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-графическое задание по дисциплине «Программирование»

Студент:
Группа ИКС-433
В. П. Попова

Преподаватель:
А.И. Вейлер

Новосибирск 2025

Содержание

1 Введение

Треугольные числа — это последовательность чисел, которые можно представить в виде равностороннего треугольника. n-ное треугольное число равно сумме натуральных чисел от 1 до n. Данная работа посвящена разработке программы на языке C, вычисляющей треугольные числа с использованием рекурсивного подхода.

1.1 Задача

Разработать программу `triangle`, вычисляющую n-ное треугольное число. Программа должна:

- Принимать в качестве аргумента командной строки значение n
- Реализовать рекурсивный алгоритм вычисления треугольного числа
- Выводить результат на экран и в файл в виде графического представления треугольника с использованием символа ‘*‘
- Обрабатывать возможные ошибки ввода

1.2 Критерии оценки

- Удовлетворительно: алгоритм без рекурсии, без динамического выделения памяти
- Хорошо: рекурсивный алгоритм с динамическим выделением памяти
- Отлично: рекурсивный алгоритм, вывод в файл графического представления

1.3 Анализ задачи

1.3.1 Математическая модель

Треугольное число T_n вычисляется по формуле:

$$T_n = \sum_{k=1}^{n} k = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$$

В программе используется рекурсивный подход для вычисления T_n.

1.3.2 Псевдокод

```
Function get_tri_num(n):
    If n <= 0:
        Return 0
    If n == 1:
        Return 1
    Else:
        Return n + get_tri_num(n - 1)

Function check_input(count, args):
    If count == 2:
        Return True
    Else:
```

```
        Return False

Function get_num(num_str):
    Return integer(num_str)

Function make_tri_file(n):
    Open file "triangle.txt" for writing
    If file opening fails:
        Print "Ошибка открытия файла triangle.txt"
        Return
    For i from 1 to n:
        For j from 0 to i-1:
            Write "* " to file
        Write newline to file
    Close file

Function show_help(name):
    Print "Используй: " + name + " <число>"

Main program:
    If not check_input(count, args):
        show_help(args[0])
        Exit with code 1
    num = get_num(args[1])
    If num <= 0:
        Print "Число должно быть больше 0"
        Exit with code 1
    tri_num = get_tri_num(num)
    Print "Треугольное число для n = " + num + ": " + tri_num
    make_tri_file(num)
    Print "Треугольник в triangle.txt"
```

1.3.3 Алгоритм

Основные этапы:

1. Получение входного параметра n
2. Проверка корректности входных данных
3. Рекурсивное вычисление треугольного числа
4. Вывод результата на экран
5. Создание графического представления в файл

2 Реализация

2.1 Структура программы

Программа состоит из следующих файлов:

- `main.c` — файл с главной функцией
- `triangle.c` — файл с функциями
- `triangle.h` — файл с заголовками функций
- `test.c` — файл для тестирования программы
- `CMakeLists.txt` — файл для сборки проекта

Программа реализует:

- Функцию для рекурсивного вычисления числа
- Функцию для проверки количества аргументов
- Функцию для преобразования аргумента в число
- Функцию для вывода треугольника в файл
- Функцию для вывода подсказки
- Главную функцию

2.2 Тестирование

№	Входные данные	Вывод программы
1	`./triangle`	Используй: ./triangle <число>
2	`./triangle abc`	Число должно быть больше 0
3	`./triangle -5`	Число должно быть больше 0
4	`./triangle 5`	Треугольное число для n = 5: 15
5	`./triangle 10`	Треугольное число для n = 10: 55

Таблица 1: Результаты тестирования

2.2.1 Результаты тестирования

- Все тесты пройдены успешно
- Обработка ошибок работает корректно
- Файл `triangle.txt` создаётся в нужном формате

2.3 Примеры работы

Пример для $n = 5$:

Вывод на экран:

Треугольное число для $n = 5 : 15$

Треугольник в triangle.txt

Содержимое файла `triangle.txt`:

```
*
* *
* * *
* * * *
* * * * *
```

2.4 Исходный код

2.4.1 main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "triangle.h"

int check_input(int count, char *args[]) {
    return count == 2;
}

int get_num(char *num_str) {
    int num = atoi(num_str);
    return (num > 0) ? num : -1;
}

void show_help(char *name) {
    printf(": %s <>\n", name);
}

int main(int count, char *args[]) {
    if (!check_input(count, args)) {
        show_help(args[0]);
        return 1;
    }

    int num = get_num(args[1]);
    if (num <= 0) {
        printf("    0\n");
        return 1;
    }

    int tri_num = get_tri_num(num);
    printf("   n = %d: %d\n", num, tri_num);
    make_tri_file(num);
```

```
    printf("  triangle.txt\n");
    return 0;
}
```

2.4.2 triangle.c

```
#include <stdio.h>
#include "triangle.h"

int get_tri_num(int n) {
    if (n <= 0) return 0;
    if (n == 1) return 1;
    return n + get_tri_num(n - 1);
}

void make_tri_file(int n) {
    FILE *file = fopen("triangle.txt", "w");
    if (file == NULL) {
        printf("   triangle.txt\n");
        return;
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j < i; j++) {
            fprintf(file, "* ");
        }
        fprintf(file, "\n");
    }
    fclose(file);
}
```

2.4.3 triangle.h

```
#ifndef TRIANGLE_H
#define TRIANGLE_H

int get_tri_num(int n);
void make_tri_file(int n);

#endif
\end{lstlisting>

\subsubsection{test.c}

\begin{lstlisting}
#include <stdarg.h>
#include <stddef.h>
#include <setjmp.h>
#include <cmocka.h>
#include <stdlib.h>
#include <stdio.h>
#include "triangle.h"

```

```c
static void test_get_tri_num(void **state) {
    (void)state;
    assert_int_equal(get_tri_num(1), 1);
    assert_int_equal(get_tri_num(5), 15);
    assert_int_equal(get_tri_num(10), 55);
}

static void test_make_tri_file(void **state) {
    (void)state;
    make_tri_file(3);
    FILE *file = fopen("triangle.txt", "r");
    assert_non_null(file);
    char line[100];
    assert_non_null(fgets(line, sizeof(line), file));
    assert_string_equal(line, "* \n");
    assert_non_null(fgets(line, sizeof(line), file));
    assert_string_equal(line, "* * \n");
    assert_non_null(fgets(line, sizeof(line), file));
    assert_string_equal(line, "* * * \n");
    fclose(file);
    remove("triangle.txt");
}

int main(void) {
    const struct CMUnitTest tests[] = {
        cmocka_unit_test(test_get_tri_num),
        cmocka_unit_test(test_make_tri_file),
    };
    return cmocka_run_group_tests(tests, NULL, NULL);
}
```

2.4.4 CMakeLists.txt

```
cmake_minimum_required(VERSION 3.10)
project(TriangleNumbers C)

set(CMAKE_C_STANDARD 99)
set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -Wall")

find_package(cmocka REQUIRED)

add_library(triangle_lib SHARED triangle.c)

add_executable(triangle main.c)
target_link_libraries(triangle triangle_lib)

add_executable(test_triangle test.c)
target_link_libraries(test_triangle triangle_lib cmocka)

enable_testing()
add_test(NAME TriangleTests COMMAND test_triangle)
```

3 Заключение

В ходе работы была разработана программа, соответствующая критериям оценки "отлично". Программа:

- Корректно вычисляет треугольные числа рекурсивным методом
- Обрабатывает ошибки ввода
- Создаёт графическое представление результата в файле `triangle.txt` с использованием символа ‘*‘