МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и
информатики»
(СибГУТИ)

Кафедра телекоммуникационных систем и вычислительных средств
(ТС и ВС)

Расчетно-графическая работа
по дисциплине
«Программирование»

Студент:
Группа ИКС-433 *М.Г. Дроздова*

Предподаватель:
Инженер преподаватель *А.И. Вейлер*

Новосибирск 2025

СОДЕРЖАНИЕ

1 ВАРИАНТ 19. ПОИСК ПАЛИНДРОМОВ В ТЕКСТЕ

1.1 Задание

Разработать программу **palindrom**, выполняющую поиск всех палиндромов в заданном тексте. Программа должна:

- Принимать имя файла с текстом в качестве аргумента командной строки
- Выводить все найденные палиндромы
- Реализовать три уровня функциональности (на разные оценки)

1.2 Методы решения

Для решения задачи используются:

- Обработка строк (удаление пробелов и знаков препинания)
- Проверка на палиндром (сравнение символов с обоих концов)
- Алгоритм Манакера для поиска всех подпалиндромов

1.3 Алгоритм Манакера

Алгоритм заключается в том, что мы обрабатываем строку символ за символом, поддерживая самый правый палиндром в нашей строке. И на каждой итерации смотрим, наш текущий элемент находится внутри границ самого правого палиндрома или нет. Если находится, то мы можем извлечь ответ из ранее посчитанных значений, путём нехитрых манипуляций с индексами. Если же не находится, то мы идём точно таким же алгоритмом: идём символ за символом и сравниваем зеркальные элементы относительно центра. Идём до тех пор, пока они равны. И не забываем обновить после этого границы самого правого найденного палиндрома.

1.4 Псевдокоды

1.4.1 №1

```
int isPalindromeSimple(char *str) {
    int len = strlen(str);
    int left = 0, right = len - 1;
    while (left < right) {
        if (tolower(str[left]) != tolower(str[right])) {
            return 0;
        }
        left++;
        right--;
    }
    return 1;
}
```

1.4.2 №2

```
ФУНКЦИЯ НайтиПалиндромы(строка S):
    // 1. Преобразование строки
    T = "$#" + ВставитьСимволыМежду(S, "#") + "@"
    n = длина(T)

    // 2. Инициализация
    P = массив размером n, заполненный 0
    C = 0  // центр текущего палиндрома
    R = 0  // правая граница

    // 3. Основной алгоритм
    ДЛЯ i ОТ 1 ДО n-2:
        mirror = 2*C - i  // зеркальное отражение i относительно C

        ЕСЛИ i < R:
            P[i] = min(R - i, P[mirror])
```

```
        // Расширяем палиндром
        ПОКА T[i + P[i] + 1] == T[i - P[i] - 1]:
            P[i] += 1

        // Обновляем центр и границу
        ЕСЛИ i + P[i] > R:
            C = i
            R = i + P[i]

    // 4. Находим максимальный палиндром
    max_len = 0
    center = 0
    ДЛЯ i ОТ 1 ДО n-2:
        ЕСЛИ P[i] > max_len:
            max_len = P[i]
            center = i

    // 5. Восстанавливаем исходный палиндром
    start = (center - max_len) // 2
    конец = start + max_len - 1
    ВЕРНУТЬ S[start..конец]
КонецФункции

ВСПОМОГАТЕЛЬНАЯ ФУНКЦИЯ ВставитьСимволыМежду(строка S, символ sep):
    результат = ""
    ДЛЯ КАЖДОГО символа c В S:
        результат += sep + c
    ВЕРНУТЬ результат + sep
КонецФункции
```

1.5 Тестовые данные

1.5.1 На оценку 3

```
Строка: radar
Это палиндром!
Строка: level/
Это НЕ палиндром!
Строка: and
Это НЕ палиндром!
```

1.5.2 На оценку 4

```
Строка: radar
Это палиндром!
Строка: level/
Это палиндром!
Строка: and
Это НЕ палиндром!
```

1.5.3 На оценку 5

```
Строка: and
Это НЕ палиндром!
Строка: eye
Это палиндром (по алгоритму Манакера)!
```

