МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №8

по дисциплине: Основы программирования тема: «Свободные массивы строк»

Выполнил: ст. группы ПВ-201 Машуров Дмитрий Русланович

Проверил:

Притчин Иван Сергеевич Брусенцева Валентина

Станиславовна

Белгород 2021 г.

**Лабораторная работа № 8**

**Свободные массивы строк**

**Цель работы**: закрепление навыков работы с массивами указателей и строками.

**Задания для подготовки к работе**

1. Описать функцию int get\_word (char \*s, int n), которая записывает в строку s слово длиной не более n, введенное с клавиатуры, и возвращает длину слова или EOF, если длина считываемого слова больше n. Словом считается последовательность символов, не содержащая пустых символов.
2. Разработать алгоритм и составить программу для решения задачи соответствующего варианта. Слова вводимого текста хранить в свободном массиве строк или в динамическом массиве структур, каждый элемент которого содержит член – указатель на слово, и члены с данными, необходимыми для решения задачи. Конец ввода – конец файла.
3. Подобрать тестовые данные.
4. Пояснения: функции создания свободного массива (create\_free\_arr) передается ориентировочный размер массива указателей k , который может быть изменен, если слов окажется больше или меньше, и максимальная длина слова len, которая определяет размер буфера. В функции создания свободного массива буфер может быть описан как локальный массив размером len+1, или такой массив можно разместить в динамической памяти, но в этом случае перед выходом из функции создания свободного массива память должна быть освобождена. Если len+1-й символ не пустой, создание свободного массива прекращается, то есть освобождается выделенная ранее память, и функция возвращает NULL, в противном случае функция создания свободного массива возвращает указатель на созданный массив указателей.

В функции создания свободного массива слова считываются буфер buf функцией get\_word, описанной в задании к лабораторной работе. Затем, если необходимо, слово помещается в массив. Для этого выделяется место в динамической памяти в соответствии с размером слова, и слово копируется в выделенную область.

**Задание варианта №17:**

Даны два текста. Определить, получен ли второй текст перестановкой в обратном порядке слов первого текста

**Выполнение:**

1. **Описание алгоритма и выделение подзадач**

Исходя из условия задачи, буду помещать первый и второй тексты в динамический массив структур text1 и text2 соответственно. Далее сравню массивы слов text1 и text2 на заданное условие: буду сравнивать слова из text1 в обычном порядке со словами из text2 в обратном порядке на соответствие

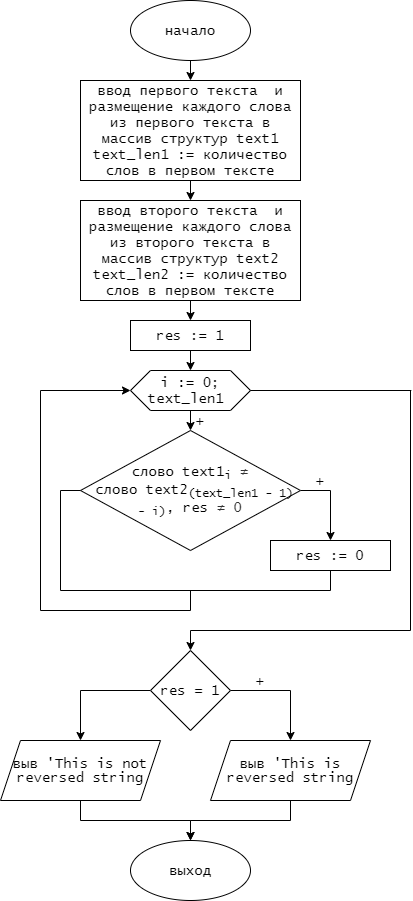
Выделение подзадач:

1. Создание массива слов из введённого с клавиатуры текста
2. Определение двух массивов слов на то, является ли второй обратной перестановкой первого
3. **Структуры данных**

Опишу структуру для хранения массива слов, где w – слово, text\_len – количество слов текста

struct text {  
 char w[255];  
 size\_t text\_len;  
};

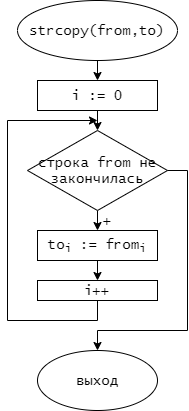
1. **Блок-схема алгоритма с укрупнёнными блоками**

