

**ESTRUCTURA DE DATOS Y ANÁLISIS DE ALGORITMOS
INFORME Y MANUAL**

Cristian Eduardo Espinoza Silva

Profesor: Alejandro Cisterna
Ayudante: Luis Migryk
Fecha de Entrega: 6 de abril de 2017

Santiago de Chile

1 - 2017

TABLA DE CONTENIDO

CAPÍTULO 1. Introducción	3
1.1 Descripción del problema.....	3
1.2 Objetivos del software	4
CAPÍTULO 2. Desarrollo.....	5
2.1 Marco teórico	5
2.2 Aspectos de implementación.....	6
2.3 Descripción del algoritmo	6
2.4 Funciones	9
2.5 Análisis de resultados.....	10
2.6 Tiempos de ejecución.....	12
CAPÍTULO 3. Conclusión	13
CAPÍTULO 4. Referencias.....	14
CAPÍTULO 5. Manual de usuario.....	15
5.1 Introducción	15
5.2 Instrucciones.....	16
5.2.1 Descarga de mingw	16
5.2.2 Compilación	16
5.2.3 Ejecución	22
5.3 Funcionalidades.....	24
5.4 exitos y Posibles errores	28

ÍNDICE DE FIGURAS

Figura 1: Laberinto.....	3
Figura 2: Struct utilizados en el software.....	5
Figura 3: Representación de una lista enlazada.....	5
Figura 4: Diagrama de abstracción del problema central.....	8
Figura 5: Salida que entrega el software con los caminos encontrados.....	11
Figura 6: Tabla de resultados $T(n)$ y $O(n)$	12
Figura 7: Ventana ejecutar en sistema operativo Windows.....	16
Figura 8: Consola de Windows.....	17
Figura 9: Crear carpeta y almacenar software dentro.....	17
Figura 10: Buscando ruta de la carpeta creada.....	18
Figura 11: Compilando software.....	18
Figura 12: Creado el ejecutable del software.....	19
Figura 13: Abrir consola de Linux o Terminal.....	19
Figura 14: Ventana de la consola de Linux o Terminal.....	20
Figura 15: Carpeta con el software dentro.....	20
Figura 16: Consola de Linux con la ruta de la carpeta que contiene el software.....	21
Figura 17: Compilación de software.....	21
Figura 18: Carpeta con el archivo ejecutable.....	22
Figura 19: Software ejecutado.....	23
Figura 20: Software ejecutado.....	23
Figura 21: Menú principal del software.....	24
Figura 22: Cargando el archivo al software.....	25
Figura 23: Mostrando el laberinto por pantalla.....	26
Figura 24: Buscando camino hacia la Llave y luego hacia la salida.....	26
Figura 25: Cerrando el software.....	26
Figura 26: Carpeta con el archivo de “salida.out” del software.....	27
Figura 27: Salida final del software.....	27
Figura 28: Posible error dentro de software.....	28

CAPÍTULO 1. INTRODUCCIÓN

En el presente informe se ha pedido a los alumnos de la universidad de Santiago de Chile 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, luego proceder encontrar el camino hacia la salida.

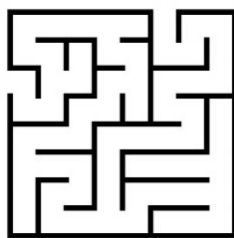


Figura 1: Laberinto

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 mínimo hacia la llave, luego encontrar el camino mínimo hacia la salida del “Laberinto”.
- Implementar listas enlazadas dentro del código a implementar.
- Entregar un archivo de texto con los dos caminos mencionados anteriormente, mostrando la serie de paso que se debe seguir para llegar al objetivo descrito.

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 MARCO TEÓRICO

- Estructuras: 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 figura 2 se puede apreciar los struct declarados.

```
typedef struct{
    int fila;
    int columna;
    char **mapa;
}Tablero;

typedef struct{
    int fila;
    int columna;
    int origen;
    int estado;
    char* camino;
}Position;

typedef struct Nodo{
    char* camino;
    int fila;
    int columna;
    struct Nodo* siguiente;
}Nodo;

typedef struct{
    int largo;
    Nodo* primero;
    Nodo* ultimo;
}Lista;
```

Figura 2: Struct utilizados en el software.

- Listas simples enlazadas: Nos permite almacenar datos de una forma organizada, al igual que los vectores, pero, a diferencia de estos, esta estructura es dinámica.

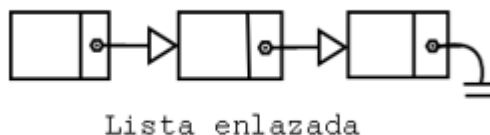


Figura 3: Representación de una lista enlazada.

2.2 ASPECTOS DE IMPLEMENTACIÓN

El problema fue abarcado con la herramienta de división en sub-problemas y recursividad, 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 el editor de texto “Sublime Text 3”, es 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 transcurso del software.

