начального значения), проверки условия повторения цикла, модификации - изменения значения параметра для следующего прохождения тела цикла

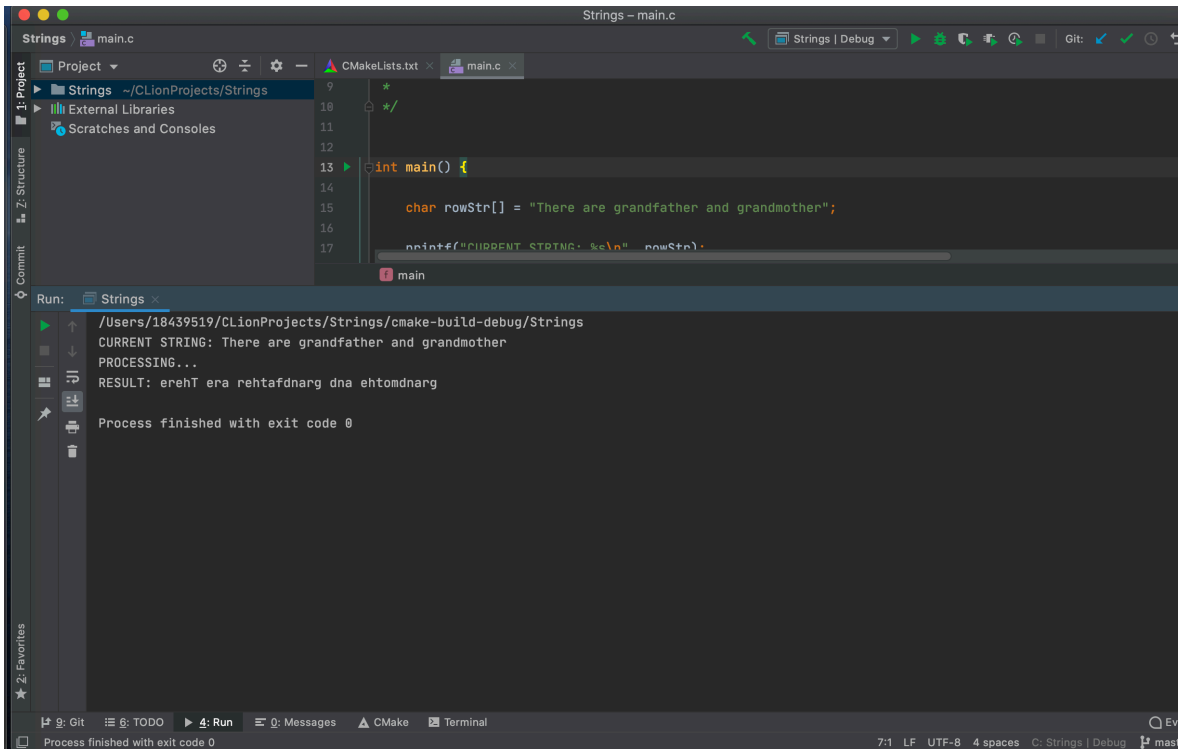
5 Пример работы программы

На рисунке 1 показана фотография дисплея компьютера с развернутым окном среды разработки JetBrains CLion после выполнения разработанного приложения. На второй строке консоли при помощи функции printf выведена обрабатываемая строка в исходном виде: «**There are grandfather and grandmother**»

На третьей строке вывод сообщения “**PROCESSING…**”, после чего начинается исполнение основного алгоритма приложения по смене положения букв в обрабатываемой строке.

На четвертой строке выведен результат выполнения алгоритма.

Рисунок 1. Фотография экрана во время выполнения приложения, разработанного согласно задачи лабораторной работы



Выводы

Изучен теоретическим материал по работе со троками в Си, который применен в практике для решения задачи в рамках данной лабораторной работы.

Успешно разработан и апробирован алгоритм смены букв в словах предложения с сохранением исходного порядка слов, однако алгоритмическая сложность разработанного решения составляет $O(n^2)$, что относит алгоритм приложения к медленным, что может быть предметов дальнейшего совершенствования приложения.

Приложение. Текст программы с комментариями

```
#include <stdio.h>
#include <string.h>

/**
 * Тема: Работа со строками в Си
 * Лабораторная №4: вариант 13
 * Задача: "Перевернуть" в строке все слова. (Например: "Жили были дед и баба" - "илиЖ илиб дед и абаб").
 * Исполнил: Аветисян Арташес Робертович
 *
 */


int main() {

    char rowStr[] = "There are grandfather and grandmother";

    printf("CURRENT STRING: %s\n", rowStr);
    printf("PROCESSING...\n");

//    Алгоритм:
//    разбить выражение на массив строк используя разделитель пробел
//    для каждого элемента в массиве выполнить обратный перебор букв
//    склеить все елементы массива с перевернутыми буквами в новуй строку

    //10 kb
    char data [100][100];
    int dataElementsLength[100];

    //длина строки для обработки
    int rowStrSize = strlen(rowStr);

    if (rowStrSize > 5000){
```

```
        printf("WARNING! Process interrupted. The length of given string is too long,
mast be less than 5000 symbols.\n");
        printf("Pleas, shorten the string and try again.\n");
        return 0;
    }

    int letterIndex = 0;
    int startIndex = 0;
    int endIndex = 0;
    int reversedWordSize = 0;
    int wordIndex = 0;

    for(int i = 0; i < rowStrSize; i++){
        //printf("%c\n", rowStr[i]);

        //условия true если встречается пробел или последняя итерация
        if ((rowStr[i] == 32) || (i == rowStrSize - 1)){
            endIndex = i - 1;
            reversedWordSize = endIndex - startIndex;
            //printf("SPACE or END OF CHAR[] '%c', wordIndex = %d, currentStartIndex =
%d, currentEndIndex = %d\n", rowStr[i], wordIndex, startIndex, endIndex);
            int dataSingleElementLetterIndex = 0;
            for (int a = endIndex; a >= startIndex; a--){
                data[wordIndex][dataSingleElementLetterIndex] = rowStr[a];
                //printf("WAS ADDED CHAR:  '%c'\n", rowStr[a]);
                dataSingleElementLetterIndex ++;
            }
            dataElementsLength[wordIndex] = reversedWordSize;

            letterIndex = 0;
            startIndex = i + 1;
            wordIndex++;
        } else {
            //printf("IN WORD, letterIndex: %d\n", letterIndex);
            letterIndex++;
        }

    }

//    for (int i = 0; i < 100; i++){
//        printf("NEW LINE IN DATA\n");
//        for (int a = 0; a < 100; a++){
//            printf("DATA letter: %c\n", data[i][a]);
//        }
//    }


    char reversedString[10000];
```

```
    int reversedStringIndex = 0;

    for (int i = 0; i < wordIndex; i++){
        //printf("wordIndex iteration = %d, \n", i);
        for(int a = 0; a <= dataElementsLength[i]; a++){
            //printf("dataElementsLength[i] = %d, \n", dataElementsLength[i]);
            reversedString[reversedStringIndex] = data[i][a];
            //printf("letter: %c, reversedStringIndex = %d\n",
reversedString[reversedStringIndex], reversedStringIndex);
            reversedStringIndex++;
        }
        reversedString[reversedStringIndex] = 32;
        reversedStringIndex++;
    }

    printf("RESULT: %s\n", reversedString);

    //TODO: алгоритмическая сложность большая O(n) + O(n*n), алгоритм
медленный

    return 0;
}
```

Защита: