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 *С.А Демин*

Преподаватель:
Преподаватель *А.И. Вейлер*

Новосибирск 2025

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ 2

1 ВВЕДЕНИЕ 3
1.1 Задача 3
1.2 Критерии оценки 3
1.3 Анализ задачи 4
1.3.1 Математическая модель 4
1.3.2 Псевдокод 4
1.3.3 Алгоритм 5

2 РЕАЛИЗАЦИЯ 6
2.1 Структура программы 6
2.2 Тестирование 6
2.2.1 Результаты тестирования 6
2.3 Примеры работы 6
2.4 Исходный код 8
2.4.1 main.c 8
2.4.2 functions.c 8
2.4.3 triangle.h 9
2.4.4 test_triangle.c 9
2.4.5 CMakeLists.txt 10

3 ЗАКЛЮЧЕНИЕ 12

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
- Осуществлять динамическое выделение памяти под входные данные
- Выводить результат на экран и в файл в виде графического представления треугольника
- 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, динамическое выделение памяти, вывод в файл графического представления

1.3 Анализ задачи

1.3.1 Математическая модель

Треугольное число T_n вычисляется по формуле:

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

1.3.2 Псевдокод

```
Function triangularNumberRecursive(n):
    If n == 1:
        Return 1
    Else:
        Return n + triangularNumberRecursive(n - 1)

Function validateInput(argc, argv):
    If argc == 2:
        Return True
    Else:
        Return False

Function parseInput(n_str):
    Return integer(n_str)

Function printTriangleToFile(n):
    Open file "triangle.txt" for writing
    If file opening fails:
        Print "Error opening triangle.txt for writing."
        Exit program with code 1

    For i from 1 to n:
        For j from 0 to i-1:
            Write "  " to file
        Write newline to file
```

```
    Close file

Function printUsage(programName):
    Print "Usage: " + programName + " <n>"

Main program:
    If not validateInput(argc, argv):
        printUsage(argv[0])
        Exit with code 1

    n = parseInput(argv[1])
    If n <= 0:
        Print "n must be a positive integer."
        Exit with code 1

    triangularNumber = triangularNumberRecursive(n)
    Print "Triangular number for n = " + n + ": " + triangularNumber

    printTriangleToFile(n)
    Print "Triangle saved to 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
5. Создание графического представления в файле

2 РЕАЛИЗАЦИЯ

2.1 Структура программы

Программа состоит из 3 файлов:

- main.c - файл с главной функцией
- functions.c - файл с функциями
- triangle.h - файл с заголовками функций

Во всех этих файлах содержится реализации:

- Функции для нахождения числа рекурсивно
- Функции для проверки количества аргументов
- Функция для преобразования аргумента в число
- Функция для вывода треугольника в файл
- Функция для вывода подсказки
- Главная функция

2.2 Тестирование

№	Входные данные	Вывод программы
1	./main	Формат ввода: ./main <n>
2	./main abc	n должно быть положительным целым числом.
3	./main -5	n должно быть положительным целым числом.
4	./main 5	Треугольное число для n = 5: 15
5	./main 10	Треугольное число для n = 10: 55

2.2.1 Результаты тестирования

- Все тесты пройдены успешно
- Обработка ошибок работает корректно
- Файл создаётся в нужном формате

2.3 Примеры работы

```
segoga@desktop:/mnt/c/Users/sergx/OneDrive/Desktop/РГР прога$ ./main
Формат ввода: ./main <n>
segoga@desktop:/mnt/c/Users/sergx/OneDrive/Desktop/РГР прога$ ./main abc
n должно быть положительным целым числом.
segoga@desktop:/mnt/c/Users/sergx/OneDrive/Desktop/РГР прога$ ./main -5
n должно быть положительным целым числом.
segoga@desktop:/mnt/c/Users/sergx/OneDrive/Desktop/РГР прога$ ./main 5
Треугольное число для n = 5: 15
Треугольник записан в файл triangle.txt
segoga@desktop:/mnt/c/Users/sergx/OneDrive/Desktop/РГР прога$ ./main 10
Треугольное число для n = 10: 55
Треугольник записан в файл triangle.txt
```

Рисунок 1 — Запуск программы в разных режимах

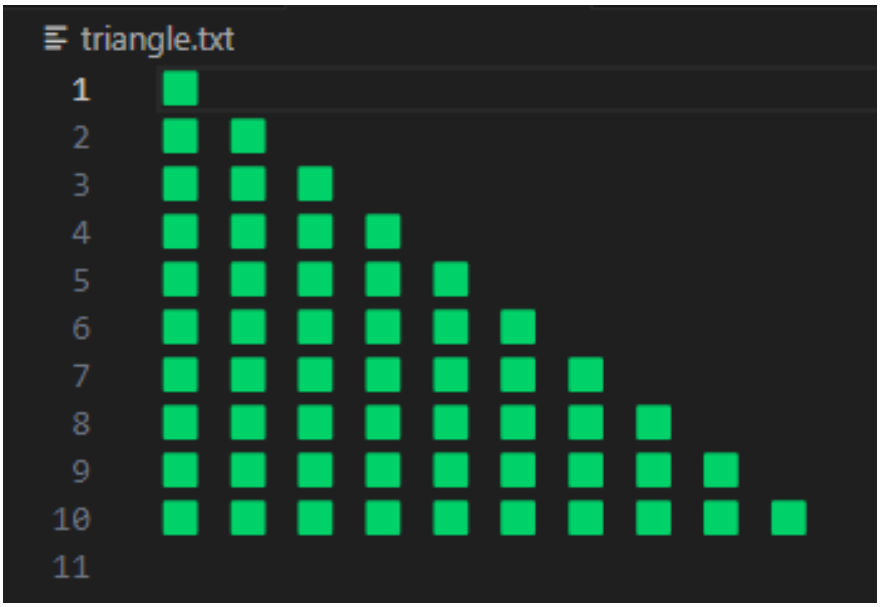

Рисунок 2 — Содержимое файла triangle.txt

