**Proyecto:** Generación y resolución automática de laberintos en C++

**Descripción general:** En este proyecto desarrollé un programa que genera automáticamente un laberinto aleatorio utilizando un algoritmo basado en **DFS** (Depth-First Search) con **backtracking**, y luego lo resuelve mediante **BFS** (Breadth-First Search) para encontrar el camino más corto desde la entrada hasta la salida.

El objetivo fue comprender y aplicar técnicas para la generar estructuras complejas, como laberintos, además de reforzar el manejo de estructuras de datos como vectores, colas, mapas y funciones recursivas en C++.

**Generación del laberinto**

Para la generación del laberinto, aplique el algoritmo DFS recursivo con retroceso (backtracking).

Cada celda del laberinto tiene 4 paredes (superior, inferior, izquierda, derecha) y un estado que indica si fue visitada.

Se inicia desde la celda (0, 0) y, en cada paso, se elige una dirección aleatoria (mezclada con el algoritmo shuffle) entre las posibles: arriba, abajo, izquierda, derecha. Si la celda vecina no ha sido visitada, se eliminan las paredes entre ambas y se continúa recursivamente desde allí.

Este enfoque garantiza que el laberinto generado sea perfecto, es decir, que tenga un único camino entre cualquier par de celdas, sin ciclos.

**Resolución del laberinto**

El laberinto fue resuelto utilizando el algoritmo BFS (Breadth-First Search). Esta elección se basó por lo siguiente:

Explorar el grafo de forma nivel por nivel, asegurando que la primera vez que se llega a la salida sea siempre a través del camino más corto.

Reconstruir ese camino a través de un mapa padre que guarda la relación entre cada celda visitada y su antecesora.

Adaptarse fácilmente a laberintos representados por celdas con paredes, verificando condiciones específicas antes de moverse.

El resultado es una ruta marcada con \* desde la entrada (E) en la esquina superior izquierda hasta la salida (S) en la inferior derecha.

**Cosas que descubrir y haría distinto**

Durante el desarrollo me enfoqué más en la lógica que en la medición de rendimiento, pero con el avanzar del proyecto me di cuenta que el algoritmo DFS es más eficiente para generar un laberinto y el BFS es más eficiente para resolver el laberinto.

En cuanto a lo que haría diferente es la implementación de visualizaciones en consola con pausas entre pasos para analizar el comportamiento de los algoritmos en tiempo real.