UNIVERSIDAD DE COSTA RICA

Escuela de Ingeniería Eléctrica

IE0117: Programación Bajo Plataformas Abiertas

Reporte

Laboratorio

Prof. Carolina Trejos Quirós

Estudiante: David Abraham Vega Naranjo, C38344

17 de mayo de 2025

Índice

1. Introducción

En este laboratorio se llevara acabo diversas tareas orientas a formas de recorrer array bidimensionales por medio de aritmetica de punteros, usar punteres, asignar sus valores y extraer su direccion, y tambien el uso de bibliotecas nuevas nesacesarias para utilizar funciones como "rand", "srand", fopen, fgets, strtok, strlen, strcmp, strcpy y tolower.

Mi repositorio link.

2. Implementación

2.1. Ejercicio 1

2.1.1. Pseudocódigo y código

```
1
2   void findLargestLine(int **matrix, int size, int * result) {
3
4       for (int f = 0; f < size; f++) {
5           int newresult = 0;
6           for (int c = 0; c < size; c++) {
7               if (*(*(matrix + f) + c) == 1) {
8                   newresult++;
9                   if (newresult > *result) {
10                      *result = newresult;
11                  }
12              } else {
13                  newresult = 0;
14              }
15          }
16      }
17  }
18
19  // funcion void de buscar secuencia mas larga de unos
20  //      parametros:
21  //      (puntero a la matriz, tamano de la matriz, puntero longitud)
22      //  for filas desde 0 hasta size
23      //      definir contador de 1s en 0
24      //      for columnas desde 0 hasta size
25      //          si el puntero de elemento de la matriz es igual a 1
26      //              se aumenta el contador de 1s
27      //              si contador de 1s > puntero longitud
28      //                  puntero longitud = contador de 1s
29      //              sino
30      //                  se reinicia contador de 1s
31      //      finfor
32      //   finfor
33  // finfuncion
```

Para "findLargestLine" se resiven tres parametros necesarios, puntero a la matriz, un entero tamano de la matriz, y puntero donde se almacena la secuencia mas larga de 1s. Se hace un for que itera en cada fila y en cada columna, luego se hace una condicional donde se compara con aritmetica de punteros si el elemento de la matriz es igual a 1, conforme la esa codicional se realiza otra donde se verifica si el tamano anteriores encontrado es mayor que el se acaba de hacer en esa iteracion, haciendo esto por fila, se encuentra la secuencia de 1s mas extensa de la matriz.

```
    void allocateMatrix(int ***matrix, int size) {
        *matrix = malloc(size * sizeof(int*));
        for (int f = 0; f < size; f++) {
            (*matrix)[f] = malloc(size * sizeof(int));
        }
    }

    // funcion void para reservar la matriz en memoria dinamica.
    //      parametros:
    //      (direccion puntero a la matriz, tamano de la matriz)
    //      puntero matriz igual a malloc(tamano de la matriz * tamano en bytes
   de puntero a entero)
    //      for filas desde 0 hasta size
    //          puntero matriz[filas] igual a malloc(tamano de la matriz * tamano
    en bytes entero)
    //      finfor
    // finfuncion
```

La finalidad de "allocateMatrix" es resivir el puntero doble de la matriz y el tamano de la matriz ingresado por el usuario para con ayuda de la funcion "malloc" de la biblioteca "stdlib" reservar la memoria dinamica. Para cada fila un puntero a entero y para cada columna un entero.

```
    void fillMatrix(int **matrix, int size) {
        for (int f = 0; f < size; f++) {
            for (int c = 0; c < size; c++) {
                *(*(matrix + f) + c) = rand() % 2;
            }
        }
    }

    // funcion void para llenar la matriz con numeros aleatorios (0s y 1s)
    //      parametros:
    //      (puntero a la matriz, tamano de la matriz)
```

```
        //  for filas desde 0 hasta size
        //      for columnas desde 0 hasta size
    //              valor elemento = rand() con rango de 0 a 1
        //      finfor
        //   finfor
    // finfuncion
```