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

2.3 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.
- Transiciones (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 4 “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.

- Struct Lista: Esta estructura contendrá a una serie de elementos dentro de ella que cada uno tendrá características distintas, además contiene 3 variables:
 - Largo: Representa cuando elementos contiene.
 - Primero: Indica el primer elemento de la lista.
 - Ultimo: Indica el último elemento de la lista.
- Struct Nodo: Esta estructura ayuda a la representación de lista enlazadas, además se encarga de tener un registro de cada una de las transiciones, es decir, dentro de ella contiene 4 variables:
 - Fila y columna: Guarda las posiciones de como representar un movimiento dentro del “Laberinto”.
 - Siguiente: Contiene en su interior el nodo siguiente que lo procede.
 - Camino: Contiene el nombre del movimiento realizado.

Ahora que conocemos esto, debemos aclarar que los Estados Pendientes, Estados Cerrados y Estado actual, son representados como un puntero a arreglo de Position y las transiciones están implementadas con listas enlazadas.

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

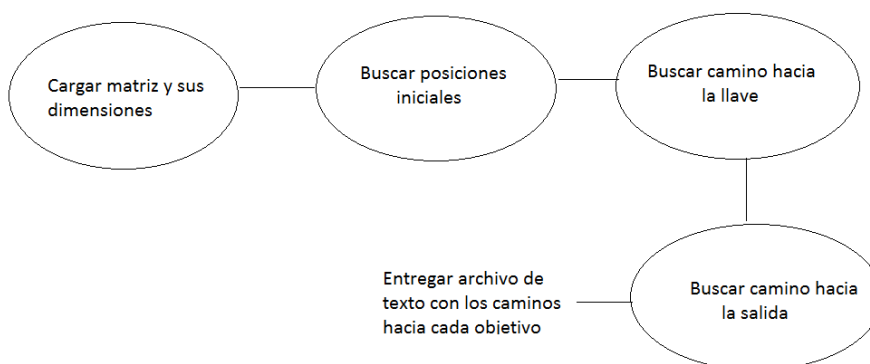


Figura 4: Diagrama de abstracción del problema central.

Primeramente, se procede a declarar una variable del tipo “Struct tablero”, luego se solicita memoria correspondiente a las filas y columnas respectiva de la matriz a leer, extraídas desde el archivo, por último, se procede a cargar la matriz que se encuentra en el almacenada en el archivo.

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 realiza 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 esta secuencia de paso es realizada para el camino hacia la Llave, luego de obtener la llave es realizado para el camino hacia la Salida del “Laberinto”.

2.4 FUNCIONES

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.
- Print: Algoritmo encargado de mostrar el laberinto leído por pantalla.
- CreateBoard: Algoritmo encargado de crear una matriz con las dimensiones que se rescataron desde el archivo de texto.
- BuscarPos: Algoritmo encargado de buscar la Entrada, Llave o Salida dentro de la matriz.
- NoEstarCerrado: Algoritmo encargado de verificar si las posiciones entregadas se encuentran dentro del arreglo de Estados Cerrados.
- 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.

- 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.
- SaveTablero: Algoritmo encargado de crear un archivo con los dos caminos encontrados.
- Verificar: Algoritmo encargado de verificar si las coordenadas de la matriz se encuentran dentro del arreglo de Camino Correcto.
- CrearLista: Algoritmo encargado de crear una lista de tipo “Struct lista”.
- CrearNodo: Algoritmo encargado de crear un elemento de tipo “Struct Nodo”.
- AgregarNodo: Algoritmo encargado de agregar un elemento a la lista enlazada que este declara en el programa.
- BuscarNodo: Algoritmo encargado de buscar un elemento dentro de una lista enlazada.
- CrearRuta: Algoritmo encargado de crear cada una de las transiciones que se pueden realizar dentro del “Laberinto”, además de agregarlas a una lista enlazada.
- 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”.

2.5 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.



```

Salida: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
*****
**XXXX**
*****X**
*****X**
**...X**
S..****

Derecha
Derecha
Derecha
Abajo
Abajo
Abajo

*****
**E..**
*****
*****
*****
*****
XXX****

Izquierda
Izquierda
Izquierda
Abajo
Izquierda
Izquierda

```

Figura 5: 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.

2.6 TIEMPOS DE EJECUCIÓN

Funciones	$T(n)$	$O(n)$
Cargar	$2*(n^2) + n + 13$	n^2
Print	$(n^2) + n + 3$	n^2
CreateBoard	$(n^2) + n + 7$	n^2
BuscarPos	$3*(n^2) + 9$	n^2
NoEstarCerrado	$2n + 7$	n^2
BuscarCamino	$13n^2 + 11n + 8$	n^2
CaminoCorrec	$7n^3 + 7n^2 + 15$	n^3
SaveTablero	$(n^2) + n + 14$	n^2
Verificar	$3n + 7$	n
Bloque Principal	$13n + 16$	n
crearLista	5	1
crearNodo	6	1
agregarNodo	5	1
buscarNodo	10	1
crearRuta	19	1

Figura 6: Tabla de resultados $T(n)$ y $O(n)$.

