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 №18

По дисциплине: «Основы программирования»

Тема: ««Обработка строк в стиле С»»

Выполнил: студент группы ВТ-231

Борченко Александр Сергеевич

Проверили:

Черников Сергей Викторович

Новожен Никита Викторович

Белгород 2024

Цель работы: решение задач на строки в стиле С.

Содержание работы:

Заголовочный файл string_.h:

```
#ifndef LAB17STR_STRING__H
#define LAB17STR_STRING__H
#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>
#include <memory.h>
#include <string.h>
#include <stdbool.h>
#include <assert.h>
#include <ctype.h>

#define MAX_STRING_SIZE 100
#define MAX_N_WORDS_IN_STRING 100
#define MAX_WORD_SIZE 20
#define English_Alphabet 26

typedef struct WordDescriptor {
    char *begin; // позиция начала слова
    char *end; // позиция первого символа, после последнего символа слова
} WordDescriptor;

typedef struct BagOfWords {
    WordDescriptor words[MAX_N_WORDS_IN_STRING];
    size_t size;
} BagOfWords;

BagOfWords _bag;
BagOfWords _bag2;


typedef enum WordBeforeFirstWordWithAReturnCode {
    FIRST_WORD_WITH_A,
    NOT_FOUND_A_WORD_WITH_A,
    WORD_FOUND,
    EMPTY_STRING
} WordBeforeFirstWordWithAReturnCode;

size_t findStrLen(const char *start);

char* find(char *begin, char *end, int ch);

char* findNonSpace(char *begin);

char *findSpace(char *begin);

char *findNonSpaceReverse(char *rbegin, const char *rend);

char *findSpaceReverse(char *rbegin, const char *rend);

char *copy(const char *beginSource,
           const char *endSource,
           char*beginDestination);

char *copyIf(char *beginSource,
             const char *endSource,
             char*beginDestination,
             int (*f)(int));

char* copyIfReverse(char *rbeginSource,
                    const char *rendSource,
                    char *beginDestination,
```

```
                    int (*f)(int));

char *getEndOfString(char *begin);

void removeNonLetters(char *s);

void assertString(const char *expected, char *got,
                  char const *fileName, char const *funcName,
                  int line);

void removeExtraSpaces(char *s);

void removeAdjacentEqualLetters(char *s);

int getWord(char *beginSearch, WordDescriptor *word);

void digitToStart(WordDescriptor word);

void digitsToStart(char *s);

void replaceDigitsToNumOfSpaces(char *s);

void replace(char *source, char *w1, char *w2);

int areWordsEqual(WordDescriptor w1, WordDescriptor w2);

bool areWordsOrdered(char *s);

void getBagOfWords(BagOfWords *bag, char *s);

char *Copy_Reverse(char *rbegin_source, const char *rend_source, char
*beginDestination);

void reverseWordsBag(char *s);

int isWordInBagOfWords(WordDescriptor word, BagOfWords bag);

BagOfWords createBagOfWordsFromString(char *s);

void wordDescriptorToString(WordDescriptor word, char *destination);

int isWordPalindrome(char *begin, char *end);

size_t howManyWordsPalindromes(char *s);

void task_9(char *str1, char *str2, char *res);

WordBeforeFirstWordWithAReturnCode getWordBeforeFirstWordWithA(char *s,
WordDescriptor *w);

WordDescriptor task_12(char *s1, char *s2);

int DuplicateWords(char *s);

void Str_parse(char *str, BagOfWords *bag);

int compareWords(char *word1, char *word2);

int task_14(BagOfWords *bag);

char *task_15(char *str);

int ThisWordInBag(BagOfWords bag, WordDescriptor word);
```

```
int isPalindrome(char *s);

void removePalindromes(char *str);

void task_18(char *s1, char *s2);

int task19(const char *word, const char *str);

#endif //LAB17STR_STRING__H
```

