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:

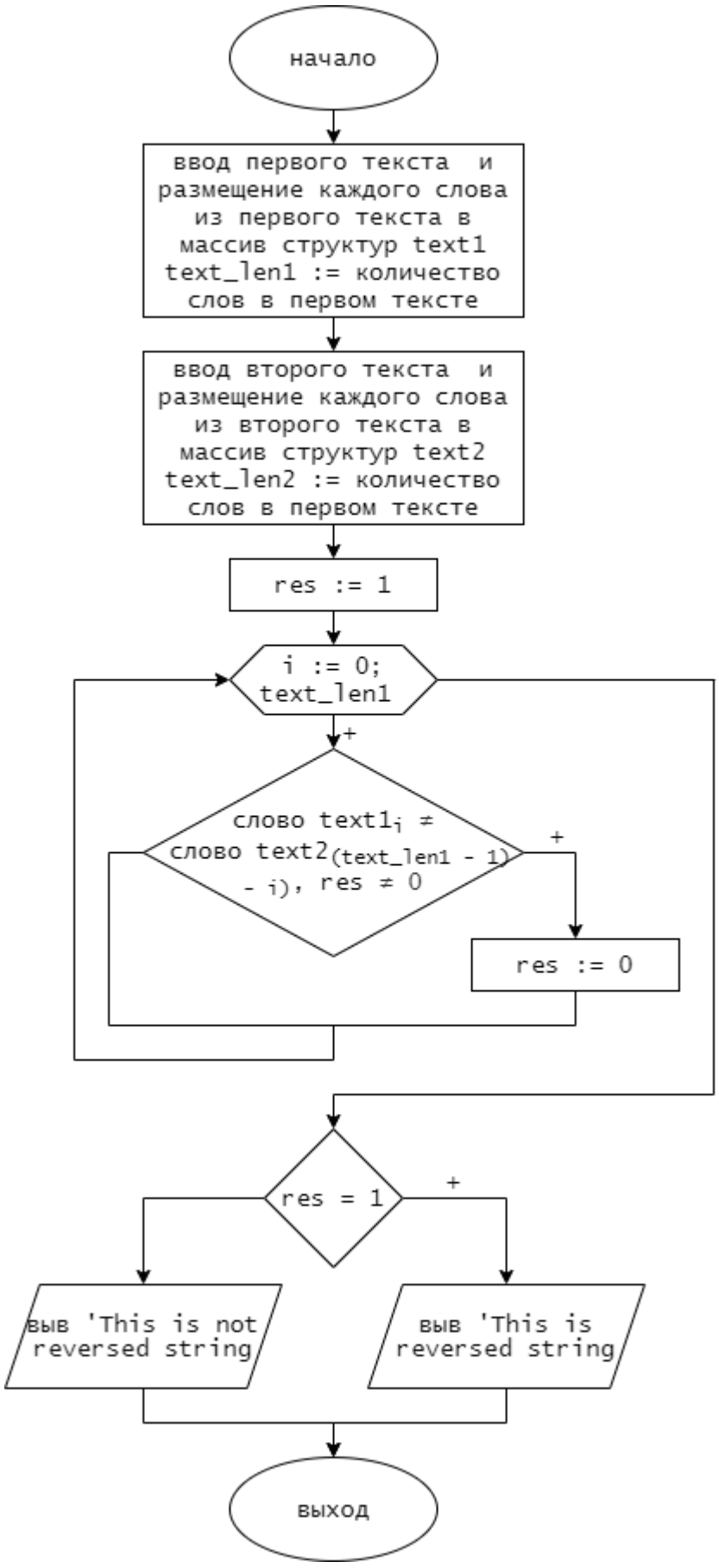
1. **Описание алгоритма и выделение подзадач**
   Исходя из условия задачи, буду помещать первый и второй тексты в динамический массив структур `text1` и `text2` соответственно. Далее сравню массивы слов `text1` и `text2` на заданное условие: буду сравнивать слова из `text1` в обычном порядке со словами 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

