EIE

Escuela de
Ingeniería Eléctrica

UNIVERSIDAD DE COSTA RICA

UNIVERSIDAD DE COSTA RICA

FACULTAD DE INGENIERIA

ESCUELA DE INGENIERIA ELÉCTRICA

PROGRAMACIÓN BAJO PLATAFORMAS ABIERTAS

LABORATORIO 4:
MATRICES CON MEMORIA DINÁMICA

ESTUDIANTE:
JESÚS ZÚÑIGA MÉNDEZ

PROFESOR:
RICARDO ROMÁN BRENES; M. SC.

I CICLO 2019

Índice

1. **Introducción** **2**

2. **matriz.h** **3**

3. **matriz.c** **3**

4. **tools.c** **6**

5. **main.c** **7**

6. **Conclusión** **12**

Índice de figuras

1. Introducción

Para este laboratorio el objetivo fue crear un programa de operaciones de matrices con el fin de aprender a utilizar memoria dinámica, a continuación se adjunta el código escrito en C con el que se soluciono el problema, en el código se encuentra la documentación necesaria para comprender que lógica siguen las funciones presentes en el programa.

2. matriz.h

```
/**
 * @file matriz.h
 * @brief Archivo que contiene lo headers de las funciones
 */
#include <stdlib.h>
#include <stdio.h>
//tools.c
void Bienvenido();
int Menu();
//matriz.c
void ImprimirMatriz(float**,int , int );
float** LlenarMatriz(float** ,int , int );
float** CrearMatriz (int, int);
float** LimpiarMatriz(float** ,int , int );
float** SumarMatrices(float** ,float** ,int , int );
float** MultiplicarMatrices(float** ,float** ,int , int ,int ,int);
float** SumarEscalar(float** ,int , int ,float );
float** MultiplicarEscalar(float** ,int , int ,float );
float** Transpuesta(float** ,int , int );
```

