Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE0117 Programación bajo plataformas abiertas

Proyecto: resolvedor de laberintos

por

Javier Monge Matamoros, carné B74888

I-2022

Índice general

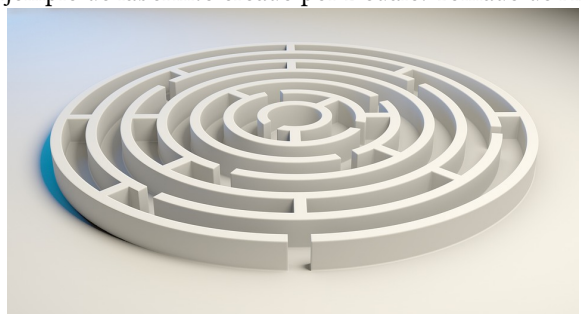
1. **Introducción** **2**
2. **Diseño general** **3**
3. **Principales retos** **4**
4. **Conclusiones** **6**

Capítulo 1

Introducción

En este informe se abarcará una solución para encontrar respuestas a laberintos mediante un programa en lenguaje C como parte del proyecto del curso de programación bajo plataformas abiertas. La estrategia más común para resolver un laberinto es la regla de la mano derecha, donde se coloca la mano derecha en la pared y se mantiene allí hasta encontrar una salida (Roberts, 2015). En la antigua Grecia, se dice que

Figura 1.1: Ejemplo de laberinto creado por Dédalo. Tomado de Anapliotis, s.f.



cuando el Minotauro nació, su madre Pasífae fue capaz de alimentarlo. Sin embargo, a medida que crecía, se volvió insaciable. Como era una criatura antinatural, un híbrido de hombre y bestia, no tenía una dieta natural y empezó a comer humanos. El rey Minos, marido de Pasífae consultó a un oráculo para determinar qué hacer con la bestia. El oráculo aconsejó un laberinto. Dédalo, maestro artesano e ingeniero que diseñó el artilugio con el que se concibió la bestia, y que más tarde crearía alas para sí mismo y para su hijo Ícaro, creó un elaborado laberinto para albergar con seguridad al Minotauro(Anapliotis, s.f.). Esta es la primera vez que se tenga registro del concepto laberinto y se muestra en la figura 1.1.

Capítulo 2

Diseño general

Para construir el laberinto es necesario leer el archivo de texto "laberinto.txt"que contiene una matriz con números, donde los 1 representan un posible camino a la solución, los 0 son paredes y el 2 es la solución del laberinto. Un ejemplo de archivo texto con un laberinto es el siguiente: El código busca el tamaño de

0	0	1	0	0
0	1	1	0	0
0	0	1	1	0
0	0	0	2	0

la matriz, que es mxn; es decir, guarda la cantidad de filas y columnas del laberinto. Luego crea la matriz y la completa con los datos leídos del archivo de texto. Luego, se recorren los bordes de la matriz para encontrar la entrada o entradas del laberinto. En este método también se llama al método que busca la solución a partir de la entrada encontrada. Este método utiliza la estrategia explicada en la introducción de este reporte, que es la regla de la mano derecha. Esta estrategia continua por un camino hasta el final, sea un 2 o un 1. Donde si es un 1 quiere decir que este camino no lleva a la solución, mientras que el 2 significa que se llegó a la solución. Estas opciones se imprimen. Si el 2 fue encontrado el programa devuelve las coordenadas de dónde se encuentra en la matriz la solución, siguiendo la siguiente enumeración: Es decir,

	0	1	2	3	4
0	0	0	1	0	0
1	0	1	1	0	0
2	0	0	1	1	0
3	0	0	0	2	0

la solución de este laberinto sería: (3,3).

Capítulo 3

Principales retos

1. Construir una matriz a partir de un archivo de texto: El reto de esta sección fue poder hacer que el código creara una matriz a partir de datos de un archivo de texto. Primero se calculó el tamaño de la matriz; es decir, conocer el número de filas y columnas. Luego, con estos datos se construye la matriz nxm para luego completarla abriendo un archivo .txt y escribiendo en cada coordenada el valor del archivo. El código también detecta si el archivo ingresado es válido, es decir cumple con las especificaciones dadas en el proyecto o se debe ingresar otro archivo.