- Se puede apreciar que el software tiene un orden de $O(n^3)$.

Se debe destacar que debe haber mejores algoritmos que tengan un orden menor y puedan cumplir con las mismas funcionalidades que este algoritmo planteado.

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.

Además, se debe destacar que los resultados del software fueron exitosos en la gran parte de pruebas, además se implementó un algoritmo de $O(n^3)$, sin embargo, se deje dejar abierta la posibilidad de que existen algoritmos que tiene un orden mejor y cumplen con las mismas funcionalidades que el algoritmo planteado.

CAPÍTULO 4. REFERENCIAS

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

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

CAPÍTULO 5. MANUAL DE USUARIO

5.1 INTRODUCCIÓN

En el siguiente documento se presenta una forma adecuada, explicativa y simple de cómo utilizar el software hecho.

El software principalmente consiste en la resolución del problema entregado en el laboratorio 1, que el objetivo principal es encontrar el camino correcto hacia la llave y luego hacia la salida del “Laberinto”.

Las reglas del “Laberinto” son diferentes a las que estamos acostumbrados, ya que se podría decir que es un “Laberinto circular”, de manera que si uno sale por un lugar inferior procede a entrar por la parte superior, así mismo si uno procede a salir por los costados entrara por el costado opuesto.

Dentro del transcurso del manual se procede a explicar desde el primer paso que se debe hacer para poder utilizar el software de una manera correcta.

Se recomienda tener poseer un lenguaje técnico mayor al promedio para poder aprovechar las ventajas que entrega la aplicación.

Por ultimo debemos destacar que el software esta implementado en un paradigma imperativo, que principalmente consta de declaraciones que se ejecutan secuencialmente.

5.2 INSTRUCCIONES

A continuación, se procede a explicar cada uno de los pasos a seguir para poder compilar el software que soluciona el “Laberinto”.

5.2.1 Descarga de mingw

Para poder compilar el software credo se necesita descargar lo siguiente, con el fin que los comandos ingresados sean conocidos por su equipo.

Se puede obtener del siguiente link:

- <https://www.fdi.ucm.es/profesor/luis/fp/devtools/MinGW.html>

5.2.2 Compilación

5.2.2.1 Windows:

1. Debemos abrir la consola de Windows, existen dos opciones las cuales son:
 - a. Apretar Windows + R
 - b. Abrir la ventana de ejecutar y escribir “cmd”.

Una de las opciones se puede apreciar en la Figura 1, que viene a continuación:

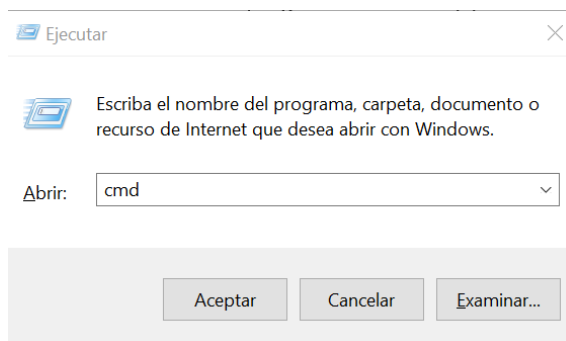


Figura 7: Ventana ejecutar en sistema operativo Windows.

2. Luego de realizar el anterior paso, se abrirá la consola como mostrará en la Figura 2:

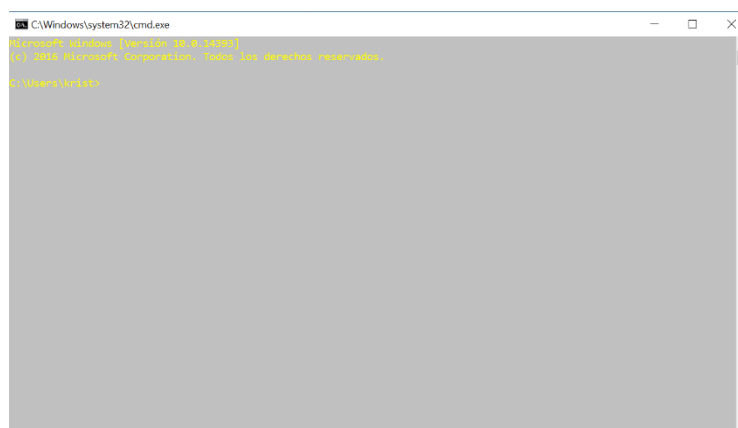


Figura 8: Consola de Windows.