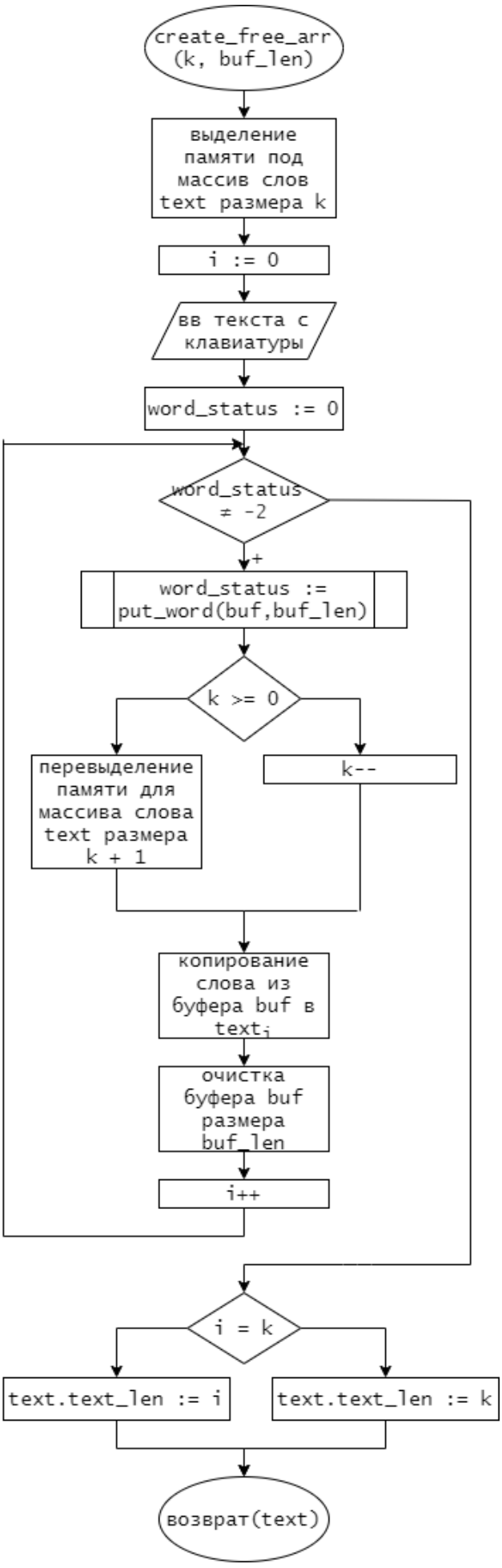
2. **Структуры данных**
   Опишу структуру для хранения массива слов, где `w` – слово, `text_len` – количество слов текста

```
struct text {
    char w[255];
    size_t text_len;
};
```

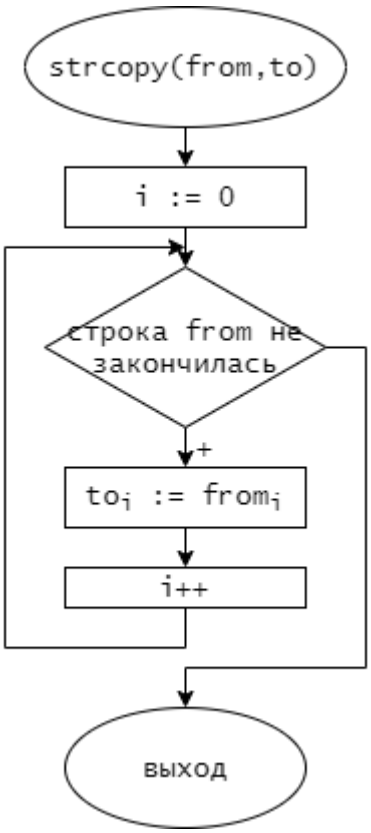
3. **Блок-схема алгоритма с укрупнёнными блоками**


начало
ввод первого текста и размещение каждого слова из первого текста в массив структур text1 text_len1 := количество слов в первом тексте
ввод второго текста и размещение каждого слова из второго текста в массив структур text2 text_len2 := количество слов в первом тексте
res := 1
i := 0; text_len1
+
слово text1i ≠ слово text2(text_len1 - 1) - i), res ≠ 0
+
res := 0
res = 1
+
выв 'This is not reversed string
выв 'This is reversed string
выход

