Universidad de Costa Rica

Estudiante: Geovanny Bonilla Espinoza

Carnet (B61141)

Programación bajo plataformas abiertas (IE0117)

Informe Técnico

Profesor: Juan Carlos Coto

Proyecto

Elaboración de Laberinto

2022

Índice

Contenido

Introducción

En la programación consiste en un extenso mundo de posibilidades, en las que se pueden desarrollar todo tipo de aplicaciones, todo esto mediante la implementación de código. Para lograr esto, es necesario dominar ciertos aspectos o temas dentro del lenguaje, el manejo de archivos, manejo de estructuras de decisiones, matrices y vectores. Logrando la compresión y manejo de estas estructuras básicas, las posibilidades son muy amplias para el desarrollo.

En este documento se realizará la explicación de como todas estas estructuras y aspectos nos pueden ayudar a realizar u programa con la función de leer un archivo, este contendrá una matriz de M x N de 0 y 1, simulando un laberinto. El programa deberá localizar la correcta solución en caso de existir. Esta explicación consistirá en la explicación del código empleado, además de sus distintas funciones cerradas para sus distintos usos dentro del funcionamiento del programa.

A parte se analizaran los principales retos encontrados a lo largo de la confección y programación de este programa, finalizando por las conclusiones finales que se obtuvieron con el desarrollo del programa.

Este documento servirá para poder entender mejor el funcionamiento de las estructuras y fundamentos de la programación, así como de generar una idea de cómo se pueden solucionar algunos problemas cotidianos con la lógica aplicada en este ejemplo.

Diseño General

El diseño de este programa se basó en la confección del archivo principal (main.c) y la elaboración de 3 archivos que le brindan las funciones a realizar:

Filacolumna.c

Borde.c

Solución.c

Estos archivos conforman la estructura del programa. En el archivo main.c se elaboró la estructura principal con las llamadas a las funciones.

```
int cont = 0;
int numColunmas = 0;
char matriz[100];
char matrizAux[100];

FILE *archivo;

archivo = fopen("./laberinto.txt", "r");


while(!feof(archivo)){
    fgets(informacion,100,archivo);
    cont ++;
}
rewind(archivo);

//Se leen las filas del archivo
for(int in = 0; !feof(archivo); in++){
    fscanf(archivo,"%s",&matriz[in]);
    matrizAux[in] = matriz[in];
}
rewind(archivo);

//Se leer el números de elementos y se divide entre las filas para obtener la columnas
while(!feof(archivo)) {
        fscanf(archivo,"%s",&matriz);
        numColunmas++;
     }
fclose(archivo);
```

En main se declara el archivo tipo “FILE” con el que se podrá abrir y leer el archivo “laberint.txt”, seguidamente se determinan el número de filas con la función “fgets”, se almacena los datos en una cadena auxiliar y mediante “fscanf” determinamos el número total de elementos del archivo leído.

```
//Se leen las filas del archivo
for(int in = 0; !feof(archivo); in++){
    fscanf(archivo,"%s",&matriz[in]);
    matrizAux[in] = matriz[in];
}
rewind(archivo);

//Se leer el números de elementos y se divide entre las filas para obtener la columnas
while(!feof(archivo)) {
        fscanf(archivo,"%s",&matriz);
        numColunmas++;
     }
fclose(archivo);

//calculos de columnas
numColunmas = numColunmas/cont;

//Asignación de filas y columnas de matriz
char Matriz[cont][numColunmas];
FilaColumna(cont, numColunmas, Matriz, matrizAux);

//Verifica bordes
if(determinaBorde(cont, numColunmas, Matriz) == 1){
    solucion(cont, numColunmas, Matriz);
}else{
    printf("No hay solucion");
}
```

Mediante la función “fscanf” y un contador se determina el número de elementos totales. Una vez determinado los números de elemento, este se divide entre el número de filas y se obtienen el número de columnas de la matriz.

Seguidamente se realizan los llamados de la funciones a utilizar, empezando por “FilaColumna”.

```
#include "filacolumna.h"

void FilaColumna(int cont, int numColunmas, char Matriz[cont][numColunmas], char *matrizAux){

    int contador = 0;
    int fila,columna;

    for( fila=0; fila<cont; fila++){
        for( columna=0; columna<numColunmas; columna++){
            Matriz[fila][columna]=  matrizAux[contador];
                contador++;
            }
    }
}
```

Esta función se encargará, mediante un bucle “for” de recorrer el arreglo con los datos y colocarlos en la posición correcta de la matriz.

Una vez definida la matriz con los datos de filas y columnas, se procede a enviarse a la función “determinaBorde”, esta se encargará de analizar la matriz y determinar si el camino empieza o termina en un borde de la matriz, esta función devuelve un int dependiendo del estado, “1” existe borde y “0” no existe borde. Esto determinará si se continúa con la siguiente función o no hay solución.