Исполняемый файл string_.c:

```c
#include "string_.h"
#include "string_.c"

#define ASSERT_STRING(expected, got) assertString(expected, got, \
__FILE__, __FUNCTION__, __LINE__)

char _stringBuffer[MAX_STRING_SIZE + 1];

/*char *getEndOfString(char *begin) {
    char *end = begin;
    while (*end != '\0')
        end++;

    return end;
}*/
//получить размер слова
int getSizeWord(WordDescriptor word) {
    return word.end - word.begin;
}

/*void removeNonLetters(char *s) {
    char *endSource = getEndOfString(s);
    char *destination = copyIf(s, endSource, s, isgraph);
    *destination = '\0';
}*/

void assertString(const char *expected, char *got,
                  char const *fileName, char const *funcName,
                  int line) {
    if (strcmp(expected, got)) {
        fprintf(stderr, "File %s\n", fileName);
        fprintf(stderr, "%s - failed on line %d\n", funcName, line);
        fprintf(stderr, "Expected: \"%s\"\n", expected);
        fprintf(stderr, "Got: \"%s\"\n\n", got);
    } else
        fprintf(stderr, "%s - You're a cool bro.\n", funcName);
}
//Удаление лишних пробелов (оставляет один пробел)
void removeExtraSpaces(char *s) {
    int i, j;
    for (i = 0, j = 0; s[i]; i++) {
        if (s[i] != ' ' || (i > 0 && s[i - 1] != ' ')) {
            s[j++] = s[i];
        }
    }

    s[j] = '\0';
}
//удаление повторяющихся символов
void removeAdjacentEqualLetters(char *s) {
    if ((s == NULL || strlen(s) == 0))
        return;
    int i, j;
    for (i = 0, j = 0; s[i] != '\0'; i++) {
        if (s[i] != s[i + 1]) {
            s[j] = s[i];
            j++;
        }
    }

    s[j] = '\0';
```

```
}
//удаляет повторяющиеся символы

int getWord(char *beginSearch, WordDescriptor *word) {
    word->begin = findNonSpace(beginSearch);
    if (*word->begin == '\0')
        return 0;
    word->end = findSpace(word->begin);
    return 1;
}

void digitToStart(WordDescriptor word) {
    char *endStringBuffer = copy(word.begin, word.end,
                                 _stringBuffer);
    char *recPosition = copyIfReverse(endStringBuffer - 1,
                                      _stringBuffer - 1,
                                      word.begin, isdigit);
    copyIf(_stringBuffer, endStringBuffer, recPosition, isalpha);
}

void digitsToStart(char *s) {
    char *beginSearch = s;
    WordDescriptor word;
    while (getWord(beginSearch, &word)) {

        digitToStart(word);
        beginSearch = word.end;
    }
}

void replaceDigitsToNumOfSpaces(char *s) {
    copy(s, getEndOfString(s), _stringBuffer);
    char *recPtr = s;
    char *readPtr = _stringBuffer;
    for (int i = 0; i < strlen(_stringBuffer); ++i) {
        if (strlen(s) >= MAX_STRING_SIZE) {
            fprintf(stderr, "Out of MAX_STRING_SIZE");
            exit(1);
        }
        if (!isdigit(_stringBuffer[i])) {
            *recPtr = *readPtr;
            recPtr++;
            readPtr++;
        } else {
            int counter = _stringBuffer[i] - '0';
            for (int j = counter; j > 0; --j) {
                *recPtr = ' ';
                recPtr++;
            }
            readPtr++;
        }
    }
    _stringBuffer[0] = '\0';
    *recPtr = '\0';
}


void replace(char *source, char *w1, char *w2) {
    size_t size_w1 = strlen(w1);
    size_t size_w2 = strlen(w2);
    WordDescriptor word1 = {w1, w1 + size_w1};
    WordDescriptor word2 = {w2, w2 + size_w2};
    char *readPtr, *recPtr;
```

```
    if (size_w1 >= size_w2) {
        readPtr = source;
        recPtr = source;
    } else {
        copy(source, getEndOfString(source), _stringBuffer);
        readPtr = _stringBuffer;
        recPtr = source;
    }
    while (*readPtr != '\0') {
        if (memcmp(readPtr, w1, size_w1) == 0) {
            for (int i = 0; i < size_w2; i++) {
                *recPtr = w2[i];
                recPtr++;
            }

            readPtr += size_w1;
        } else {
            *recPtr = *readPtr;
            readPtr++;
            recPtr++;
        }
    }
    *recPtr = '\0';
}

//Два слова одинаковые
int areWordsEqual(WordDescriptor w1, WordDescriptor w2) {
    char *ptr1 = w1.begin;
    char *ptr2 = w2.begin;
    while (ptr1 <= w1.end && ptr2 <= w2.end) {
        if (*ptr1 != *ptr2)
            return 0;

        ptr1++;
        ptr2++;
    }

    if (ptr1 > w1.end && ptr2 > w2.end)
        return 1;
    else
        return 0;
}
//Определяет, упорядочены ли лексикографически слова данного предложения №6
(b)
bool areWordsOrdered(char *s) {
    WordDescriptor word1;
    WordDescriptor word2;
    if (getWord(s, &word1)) {
        word2 = word1;
        while (getWord(s, &word1)) {
            if (areWordsEqual(word1, word2) == 0)
                return false;
            word2 = word1;
            s = word1.end;
        }
        return true;

    } else
        return true;
}

void getBagOfWords(BagOfWords *bag, char *s) {
    WordDescriptor word;
```

```
    bag->size = 0;
    while (getWord(s, &word)) {
