Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

Programación bajo plataformas abiertas

Profesor: Juan Carlos Coto Ulate

Proyecto final: solución de laberinto en C

Edwar Jimenez Padilla B63653

Primer semestre del año 2022

Tabla de contenido

Introducción

Aunque en la actualidad existen diferentes leguajes de programación con diferentes utilidades y con interfaces e instrucciones más amigables y fáciles de entender, como es el lenguaje Python o R, lo cierto es que el lenguaje maquina si es muy complicado de tratar, y cando es requerida una comunicación más rápida con la máquina, es mejor utilizar lenguajes más antiguos y creados con estas capacidades como lo es el lenguaje C.

El objetivo del presente proyecto es, mediante la creación de un código solucionador de laberintos en el lenguaje C, poner en práctica los conocimientos presentados durante el curso, utilizando bibliotecas, funciones, variables, punteros, arreglos, memoria dinámica, archivos, entro otros conceptos, que permitan familiarizarse con la sintaxis de C mientras se desarrolla la lógica de programación, la cual será de mucha utilidad durante toda la vida universitaria y profesional.

Diseño general

El siguiente esquema describe superficialmente el código y las funciones más importantes del programa.

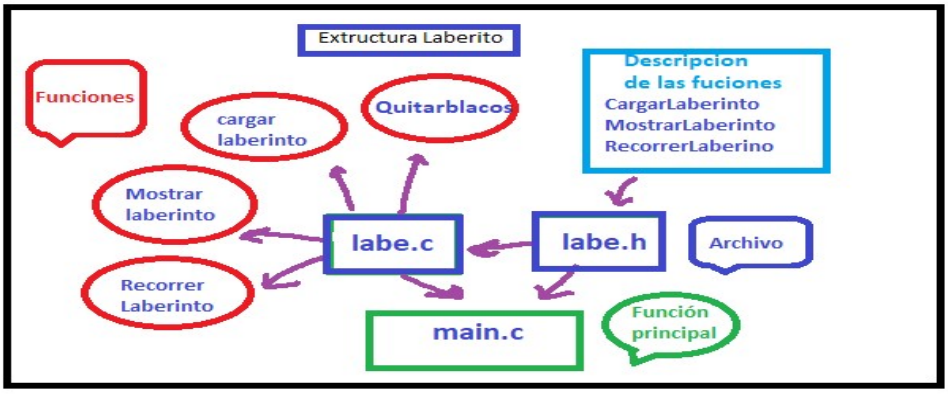

Figura 1: Esquema del código que debe resolver un laberinto.

Se implementan bibliotecas comunes de cómo, stdio.h, ctype.h, stdlib.h, string.h y unistd.h, además **labe.h**, que se crea como archivo junto a **labe.c,** pero solo contiene las descripciones de las funciones, **CargarLaberinto, MostrarLaberinto y RecorrerLaberinto**, el archivo **labe.c** además de contener el código de las tres funciones mencionas, en él se crea la función **quitarBlancos**, que sirve de apoyo para quitar los espacios blancos de la matriz que se debe leer. Las otras funciones se describen a continuación.

Como idea principal es necesario que el código lea un archivo en formato .txt que contiene una matriz con valores enteros, 0(pared),1(caminos) y 2(solución) y que al encontrar la solución imprima “solucionado”, en caso contrario se debe imprimir “no hay solución”

CargarLaberito:

Su función es leer la matriz, esto lo hace con la estructura de tipo **FILE** que abre el archivo con la función **fopen(),** luego elimina los espacios en blanco de la matriz , luego para determinar el tamaño de la matriz usando la función **mallo()** y **sizeof()**, para esto se vuelve a abrir el archivo para llenarlo con la información por medio del ciclo **while(){}** y el condicional **if()** y funciones como **feof(),fgetc()** y **isspace(),** luego se crea la matriz y se cierra.

MostrarLaberinto:

se encarga de mostrar el laberinto imprimiendo los valores enteros presentes en el documento **.txt**, esto lo hace con dos ciclos **for(){}** , uno de ellos anidado. Aunque esta función no es requerida para el presente proyecto es muy útil para saber cómo funciona el código durante el proceso de compilación.

RecorrerLaberinto:

se encarga de recorrer la matriz en busca de la solución, siguiendo las condiciones que son impuestas por las paredes, estas condiciones se establecen por medio del condicional **if(){}** he implementado la función MostrarLaberinto.

Main:

Esta es la función principal, en esta se lleva a cabo la ejecución del programa y se implementan las funciones presentes en el archivo labe.c por medio de labe.h. Aquí se define el archivo como una macro, y se crea el puntero labe que recibe la función CargarLaberinto luego de haber pasado por parámetros, el nombre del archivo, el número de filas y el número de columnas. Para después imprimir la matriz, ejecutar la función MostrarLaberinto y finalmente se establen y se recorren los bordes de la matriz buscando entradas validas, y en caso de encontrarla ejecuta la función RecorrerLaberito, para que se encargue de encontrar la solución en caso de que la haya.