1.6 Скриншоты с результатами

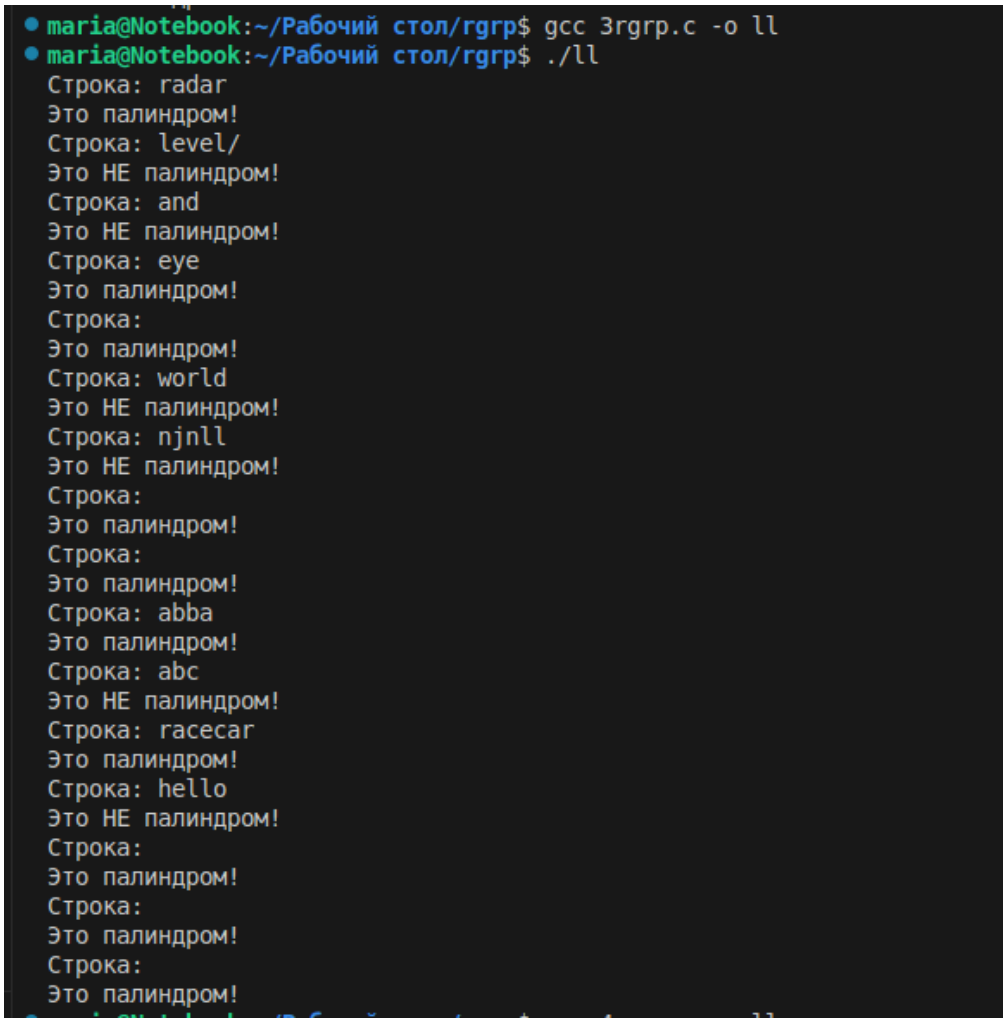

Рисунок 1 — На оценку 3, файл 3rgrp.c

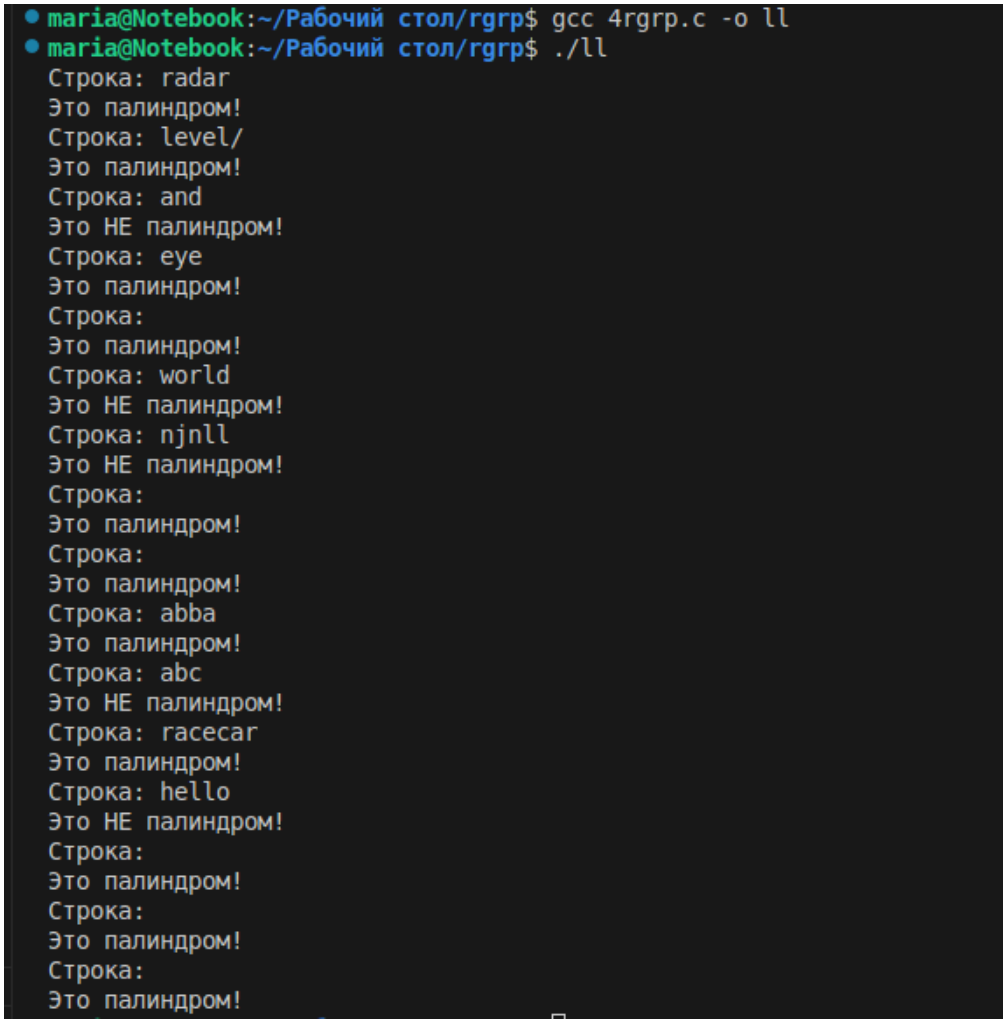

Рисунок 2 — На оценку 4, файл 4rgrp.c

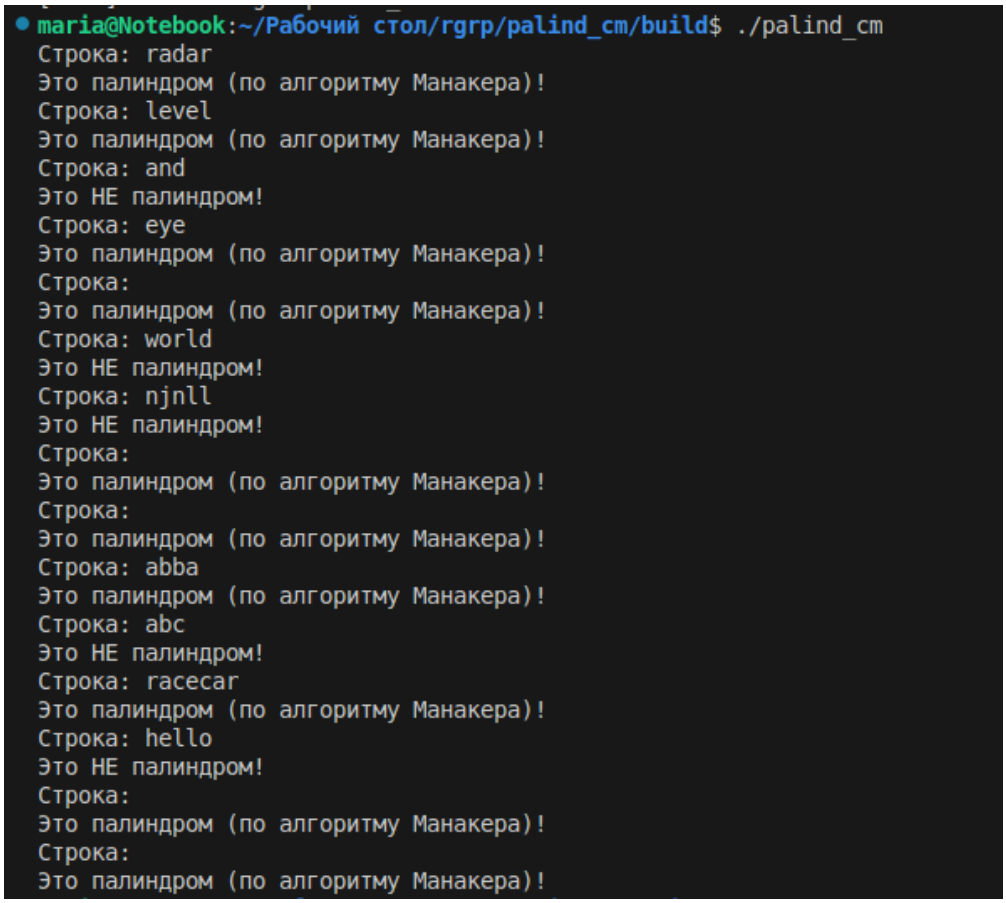

Рисунок 3 — На оценку 5, Cmake файл

Рисунок 4 — Unit-tests

1.7 Структура CMake файла

1. palind.h - заголовочный файл
2. palind.c - файл с функциями
3. main.c - файл с главной функцией
4. CMakeLists.txt - файл для сборки проекта
5. testpalind.c - файл с Unit тестами

1.8 Листинг программы

1.8.1 3rgrp.c

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int palind(char *str) {
    int len = strlen(str);
    int left = 0, right = len - 1;
    while (left < right) {
        if (tolower(str[left]) != tolower(str[right])) {
            return 0;}
        left++;
        right--;}
