Instituto Politécnico Nacional

Escuela Superior de Cómputo

Unidad de aprendizaje:
Fundamentos de programación

Proyecto:
Generador de laberintos

Alumno:

González Cárdenas Ángel Aquilez

Boleta: 2016630152

Grupo: 1CV7

Profesor: Gutiérrez Aldana Eduardo

Índice

1. Introducción

El presente escrito documenta el proceso de resolución de una problemática particular presentada como parte del proyecto de evaluación correspondiente al semestre *2024-1* de la unidad de aprendizaje *Fundamentos de programación* para la carrera de *Ingeniería en Sistemas Computacionales* del programa académico *ISC2020*.

2. Problemática a resolver

¿Cómo se puede diseñar e implementar un algoritmo que genere laberintos aleatorios con diferentes niveles de dificultad, utilizando matrices bidimensionales como estructura de datos y el lenguaje de programación C?

Esta problemática plantea el reto de crear un programa que pueda generar laberintos de forma dinámica, sin tener que definirlos previamente de forma estática. Además, se debe considerar el nivel de dificultad del laberinto, que puede depender de factores como el tamaño de la matriz, el número y la posición de las paredes, la existencia o no de múltiples soluciones, etc. El programa debe ser capaz de mostrar el laberinto generado en la pantalla, así como permitir al usuario interactuar con él para intentar resolverlo. El lenguaje de programación C se elige por su eficiencia y su facilidad para trabajar con matrices y punteros.

El programa ya realizado deberá realizar las siguientes tareas:

A. Inicialización: Comienza con una cuadrícula (matriz bidimensional) de celdas. Inicialmente, todas las celdas se consideran paredes. Elige una celda inicial al azar y márcala como un pasillo. Agrega esta celda a una pila o lista.

B. Bucle principal: Mientras la pila no esté vacía, realiza los siguientes pasos:

   a Paso a la siguiente celda: Saca la celda superior de la pila. Esta será tu celda actual.

   b Obtén vecinos no visitados: Encuentra los vecinos no visitados (celdas adyacentes) de la celda actual.

   c Si no hay vecinos no visitados: Retrocede a la celda anterior sacada de la pila. Esto es lo que hace que el algoritmo sea "backtracking" (retroceso).

   d Si hay vecinos no visitados: Elige al azar uno de los vecinos no visitados. Marca la celda vecina como pasillo y la celda actual como visitada. Agrega la celda actual a la pila.

C. Terminación: Continúa el proceso hasta que todas las celdas hayan sido visitadas. En este punto, habrás creado un laberinto con pasillos y paredes.

El algoritmo "Maze Carver" es recursivo en naturaleza debido a la pila o lista utilizada para rastrear las celdas visitadas. En cada paso, se avanza y retrocede en la estructura de datos para construir gradualmente el laberinto.

Este algoritmo tiene la ventaja de que puede generar laberintos con caminos largos y rutas interesantes. Además, es relativamente fácil de implementar y entender. Sin embargo, puede generar laberintos con muchos pasillos rectos y corredores largos, lo que puede no ser estéticamente agradable para algunos casos de uso.