****

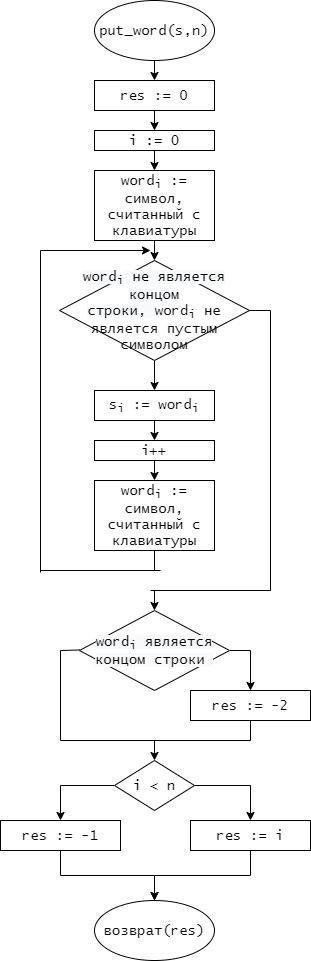
1. **Описание подпрограмм**
   1. Создание массива слов из введённого с клавиатуры текста
      1. Выделение подзадач:
         1. Копирование из одной строки в другую
         2. Запись слова в буфер
      2. Заголовок: struct text \* create\_free\_arr(int k, int buf\_len)
      3. Назначение: возвращает указатель на массив слов длины k, buf\_len – размер буфера для слова
      4. Блок-схема:

****

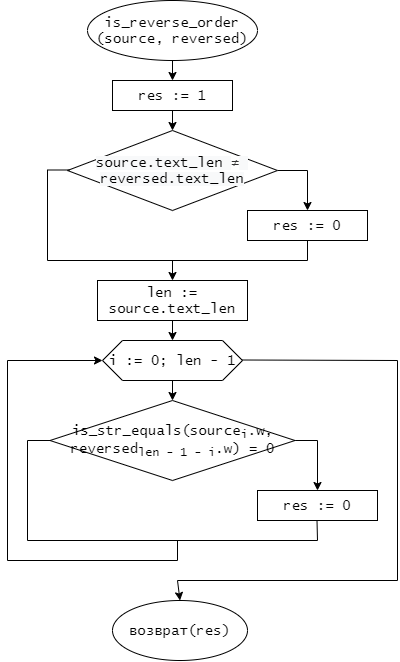
1. Копирование из одной строки в другую
   1. Заголовок: void strcopy(const char from[], char to[])
   2. Назначение: копирует строку from в строку to
   3. Блок-схема:

****

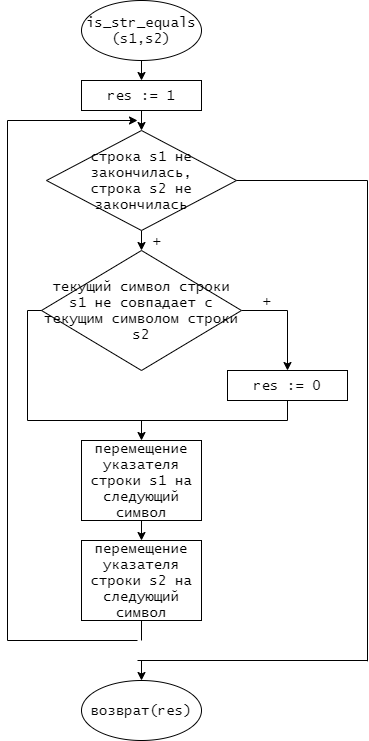
1. Запись слова в буфер
   1. Заголовок: int put\_word(char \*s, int n)
   2. Назначение: возвращает длину считанного с клавитуры слова строки s, если она меньше, чем n; возвращает ‘-2’, если строка s закончилась; возвращает ‘EOF’, если считанное слово оказалось больше, чем n
   3. Блок-схема:

****

* 1. Определение двух массивов слов на то, является ли второй обратной перестановкой первого
     1. Выделение подзадач:
        1. Определение равенства двух строк
     2. Заголовок: int is\_reverse\_order(struct text source[], struct text reversed[])
     3. Назначение: возвращает ‘1’, если массив слов reversed – обратная перестановка массива слов source, иначе – ‘0’
     4. Блок-схема