3. Almacenar el software dentro de una carpeta para poder tener todo lo necesario dentro de ella, lo recomendado es crear dicha carpeta en el escritorio de su máquina, como se muestra en la Figura 3:

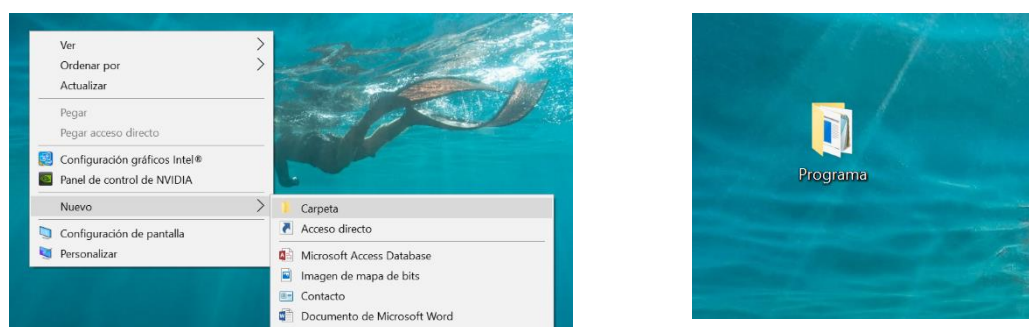
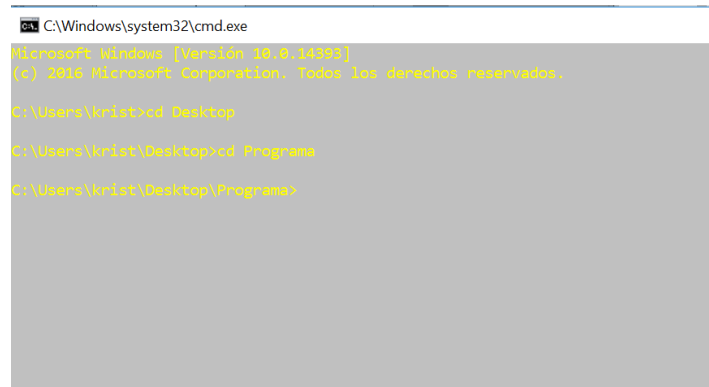


Figura 9: Crear carpeta y almacenar software dentro.

4. Luego volvemos a la consola de Windows, para poder ejecutar el software y seguir los siguientes pasos:
 - a. Primer paso, dejamos la consola de Windows en la ruta donde tenemos almacenado nuestro programa, de la manera que muestra en la Figura 4:



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\kr1st>cd Desktop

C:\Users\kr1st\Desktop>cd Programa

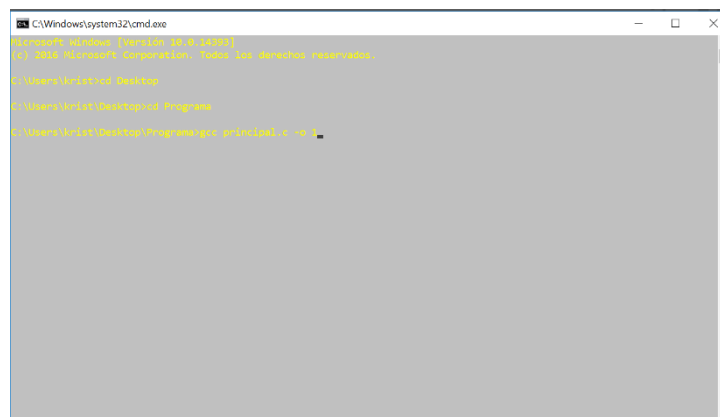
C:\Users\kr1st\Desktop\Programa>
```

Figura 10: Buscando ruta de la carpeta creada.

b. Luego comenzamos la compilación, escribiendo lo siguiente en la consola de Windows:

- gcc (Nombre del archivo).c -o (Nombre del ejecutable)
- (Nombre del ejecutable).exe

La Figura 5 entrega las instrucciones con mayor claridad:



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\kr1st>cd Desktop

C:\Users\kr1st\Desktop>cd Programa

C:\Users\kr1st\Desktop\Programa>gcc principal.c -o 1_
```

Figura 11: Compilando software

- c. Luego se crea el archivo ejecutable en la carpeta donde tenemos almacenado el software, como muestra la Figura 6:

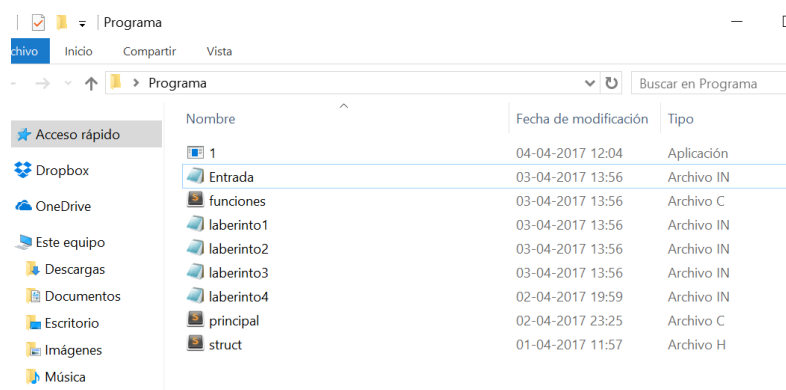


Figura 12: Creado el ejecutable del software.