        bag->words[bag->size] = word;
        bag->size++;
        s = word.end;
    }
}
//вспомогательная функция, не такая как char* copyIfReverse:
char *Copy_Reverse(char *rbegin_source, const char *rend_source, char
*beginDestination) {
    while (rbegin_source != rend_source)
        (*beginDestination++) = *rbegin_source--;

    return beginDestination;
}


void reverseWordsBag(char *s) {
    *copy(s, getEndOfString(s), _stringBuffer) = '\0';
    getBagOfWords(&_bag, _stringBuffer);
    char *copy_str = s;
    for (int i = 0; i < _bag.size; i++) {
        copy_str = Copy_Reverse(_bag.words[i].end - 1, _bag.words[i].begin -
                                                    1, copy_str);
        *copy_str++ = ' ';
    }
    if (copy_str != s)
        copy_str--;

    *copy_str = '\0';
}
//для 12 задания
int isWordInBagOfWords(WordDescriptor word, BagOfWords bag) {
    for (size_t i = 0; i < bag.size; ++i) {
        if (strncmp(word.begin, bag.words[i].begin, word.end - word.begin) ==
0)
            return 1;
    }

    return 0;
}
//для 12 задания
BagOfWords createBagOfWordsFromString(char *s) {
    BagOfWords bag;
    bag.size = 0;

    char *wordBegin = s;
    for (; *s; s++) {
        if (isspace(*s)) {
            if (s > wordBegin) {
                bag.words[bag.size].begin = wordBegin;
                bag.words[bag.size].end = s;
                bag.size++;
            }

            wordBegin = s + 1;
        }
    }

    if (s > wordBegin) {
        bag.words[bag.size].begin = wordBegin;
        bag.words[bag.size].end = s;
        bag.size++;
```

```
    }

    return bag;
}
//для 12 задания
void wordDescriptorToString(WordDescriptor word, char *destination) {
    int length = word.end - word.begin;
    strncpy(destination, word.begin, length);
    destination[length] = '\0';
}
//Вспомогательная функция на проверку слова-палиндрома
int isWordPalindrome(char *begin, char *end) {
    while (begin < end) {
//Пропуск пробелов и знаков пунктуации
        if (!isalpha(*begin)) {
            begin++;
        } else if (!isalpha(*end)) {
            end--;
        } else {
//Сравниваю символы
            if (tolower(*begin) != tolower(*end)) {
                return 0; // Не палиндром
            }
            begin++;
            end--;
        }
    }
    return 1; // Палиндром
}

size_t howManyWordsPalindromes(char *s) {
    char *end_str = getEndOfString(s);
    char *beginSearch = findNonSpace(s);
    int countPalindromes = 0;
    char *position_first_comma = find(beginSearch, end_str, ',');
    int last_comma = *position_first_comma == '\0' && end_str - beginSearch
!= 0;

    while (*position_first_comma != '\0' || last_comma) {
        beginSearch = findNonSpace(beginSearch);
        countPalindromes += isWordPalindrome(beginSearch,
position_first_comma);
        beginSearch = position_first_comma + 1;
        if (last_comma)
            break;

        position_first_comma = find(beginSearch, end_str, ',');
        last_comma = *position_first_comma == '\0';
    }

    return countPalindromes;
}
void task_9(char *str1, char *str2, char *res) {
    char *word1 = strtok(str1, " ");
    char *word2 = strtok(str2, " ");

    while (word1 != NULL || word2 != NULL) {
        if (word1 != NULL) {
            strcat(res, word1);
            strcat(res, " ");
            word1 = strtok(NULL, " ");
        }
        if (word2 != NULL) {
```

```
            strcat(res, word2);
            strcat(res, " ");
            word2 = strtok(NULL, " ");
        }
    }
}
/*void task_11(char *s) {
    char *word = NULL;
    char *key = strtok(s, " ");

    while (key != NULL) {
        int foundA = 0;
        for (int i = 0; key[i] != '\0'; i++) {
            if (tolower(key[i]) == 'a') {
                foundA = 1;
                break;
            }
        }

        if (foundA) {
            if (word != NULL) {
                printf("%s\n", word);
                return;
            }
        } else {
            word = key;
        }

        key = strtok(NULL, " ");
    }
}*/
//как бы продолжение 11 номера
WordBeforeFirstWordWithAReturnCode getWordBeforeFirstWordWithA(char *s,
WordDescriptor *w) {
    if (s == NULL || strlen(s) == 0)
        return EMPTY_STRING;

    char *wordBegin = NULL;
    char *wordEnd = NULL;
    char *key = strtok(s, " ");

    while (key != NULL) {
        int found_a = 0;
        for (int i = 0; key[i] != '\0'; i++)
            if (tolower(key[i]) == 'a') {
                found_a = 1;
                break;
            }

        if (found_a) {
            if (wordBegin != NULL) {
                w->begin = wordBegin;
                w->end = wordEnd;
                return WORD_FOUND;
            } else
                return FIRST_WORD_WITH_A;

        } else {
            wordBegin =key;
            wordEnd = key + strlen(key);
        }

        key = strtok(NULL, " ");
```

```
    }

    return NOT_FOUND_A_WORD_WITH_A;
}
WordDescriptor task_12(char *s1, char *s2) {