1. Сравнение двух строк
   1. Заголовок: int is\_str\_equals(char \*s1, char \*s2)
   2. Назначение: возвращает ‘1’, если строка s1 совпадает со строкой s2, иначе – ‘0’
   3. Блок-схема:



1. **Тестовые данные**

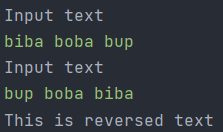
|  |  |  |
| --- | --- | --- |
| **№** | **Вход** | **Выход** |
| **1** | “biba boba bup”  “bup boba biba” | Текст является перевёрнутым |
| **2** | “ora ora ora ora”  “muda muda muda” | Текст не является перевёрнутым |

1. **Текст программы**

#include <stdio.h>  
#include <malloc.h>  
  
/\* структура, описывающая массив слов text \*/  
struct text {  
 char w[255];  
 size\_t text\_len;  
};  
  
/\* копирует строку from в строку to \*/  
void strcopy(const char from[], char to[]) {  
 size\_t i = 0;  
 while (from[i] != '\0') {  
 to[i] = from[i];  
 i++;  
 }  
  
 to[i] = 0;  
}  
  
/\* возвращает длину считанного c клавиатуры слова строки s, если она меньше, чем n;  
 \* возвращает '-2', если строка s закончилась;  
 \* возвращает 'EOF', если считанное слово оказалось больше, чем n \*/  
int put\_word(char \*s, int n) {  
 char word[n];  
  
 size\_t i = 0;  
 word[i] = getchar();  
 while (word[i] != '\n' && word[i] != ' ' && word[i] != '\0') {  
 s[i] = word[i];  
 i++;  
 word[i] = getchar();  
 }  
  
 if (word[i] == '\n') {  
 return -2;  
 }  
  
 if (i < n) {  
 word[i] = 0;  
 return i;  
 } else return EOF;  
}  
  
/\*возвращает '1', если строка s1 совпадает со строкой s2, иначе - '0'\*/  
int is\_str\_equals(char \*s1, char \*s2) {  
 while (\*s1 != '\0' && \*s2 != '\0') {  
 if (\*s1 != \*s2) {  
 return 0;  
 }  
  
 s1++; s2++;  
 }  
  
 return 1;  
}  
  
/\* возвращает указатель на массив слов длины k,  
 \* buf\_len - размер буфера для слова \*/  
struct text \*create\_free\_arr(int k, int buf\_len) {  
 struct text \*text = (struct text \*) calloc(sizeof(struct text), k);  
 size\_t i = 0;  
 char buf[buf\_len + 1];  
  
 printf("Input text\n");  
 int word\_status = 0;  
  
 while (word\_status != -2) {  
 word\_status = put\_word(buf, buf\_len);  
  
 if (k >= 0)  
 k--;  
 else  
 realloc(text, ++k);  
  
 strcopy(buf, text[i].w);  
 for (size\_t j = 0; j < buf\_len; ++j) {  
 buf[j] = 0;  
 }  
 i++;  
 }  
  
 if (i == k) {  
 text->text\_len = k;  
 } else text->text\_len = i;  
  
  
 return text;  
}  
  
/\* возвращает '1', если масси слов reversed - обратная перестановка  
 \* массива слова source, иначе - '0' \*/  
int is\_reverse\_order(struct text source[], struct text reversed[]) {  
 if (source->text\_len != reversed->text\_len) {  
 return 0;  
 }  
  
 size\_t len = source->text\_len;  
 for (size\_t i = 0; i < len; ++i) {  
 if (!(is\_str\_equals(source[i].w, reversed[(len - 1) - i].w))) {  
 return 0;  
 }  
 }  
  
 return 1;  
}  
  
int main() {  
 struct text \*text1 = create\_free\_arr(5, 255);  
 struct text \*text2 = create\_free\_arr(5, 255);  
  
 if (is\_reverse\_order(text1, text2)) {  
 printf("This is reversed text");  
 } else printf("This is not reversed text");  
}

1. **Результаты работы:**

*Пример №1:*

**

*Пример №2:*