"fillMatrix" lo que realiza es iterar con aritmetica de punteros en cada espacio reservado en memoria para cada elemento de la matriz e ir rellenando con un entero entre 0 y 1 de manera aleatoria con ayuda de la funcion " rand".

```
    int main() {

    srand(time(NULL));

    int size, largestLine = 0;
    int **matrix = NULL;
    printf("Ingrese el tamano de la matriz cuadrada: ");
    scanf("%d", &size);

    allocateMatrix(&matrix, size);

    fillMatrix(matrix, size);

    printMatrix(matrix, size);

    findLargestLine(matrix, size, &largestLine);

    freeMatrix(matrix, size);

    printf("\nEl tamano de la secuencia de 1s mas grande es: %d\n", largestLine);
    return 0;

    }

    // inicio
    //      cambiar la "semilla" conforme el tiempo
    //      definir size y tamano de la secuencia en cero
    //      definir el puntero de la matriz como nulo
    //      solicitar al usuario el tamano de la matriz
    //      ejecutar la funcion de reservar memoria dinamica de la matriz
    //      ejecutar la funcion de llenar la matriz con 0s y 1s aleatorios
    //      ejecutar la funcion de imprimir matriz
    //      ejecutar la funcion de buscar la secuencia de 1s mas grande de cada
   fila.
    //      ejecutar la funcion para liberar memoria dinamica.
    //      imprimir la secuencia
    //      retornar 0
    // fin
```

En la funcion prtincipal "main" se define la "semilla" que ocupa "rand" para la aleatoriedad, y tambien el entero "size" y "largestLine" para el tamano de la matriz y longitud de 1s por fila, se define el puntero doble de la matriz como NULL, porque aun no esta en memoria.

Por ultimo se pide al usuario que ingrese el tamano de la matriz para asi ejecutar las funciones explicadas anteriormente y imprimir la longitud de unos mas larga de todas las filas.

2.2. Ejercicio 2

2.2.1. Pseudocódigo

```
// inicio
//      abrir archivo en modo lectura con "fopen"
//      si "fopen" retorna diferente a "NULL"
//          definir char "lineas" tamano 100
//          definir char "palabra_invertida" tamano 100
//          definir char "palindromo" tamano 100
//          definir tamano_palindromo igual a 0
//          while la linea sea diferente de "null"
//              separar las lineas por palabras sin signos
//              while la palabra sea diferente a "null"
//                  for cada caracter de la palabra hasta "\0"
//                      caracter igual a "tolower(palabra)"
//              //      finfor
//                  for cada indice desde 0 hasta "strlen(palabra)"
//                      palabra_invertida[indice] = palabra[strlen(palabra)-1 - indice]
//              //      finfor
//                  definir ultimo caracter de palabra_invertida igual a "\0"
//                  si ("palabra" y "palabra_invertida" son igual) y (tamano_palabra > tamano_palindromo)
//                      tamano_palindromo es igual a tamano_palabra
//                      palindromo es igual a palabra
//          cerrar archivo
//          imprimir palindromo
//      sino error al abrir archivo
// fin
```

2.2.2. Código

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main() {
    FILE *archivo = fopen("input.txt", "r");
    if (archivo != NULL) {
```

```
        char lineas[100];
        char palabra_invertida[100];
        char palindromo[100];
        int tamano_palindromo = 0;

        while (fgets(lineas, sizeof(lineas), archivo) != NULL) {

            char *palabra = strtok(lineas, ".,;:¡!¿?()[]{}\"'«»---_\\/|@#$%^&*~‘+=<> ");

            while (palabra != NULL) {

                for (int c = 0; palabra[c] != '\0'; c++) {
                    palabra[c] = tolower(palabra[c]);
                }

                for (int i = 0; i < strlen(palabra); i++) {
                    palabra_invertida[i] = palabra[strlen(palabra) - 1 - i];
                }

                palabra_invertida[strlen(palabra)] = '\0';

                if (strcmp(palabra, palabra_invertida) == 0 && strlen(palabra) >
                    tamano_palindromo) {

                        tamano_palindromo = strlen(palabra);
                        strcpy(palindromo, palabra);

                }
                palabra = strtok(NULL, ".,;:¡!¿?()[]{}\"'«»---_\\/|@#$%^&*~‘+=<> ");
            }

        }

        fclose(archivo);
        printf("El palíndromo mas largo encontrado es: '%s'\n", palindromo);

    } else {
        printf("Error al abrir el archivo.\n");
    }
    return 0;
}
```

2.2.3. Explicacion

Para este programa primero es necesario agregar las bibliotecas como "string" para manejo de archivos y cadenas, y "ctype" para caracteres. Dentro de "main" se abre el archivo "input.txt" con

"fopen" en modo lectura, luego se definen una serie de variables tipo
char y int necesarias mas adelante.

Seguidamente con el uso de while se lee el archivo linea por linea
usando "fgets", cada linea se divide en palabras con "strtok"
evitando signos de puntuacion y con "tolower" cada caracter de la
palabra se pasa a minuscula.

Después se invierte la palabra de esa iteracion y se copia a "palabra_invertida"
agregandole como ultimo caracter "\0" para que sea valida como cadena. Una
vez invertida se compara la palabra invertida con la palabra original
con ayuda de "strcmp", para poder condicionar que si coinciden las cadenas
y también es la cadena mas larga guardada hasta el momento, "palidromo" pasaria
a ser "palabra" copiando la cadena con "strcpy". Por ultimo luego de leer
todo el texto se cierra con "fclose" verficando que no haya errrores.

3. Resultados

3.1. Ejercicio 1

3.1.1. Ejemplos y Salidas