Interacción con el Usuario:

Sobre la forma en la que el usuario interacciona con el programa, este solo debe ingresar un archivo con una matriz con nombre laberinto.txt, para que sea reconocida por el código o puede ir directamente al código y cambiar el nombre laberiento.txt por el nombre del laberinto que desea solucionar.

También quiero mencionar que en el código se trata de dar la mayor información de los elementos utilizados y sus funciones, con un propósito de recordar las distintas funciones de los elementos utilizados.

¿Por qué se decide crear el código de esta manera?

Se decide crear de esta manera para seguir las instrucciones dadas para este proyecto, con el propósito de practicar los conocimientos dados en clases, ejemplo de esto es la utilización de memoria dinámica o la separación del código en archivos, que aunque en códigos pequeños como los de este proyecto no tienen mucha diferencia en usarlos o no, pero lo importante es que el concepto se extrapola a situaciones mas complejas y presentes en la vida diaria como es el análisis de datos

u otras áreas que manejan mucha información y por ejemplo en este caso no es posible determinar el tamaño de una matriz, sino que es útil usar memoria dinámica para recibir cualquier tipo de matriz sin desperdiciar recursos computacionales.

Principales retos encontrados.

1. Durante el proceso de creación del código se aprende mucho y se desarrolla la lógica de programación, pero este un proceso complicado, de análisis y compresión basada mayormente en la metodología de prueba y error, entonces todo el proceso desde buscar las bibliotecas, funciones utilices, aplicación de conocimiento anteriores, aplicación de nuevos conocimientos es un gran reto.

2. Otro de los principales retos fue la implementación de memoria dinámica, ya que la función **malloc()** en matrices es más complicada de trabajar y de ahí pueden surgir distintos errores como los desbordes de memoria.

3. Se presentaron muchos problemas en cuanto a la elección del compilador, ya que en algunos el programa de compilación como “Dev-c++” el código compilaba bien, pero al intentar compilarlo en “Virtual Studio code” el código daba el error, “no encontrar el directorio” o en algunas ocasiones compilaba bien, pero imprimir nada.

d. Surgieron problemas debido a la capacidad de computo de mi equipo, esto porque al tratar de crear el código en Linux, algunas funciones diferían de las que se utilizan en Windows y al usar la maquina virtual para utilizar Debía, este comenzaba a trabar todo el sistema hasta el punto de tener que apagar el computador directamente, el problema tuvo solución gracias a la sugerencias del profesor de utilizar la pagina www.Replit.com para compilar el código sin necesidad de tener un sistema Linux.

e. Se presento el problema “segmentation fault(core dumped)” que, aun que es común puede ser provocado por diversas razones y es difícil determinar la solución. Este problema se solucionó, otra vez, gracias a una rápida revisión del profesor que determino que hacían falta algunos caracteres en el nombre del archivo que se quería leer.

f. El programa aun presenta algunos errores de compilación que al momento no soy capaz de resolver, entre estos está que el código solo envía que no hay solución cuando no hay entradas, en otro caso sigue buscando la solución, aunque no haya un camino valido, además, pienso, derivado de esto el código puede quedar atrapado en un bucle si se aleja mucho del camino de la solución, en cuyo caso no se detiene y por tanto no arroja ninguna solución.

Conclusiones

Como principal conclusión se tiene que la creación de código fue muy útil para introducirse en la lógica de programación, además de permitir poner en practica muchos de los conceptos aprendidos en clase.

Aunque el código se puede mejorar, es importante destacar que el proceso es de gran enseñanza y que da motivación para tratar de solucionar los problemas que aun presenta la compilación de este código o incluso adentrarse en otros proyectos más viciosos.

Preguntas que quedan abiertas para mí son las siguientes:

¿Cómo evitar que el código repita una trayectoria que ya reviso y que evitaría que la compilación se trabe en un ciclo?

¿Existen librerías en C o variables en C que permitan hacer el juego de una manera más eficiente tanto en código como uso de recursos computacionales?

¿Existe un compilador de C, que me permita compilar el programa por bloques o que brinde información específica de los errores que se generan, como es el caso de otros lenguajes como Python?

¿Por qué si el Lenguaje C++ es posterior al lenguaje C, y se creo para mayor simplicidad, se dice que es más extenso y difícil?

Algunas fuentes y enlaces consultadas

1. Juan.C.C.Ulate, Notas y video, Programacion bajo plataformas abiertas,2022.
2. Brian W. Kernighan, Dennis M. Ritchie. El lenguaje de programación C.Hall Hispanoamericana, 1991.
3. https://pablohaya.com/2013/10/12/diferencia-entre-scanf-gets-y-fgets.
4. www.replit.com