@mainpage

# Лабораторна робота № 11

### 1.1 Розробник

@author Іовов Микита ( КН-923Б)

@date 20.05.2024 @version 1.8.17

### 1.2 Загальне завдання

Сформувати трикутник Паскаля ітеративним та рекурсивним методом.

## 2 Опис програми

### 2.1 Функціональне призначення

Вивід трикутника Паскаля ітеративним та рекурсивним методами.

### 2.2 Обмеження на застосування

Програма може бути обмежена обробкою певних типів даних або масивів конкретних розмірів.

### 2.3 Опис логічної структури

* main.c: Основний файл програми, де ініціалізуються функції printPascalIterative() і printPascalRecursive():

/\*\*\*  
  
 \* @file main.c  
  
 \* @brief Основна функція програми.  
  
 \*  
  
 \*/  
  
 #include "lib.h"  
  
  
  
/\*\*  
  
 \* @brief Функція для введення кількості строк та вивід трикутника Паскаля обома методами.  
  
 \* @return 0 у випадку успішного завершення програми.  
  
 \*  
  
 \*/  
  
  
  
int main()  
  
{  
 printf("Автор: Микита Іовов\nГрупа: КН-923Б\nТема: Взаємодія з користувачем шляхом механізму введення/виведення\n");  
 int rows = 0;  
 printf("Введіть кількість строк: ");  
 scanf("%d", &rows);  
 printf("Трикутник Паскаля ітеративним методом: \n");  
 printPascalIterative(rows);  
 printf("Трикутник Паскаля рекурсивним методом: \n");  
 printPascalRecursive(rows);  
 return 0;  
  
}

* lib.с: Файл з функціями для виводу трикутника Паскаля ітеративним та рекурсивними методами:

/\*\*  
  
 \* @file lib.c  
  
 \* @brief Функції для обчислення та виводу трикутника Паскаля.  
  
 \*  
  
 \* @details Цей файл містить функції для обчислення біноміального коефіцієнта,  
  
 \* а також для рекурсивного та ітеративного виводу трикутника Паскаля.  
  
 \*/  
  
  
  
#include "lib.h"  
  
  
  
  
  
  
  
/\*\*  
  
 \* @brief Обчислення біноміального коефіцієнта.  
  
 \*  
  
 \* @details Ця функція обчислює біноміальний коефіцієнт C(n, k) за допомогою рекурсивного методу.  
  
 \*  
  
 \* @param n Розмір множини.  
  
 \* @param k Розмір підмножини.  
  
 \* @return Значення біноміального коефіцієнта C(n, k).  
  
 \*/  
  
int binomialCoeff(int n, int k)  
  
{  
  
 if (k == 0 || k == n)  
  
 {  
  
 return 1;  
  
 }  
  
 return binomialCoeff(n - 1, k - 1) + binomialCoeff(n - 1, k);  
  
}  
  
  
  
/\*\*  
  
 \* @brief Рекурсивний вивід значень в рядку трикутника Паскаля.  
  
 \*  
  
 \* @param line Номер рядка.  
  
 \* @param i Позиція в рядку.  
  
 \*/  
  
void printPascalLine(int line, int i)  
  
{  
  
 if (i > line)  
  
 {  
  
 return;  
  
 }  
  
 printf("%d ", binomialCoeff(line, i));  
  
 printPascalLine(line, i + 1);  
  
}  
  
  
  
/\*\*  
  
 \* @brief Рекурсивний вивід пробілів перед рядком трикутника Паскаля.  
  
 \*  
  
 \* @param spaces Кількість пробілів для виводу.  
  
 \*/  
  
void printSpaces(int spaces)  
  
{  
  
 if (spaces <= 0)  
  
 {  
  
 return;  
  
 }  
  
 printf(" ");  
  
 printSpaces(spaces - 1);  
  
}  
  
  
  
/\*\*  
  
 \* @brief Допоміжна функція для рекурсивного виводу трикутника Паскаля.  
  
 \*  
  
 \* @param n Кількість рядків трикутника Паскаля для виводу.  
  
 \* @param line Поточний рядок для виводу.  
  
 \*/  
  
void printPascalHelper(int n, int line)  
  
{  
  
 if (line >= n)  
  
 {  
  
 return;  
  
 }  
  
 printSpaces(n - line - 1);  
  
 printPascalLine(line, 0);  
  
 printf("\n");  
  
 printPascalHelper(n, line + 1);  
  
}  
  
  
  
/\*\*  
  
 \* @brief Вивід трикутника Паскаля рекурсивним методом.  
  
 \*  
  
 \* @param n Кількість рядків трикутника Паскаля для виводу.  
  
 \*/  
  
void printPascalRecursive(int n)  
  
{  
  
 printPascalHelper(n, 0);  
  
}  
  
  
  
  
  
  
  
  
  
  
  
/\*\*  
  
 \* @brief Вивід трикутника Паскаля ітеративним методом.  
  