```
lincoln@lincoln-pc:~/Desktop/C$ valgrind --leak-check=full ./4ej1
==2861== Memcheck, a memory error detector
==2861== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==2861== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright i
==2861== Command: ./4ej1
==2861==
Ingrese el tamano de la matriz cuadrada: 8
Matrix (8x8):
1 1 0 0 0 1 1 0
1 1 0 1 0 0 0 1
0 1 1 0 0 1 1 1
1 0 0 1 0 0 1 1
0 0 1 0 1 0 0 0
1 0 1 1 0 0 1 1
1 0 1 1 1 0 1 0
1 1 0 1 0 1 0 0

El tamano de la secuencia de 1s mas grande es: 3
==2861==
==2861== HEAP SUMMARY:
==2861==     in use at exit: 0 bytes in 0 blocks
==2861==   total heap usage: 11 allocs, 11 frees, 2,368 bytes allocated
==2861==
==2861== All heap blocks were freed -- no leaks are possible
==2861==
==2861== For lists of detected and suppressed errors, rerun with: -s
==2861== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Figura 1: Salida Valgrind

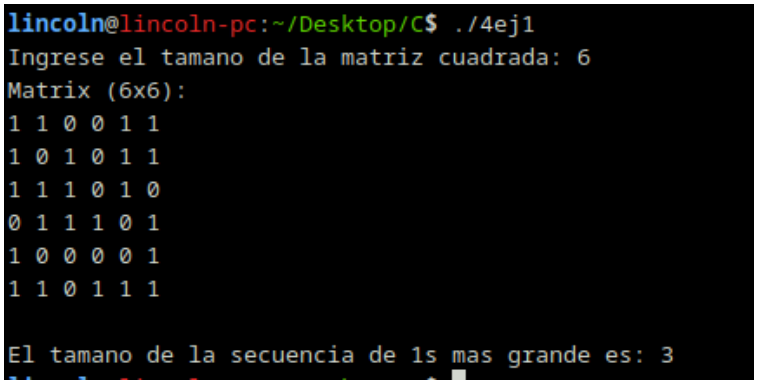

Figura 2: Ejemplo de matriz 6x6

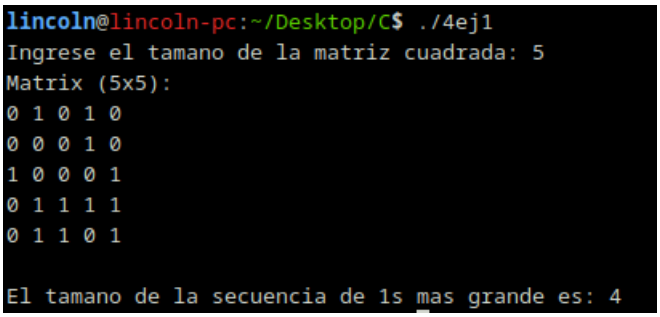

Figura 3: Ejemplo de matriz 5x5

La salida de "valgrind" se ve que "malloc" uso memoria dinamica, pero se libero correctamente con ayuda de "free". Además "==2861== All heap blocks were freed -- no leaks are possible" indica que no hubo memory leaks. Gracias a los ejemplos proporcionados se corrobora que el programa cumple con su objetivo imprimendo la secuencia mas larga de 1s de todas las filas.

3.2. Ejercicio 2

3.2.1. Salidas

```
Ana, casa Otto; radar. perro oso! kayak? solos:
Bob–menem(ara) Salas[reconocer]{oro}
Ava#stats@non~civic*dud&madam=wow+noon%level
/rotor\refer|deed¡peep¿mum^pop<tit>gig`nun'abba"
eve«eye»wow_sagas-tenet—minim gato mesa sol
cielo agua río botón lugar café noche libro reloj.

Palindromos:

Ana, Otto, radar, oso, kayak, solos, Bob, menem
ara, Salas, reconocer, oro, Ava, stats, non, civic
dud, madam, wow, noon, level, rotor, refer, deed, peep
mum, pop, tit, gig, nun, abba, eve, eye, sagas, tenet, minim
```

