UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA EN INFORMÁTICA

ESTRUCTURA DE DATOS Y ANÁLISIS DE ALORITMOS

INFORME 1

Cristian Eduardo Espinoza Silva

Profesor: Alejandro Cisterna
Ayudante: -
Fecha de Entrega: 6 de abril de 2017

Santiago de Chile

1 - 2017

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

CAPÍTULO 1. INTRODUCCIÓN

En el presente informe se ha pedido a los alumnos de la asignatura Estructura de datos y análisis de algoritmos que lleven a cabo un proyecto orientado a la resolución de un “Laberinto” en particular.

Respecto a la solución única del “Laberinto” que recae en encontrar la salida, lo que conlleva a los alumnos de la asignatura crear un software que sea capaz de poder cumplir con dicho objetivo planteado anteriormente.

En el transcurso del informe se dará a conocer el desarrollo y la solución implementada para dicho problema, es decir, se describirá paso a paso el desarrollo de la solución que logro acabar con cada una de las problemáticas planteadas.

1.1 DESCRIPCIÓN DEL PROBLEMA

El problema consiste en diseñar un software que se capaz de resolver un “Laberinto”, el cual tiene una temática particular la cual recae en resolver un laberinto circular. Se lee una matriz, además de las filas y columnas que tendrá. El jugador se encarga de manejar el menú de inicio, con el fin de poder dar una interacción con el usuario.

El principal objetivo es encontrar el camino hacia la llave en primera instancia para proceder encontrar el camino hacia la salida.

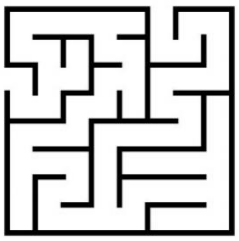

Figura 1: Laberinto

1.2 OBJETIVOS DEL SOFTWARE

El objetivo es realizar un software capaz de solucionar el “Laberinto”, desarrollado en lenguaje de programación C, ocupando el paradigma de programación imperativo.

Los objetivos del software son implementar las siguientes soluciones:

- Cargar la matriz y sus dimensiones entregadas en un archivo de texto.

- Encontrar el camino hacia la llave, luego encontrar el camino hacia la salida del “Laberinto”.
- Entregar un archivo de texto con los dos caminos mencionados anteriormente.

1.3 ASPECTOS DE IMPLEMENTACIÓN

El problema fue abarcado con la herramienta de división en sub-problemas, con el fin de poder abarcar en su totalidad cada de sus dificultades.

El principal alcance que se presenta es el cumplimiento del desarrollo del software.

Se lleva a cabo el software en el lenguaje de programación C, desarrollado en un editor de texto encarecido, es decir, funciona como una ayuda con la sintaxis propia del lenguaje destacando estructuras, funciones, declaraciones, etc.

La estructura del software se organiza en tres archivos, los cuales son: El archivo ejecutable que contiene el “main”, un archivo “.c” que contiene las funciones que se ocupan y por último el “.h” que contiene las estructuras que se utilizan en el trascurso del software.

Se importan distintas tipas de bibliotecas, con el fin de poder controlar e solucionar problemas que se fueron descubriendo en el transcurso del software.

1.4 MARCO TEÓRICO

- Estructuras (struct): Representan colecciones de variables que son diferenciadas cada una de ellas por el tipo de variable y su respectivo nombre, estas pueden contener variables de distintos tipos.
  Ocupan la palabra reservada “Typedef struct”, en la siguiente imagen se está declarando una estructura llamada Tablero.

```
typedef struct{

    int fila;
    int columna;
    char **mapa;

}Tablero;
```

Figura 2: Struct Tablero.

CAPÍTULO 2. DESARROLLO

Dentro del siguiente capítulo se otorgará una visión general del software, como fue programado y que problemas resuelven sus partes.

2.1 DESCRIPCIÓN DEL ALGORITMO

Para la solución del problema planteado se aplica la técnica de espacio de estados modificada en ciertos aspectos, los conceptos claves que se ocupan dentro del algoritmo son los siguientes:

- Estado actual: Se caracteriza por ser la posición en la cual se está actualmente dentro del laberinto, estos dentro del código
- Estados pendientes: Son las posiciones que se van abriendo y aún no han sido utilizadas.
- Estados cerrados: Son las posiciones por las cuales ya se pasó en su determinado tiempo.
- Ruta: Son todas las posibles combinaciones que se pueden realizar dentro del laberinto. En las cuales se encuentran las siguientes posibles combinaciones:
    - Abajo
    - Arriba
    - Derecha
    - izquierda

El algoritmo tiene una base sólida debido a la implementación de dos “Struct”, los cuales serán explicados a continuación:

- Struct Tablero: Esta estructura se encarga de almacenar la matriz y sus dimensiones que contendrá, es decir, dentro de ella contiene 3 variables:
    - Fila y columna: Las cuales se encargan de almacenar las dimensiones que contendrá dicha matriz.
    - Matriz: Es el cuerpo del software, ya que es donde se almacena la matriz leída desde el archivo de texto.