 \*  
  
 \* @details Ця функція виводить трикутник Паскаля до n рядків, використовуючи ітеративний метод  
  
 \* для обчислення біноміальних коефіцієнтів та динамічне виділення пам'яті.  
  
 \*  
  
 \* @param n Кількість рядків трикутника Паскаля для виводу.  
  
 \*/  
  
   
  
   
  
  
  
  
  
void printPascalIterative(int n) {  
  
 // Виділення пам'яті для двовимірного масиву  
  
 int \*\*arr = (int \*\*)malloc((size\_t)n \* sizeof(int \*));  
  
   
  
 // Заповнення масиву значеннями трикутника Паскаля  
  
 for (int i = 0; i < n; i++) {  
  
 \*(arr + i) = (int \*)malloc(((size\_t)i + 1) \* sizeof(int));  
  
 }  
  
  
  
 for (int line = 0; line < n; line++) {  
  
 for (int i = 0; i <= line; i++) {  
  
 // Перший або останній елемент ряду має значення 1  
  
 if (line == i || i == 0) {  
  
 \*(\*(arr + line) + i) = 1;  
  
 } else {  
  
 // Обчислення біноміального коефіцієнта для решти елементів  
  
 \*(\*(arr + line) + i) = \*(\*(arr + line - 1) + i - 1) + \*(\*(arr + line - 1) + i);  
  
 }  
  
 }  
  
 }  
  
  
  
 // Вивід значень трикутника Паскаля у вигляді рівнобедреного трикутника  
  
 for (int line = 0; line < n; line++) {  
  
 // Додавання пробілів для вирівнювання трикутника по центру  
  
 for (int space = 0; space < n - line - 1; space++) {  
  
 printf(" ");  
  
 }  
  
 for (int i = 0; i <= line; i++) {  
  
 printf("%d ", \*(\*(arr + line) + i));  
  
 }  
  
 printf("\n");  
  
 }  
  
  
  
 // Звільнення пам'яті  
  
 for (int i = 0; i < n; i++) {  
  
 free(\*(arr + i));  
  
 }  
  
 free(arr);  
  
}

* lib.h: Заголовочний файл, що містить прототипи функцій для формування трикутника Паскаля ітеративним та рекурсивним методами:

/\*\*  
  
\*@file lib.h  
  
\*@brief Заголовочний файл, який містить прототипи функції для формування трикутника Паскаля  
  
\*/  
  
#ifndef LIB\_H  
  
#define LIB\_H  
  
  
  
#include <stdio.h>  
  
#include <stdlib.h>  
  
  
  
/\*   
  
\*   
  
\* @brief Прототип функції для обчислення біноміального коефіцієнта.  
  
\*   
  
\* @param n Кількість елементів.  
  
\* @param k Кількість обраних елементів.  
  
\* @return Значення біноміального коефіцієнта.  
  
\*/  
  
int binomialCoeff(int n, int k);  
  
  
  
/\*   
  
\* @brief Прототип функції для виведення трикутника Паскаля рекурсивним методом.  
  
\*   
  
\* @param n Кількість строк трикутника Паскаля.  
  
\*/  
  
void printPascalRecursive(int n);  
  
  
  
/\*   
  
\* @brief Прототип функції для виведення трикутника Паскаля ітеративним методом.  
  
\*   
  
\* @param n Кількість строк трикутника Паскаля.  
  
\*/  
  
void printPascalIterative(int n);  
  
  
  
#endif

* test.c: Містить тестовий запуск функцій для завідомо відомих даних:

#include <check.h>  
  
#include "lib.h"  
  
  
  
START\_TEST(test\_binomialCoeff) {  
  
 ck\_assert\_int\_eq(binomialCoeff(0, 0), 1);  
  
 ck\_assert\_int\_eq(binomialCoeff(1, 0), 1);  
  
 ck\_assert\_int\_eq(binomialCoeff(1, 1), 1);  
  
 ck\_assert\_int\_eq(binomialCoeff(2, 1), 2);  
  
 ck\_assert\_int\_eq(binomialCoeff(3, 1), 3);  
  
 ck\_assert\_int\_eq(binomialCoeff(4, 2), 6);  
  
 ck\_assert\_int\_eq(binomialCoeff(5, 2), 10);  
  
}  
  
  
  
START\_TEST(test\_printPascalRecursive) {  
  
 // Перенаправити стандартний вивід до тимчасового файлу  
  
 freopen("temp.txt", "w", stdout);  
  
  
  
   
  
 printPascalRecursive(5);  
  
  
  
   
  
 fclose(stdout);  
  
  
  
 // Відкриття тимчасового файлу та перевірка його вмісту  
  
 FILE \*fp = fopen("temp.txt", "r");  
  
 char buffer[1024];  
  
 int lines = 0;  
  
 while (fgets(buffer, sizeof(buffer), fp) != NULL) {  
  
 lines++;  
  
   
  
 }  
  
 fclose(fp);  
  
  
  
 // Перевірка кількості рядків, які виведено  
  
 ck\_assert\_int\_eq(lines, 5);  
  