Figura 4: input.txt

Texto de ejemplo para verificar si "strtok" esta separando las palabras correctamente sin signos de puntuacion y otros varios, ademas pasar las palabras a minusculas y ver si encuentra las palabras palindromas.

```
lincoln@lincoln-pc:~/Desktop/C$ gcc -Wall 4ejercicio2.c -o 4ej2
lincoln@lincoln-pc:~/Desktop/C$ valgrind --leak-check=full ./4ej2
==2961== Memcheck, a memory error detector
==2961== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==2961== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==2961== Command: ./4ej2
==2961==
El palíndromo mas largo encontrado: reconocer==2961==
==2961== HEAP SUMMARY:
==2961==     in use at exit: 0 bytes in 0 blocks
==2961==   total heap usage: 3 allocs, 3 frees, 5,592 bytes allocated
==2961==
==2961== All heap blocks were freed -- no leaks are possible
==2961==
==2961== For lists of detected and suppressed errors, rerun with: -s
==2961== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Figura 5: Salida Valgrind

Salida de "valgrind" donde se ve que "fopen, fgets, strtok, strlen, strcmp, strcpy, tolower" usaron memoria dinamica, pero se libero correctamente. Además "==2961== All heap blocks were freed -- no leaks are possible" indica que no hubo memory leaks.

```
lincoln@lincoln-pc:~/Desktop/C$ ./4ej2
ana, otto, radar, oso, kayak, solos,
, bob, menem, ara, salas, reconocer, oro,
, ava, stats, non, civic, dud, madam, wow, noon, rotor, refer, deed, peep, mum, pop, tit, gig, nun, abba,
, eve, eye, wow, sagas, tenet,
```

Figura 6: Salida de todos los palíndromos

Esta figura indica que el programa es capaz de dividir lineas en palabras, separarlas sin ningun tipo de signo, y distinguir cuales palabras son palindromos y cuales no.

```
lincoln@lincoln-pc:~/Desktop/C$ gcc -Wall 4ejercicio2.c -o 4ej2
lincoln@lincoln-pc:~/Desktop/C$ ./4ej2
El palíndromo mas largo encontrado es: 'reconocer'
```

Figura 7: Salida final

Con esta salida del codigo se verifica por ultimo que cumple con encontrar el palindromo mas extenso.

4. Conclusiones y recomendaciones

Gracias a la implementación de cada uno de los programas anteriores, pude aprender una buena base sobre cómo recorrer los arrays con aritmetica de punteros, tambien lo util que fueron los punteros, al hallar la direccion de las variables y asi no usar variables globales.

Ademas el uso de funciones para reservar la memoria dinamica y como liberarla. En esta ocasion tambien se utilizó la función "rand" y "srand" para generar matrices aleatorias, y c funciones para trabajar con archivos y cadenas de textos.

Referencias


```
@manual{ibmRand,
  author       = {IBM},
  title        = {rand(), rand\_r() - Generate a random number},
  year         = {s.f.},
  url          = {https://www.ibm.com/docs/es/i/7.5.0?topic=functions-rand-rand-r-generate-
  note         = {Consultado en mayo de 2025}
}

@manual{ibmSrand,
  author       = {IBM},
  title        = {srand() - Set the seed for the rand() function},
  year         = {s.f.},
  url          = {https://www.ibm.com/docs/es/i/7.5.0?topic=functions-srand-set-seed-rand-fu
  note         = {Consultado en mayo de 2025}
}


@manual{ibmStrtok,
  author       = {IBM},
  title        = {strtok() - Tokenize a string},
  year         = {s.f.},
  url          = {https://www.ibm.com/docs/es/i/7.5.0?topic=functions-strtok-tokenize-string
  note         = {Consultado en mayo de 2025}
}


@manual{ibmTolower,
  author       = {IBM},
  title        = {tolower(), toupper() - Convert character case},
  year         = {s.f.},
  url          = {https://www.ibm.com/docs/es/i/7.5.0?topic=functions-tolower-toupper-conver
  note         = {Consultado en mayo de 2025}
}


@manual{ibmStrcmp,
  author       = {IBM},
  title        = {strcmp() - Compare strings},
  year         = {s.f.},
  url          = {https://www.ibm.com/docs/es/i/7.5.0?topic=functions-strcmp-compare-strings
  note         = {Consultado en mayo de 2025}
}
```

```
@manual{ibmStrcpy,
  author       = {IBM},
  title        = {strcpy() - Copy strings},
  year         = {s.f.},
  url          = {https://www.ibm.com/docs/es/i/7.5.0?topic=functions-strcpy-copy-strings#st
  note         = {Consultado en mayo de 2025}
}
```