    BagOfWords bag = createBagOfWordsFromString(s2);
    WordDescriptor lastWord = {NULL, NULL};

    BagOfWords wordsInS1 = createBagOfWordsFromString(s1);
    for (size_t i = 0; i < wordsInS1.size; ++i) {
        if (isWordInBagOfWords(wordsInS1.words[i], bag))
            lastWord = wordsInS1.words[i];
    }

    return lastWord;
}
int DuplicateWords(char *s) {
    char *words[100];
    int wordCount = 0;

    char *word = strtok(s, " ");
    while (word != NULL) {
        words[wordCount] = word;
        wordCount++;
        word = strtok(NULL, " ");
    }

    for (int i = 0; i < wordCount; i++)
        for (int j = i + 1; j < wordCount; j++)
            if (strcmp(words[i], words[j]) == 0)
                return 1;

    return 0;
}
//Дробление строки
void Str_parse(char *str, BagOfWords *bag) {
    char *key= strtok(str, " ");
    while (key != NULL) {
        bag->words[bag->size].begin = key;
        bag->words[bag->size].end = key + strlen(key);
        bag->size++;
        key = strtok(NULL, " ");
    }
}
//сравнение слов (английских)
int compareWords(char *word1, char *word2) {
    int count1[English_Alphabet] = {0};
    int count2[English_Alphabet] = {0};

    size_t len_word_1 = strlen(word1);
    size_t len_word_2 = strlen(word2);

    if (len_word_1 != len_word_2) {
        return 0;
    }

    for (int i = 0; i < len_word_1; i++) {
        count1[tolower(word1[i]) - 'a']++;
        count2[tolower(word2[i]) - 'a']++;
    }

    for (int i = 0; i < English_Alphabet; i++) {
        if (count1[i] != count2[i]) {
```

```
            return 0;
        }
    }

    return 1;
}

int task_14(BagOfWords *bag) {
    for (int i = 0; i < bag->size - 1; i++) {
        for (int j = i + 1; j < bag->size; j++) {
            if (compareWords(bag->words[i].begin, bag->words[j].begin)) {
                char result1[MAX_WORD_SIZE], result2[MAX_WORD_SIZE];
                wordDescriptorToString(bag->words[i], result1);
                wordDescriptorToString(bag->words[j], result2);
                return 1;
            }
        }
    }

    return 0;
}
char *task_15(char *str) {
    char *last_space = strrchr(str, ' ');
    if (last_space != NULL)
        *last_space = '\0';

    return str;
}
int ThisWordInBag(BagOfWords bag, WordDescriptor word) {
    for (int i = 0; i < bag.size; i++) {
        if (strncmp(word.begin, bag.words[i].begin, getSizeWord(word)) == 0)
{
            return 1;
        }
    }
    return 0;
}

WordDescriptor task_16(char *s1, char *s2) {
    getBagOfWords(&_bag, s1);
    getBagOfWords(&_bag2, s2);

    for (int i = 0; i < _bag.size; i++) {
        if (ThisWordInBag(_bag2, _bag.words[i])) {
            if (i != 0) {
                return  _bag.words[i - 1];
            } else {
                break;
            }
        }
    }

    WordDescriptor NULL_word = {NULL, NULL};

    return NULL_word;
}
int isPalindrome(char *s) {
    size_t length = strlen(s);
    for (int i = 0; i < length / 2; i++)
        if (tolower(s[i]) != tolower(s[length - i - 1]))
            return 0;

    return 1;
```

```
}

void removePalindromes(char *str) {
    char *token = strtok(str, " ");
    char result[1000] = "";

    while (token != NULL) {
        if (!isPalindrome(token)) {
            strcat(result, token);
            strcat(result, " ");
        }

        token = strtok(NULL, " ");
    }

    strcpy(str, result);
}
void task_18(char *s1, char *s2) {
    char *write_ptr;

    getBagOfWords(&_bag, s1);
    getBagOfWords(&_bag2, s2);

    int large_string_1;

    if(_bag.size > _bag2.size){
        large_string_1 = 1;
    } else {
        large_string_1 = 0;
    }

    int diff_size;
    if (large_string_1) {
        diff_size = _bag.size - _bag2.size;
    } else {
        diff_size = _bag2.size - _bag.size;
    }

    if (large_string_1) {
        write_ptr = _bag2.words[_bag2.size - 1].end;

        for (int i = 0; i < diff_size; i++) {
            *(write_ptr++) = ' ';
            write_ptr = copy(_bag.words[_bag2.size + i].begin,
_bag.words[_bag2.size + i].end, write_ptr);
        }

    } else {
        write_ptr = _bag.words[_bag.size - 1].end;
        for (int i = 0; i < diff_size; i++) {
            *(write_ptr++) = ' ';
            write_ptr = copy(_bag2.words[_bag.size + i].begin,
_bag2.words[_bag.size + i].end, write_ptr);
        }
    }

    *write_ptr = '\0';
}
int task19(const char *word, const char *str) {
    int letters[English_Alphabet] = {0};

    for (; *str; ++str) {
        if (*str >= 'a' && *str <= 'z') {
```

```
            letters[*str - 'a'] = 1;
        } else if (*str >= 'A' && *str <= 'Z') {
            letters[*str - 'A'] = 1;
        }
    }

    for (; *word; ++word) {
        if (*word >= 'a' && *word <= 'z') {
            if (!letters[*word - 'a']) {
                return 0;
            }
        } else if (*word >= 'A' && *word <= 'Z') {
            if (!letters[*word - 'A']) {
                return 0;
            }
        }
    }

    return 1;
}
```