5.2.2.2 Linux

1. Debemos abrir la consola de Linux o también llamada terminal, existen dos opciones las cuales son:
 - a. Apretar clic derecho y colocar la opción abrir terminal.
 - b. Buscar en los programas de Linux y colocar “Terminal”

Una de las opciones se puede apreciar en la Figura 7 que se observa abajo:

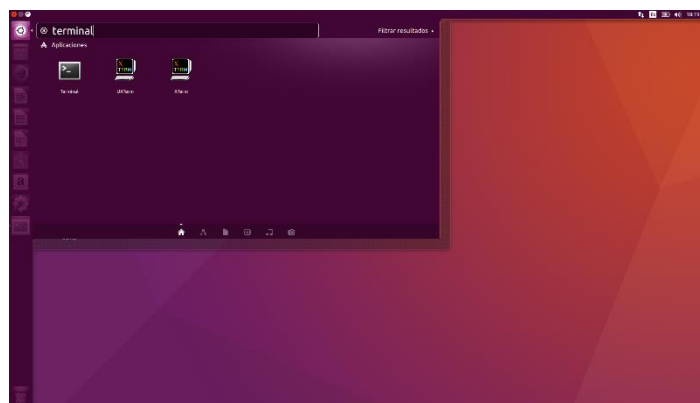


Figura 13: Abrir consola de Linux o Terminal.

2. Luego de realizar el anterior paso, se abrirá la consola como mostrará la Figura 8:

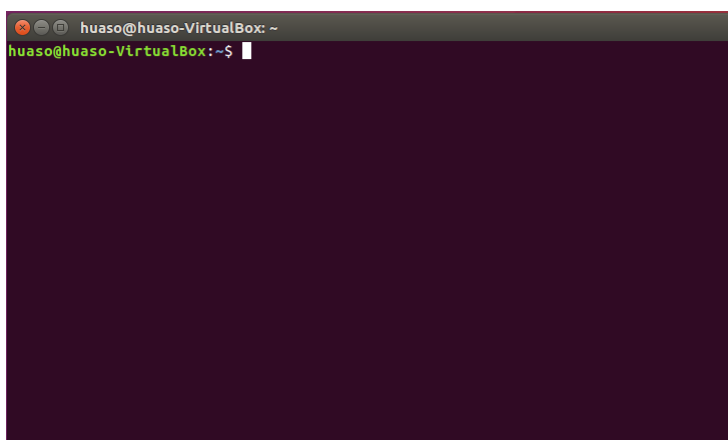


Figura 14: Ventana de la consola de Linux o Terminal.

3. Almacenar el software dentro de una carpeta para poder tener todo lo necesario dentro de ella, lo recomendado es crear dicha carpeta en el escritorio de su máquina, como se muestra en la Figura 9:

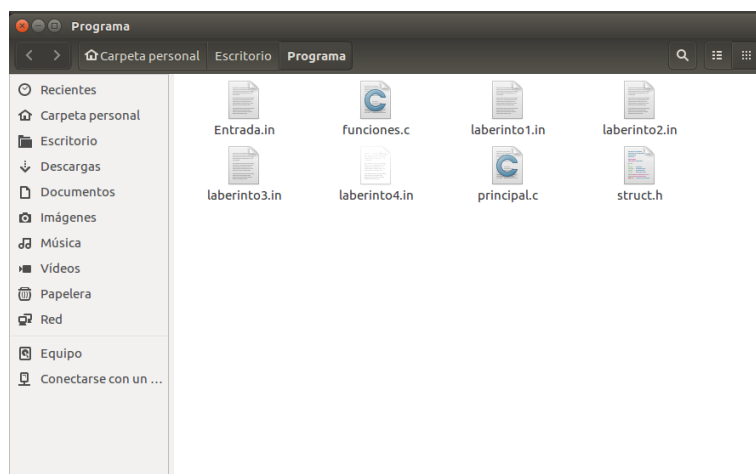
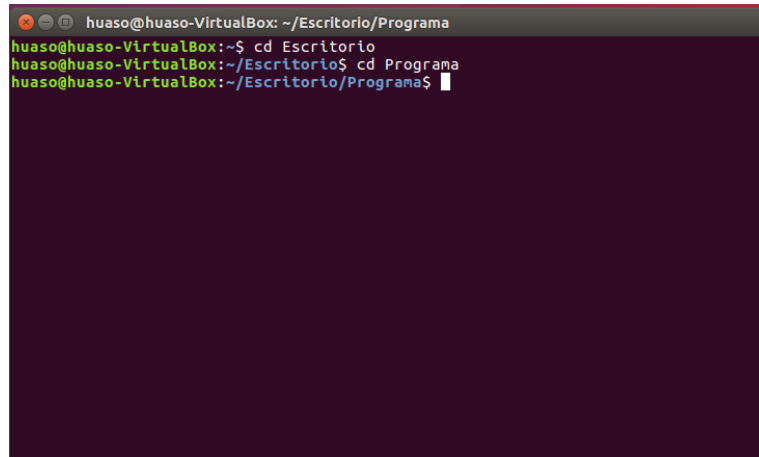


Figura 15: Carpeta con el software dentro.

