Laboratorio Semana 4

Ejercicio: “El camino más corto en un laberinto”

Aplicar el uso de std::vector para almacenar estructuras de datos dinámicas y utilizar recursión para explorar todas las posibles rutas dentro de un laberinto representado como una matriz.

Descripción del problema

Dado un laberinto representado como una matriz de enteros:

0 representa un camino libre.
1 representa una pared o celda bloqueada.
El objetivo es encontrar la ruta más corta desde la esquina superior izquierda (0,0) hasta la esquina inferior derecha (n-1, n-1), moviéndose solo arriba, abajo, izquierda o derecha.

Requerimientos

Implementar el laberinto con un std::vector<std::vector<int>>.
Usar una función recursiva para explorar todos los caminos posibles desde la posición actual.
Mantener un std::vector<std::pair<int,int>> con el camino actual y otro con el mejor (más corto) camino encontrado hasta el momento.
Evite ciclos infinitos (no se puede visitar dos veces la misma celda).
Al final, mostrar:
El laberinto original.
La longitud del camino más corto.
Las coordenadas del camino más corto.

Ejemplo de entrada

```
std::vector<std::vector<int>> laberinto = {

    {0, 1, 0, 0},

    {0, 0, 0, 1},

    {1, 0, 1, 0},

    {0, 0, 0, 0}

};
```

Salida esperada

Camino más corto encontrado: 6 pasos

Ruta: (0,0) -> (1,0) -> (1,1) -> (1,2) -> (2,3) -> (3,3)

Sugerencias

Crear una función recursiva buscarCamino(...) que reciba:
El laberinto
La posición actual (x, y)
El camino actual (std::vector<std::pair<int,int>>)
El mejor camino hasta el momento (referencia)
Antes de entrar a una celda, verificar que:
Está dentro de los límites.
No es una pared.
No fue visitada.
Usar backtracking para desmarcar las celdas visitadas al retroceder.