Тесты:

```
void test_removeExtraSpaces() {
    char str[] = "What    time    is it    , bro      ?";
    char exp[] = "What time is it , bro ?";
    removeExtraSpaces(str);

    ASSERT_STRING(exp, str);

    char str1[] = "                          ";
    char exp1[] = "";
    removeExtraSpaces(str1);

    ASSERT_STRING(exp1, str1);

    char str2[] = "sentence without unnecessary spaces";
    char exp2[] = "sentence without unnecessary spaces";
    removeExtraSpaces(str2);

    ASSERT_STRING(exp2, str2);
}

void test_removeAdjacentEqualLetters() {
    char str[] = "88005553535";
    char exp[] = "8053535";
    removeAdjacentEqualLetters(str);

    ASSERT_STRING(exp, str);

    char str1[] = "GGGGGggggg";
    char exp1[] = "Gg";
    removeAdjacentEqualLetters(str1);

    ASSERT_STRING(exp1, str1);

    char str2[] = "";
    char exp2[] = "";
    removeAdjacentEqualLetters(str2);

    ASSERT_STRING(exp2, str2);

    char str3[] = "Walter";
    char exp3[] = "Walter";
    removeAdjacentEqualLetters(str3);

    ASSERT_STRING(exp3, str3);
}
void test_digitsToStart() {
    char str[] = "9AbdSH7H6j";
    char exp[] = "679AbdSHHj";
    digitsToStart(str);

    ASSERT_STRING(exp, str);

    char str1[] = "";
    char exp1[] = "";
    digitsToStart(str1);

    ASSERT_STRING(exp1, str1);

    char str2[] = "1b2im_1b2im3_b3am5_b34am";
    char exp2[] = "435332121bimbimbambam";
    digitsToStart(str2);
```

```
    ASSERT_STRING(exp2, str2);
}

void test_replaceDigitsToNumOfSpaces() {
    char str[MAX_STRING_SIZE] = "";
    char exp[] = "";
    replaceDigitsToNumOfSpaces(str);

    ASSERT_STRING(exp, str);

    char str1[MAX_STRING_SIZE] = "Bo4bR";
    char exp1[] = "Bo    bR";//4 пробела
    replaceDigitsToNumOfSpaces(str1);

    ASSERT_STRING(exp1, str1);

    char str2[MAX_STRING_SIZE] = "123";
    char exp2[] = "      ";//1+2+3=6 пробелов
    replaceDigitsToNumOfSpaces(str2);

    ASSERT_STRING(exp2, str2);

    char str3[MAX_STRING_SIZE] = "sentence without numbers";
    char exp3[] = "sentence without numbers";
    replaceDigitsToNumOfSpaces(str3);

    ASSERT_STRING(exp3, str3);
}

void test_replace() {
    char str[MAX_STRING_SIZE] = " ";
    char w1[] = " ";
    char w2[] = "background-image";
    replace(str, w1, w2);
    char exp[MAX_STRING_SIZE] = "background-image";

    ASSERT_STRING(exp, str);

    char str1[MAX_STRING_SIZE] = "I love water";
    char w1_1[] = "water";
    char w2_1[] = "beer";
    replace(str1, w1_1, w2_1);
    char exp1[MAX_STRING_SIZE] = "I love beer";

    ASSERT_STRING(exp1, str1);

    char str2[MAX_STRING_SIZE] = "";
    char w1_2[] = "";
    char w2_2[] = "";
    replace(str2, w1_2, w2_2);
    char exp2[MAX_STRING_SIZE] = "";

    ASSERT_STRING(exp2, str2);

    char str3[MAX_STRING_SIZE] = "SeNteNcE CAPs";
    char w1_3[] = "SeNteNcE CAPs";
    char w2_3[] = "Sentence caps";
    replace(str3, w1_3, w2_3);
    char exp3[MAX_STRING_SIZE] = "Sentence caps";

    ASSERT_STRING(exp3, str3);
}
```

```
void test_areWordsEqual() {
    WordDescriptor w1 = {"bimbim", "bambam"};
    WordDescriptor w2 = {"bimbim", "bambam"};

    if(areWordsEqual(w1, w2)) {
        printf("You're a cool bro\n");
    } else {
        printf("You failed bro\n");
    }

    WordDescriptor w1_1 = {"", ""};
    WordDescriptor w2_1 = {"", ""};

    if(areWordsEqual(w1_1, w2_1)) {
        printf("You're a cool bro\n");
    } else {
        printf("You failed bro\n");
    }

    WordDescriptor w1_2 = {"777", "666"};
    WordDescriptor w2_2 = {"777", "666"};

    if(areWordsEqual(w1_2, w2_2)) {
        printf("You're a cool bro\n");
    } else {
        printf("You failed bro\n");
    }
}

void test_areWordsOrdered() {
    char s[] = "abc";
    assert(areWordsOrdered(s) == true);

    char s2[] = "b a c";
    assert(areWordsOrdered(s2) == false);

    char s3[] = "a";
    assert(areWordsOrdered(s3) == true);

    char s4[] = "";
    assert(areWordsOrdered(s4) == true);

    char s5[] = "aa";
    assert(areWordsOrdered(s5) == true);

    char s6[] = "a!b?c.";
    assert(areWordsOrdered(s6) == true);
}

void test_getBagOfWords() {
//Тестирование, основанное на примере из пособия
    BagOfWords bag;
    bag.size = 0;
    char s[] = "a bc  d";

    getBagOfWords(&bag,s);

    for (int i = 0; i < bag.size; i++) {
        printf("letter %d - %.*s\n", i+1, (int)(bag.words[i].end -
bag.words[i].begin), bag.words[i].begin);
    }

    return;
```

```
}

void test_reverseWordsBag() {
    char s[MAX_STRING_SIZE] = "";
    reverseWordsBag(s);

    ASSERT_STRING("", s);

    char s1[MAX_STRING_SIZE] = "Sasha";
    reverseWordsBag(s1);

    ASSERT_STRING("ahsaS", s1);

    char s2[MAX_STRING_SIZE] = "ABCD";
    reverseWordsBag(s2);

    ASSERT_STRING("DCBA", s2);

    char s3[MAX_STRING_SIZE] = "123";
    reverseWordsBag(s3);

    ASSERT_STRING("321", s3);
}
void test_howManyWordsPalindromes() {
    char s[] = "";
    assert(howManyWordsPalindromes(s) == 0);

    char s1[] = "ANNA";