- Struct Position: Esta estructura se encarga de tener un registro de cada una de las coordenadas de la matriz, es decir, dentro de ella contiene 4 variables:

- Fila y columna: Guarda las coordenadas de la posición que se está trabajando.
- Estado: Mantiene una vigilancia, es decir, entrega información si la casilla ya fue abierta anteriormente.
- Origen: Nos permite saber de dónde proviene cada una de las posiciones que se van agregando a los estados pendientes.

Ahora que conocemos esto, debemos aclarar que los Estados Pendientes, Estados Cerrados y Estado actual, son representados como un puntero a arreglo de Position.

Para poder llegar a la solución final del problema se realiza una secuencia de pasos que se puede apreciar en el siguiente diagrama:

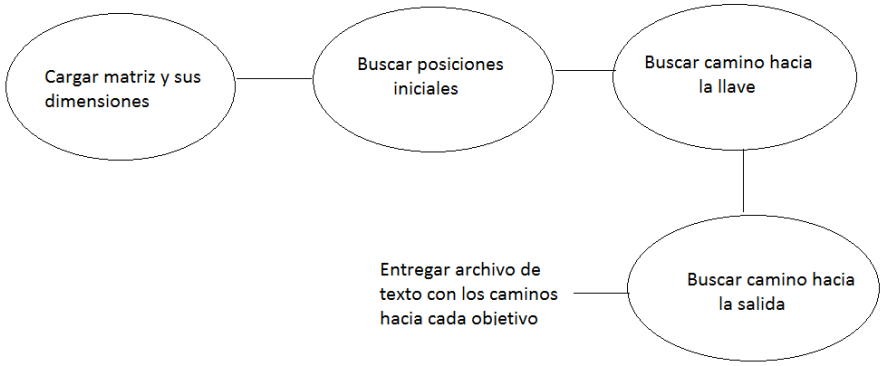

Figura 3: Diagrama de abstracción del problema central.

Una vez ya cargada la matriz y sus respectivas dimensiones, se procede hacer el siguiente paso descrito:

- Buscar posiciones iniciales: Se refiere a buscar las posiciones donde se encuentra la Entrada, Llave y Salida.
- Buscar camino hacia la llave: Se encarga de encontrar el camino correcto hacia la posición donde se encuentra la llave, este procedimiento se lleva a cabo de la siguiente manera:
  - Primero se asigna como posición actual las coordenadas de donde se encuentra la Entrada del laberinto, además de agregar el origen de dicha coordenada.
  - Luego, se toma dicha coordenada y se le aplica cada una de los posibles movimientos que se pueden efectuar dentro del laberinto

también llamado Ruta anteriormente, además de agregar cada coordenada posible al arreglo de Estados pendientes.
    - Después, se procede a cerrar el Estado Actual y a tomar el primer elemento del arreglo Estados pendientes el cual pasará a ser el nuevo Estado Actual dentro del "Laberinto", además de agregar el origen para tener conocimiento el origen de donde proviene dicha posición, con el objetivo de poder reunir el camino final.
    - Por último, se realizada un llamado recursivo repitiendo cada uno de los pasos expuestos anteriormente hasta que se llega al caso base, el cual es cuando el Estado Actual es el objetivo final, es decir, las coordenadas de la llave o salida del "Laberinto".
- Al tener ya el camino final hacia el objetivo pedido, se procede a reunir cada una de las coordenadas e ir retrocediendo a la coordenada de donde proviene, así obteniendo el camino final hacia el objetivo pedido.

Tener en cuenta que este paso es realizado para el camino hacia la Llave y luego de obtener la llave es realizado para el camino hacia la Salida del "Laberinto".

- Por último, se procede a entregar una salda en forma de un archivo de texto con extensión ".in", el cual contiene el camino hacia la Llave y debajo de este el camino hacia la Salida del "Laberinto".

2.2 ANÁLISIS DE RESULTADOS

Se ha generado el código acorde al algoritmo descrito anteriormente, solucionando en su totalidad la problemática propuesta.