4. Luego volvemos a la consola de Linux, para poder ejecutar el software como muestra en la Figura 10:

- a. Primer paso, dejamos la consola de Linux en la ruta donde tenemos almacenado nuestro programa, de la manera que muestra la siguiente imagen:

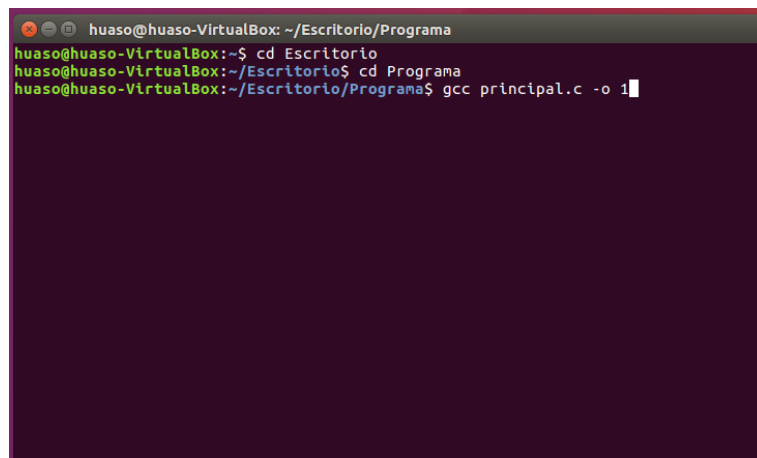


```
huaso@huaso-VirtualBox: ~/Escritorio/Programa
huaso@huaso-VirtualBox:~$ cd Escritorio
huaso@huaso-VirtualBox:~/Escritorio$ cd Programa
huaso@huaso-VirtualBox:~/Escritorio/Programa$
```

Figura 16: Consola de Linux con la ruta de la carpeta que contiene el software.

- b. Luego comenzamos la compilación, escribiendo lo siguiente en la consola de Linux:
- gcc (Nombre del archivo).c -o (Nombre del ejecutable)
 - ./(Nombre del ejecutable)

La Figura 11 entrega las instrucciones con mayor claridad:



```
huaso@huaso-VirtualBox: ~/Escritorio/Programa
huaso@huaso-VirtualBox:~$ cd Escritorio
huaso@huaso-VirtualBox:~/Escritorio$ cd Programa
huaso@huaso-VirtualBox:~/Escritorio/Programa$ gcc principal.c -o 1
```

Figura 17: Compilación de software.

- c. Luego se crea el archivo ejecutable en la carpeta donde tenemos almacenado el software, como muestra Figura 12:

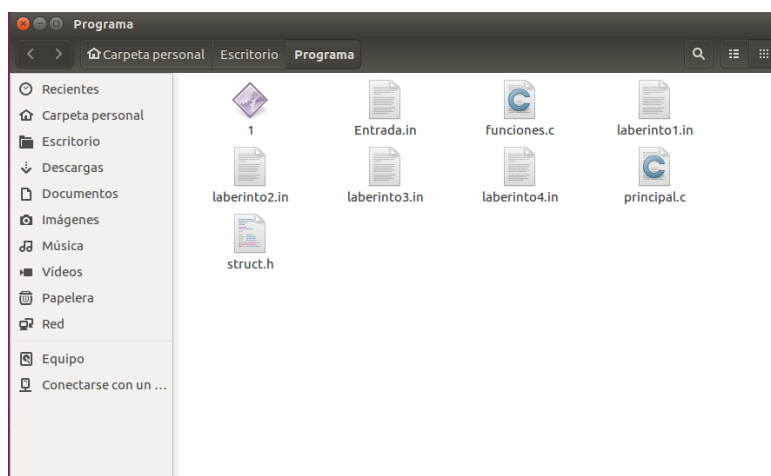


Figura 18: Carpeta con el archivo ejecutable

5.2.3 Ejecución

Teniendo el archivo “.exe” o “.out” en Windows y Linux respectivamente se procede a ejecutarlo cada uno en su ambiente.

Antes de llevar a cabo dicha acción se debe verificar que los caminos de los “Laberinto” estén en la misma carpeta donde se tiene almacenado dicho software, además de corroborar que los archivos donde se tiene almacenado cada uno de los caminos estén con la extensión.in.

