МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТЕФАНИКА

Кафедра комп’ютерних наук та інформаційних систем

**Лабораторна робота № 1**

з курсу Предмет

на тему:

«Тема»

**Виконав(-ла):**

студент(-ка) групи ШИФР-1

Рудка.А.В.

«\_\_\_» 25.09\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

**Перевірив:**

ст. викладач кафедри КНІС

к.т.н. Ізмайлов А. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2023 р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оцінка, підпис)

м. Івано-Франківськ

2023

**Лабораторна робота №1**

**Рекурсивні функції Ч.1**

**Мета роботи:** Навчитись розв’язувати прикладні задачі із застосуванням рекурсивних процедур.

**1.1 Теоретичні відомості**

Рекурсією називається конструкція, де підпрограма (функція) викликає сама себе. Багато математичних функцій (послідовностей) можна подати у вигляді рекурсивних, тобто таких, що виражаються через попереднє (одно чи два чи й більше) значення.  
 **1.2 Хід роботи**

1.2.1 Створити функцію Task1, яка приймає у якості аргумента кількість елементів (чисел) послідовності Фібоначчі, які потрібно вивести (не більше 40). Вивід слід здійснювати на рядковий потік (stringstream), який повернути з функції у вигляді рядка. Знаходження чисел Фібоначчі реалізувати за допомогою рекурсивної функції FibonacciTask1, яка реалізує обчислення на основі безпосереднього визначення послідовності Фібоначчі (спосіб а) у теоретичних відомостях).

#include <iostream>

#include <sstream>

int FibonacciTask1(int n)

{

if (n == 0)

return 0;

if (n == 1)

return 1;

if(n >= 2)

return FibonacciTask1(n - 2) + FibonacciTask1(n - 1);

}

std::string Task1(int n)

{

std::stringstream ss;

for (size\_t i = 0; i < n; i++)

{

ss << FibonacciTask1(i);

ss << '\n';

}

return ss.str();

}

int main()

{

int n;

std::cin >> n;

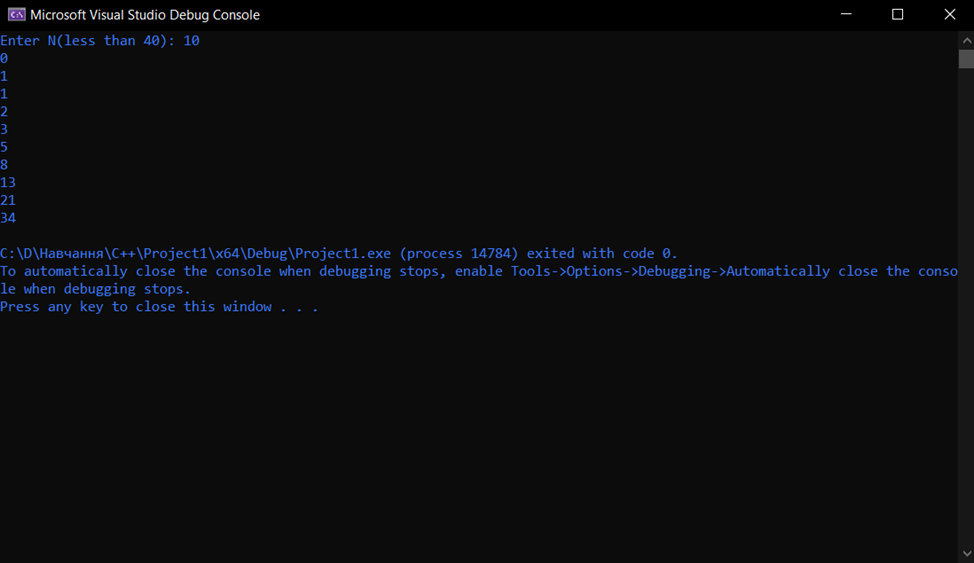
if (n > 40)

return 1;

std::cout << Task1(n);

return 0;