```
segoga@desktop:/mnt/c/Users/sergx/OneDrive/Desktop/РГР прога/build$ ./test_triangle
[==========] tests: Running 3 test(s).
[ RUN      ] test_calculation
[       OK ] test_calculation
[ RUN      ] test_validation
[       OK ] test_validation
[ RUN      ] test_parsing
[       OK ] test_parsing
[==========] tests: 3 test(s) run.
[  PASSED  ] 3 test(s).
segoga@desktop:/mnt/c/Users/sergx/OneDrive/Desktop/РГР прога/build$ ctest
Test project /mnt/c/Users/sergx/OneDrive/Desktop/РГР прога/build
    Start 1: run_tests
1/1 Test #1: run_tests ........................   Passed    0.00 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =   0.02 sec
```

Рисунок 3 — Запуск 2 версии тестов

2.4 Исходный код

2.4.1 main.c

```
#include "triangle.h"

int main(int argc, char *argv[]) {
    if (!validateInput(argc, argv)) {
        printUsage(argv[0]);
        return 1;}

    int n = parseInput(argv[1]);
    if (n <= 0) {
        fprintf(stderr, "n                          .\n");
        return 1;}

    int triangularNumber = triangularNumberRecursive(n);
    printf("              n = %d: %d\n", n, triangularNumber);

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

2.4.2 functions.c

```
#include "triangle.h"

int triangularNumberRecursive(int n) {
    if (n == 1) {
        return 1;}
    return n + triangularNumberRecursive(n - 1);}

bool validateInput(int argc, char *argv[]) {
    return argc == 2;}

int parseInput(char *n_str) {
```

```
    return atoi(n_str);}

void printTriangleToFile(int n) {
    FILE *fp = fopen("triangle.txt", "w");
    if (fp == NULL) {
        fprintf(stderr, "              triangle.txt       .\n");
        exit(1);}

    for (int i = 1; i <= n; ++i) {
        for (int j = 0; j < i; ++j) {
            fprintf(fp, "  ");}
        fprintf(fp, "\n");}
    fclose(fp);}

void printUsage(char *programName) {
    fprintf(stderr, "       : %s <n>\n", programName);}
```

2.4.3 triangle.h

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdbool.h>

int triangularNumberRecursive(int n);
bool validateInput(int argc, char *argv[]);
int parseInput(char *n_str);
void printTriangleToFile(int n);
void printUsage(char *programName);
int triangularNumberRecursive(int n);
```

2.4.4 test_triangle.c

```
#include <stdarg.h>
#include <stddef.h>
```

```
#include <setjmp.h>
#include <cmocka.h>
#include "triangle.h"

static void test_calculation(void **state) {
    (void)state;
    assert_int_equal(triangularNumberRecursive(1), 1);
    assert_int_equal(triangularNumberRecursive(5), 15);
    assert_int_equal(triangularNumberRecursive(10), 55);}

static void test_validation(void **state) {
    (void)state;
    char *valid[] = {"program", "5"};
    assert_true(validateInput(2, valid));
    char *invalid[] = {"program"};
    assert_false(validateInput(1, invalid));}

static void test_parsing(void **state) {
    (void)state;
    assert_int_equal(parseInput("5"), 5);
    assert_int_equal(parseInput("0"), 0);
    assert_int_equal(parseInput("abc"), 0);}

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

2.4.5 CMakeLists.txt

```
cmake_minimum_required(VERSION 3.10)
project(triangular_project)
```

```
enable_testing()

add_library(triangular_lib SHARED functions.c)

add_executable(main main.c)
target_link_libraries(main triangular_lib)

find_package(cmocka REQUIRED)

add_executable(test_triangle test_triangle.c)
target_link_libraries(test_triangle triangular_lib cmocka)

add_test(NAME tests_of_code 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