- Manejo de archivos: Se implementó dos funciones que cumplen las funcionalidades de leer y crear un archivo con el camino hacia los objetivos del "Laberinto". En la siguiente imagen se puede apreciar como se muestra la salida del software.

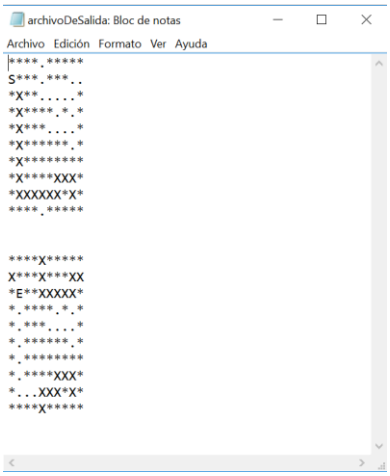

Figura 4: Salida que entrega el software con los caminos encontrados.

Se puede apreciar que el resultado es exitoso, es decir, se puede afirmar que se cumplió el objetivo del problema planteado.

2.3 TIEMPOS DE EJECUCIÓN

Como el software está desarrollado en base a la herramienta de división en sub-problemas, se tiene diversos algoritmos que abarcan cada uno de los sub-problemas.

A continuación, se darán a conocer cada una de las funciones que se implementaron dentro del código:

- Cargar: Algoritmo encargado de leer la matriz y sus dimensiones, además de almacenar la matriz dentro de la memoria.
  - T(n): 2*(n^2) + n + 13
  - O(n^2).

- Print: Algoritmo encargado de mostrar el laberinto leído por pantalla.
  - T(n): n^2 + n + 3
  - O(n^2).

- CreateBoard: Algoritmo encargado de crear una matriz con las dimensiones que se rescataron desde el archivo de texto.
  - T(n): n^2 + n + 7
  - O(n^2).

- BuscarPos: Algoritmo encargado de buscar la Entrada, Llave o Salida dentro de la matriz.
    - T(n): 3*n(^2) + 9
    - O(n^2).

- NoEstarCerrado: Algoritmo encargado de verificar si las posiciones entregadas se encuentran dentro del arreglo de Estados Cerrados.
    - T(n): 2n + 7
    - O(n).

- BuscarCamino: Algoritmo encargado de buscar el camino hacia el objetivo que se esté buscando en esa oportunidad. Se puede decir que esta es la función central del programa, ya que se encarga de encontrar el camino correcto hacia el objetivo.
    - O(n^2).

- CaminoCorrec: Algoritmo encargado de unir todas las posiciones que se utilizan para llegar al objetivo de tal manera formando el camino correcto, esta función es utilizada después de encontrar el camino hacia el objetivo.
    - O(n^3).

- SaveTablero: Algoritmo encargado de crear un archivo con los dos caminos encontrados.
    - T(n): n^2 + n + 14
    - O(n^2).

- Verificar: Algoritmo encargado de verificar si las coordenadas de la matriz se encuentran dentro del arreglo de Camino Correcto.
    - T(n): 3n + 7
    - O(n).

- Bloque Principal: Función que se encarga de llamar a todas las funciones descritas anteriormente en un orden adecuado para poder llegar de buena

manera al objetivo central del problema, el cual recae en encontrar el camino hacia la Llave y luego hacia la Salida del “Laberinto”.

- T(n): 16 + 13n
- O(n).

Concluyendo con lo anterior entonces podemos decir que el software tiene un tiempo de orden O(n^3).

CAPÍTULO 3. CONCLUSIÓN

En este presente “Laboratorio” se pudo llevar a practica la teoría que se aprendió en la catedra, pudiendo tener una visión del cómo llevar a la práctica cada uno de los elementos vistos y estudiados.

Como se mencionó en la sección de análisis de resultados se ha podido lograr el objetivo, que era solucionar el problema planteado, que en esta ocasión era llegar encontrar la Llave y luego proceder a encontrar la salida del “Laberinto”.

La mayor dificultad dentro del desarrollo del software fue la organización del código para obtener la solución requerida.

Para llegar a cada una de las soluciones finales se crearon varios algoritmos que no cumplían con lo que se buscaba obtener, estos han sido descartados, pero dejando una gran idea para poder llegar al algoritmo correcto, que cumplía con todas las necesitas que se buscaba.

CAPÍTULO 4. REFERENCIAS

Anomino. (Julio de 2007). *C con clase.*

Borjas, V. (2013). *Estructuras en C.*