}



1.2.2 Створити функцію Task2, яка приймає у якості аргумента кількість елементів (чисел) послідовності Фібоначчі, які потрібно вивести (не більше 40). Вивід слід здійснювати на рядковий потік (stringstream), який повернути з функції у вигляді рядка. Знаходження чисел Фібоначчі реалізувати за допомогою рекурсивної функції FibonacciTask2, яка реалізує обчислення на основі удосконаленої форми обчислення послідовності Фібоначчі (спосіб б) у теоретичних відомостях)

#include <iostream>

#include <sstream>

int FibonacciTask2(int k, int n, int prev1, int prev2)

{

if (k == n)

return prev2;

if( k < n)

return FibonacciTask2(k + 1, n, prev2, prev1 + prev2);

}

std::string Task2(int k, int n, int prev1, int prev2)

{

std::stringstream ss;

for (size\_t i = 0; i < n; i++)

{

ss << FibonacciTask2(k, i, prev1, prev2);

ss << '\n';

}

return ss.str();

}

int main()

{

int n;

std::cout << "Enter N(less than 40): ";

std::cin >> n;

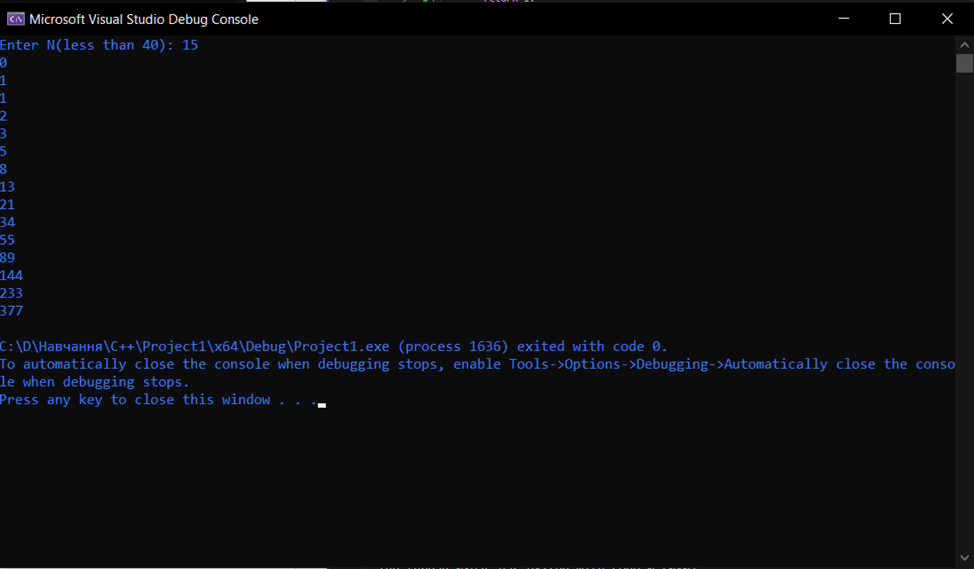
if (n > 40)

return 1;

std::cout << Task2(1, n, 0, 1);

return 0;

}



1.2.3 Створити функцію Task3, яка приймає у якості аргументів: – рядок адреси (назви) файлу textFile, який містить карту лабіринту (інформація подана у вигляді двовимірного масиву 0, 1 та 9, розділених пробілами та переходами на новий рядок, при цьому на карті прийняті такі позначення: 0 – доступні для проходу комірки, 1 – стіни, 9 – вихід (він завжди є і єдиний)); – розміри карти масиву: rowsCount (кількість рядків) та colsCount (кількість стовпців); – початкові координати: startRow (номер стартового рядка, нумерація від 0, як у масивах) та startCol (номер стартового стовпця, нумерація від 0, як у масивах). Функція повинна виводити довжину найкоротшого шляху від заданої початкової точки до виходу (кількість кроків, рухатись можна вперед, назад, вправо та вліво) на рядковий потік (stringstream), який необхідно повернути з функції у вигляді рядка. Обчислення шляху (кількості кроків) реалізувати за допомогою рекурсивної функції CalculationsTask3.