    return 1;}
int main() {
    FILE *file = fopen("rgrp.txt", "r");
    if (file == NULL) {
        printf("Ошибка открытия файла!\n");
        return 1;}
    char line[1000];
    while (fgets(line, sizeof(line), file)) {
        line[strcspn(line, "\n")] = '\0';
        printf("Строка: %s\n", line);
        if (palind(line)) {
            printf("Это палиндром!\n");
```

```
        } else {
            printf("Это НЕ палиндром!\n");}
    }
    fclose(file);
    return 0;
}
```

1.8.2 4rgrp.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int palind(char *str) {
    int len = strlen(str);
    char *cleaned = malloc(len + 1);
    int j = 0;
    for (int i = 0; i < len; i++) {
        if (isalnum(str[i])) {
            cleaned[j] = tolower(str[i]);
            j++;}
    }
    cleaned[j] = '\0';
    int left = 0, right = j - 1;
    while (left < right) {
        if (cleaned[left] != cleaned[right]) {
            free(cleaned);
            return 0;}
        left++;
        right--;}
    free(cleaned);
    return 1;}
int main() {
    FILE *file = fopen("rgrp.txt", "r");
    if (file == NULL) {
```

```
        printf("Ошибка открытия файла!\n");
        return 1;}
    char line[1000];
    while (fgets(line, sizeof(line), file)) {
        line[strcspn(line, " \n")] = '\0';
        printf("Строка: %s\n", line);
        if (palind(line)) {
            printf("Это палиндром!\n");
        } else {
            printf("Это НЕ палиндром!\n");}
    }
    fclose(file);
    return 0;}
```

1.8.3 5

1.8.4 palindcm/include/palind.h

```
#ifndef PALIND_H
#define PALIND_H
int palind(char *str);
#endif
```