    assert(howManyWordsPalindromes(s1) == 1);

    char s2[] = "ANNA, level";
    assert(howManyWordsPalindromes(s2) == 2);

    char s3[] = "p";
    assert(howManyWordsPalindromes(s3) == 1);

    char s4[] = "ANNA, Sasha";
    assert(howManyWordsPalindromes(s4) == 1);

    char s5[] = "biMBIMbamBAAAAM";
    assert(howManyWordsPalindromes(s5) == 0);
}

void test_task_9() {
    char s1[] = "1 3 5 7";
    char s2[] = "2 4 6 8";
    char result[200] = "";

    task_9(s1, s2, result);

    ASSERT_STRING("1 2 4 6 8 ", result);

    char s1_1[] = "Hi, how are you";
    char s2_1[] = "smart peter ";
    char result1[200] = "";

    task_9(s1_1, s2_1, result1);

    ASSERT_STRING("Hi, smart peter ", result1);

    char s1_2[] = "";
    char s2_2[] = "";
    char result2[200] = "";
```

```
    task_9(s1_2, s2_2, result2);

    ASSERT_STRING("", result2);
}
void testAll_getWordBeforeFirstWordWithA() {
    WordDescriptor word;
    char s1[] = "";
    assert(getWordBeforeFirstWordWithA(s1, &word) == EMPTY_STRING);
    char s2[] = "ABC";
    assert(getWordBeforeFirstWordWithA(s2, &word) == FIRST_WORD_WITH_A);
    char s3[] = "BC A";
    assert(getWordBeforeFirstWordWithA(s3, &word) == WORD_FOUND);

    char s4[] = "B Q WE YR OW IUWR";
    assert(getWordBeforeFirstWordWithA(s4, &word) ==
NOT_FOUND_A_WORD_WITH_A);
}
void test_task_12() {
    char s1[] = "";
    char s2[] = "";
    WordDescriptor word1 = task_12(s1, s2);
    char str1[MAX_WORD_SIZE];

    wordDescriptorToString(word1, str1);

    ASSERT_STRING("", str1);

    char s1_1[] = "bimbim bam bam";
    char s2_1[] = "bum bam";
    WordDescriptor word2 = task_12(s1_1, s2_1);
    char str2[MAX_WORD_SIZE];

    wordDescriptorToString(word2, str2);

    ASSERT_STRING("bam", str2);

    char s1_2[] = "123 456 789";
    char s2_2[] = "098 456 586";
    WordDescriptor word3 = task_12(s1_2, s2_2);
    char str3[MAX_WORD_SIZE];

    wordDescriptorToString(word3, str3);

    ASSERT_STRING("456", str3);
}void test_hasDuplicateWords() {
    char str1[] = "";
    assert(DuplicateWords(str1) == false);

    char str2[] = "Hi friend";
    assert(DuplicateWords(str2) == false);

    char str3[] = "bim bim";
    assert(DuplicateWords(str3) == true);

    char str4[] = "123 321 123";
    assert(DuplicateWords(str4) == true);
}
void test_task_14() {
    BagOfWords bag1;
    bag1.size = 0;
    Str_parse("", &bag1);
    assert(task_14(&bag1) == 0);
```

```
    BagOfWords bag2;
    bag2.size = 0;
    Str_parse("123 321 123 123", &bag2);
    assert(task_14(&bag2) == 4);

    BagOfWords bag3;
    bag3.size = 0;
    Str_parse("hello world olleh dlrow", &bag3);
    assert(task_14(&bag3) == 1);

}
void test_task_15() {
    char str[] = "Hello world btw";
    char *exp = task_15(str);

    ASSERT_STRING(exp, "Hello world");

    char str1[] = "";
    char *exp1 = task_15(str1);

    ASSERT_STRING(exp1, "");

    char str2[] = "123 321 5";
    char *exp2 = task_15(str2);

    ASSERT_STRING(exp2, "123 321");
}

void test_task_16() {
    char s1[] = "aa boba NNN";
    char s2[] = "xx boba NNN";

    WordDescriptor res_word = task_16(s1, s2);
    char *yres[MAX_STRING_SIZE];
    wordDescriptorToString(res_word, yres);
    ASSERT_STRING("aa", yres);

    char s1_1[] = "";
    char s2_1[] = "";

    WordDescriptor res = task_16(s1_1, s2_1);
    char *Rres[MAX_STRING_SIZE];
    wordDescriptorToString(res, Rres);
    ASSERT_STRING("", Rres);
}

void test_remove_palindromes() {
    char str[] = "mamam hi mamam";
    removePalindromes(str);
    ASSERT_STRING("hi ", str);

    char str1[] = "";
    removePalindromes(str1);
    ASSERT_STRING("", str1);

    char str2[] = "1221 bambim 1221";
    removePalindromes(str2);
    ASSERT_STRING("bambim ", str2);
}
void test_task18() {
    char s1[MAX_STRING_SIZE] = "the sock";
    char s2[MAX_STRING_SIZE] = "Hello Patrick";
```

```
    task_18(s1, s2);

    ASSERT_STRING("Hello Patrick", s2);

    char s1_1[MAX_STRING_SIZE] = "bim bim";
    char s2_1[MAX_STRING_SIZE] = "bam bam";
    task_18(s1_1, s2_1);

    ASSERT_STRING("bam bam", s2_1);

}
void test_task19() {
    char word[] = "";
    char str[] = "";

    assert(task19(word, str) == 1);

    char word1[] = "JoJo";
    char str1[] = "JoJo for art Lovers";

    assert(task19(word1, str1) == 1);

    char word1_2[] = "Hey";
    char str1_2[] = "bimbimbambamovich";

    assert(task19(word1_2, str1_2) == 0);
}
void tests() {
    test_removeExtraSpaces();
    test_removeAdjacentEqualLetters();
    test_digitsToStart();
    test_replaceDigitsToNumOfSpaces();
    test_replace();
    test_areWordsEqual();
    test_areWordsOrdered();
    test_getBagOfWords();
    test_reverseWordsBag();
    test_howManyWordsPalindromes();
    test_task_9();
    testAll_getWordBeforeFirstWordWithA();
    test_task_12();
    test_hasDuplicateWords();
    test_task_14();
    test_task_15();
    test_task_16();
    test_remove_palindromes();
    test_task18();
    test_task19();
}

int main() {
    tests();
}
```