5.2.3.1 Windows

1. Se vuelve a la consola de comandos de Windows, colocando la siguiente línea para efectuar la ejecución, en la Figura 13 se puede apreciar de mejor forma:

```

C:\Windows\system32\cmd.exe - 1.exe
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\krist>cd Desktop

C:\Users\krist\Desktop>cd Programa

C:\Users\krist\Desktop\Programa>gcc principal.c -o 1

C:\Users\krist\Desktop\Programa>1.exe

1. Cargar Tablero
2. Mostrar Tablero cargado
3. Buscar Llave
4. Buscar Salida
5. Creditos
6. Salir

Introduzca opcion (1-6): █

```

Figura 19: Software ejecutado.

- Al terminar el proceso anterior, se puede observar en la imagen anterior que el software está ejecutado y listo para ser manipulado.

5.2.3.2 Linux

1. Se vuelve a la consola de comandos de Linux, colocando la siguiente línea para efectuar la ejecución, en la Figura 14 se puede apreciar de mejor forma:

```

huaso@huaso-VirtualBox: ~/Escritorio/Programa
huaso@huaso-VirtualBox:~$ cd Escritorio
huaso@huaso-VirtualBox:~/Escritorio$ cd Programa
huaso@huaso-VirtualBox:~/Escritorio/Programa$ gcc principal.c -o 1
huaso@huaso-VirtualBox:~/Escritorio/Programa$ ./1

1. Cargar Tablero
2. Mostrar tablero cargado
3. Buscar Llave
4. Buscar Salida
5. Creditos
6. Salir

Introduzca opcion (1-6): █

```

Figura 20: Software ejecutado.

- Al terminar el proceso anterior, se puede observar en la imagen anterior que el software está ejecutado y listo para ser manipulado.

5.3 FUNCIONALIDADES

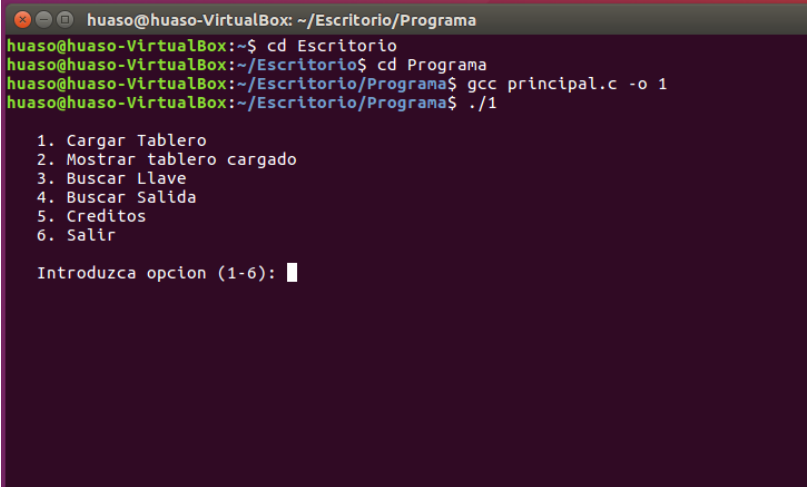
El software tiene la funcionalidad de encontrar el camino mínimo que se tiene para resolver el “Laberinto”, esto en dos oportunidades, primeramente, para encontrar el camino hacia la llave y a continuación hacia la salida del “Laberinto”.

El usuario tiene interacción con el software que recae en ingresar el nombre del “Laberinto” que se desea abrir, tener en cuenta que este archivo tendrá una entrada “.in”.

El resultado entregado por el software se verá expuesto en un archivo de salida con extensión “.out”.

A continuación, se realizará un procedimiento de los pasos a seguir para hacer uso del software de una buena manera:

- Primero debemos volver a la consola y ejecutar el software, entregando una ventana como muestra la Figura 15:



```
huaso@huaso-VirtualBox: ~/Escritorio/Programa
huaso@huaso-VirtualBox:~$ cd Escritorio
huaso@huaso-VirtualBox:~/Escritorio$ cd Programa
huaso@huaso-VirtualBox:~/Escritorio/Programa$ gcc principal.c -o 1
huaso@huaso-VirtualBox:~/Escritorio/Programa$ ./1

1. Cargar Tablero
2. Mostrar tablero cargado
3. Buscar Llave
4. Buscar Salida
5. Creditos
6. Salir

Introduzca opcion (1-6):
```

Figura 21: Menú principal del software.

- Luego de eso debemos seleccionar la opción de ingresar el nombre del tablero, como se muestra en la Figura 16:

```

huaso@huaso-VirtualBox: ~/Escritorio/Programa
huaso@huaso-VirtualBox:~/Escritorio/Programa$ gcc principal.c -o 1
huaso@huaso-VirtualBox:~/Escritorio/Programa$ ./1

1. Cargar Tablero
2. Mostrar tablero cargado
3. Buscar Llave
4. Buscar Salida
5. Creditos
6. Salir

Introduzca opcion (1-6): 1
Ingrese el nombre del archivo del laberinto Entrada.in

```

Figura 22: Cargando el archivo al software.