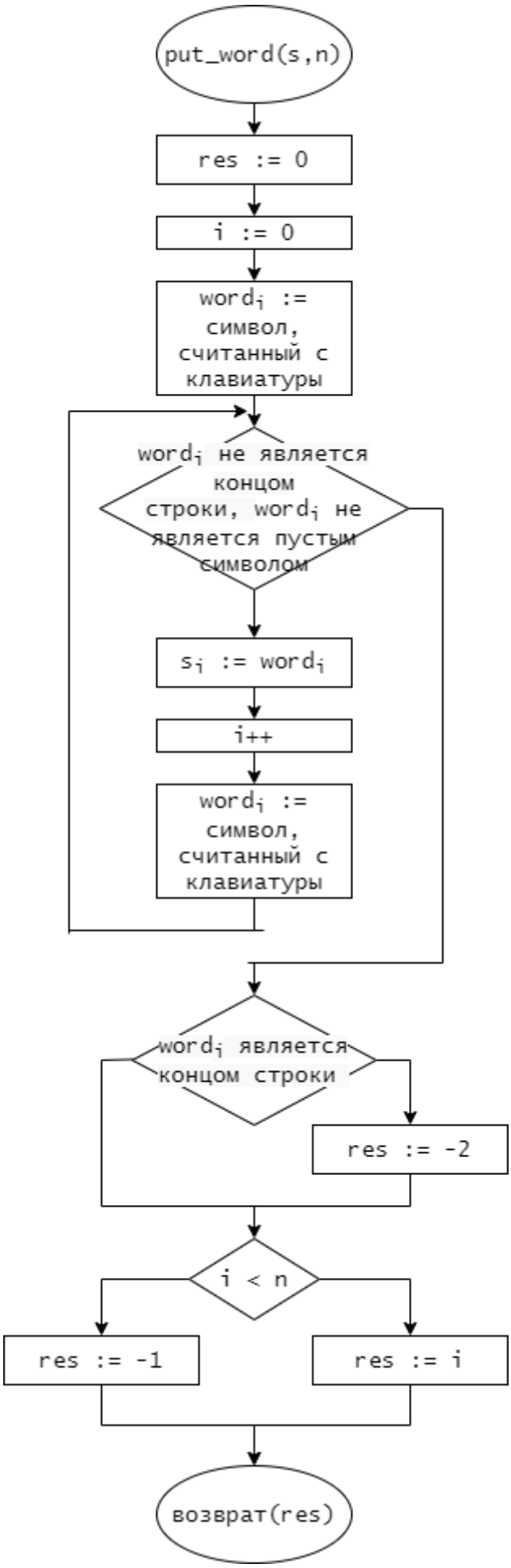
4. **Описание подпрограмм**
    a. Создание массива слов из введённого с клавиатуры текста
        i. Выделение подзадач:
            1. Копирование из одной строки в другую
            2. Запись слова в буфер
        ii. Заголовок: `struct text * create_free_arr(int k, int buf_len)`
        iii. Назначение: возвращает указатель на массив слов длины `k`, `buf_len` – размер буфера для слова
        iv. Блок-схема:


create_free_arr
(k, buf_len)
выделение
памяти под
массив слов
text размера k
i := 0
вв текста с
клавиатуры
word_status := 0
word_status
≠ -2
+
word_status :=
put_word(buf,buf_len)
k >= 0
перевыделение
памяти для
массива слова
text размера
k + 1
k--
копирование
слова из
буфера buf в
text[i]
очистка
буфера buf
размера
buf_len
i++
i = k
text.text_len := i
text.text_len := k
возврат(text)