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

int CalculationsTask3(std::vector<std::vector<int>>& maze, int row, int col, int steps) {

size\_t rowsCount = maze.size();

size\_t colsCount = maze[0].size();

if (row < 0 || row >= rowsCount || col < 0 || col >= colsCount || maze[row][col] == 1) {

return INT\_MAX;

}

if (maze[row][col] == 9) {

return steps;

}

maze[row][col] = 1;

int up = CalculationsTask3(maze, row - 1, col, steps + 1);

int down = CalculationsTask3(maze, row + 1, col, steps + 1);

int left = CalculationsTask3(maze, row, col - 1, steps + 1);

int right = CalculationsTask3(maze, row, col + 1, steps + 1);

int minSteps = std::min(std::min(up, down), min(left, right));

maze[row][col] = 0;

return minSteps;

}

std::string Task3(std::string fileName, int rowsCount, int colsCount, int startRow, int startCol) {

std::ifstream inputFile(fileName);

if (!inputFile) {

return "Error: Failed to open the file.";

}

std::vector<std::vector<int>> maze(rowsCount, std::vector<int>(colsCount));

for (int i = 0; i < rowsCount; i++) {

for (int j = 0; j < colsCount; j++) {

if (!(inputFile >> maze[i][j])) {

return "Error: Invalid input format in the file.";

}

}

}

inputFile.close();

int shortestPath = CalculationsTask3(maze, startRow, startCol, 0);

if (shortestPath == INT\_MAX) {

return "No path to the exit.";

}

else {

std::stringstream ss;

ss << "Shortest path to the exit: " << shortestPath << " steps.";

return ss.str();

}

}

int main() {

std::string fileName = "maze.txt";

int rowsCount = 5;

int colsCount = 5;

int startRow = 0;

int startCol = 0;

std::string result = Task3(fileName, rowsCount, colsCount, startRow, startCol);

std::cout << result << std::endl;

return 0;

}

maze.txt

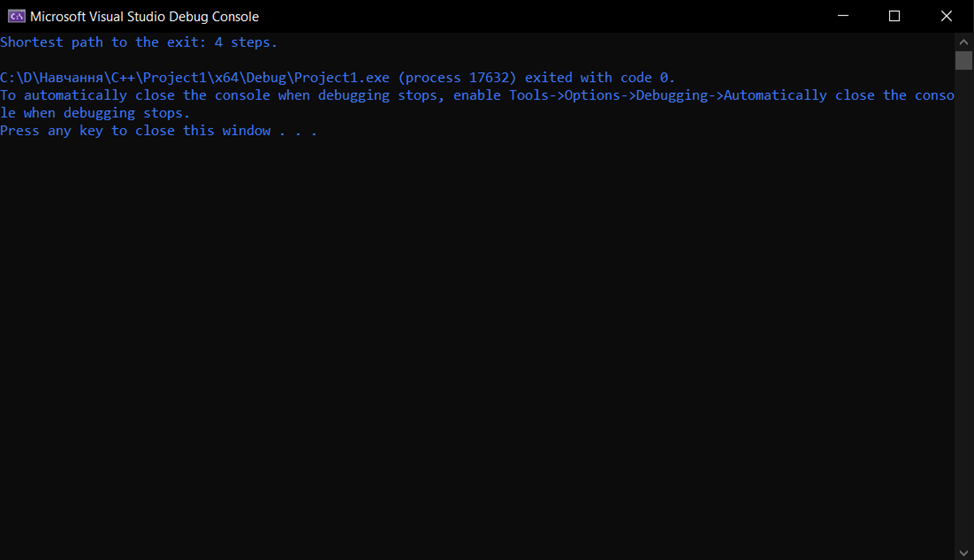
0 1 1 1 1

0 1 1 1 1

0 0 9 0 1

1 1 1 0 1

0 0 0 0 1



**Висновок** Рекурсія - потужний інструмент у програмуванні, який дозволяє розв'язувати складні задачі шляхом розбиття їх на більш прості підзавдання. Важливо правильно визначити базовий випадок для завершення рекурсивного виконання і уникнути безкінечної рекурсії. Треба також враховувати ефективність та споживану пам'ять при застосуванні рекурсії та тестувати її на різних вхідних даних для впевненості в її правильності.