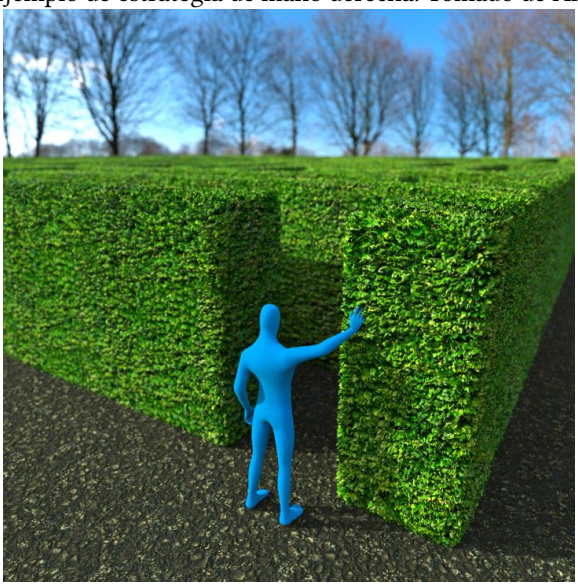
2. Encontrar la estrategia de solución: Después de mucho indagar, la mejor estrategia, como se ha mencionado, fue la regla de la mano derecha. La teoría dice que es posible encontrar la salida a un laberinto manteniendo la mano derecha pegada a la pared, como la figura 3.1. Por lo tanto, el código debe buscar primero si hay camino a mano derecha, si no fuera un camino, busca abajo y si tampoco encuentra busca al lado izquierdo. Si estas 3 opciones no son caminos, debe devolverse para atrás.

3. Programar el algoritmo: Como se menciona en punto anterior, un reto fue programar la lógica detrás de la estrategia de la mano derecha. El principal obstáculo de esta etapa era como hacer que el programa supiera un camino ya ha sido recorrido que se logró satisfactoriamente y si existiese más de un camino con solución se imprimen todas las entradas posibles.

4. Compilación: Debido a la arquitectura de la computadora en la que fue desarrollado este proyecto, se presentaron muchas limitantes por cuestiones de arquitectura e incompatibilidades. Sin embargo, dentro de las herramientas disponibles para MacOS fue posible compilar y probar el código elaborado.

5. Estructura: un gran reto fue mantener un main() bastante limpio para evitar tener código innecesario en él. Por lo tanto se desarrolló todo el proyecto en métodos que se llaman durante la ejecución del código y en el main solo se debe indicar el nombre del archivo de texto que contiene el laberinto que se desea resolver.

Figura 3.1: Ejemplo de estrategia de mano derecha. Tomado de Anapliotis, s.f.



Capítulo 4

Conclusiones

Poder construir un resolvedor de laberintos fue un reto bastante complejo, donde se debían integrar muchos conocimientos del curso, además de establecer una estrategia con una lógica que pudiera ser útil para resolver cualquier laberinto. A pesar de los retos expuestos en la sección anterior, fue posible crear un programa capaz de resolver un laberinto contenido en un archivo de texto y devolver la ubicación de la solución, si existiese. Con este proyecto se profundizó el conocimiento en lectura y escritura de archivos mediante código, estrategias para resolver problemas y el manejo de matrices. A pesar de las múltiples pruebas al código, sigue existiendo la duda de si realmente esta lógica puede resolver cualquier programa.

Bibliografía

Anapliotis, Harry (s.f.). *Everything about the Mythical Labyrinth of the Minotaur (Minoan Maze)*. URL: https://www.rental-center-crete.com/blog/labyrinth-of-the-minotaur/.
Roberts, Eric (2015). "Recursive Backtracking Recursive Backtracking Solving a Maze". En.