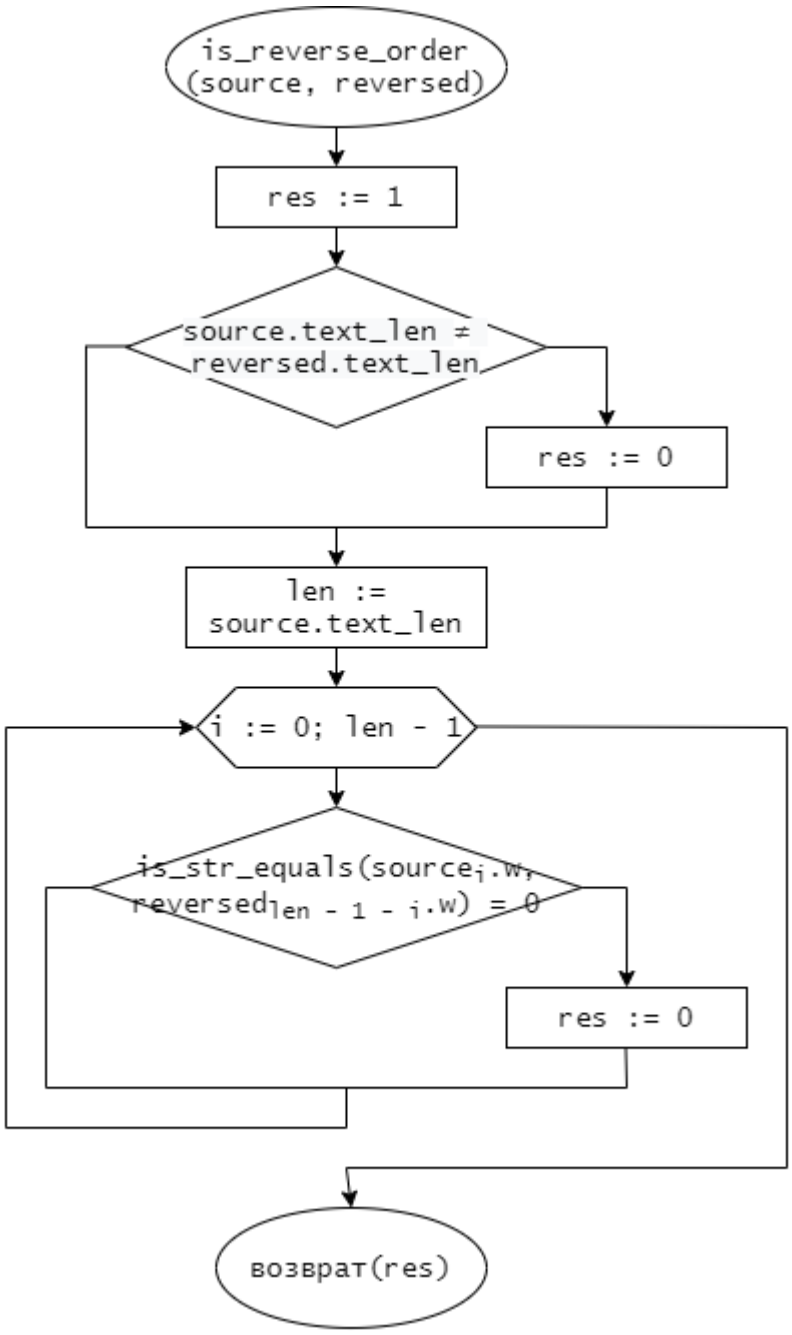
i. Копирование из одной строки в другую
   a. Заголовок: `void strcopy(const char from[], char to[])`
   b. Назначение: копирует строку `from` в строку `to`
   c. Блок-схема:



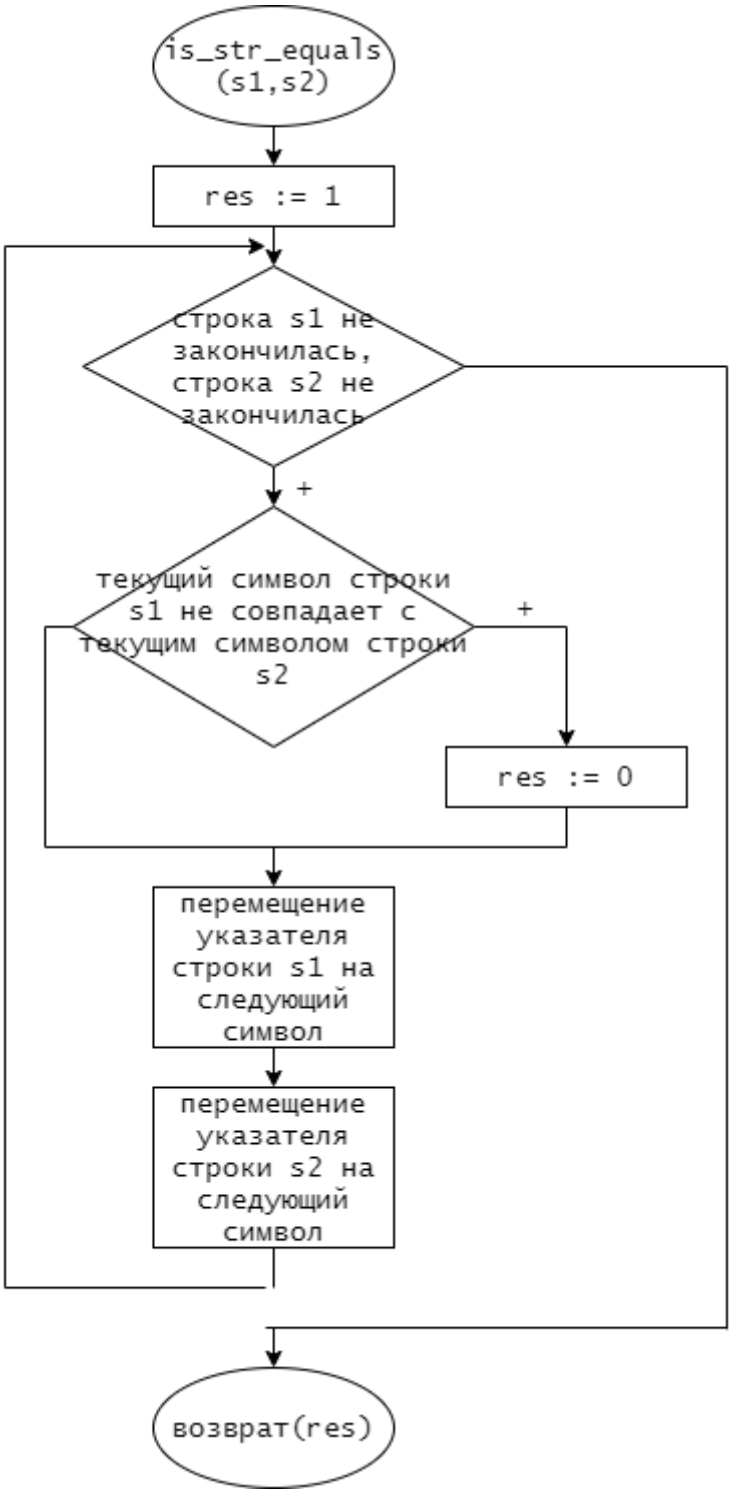
ii. Запись слова в буфер
   a. Заголовок: `int put_word(char *s, int n)`
   b. Назначение: возвращает длину считанного с клавитуры слова строки `s`, если она меньше, чем `n`; возвращает ‘-2’, если строка `s` закончилась; возвращает ‘`EOF`’, если считанное слово оказалось больше, чем `n`
   c. Блок-схема:


put_word(s,n)
res := 0
i := 0
word$_i$:= символ, считанный с клавиатуры
word$_i$ не является концом строки, word$_i$ не является пустым символом
s$_i$:= word$_i$
i++
word$_i$:= символ, считанный с клавиатуры
word$_i$ является концом строки
res := -2
i < n
res := -1
res := i
возврат(res)

b. Определение двух массивов слов на то, является ли второй обратной перестановкой первого
   i. Выделение подзадач:
      1. Определение равенства двух строк
   ii. Заголовок: `int is_reverse_order(struct text source[], struct text reversed[])`
   iii. Назначение: возвращает ‘1’, если массив слов `reversed` – обратная перестановка массива слов `source`, иначе – ‘0’
   iv. Блок-схема



i. Сравнение двух строк
   a. Заголовок: `int is_str_equals(char *s1, char *s2)`
   b. Назначение: возвращает ‘1’, если строка `s1` совпадает со строкой `s2`, иначе – ‘0’
   c. Блок-схема:



5. 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

6. 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 с 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");
}
```

7. Результаты работы:

Пример №1:

```
Input text
biba boba bup
Input text
bup boba biba
This is reversed text
```

Пример №2:

```
Input text
ora ora ora ora
Input text
muda muda muda
This is not reversed text
```