Результат тестирования:

```
test_removeExtraSpaces - You're a cool bro.
test_removeExtraSpaces - You're a cool bro.
test_removeExtraSpaces - You're a cool bro.
test_removeAdjacentEqualLetters - You're a cool bro.
test_removeAdjacentEqualLetters - You're a cool bro.
test_removeAdjacentEqualLetters - You're a cool bro.
test_removeAdjacentEqualLetters - You're a cool bro.
test_digitsToStart - You're a cool bro.
test_digitsToStart - You're a cool bro.
test_digitsToStart - You're a cool bro.
test_replaceDigitsToNumOfSpaces - You're a cool bro.
test_replace - You're a cool bro.
test_replace - You're a cool bro.
test_replace - You're a cool bro.
test_replace - You're a cool bro.
```

```
You're a cool bro
You're a cool bro
You're a cool bro
letter 1 - a
letter 2 - bc
letter 3 - d
test_reverseWordsBag - You're a cool bro.
test_reverseWordsBag - You're a cool bro.
test_reverseWordsBag - You're a cool bro.
test_reverseWordsBag - You're a cool bro.
test_task_9 - You're a cool bro.
test_task_9 - You're a cool bro.
test_task_9 - You're a cool bro.
test_task_9 - You're a cool bro.
test_task_12 - You're a cool bro.
test_task_12 - You're a cool bro.
test_task_12 - You're a cool bro.
```

```
C:\Users\Александр\CLionProjects\Lab17str\Lab18.exe

Process finished with exit code 0
```