1.8.5 palindcm/src/palind.c

```
#include "palind.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
int palind(char *str) {
    int len = strlen(str);
    int t_len = 2 * len + 3;
    char *T = malloc(t_len);
    T[0] = '$';
```

```
    T[t_len - 1] = '@';
    for (int i = 0; i < len; i++) {
        T[2 * i + 1] = '#';
        T[2 * i + 2] = tolower(str[i]);}
    T[2 * len + 1] = '#';
    T[t_len] = '\0';
    int *P = malloc(t_len * sizeof(int));
    for (int i = 0; i < t_len; i++) P[i] = 0;
    int C = 0, R = 0;
    for (int i = 1; i < t_len - 1; i++) {
        int mirror = 2 * C - i;
        if (i < R) P[i] = (R - i < P[mirror]) ? R - i : P[mirror];
        while (T[i + P[i] + 1] == T[i - P[i] - 1]) P[i]++;
        if (i + P[i] > R) {
            C = i;
            R = i + P[i];}
    }
    int palind = 0;
    for (int i = 1; i < t_len - 1; i++) {
        if (P[i] >= len) {
            palind = 1;
            break;}
    }
    free(T);
    free(P);
    return palind;}
```

1.8.6 palindcm/src/main.c

```
#include "palind.h"
#include <stdio.h>
#include <stdlib.h>
int main() {
    FILE *file = fopen("rgr1.txt", "r");
    if (file == NULL) {
```

```
        printf("Ошибка открытия файла!\n");
        return 1;}
    char line[1000];
    while (fgets(line, sizeof(line), file)) {
        line[strcspn(line, "\n")] = '\0';
        printf("Строка: %s\n", line);
        if (palind(line)) {
            printf("Это палиндром (по алгоритму Манакера)!\n");
        } else {
            printf("Это НЕ палиндром!\n");}
    }
    fclose(file);
    return 0;}
```

1.8.7 palindcm/CMakeLists.txt

```
cmake_minimum_required(VERSION 3.10)
project(palind_cm)
set(CMAKE_C_STANDARD 11)
include_directories(include)
add_executable(palind_cm src/palind.c src/main.c)
add_executable(test_palind src/palind.c test/test_palind.c)
configure_file(
    "${CMAKE_SOURCE_DIR}/rgr1.txt"
    "${CMAKE_BINARY_DIR}/rgr1.txt"
    COPYONLY
)
```

1.8.8 palindcm/test/testpalind.c

```
#include <stdio.h>
#include <assert.h>
#include "palind.h"

void test_palind() {
```

```
    printf("Запуск тестов...\n");
    // Тест 1: Палиндром
    char *str1 = "abba";
    int result1 = palind(str1);
    printf("Тест 1: '%s' - результат %d\n", str1, result1);
    assert(result1 == 1);
    // Тест 2: Не палиндром
    char *str2 = "hello";
    int result2 = palind(str2);
    printf("Тест 2: '%s' - результат %d\n", str2, result2);
    assert(result2 == 0);

    printf("Все тесты пройдены успешно!\n");
}
int main() {
    test_palind();
    return 0;
}
```

2 ЗАКЛЮЧЕНИЕ ПО ВЫПОЛНЕННОЙ РАБОТЕ

2.1 Выполненные требования:

2.1.1 Динамическое выделение памяти

- Реализовано для хранения преобразованной строки (T) и массива длин палиндромов (P).
- Использованы функции malloc и free для управления памятью.

2.1.2 Алгоритм Манакера

- Корректно реализован алгоритм поиска палиндромов за линейное время O(n)O(n).
- Обработка строки включает:
  1. Вставку разделителей между символами.
  2. Учет границ строки $ и @.
  3. Поиск максимального палиндрома через массив радиусов P.

2.1.3 Чтение входных данных

- Программа принимает имя файла через жестко заданный путь в коде.
- Чтение выполняется построчно с обработкой символов перевода строки.

2.1.4 Тестирование

- Написаны юнит-тесты для проверки корректности работы функции palind.
- Проверены граничные случаи: пустая строка, палиндромы с четным/нечетным числом символов, регистронезависимость.

- Когда вы увлеклись программированием?
- В старших классах.
- С таким опытом, у вас уже наверное очень хорошо получается?