- Luego de eso debemos volver a ingresar una opción, en este caso ocuparemos la opción número 2 que nos muestra el tablero leído, se debe hacer como se muestra en la Figura 17:

```

1. Cargar Tablero
2. Mostrar tablero cargado
3. Buscar Llave
4. Buscar Salida
5. Creditos
6. Salir

Introduzca opcion (1-6): 2
*.*.*.*.*
*.S...K*
*.*****
*.*****
*...E*.*
*.*****
*.....*
*.*****
*...*.*
*.*.*.*.*
*.*.*.*.*

1. Cargar Tablero
2. Mostrar tablero cargado
3. Buscar Llave
4. Buscar Salida
5. Creditos
6. Salir

Introduzca opcion (1-6):

```

Figura 23: Mostrando el laberinto por pantalla.

- Luego debemos realizar los pasos 3 para buscar el camino hacia la llave y en seguida colocar 4 para buscar el camino hacia la salida, tener en cuenta que, si uno revisa el archivo creado antes de terminar las dos opciones, solo se verá el camino que solicito buscar, como se aprecia en la Figura 18:

```

1. Cargar Tablero
2. Mostrar tablero cargado
3. Buscar Llave
4. Buscar Salida
5. Creditos
6. Salir

Introduzca opcion (1-6): 3

1. Cargar Tablero
2. Mostrar tablero cargado
3. Buscar Llave
4. Buscar Salida
5. Creditos
6. Salir

Introduzca opcion (1-6): 4

1. Cargar Tablero
2. Mostrar tablero cargado
3. Buscar Llave
4. Buscar Salida
5. Creditos
6. Salir

Introduzca opcion (1-6): █

```

Figura 24: Buscando camino hacia la Llave y luego hacia la salida.

- Finalmente, para salir del software se selecciona la opción número 6, como muestra la Figura 19:

```

1. Cargar Tablero
2. Mostrar tablero cargado
3. Buscar Llave
4. Buscar Salida
5. Creditos
6. Salir

Introduzca opcion (1-6): 6
huaso@huaso-VirtualBox:~/Escritorio/Programa$ █

```

Figura 25: Cerrando el software.

- Luego el software se cierra y si nos dirigimos a la carpeta donde tenemos almacenado el programa no aparece el archivo con la salida, como se puede observar en la Figura 20:

5.4 EXITOS Y POSIBLES ERRORES

En ejecución la aplicación paso todas las pruebas que se le realizaron, tener en cuenta que fue sometida alrededor a distintos “Laberintos” teniendo siempre un resultado exitoso.

Algunos de los errores posibles que puede hacer que la aplicación son los siguientes:

- Cuando el “Laberinto” no cuenta con una salida, entrada o llave dentro de su mapa.
- Cuando el “Laberinto” no tiene un camino posible para poder llegar a uno de los objetivos mencionados en el punto anterior.
- Cuando el usuario modifica alguna línea del código planteado.
- Cuando las entradas no son las que el software espera.
- Cuando el disco se queda sin memoria.

En la Figura 22, se puede observar como el software deja de funcionar cuando se encuentra con uno de los errores mencionados:

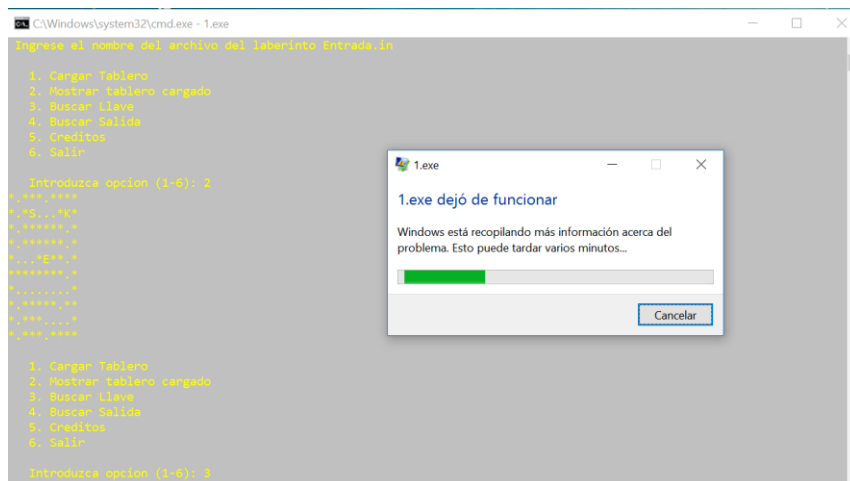


Figura 28: Posible error dentro de software.