Результат выполнения (комиты):

```
commit c9fc6053fd3ea7e60adfee08f26a131e8de87ea7 (HEAD -> master, origin/master)
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Fri Apr 19 11:09:33 2024 +0300

    1028строк ада и боли

 Lab18.c   | 145 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++---------
 Lab18.exe | Bin 109616 -> 111282 bytes
 string_.h |  57 +++++++++++++++++++++++++
 3 files changed, 182 insertions(+), 20 deletions(-)

commit 89dbb28eca022a1a31b8e1e54ac9cdeaebcb0c13
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Thu Apr 18 22:41:08 2024 +0300

    task17/test/сегодня стопаю

 Lab18.c   |  43 ++++++++++++++++++++++++++++++++++++++++++++-
 Lab18.exe | Bin 108786 -> 109616 bytes
 2 files changed, 42 insertions(+), 1 deletion(-)

commit b5b2e0052cd36c311e4a23e3077fd2e2887ae3d7
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Thu Apr 18 19:17:23 2024 +0300

    task16/test

 Lab18.c   |  52 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-
 Lab18.exe | Bin 107648 -> 108786 bytes
 2 files changed, 51 insertions(+), 1 deletion(-)

commit 62e6d13ad290683ea06552f9268698c4eed87401
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Thu Apr 18 18:30:17 2024 +0300

    task15/test/приятный номер

 Lab18.c   |  28 ++++++++++++++++++++++++++-
 Lab18.exe | Bin 106850 -> 107648 bytes
 2 files changed, 27 insertions(+), 1 deletion(-)

commit 654d8cfb1944390fed6f85e42117b9fce1497829
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Thu Apr 18 17:21:30 2024 +0300

    Помощь.. мне нужна помощь...

 Lab18.c   |  73 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++--
 Lab18.exe | Bin 105716 -> 106850 bytes
 string_.h |   1 +
 3 files changed, 72 insertions(+), 2 deletions(-)

commit b80b9c329a758e9ae58e08335a782137c1f68151
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Thu Apr 18 16:06:21 2024 +0300

    task13/test (самая приятная функция)

 Lab18.c   |  36 ++++++++++++++++++++++++++++++++++-
 Lab18.exe | Bin 104618 -> 105716 bytes
 2 files changed, 35 insertions(+), 1 deletion(-)

commit 64e2660a2bf77183eef4afe370ddf1b632a2cd97
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Thu Apr 18 14:46:45 2024 +0300

    task12/test+3доп функции

 Lab18.c   |  86 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-
 Lab18.exe | Bin 101165 -> 104618 bytes
 2 files changed, 85 insertions(+), 1 deletion(-)
:
```

```
 2 files changed, 85 insertions(+), 1 deletion(-)

commit 9956c00fb5d1b84dadee30deb2c6a4b1ddb597d9
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Wed Apr 17 17:38:48 2024 +0300

    Сделал 11 номер/тесты с пособия/10 не смог

 Lab18.c    |  76 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-
 Lab18.exe  | Bin 100535 -> 101165 bytes
 string_.h  |   8 +++++++
 3 files changed, 83 insertions(+), 1 deletion(-)

commit 39d5dfdfa0c1ea7fcfac4c25017380235783d629
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Wed Apr 17 11:47:35 2024 +0300

    task9/test

 Lab18.c    |  47 ++++++++++++++++++++++++++++++++++++++++++++++-
 Lab18.exe  | Bin 98568 -> 100535 bytes
 2 files changed, 46 insertions(+), 1 deletion(-)

commit 31cfa183621bb3287c40f6318cc364c449e49dc2
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Tue Apr 16 20:02:03 2024 +0300

    Пункт 8 сделал - сегодня стопну, тяжко

 Lab18.c    | 111 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-
 Lab18.exe  | Bin 95571 -> 98568 bytes
 2 files changed, 110 insertions(+), 1 deletion(-)

commit cd9c02958c7a92ed3a0dfb36adbde8ab51396db2
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Tue Apr 16 19:14:15 2024 +0300

    getagfords/test (без реверса)

 Lab18.c    |  27 ++++++++++++++++++++++++++-
 Lab18.exe  | Bin 95571 -> 95571 bytes
 2 files changed, 26 insertions(+), 1 deletion(-)

commit 9672dfbfe63c1ff86c69243dc890433d09eb2d22
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Tue Apr 16 18:58:37 2024 +0300

    Переделан позорный тест, простите

 Lab18.c    |  24 ++++++++++++++----------
 Lab18.exe  | Bin 95499 -> 95571 bytes
 string_.h  |   7 +++++++
 3 files changed, 21 insertions(+), 10 deletions(-)

commit 22fac1a926590e9da76216700b099ad6b2508acf
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Tue Apr 16 18:42:18 2024 +0300

    areordsrdered/test?

 Lab18.c    |  39 +++++++++++++++++++++++++++++++++++++--
 Lab18.exe  | Bin 94369 -> 95499 bytes
 2 files changed, 37 insertions(+), 2 deletions(-)

commit c52e51a8a0abd0ca2e08e90a3ef976a846aa8202
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Tue Apr 16 17:48:51 2024 +0300

    func areordsqual/test

 Lab18.c    |  49 ++++++++++++++++++++++++++++++++++++++++++++++++-
:
```

```
commit 845cadccc161b523270f4cf3a96789634fc4262e
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Tue Apr 16 11:45:02 2024 +0300

    func replace+test

 Lab18.c   |  68 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++-
 Lab18.exe | Bin 91995 -> 93788 bytes
 2 files changed, 67 insertions(+), 1 deletion(-)

commit c6846ab405d34e5d9f52726009c862e40a59477d
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Mon Apr 15 19:40:25 2024 +0300

    Новая функция+тест. На сегодня все...

 Lab18.c   |  57 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++-
 Lab18.exe | Bin 90331 -> 91995 bytes
 string_.h |   1 +
 3 files changed, 57 insertions(+), 1 deletion(-)

commit f527602a6dc33d02a2d831b622c4d8f74c8d636c
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Mon Apr 15 18:48:26 2024 +0300

    digitToStart/test/сложно

 Lab18.c   |  58 +++++++++++++++++++++++++++++++++++++++++++++++++++++++---
 Lab18.exe | Bin 88772 -> 90331 bytes
 string_.h |   5 +++++
 3 files changed, 60 insertions(+), 3 deletions(-)

commit 9bacf8c6adc6cc1c6d907c93e993104e1f397c70
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Mon Apr 15 18:09:33 2024 +0300

    removeAdjacentEqualLetters/test

 Lab18.c   |  45 +++++++++++++++++++++++++++++++++++++++++++--
 Lab18.exe | Bin 88132 -> 88772 bytes
 string_.h |  13 +++++++++++++
 3 files changed, 56 insertions(+), 2 deletions(-)

commit 5c4f67861f5656739ed9e49239093182d2a3f557
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Mon Apr 15 17:41:45 2024 +0300

    написал больше тестов с различными случиями

 Lab18.c   |  16 ++++++++++++++--
 Lab18.exe | Bin 88133 -> 88132 bytes
 2 files changed, 14 insertions(+), 2 deletions(-)

commit 53390dd2664f52137ef2c3cf1d533252514d83f6
Author: Александр <alexanders.borchenko@gmail.com>
Date:   Mon Apr 15 17:34:43 2024 +0300

    Начал 18 / сделал, что понял / тест

 CMakeLists.txt                                         |   3 +-
 Lab18.c                                                |  55 +++++++++++++++++++++
 Lab18.exe                                              | Bin 0 -> 88133 bytes
 ...json => codemodel-v2-d3a698ee0e395d7171aa.json}     |   2 +-
 ...45.json => index-2024-04-15T08-56-14-0036.json}     |   4 +-
 ...arget-Lab17str-Debug-d2256fb134de540ba191.json}     |  12 ++++-
 cmake-build-debug/CMakeFiles/clion-Debug-log.txt       |   2 +-
 cmake-build-debug/build.ninja                          |   8 ++-
:
```

Вывод: получил кисту в области мозжечка, думал над смыслом бытия и сожалел о поступлении, заинтересовался фронтендом и решил задачи на строки в стиле C.