}  
  
  
  
START\_TEST(test\_printPascalIterative) {  
  
 // Перенаправити stdout до тимчасового файлу  
  
 freopen("temp.txt", "w", stdout);  
  
  
  
 // Виклик функції  
  
 printPascalIterative(5);  
  
  
  
 // Відновлення stdout  
  
 fclose(stdout);  
  
  
  
 // Відкриття тимчасового файлу та перевірка його вмісту  
  
 FILE \*fp = fopen("temp.txt", "r");  
  
 char buffer[1024];  
  
 int lines = 0;  
  
 while (fgets(buffer, sizeof(buffer), fp) != NULL) {  
  
 lines++;  
  
  
  
 }  
  
 fclose(fp);  
  
  
  
 // Перевірка кількості рядків, які виведено  
  
 ck\_assert\_int\_eq(lines, 5);  
  
}  
  
  
  
Suite\* lib\_suite(void) {  
  
 Suite \*s;  
  
 TCase \*tc\_core;  
  
  
  
 s = suite\_create("Lib");  
  
 tc\_core = tcase\_create("Core");  
  
  
  
 tcase\_add\_test(tc\_core, test\_binomialCoeff);  
  
 tcase\_add\_test(tc\_core, test\_printPascalRecursive);  
  
 tcase\_add\_test(tc\_core, test\_printPascalIterative);  
  
  
  
 suite\_add\_tcase(s, tc\_core);  
  
  
  
 return s;  
  
}  
  
  
  
int main(void) {  
  
 int number\_failed;  
  
 Suite \*s;  
  
 SRunner \*sr;  
  
  
  
 s = lib\_suite();  
  
 sr = srunner\_create(s);  
  
  
  
 srunner\_run\_all(sr, CK\_NORMAL);  
  
 number\_failed = srunner\_ntests\_failed(sr);  
  
 srunner\_free(sr);  
  
  
  
 return (number\_failed == 0) ? 0 : 1;  
  
}

### 2.2.3 Структура проекту

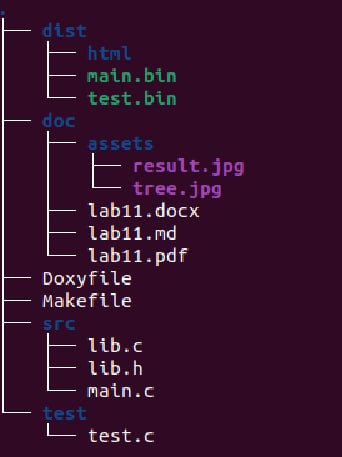


Рисунок 1 - Структура проекту

### 2.4 Важливі фрагменти програми

* Включення заголовочних файлів.
* Створення тестового набору.
* Запуск тестів та вивід результатів.

# 2 Варіанти використання

Для представлення виконання кожного завдання використовується: - послідовне виконання програми в інструменті lldb; - виведення результатів у консоль за допомогою функції виводу; - юніт-тест;

### Варіант використання 1

* Формування трикутника Паскаля.

Інструкція для запуску програми за допомогою юніт-тесту: - Виклик юніт тесту за допомогою команди ./test.bin.

Інструкція для запуску програми: - Виклик програми за допомого команди ./main.bin. - Ввести потрібну кількість рядків для трикутника Паскаля.

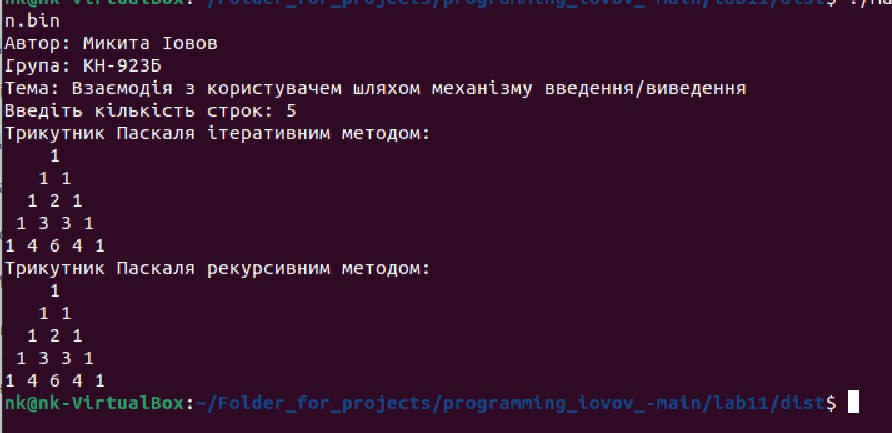


Рисунок 2 - Скріншот виводу програми

### Результат:

Як бачимо, програма коректно виводить трикутник Паскаля у консоль двома методами: ітеративним і рекурсивним.

# Висновок

Наданий код виконує функцію створення трикутника Паскаля ітеративним та рекурсивним методами. Ця програма реалізована мовою програмування C і використовує вбудовану функцію printf() для форматованого виводу.