En caso de que el estado sea “1” se llamará a la función “Solucion”.

```
void solucion(int cont, int numColunmas, char Matriz[cont][numColunmas]){
    int fila,columna;
    int x,y;
    int solucion = 0;

    for( fila=0; fila<cont; fila++){
        for( columna=0; columna<numColunmas; columna++){
            if(Matriz[fila][columna] == '2'){
                x=fila;
                y=columna;
                solucion = buscaSolucion(x, y, cont, numColunmas, Matriz);
                if(solucion == 1){
                    fila = cont;
                    printf("Solucion (%d,%d)\n", x, y);
                }

            }
        }
    }

    if(solucion != 1){
        printf("No hay Solucion\n");
    }

}
```

Esta función recorre a la matriz en busca de la solución (“2”), cada vez que encuentre una solución, procederá a guardar su coordenada “X” y “Y”, estas las enviará a la función “buscaSolucion” la cual se encargará de recorrer la matriz mediante un ciclo “while” con condicionales “if” anidados, los cuales se encargaran de mover la coordenada en busca de valores “1”, esto para buscar el camino desde la solución hasta un borde. Si el camino está completo, entonces se retornará la variable fin, esta tendrá el estado “1” o “0” y con esto se decidirá si el laberinto posee o no solución.

```
buscaSolucion(int x, int y, int cont, int numColunmas, char Matriz[cont][numColunmas]){

int fin = 0;
int xmenos = 0;
int xmas = 0;
int ymenos = 0;
int ymas = 0;
int vacio = 0;

do{
        if(Matriz[x][y + 1] == '1' && ymenos == 0){
            ymas = 1;
            xmenos = 0;
            xmas = 0;
            y = y + 1;
        }else{
            if(Matriz[x + 1][y] == '1' && xmenos == 0){
                ymenos = 0;
                ymas = 0;
                xmas = 1;
                x = x + 1;
            }else{
                if(Matriz[x][y - 1] == '1' && ymas == 0){
                    ymenos = 1;
                    xmenos = 0;
                    xmas = 0;
                    y = y - 1;
                }else{
                    if(Matriz[x][y - 1] == '1' && ymas == 0){
                        ymenos = 1;
                        xmenos = 0;
                        xmas = 0;
                        y = y - 1;
                    }else{
                        if(Matriz[x - 1][y] == '1' && xmas == 0){
                                ymenos = 0;
                                ymas = 0;
                                xmenos = 1;
                                x = x - 1;
                        }else{
                            vacio = 1;
                        }
                    }
                }
            }
            if(x == 0 || x == cont){
                fin = 1;
            }
            if(y == 0 || y == numColunmas){
                fin = 1;
            }

    }while(vacio == 0);
    return fin;
}
```

Principales Retos

Determinar la dimensión de la matriz representó un reto debido a que no es posible la asignación dinámica de una matriz por lo que se tuvo que leer el archivo, pasar la información a una cadena y luego asignar cada elemento de la cadena en la posición (x,y) de la matriz.

Manejar el envío de matrices a funciones presentó algunos problemas ya que no solamente se debe enviar la matriz, esta debe ser enviada con sus dimensiones o punteros.

Transformar un arreglo a matriz se debió realizar con la ayuda de un contador para poder mover las posiciones del arreglo con los datos de la matriz.

Realizar la búsqueda del camino en la matriz significó un reto, muchos de los intentos por encontrar el camino resultó en un bucle infinito, por este motivo se tuvo que bloquear la búsqueda de la dirección contraria si el camino era recto, esto permitió que el ciclo no estuviera devolviéndose una y otra vez.

Poca agilidad del lenguaje de programación, C es un lenguaje que por sus años, es poco amigable con el programador, no ofrece tantas facilidades como otros lenguajes. Aspecto como el manojo de punteros o la asignación dinámica, resultaron un reto para el manejo correcto del programa.

Conclusiones

La elaboración de este programa, permitió poder comprender mejor el manejo que un lenguaje como C posee con las matrices, teniendo que definirlas de una manera un poco más compleja si no se tiene el dato exacto de la dimensión a utilizar. Además este programa permitió poder desarrollar una lógica al momento de tener que solucionar problemas similares, en donde el uso de matrices y el recorrido de estas se tengan que implementar.

A pesar de llegar a la solución del problema, surgen preguntas como si es posible poder asignar más fácilmente la asignación dinámica en una matriz o si se puede asignar el contenido de un archivo directamente en una matriz.

La elaboración de este tipo de ejercicios potencia la lógica y permiten poder entender como funciona el lenguaje, además de ayudar en facilitar futuros problemas a solucionar por medio de un programa.