3. matriz.c

```
/**
 * @file matriz.c
 * @brief Archivo que contiene las funciones relacionadas a las matrices
 */
#include "../incl/matriz.h"
/**
 * @brief Funcion que crea una matriz del tamaño deseado por el usuario
 * @param filas es el tamaño de filas de la matriz
 * @param columnas es el tamaño de columnas de la matriz
 * @return devuelve un puntero con la matriz creada
 */
float** CrearMatriz (int filas, int columnas){
    //creamos el puntero con "forma de matriz" y le asignamos el tamaño a las filas
    float** matriz = (float**) malloc(filas * sizeof(float*));
    //este ciclo toma el puntero lo "recorre" por filas y
    //le asigna el tamaño de las columnas
    for (int i = 0; i < filas; i++){
        *(matriz+i) = (float*) malloc(columnas * sizeof(float));
    }
    return matriz;
}
/**
 * @brief Funcion que permite imprimir una matriz
 * @param recibe un puntero a la matriz el numero de filas y de columnas
 * @return no devuelve ningun valor
 */
void ImprimirMatriz(float** matriz,int filas, int columnas){
    float numero = 0;
    printf("\n\n\n");
```

```
    for(int f = 0; f < filas; f++){
        for(int c = 0; c < columnas; c++){
            numero = *(*(matriz+f)+c);
            printf("%f     ",numero);
        }
    printf("\n");
    }
}
/**
 * @brief Funcion que permite llenar una matriz
 * @param recibe un puntero a la matriz el numero de filas y de columnas
 * @return devuelve la matriz llena
 */
float** LlenarMatriz(float** matriz,int filas, int columnas){
    float numero = 0;
    int s = 0;
    for(int f = 0; f < filas; f++){
        for(int c = 0; c < columnas; c++){
            printf("Digite la fila %d, columna %d\n",(f+1),(c+1));
            s = scanf("%f",&numero);
            *(*(matriz+f)+c) = numero;
        }
    }
    return matriz;
}
/**
 * @brief Funcion que permite limpiar una matriz, es decir llenarla de ceros
 * @param recibe un puntero a la matriz el numero de filas y de columnas
 * @return devuelve la mnatriz llena
 */
float** LimpiarMatriz(float** matriz,int filas, int columnas){
    float numero = 0;
    for(int f = 0; f < filas; f++){
        for(int c = 0; c < columnas; c++){
            *(*(matriz+f)+c) = numero;
        }
    }
    return matriz;
}
/**
 * @brief Funcion que permite sumar dos matrices
 * @param recibe un puntero a cada una de las matrices y el
 //tamaño en filas y columnas del mismo
 * @return devuelve la mnatriz sumada
 */
float** SumarMatrices(float** matrizA,float** matrizB,int filas, int columnas){
    float numero = 0;
    float** matrizResultado = matrizA;
    for(int f = 0; f < filas; f++){
        for(int c = 0; c < columnas; c++){
            numero = *(*(matrizA+f)+c) + *(*(matrizB+f)+c);
            *(*(matrizResultado+f)+c) = numero;
        }
    }
```

```
    return matrizResultado;
}
/**
 * @brief Funcion que permite multiplicar dos matrices
 * @param recibe un puntero a cada una de las matrices y el
 //tamaño en filas y columnas del mismo
 * @return devuelve la mnatriz multiplicada
 */
float** MultiplicarMatrices(float** matrizA,float** matrizB,int filasA, int columnasA,int
    float numero = 0;
    int filasR = filasA;
    int columnasR = columnasB;
    float** matrizResultado = CrearMatriz(filasR,columnasR);
    float** filaMatrizA = CrearMatriz(1,columnasA);
    float** columnaMatrizB = CrearMatriz(1,columnasA);
    float** filaMatrizR = CrearMatriz(1,columnasR);
    //se hace el calculo de lo que va en la entrada correspondiente de la mariz resultado
    for(int fmA = 0; fmA < filasA; fmA++){
        for(int cmA = 0; cmA < columnasA; cmA++){
            //llenamos el "vector" con la fila de la matriz A con la que vamos a trabajar
            *(*(filaMatrizA)+cmA) = *(*(matrizA+fmA)+cmA);
        }
        for (int cmB = 0; cmB < columnasB; cmB++){
            //recorremos la matriz b por columnas para ir sacando cada columna
            for (int fmB = 0; fmB < filasB; fmB++){
                //recorremos la matriz b por filas  sacando cada columna
                *(*(columnaMatrizB)+fmB) = *(*(matrizB+fmB)+cmB);
            }
            //despues de tener la fila y la columna para trabajar calculamos la
            //entrada de la nueva matriz
            for (int eV = 0; eV < columnasA; eV++){
                numero = numero + (*(*(filaMatrizA)+eV) * *(*(columnaMatrizB)+eV));
            }
            //asignamos el valor de numero al "vector" que tendra
            // los datos de cadafila de la matriz resultado y limpiamos
            *(*(filaMatrizR)+cmB) = numero;
            numero = 0;
        }
        //asignamos lo calculado a la matriz resultado
        for (int cmR=0; cmR < columnasR; cmR++){
            *(*(matrizResultado+fmA)+cmR) = *(*(filaMatrizR)+cmR);
        }
    }
    free(filaMatrizA);
    free(filaMatrizR);
    free(columnaMatrizB);
    return matrizResultado;
}
/**
 * @brief Funcion que suma un escalar a ua matriz
 * @param recibe un puntero a la matriz y el escalar
 * @return devuelve la mnatriz sumada
 */
float** SumarEscalar(float** matriz,int filas, int columnas,float escalar){
```

```
    float** matrizResultado = matriz;
    for(int f = 0; f < filas; f++){
        for(int c = 0; c < columnas; c++){
            *(*(matrizResultado+f)+c) = *(*(matrizResultado+f)+c) + escalar;
        }
    }
    return matrizResultado;
}
/**
 * @brief Funcion que multiplica un escalar a ua matriz
 * @param recibe un puntero a la matriz y el escalar
 * @return devuelve la mnatriz sumada
 */
float** MultiplicarEscalar(float** matriz,int filas, int columnas,float escalar){
    float** matrizResultado = matriz;
    for(int f = 0; f < filas; f++){
        for(int c = 0; c < columnas; c++){
            *(*(matrizResultado+f)+c) = *(*(matrizResultado+f)+c) * escalar;
        }
    }
    return matrizResultado;
}
/**
 * @brief Funcion que transpone una matriz
 * @param recibe un puntero a la matriz, las filas y las columnas
 * @return devuelve la mnatriz traspuesta
 */
float** Transpuesta(float** matriz,int filas, int columnas){
    int filasR = columnas;
    int columnasR = filas;
    float** matrizResultado = CrearMatriz(filasR,columnasR);
    for(int f = 0; f < filas; f++){
        for(int c = 0; c < columnas; c++){
            *(*(matrizResultado+c)+f) = *(*(matriz+f)+c);
        }
    }
    return matrizResultado;
}
```

4. tools.c

```
/**
 * @file tools.c
 * @brief Archivo que contiene las funciones a las cuales invoca main.c
 */
# include "../incl/matriz.h"
/**
 * @brief Funcion que imprime el rotulo de bienvenida
 * @param no recibe parametros
 * @return no devuelve ningun valor
 */
void Bienvenido(){
    printf("  ____   _                              _      _          _  _  _ \n |  _ \\
```

```
    printf("\n");
    printf("\n");
}
/**
 * @brief Funcion que imprime el menu
 * @param no recibe parametros
 * @return devuelve la opcion escogida por el usuario
 */
int Menu(){
    printf("Digite el numero de la opcion que quiere realizar.");
    printf("3. Borrar una matriz.\t\n");
    printf("4. Sumar matrices.\t\n");
    printf("5. Multiplicar matrices.\t\n");
    printf("6. Sumar escalar y matriz.\t\n");
    printf("7. Multiplicar escalar y matriz.\t\n");
    printf("8. Transponer una matriz.\t\n");
    printf("9. Imprimir una matriz.\t\n");
    printf("0. SALIR.\t\n\n\n");
    int seleccion = 0;
    int s = scanf("%d",&seleccion);
    return seleccion;
}
```

5. main.c

```
/**
 * @file main.c
 * @author Jesus Zuñiga Mendez
 * @brief Archivo pricipal, laboratorio que trata sobrememoria dinamica para la creacion de
 * @version 1
 * @date 27 de mayo de 2019
 * @copyright Copyleft (l) 2019
 */
#include "./incl/matriz.h"
/**
 * @brief Funcion pricipal donde se llmaran las funciones creadas en matriz.c
 * @param argc cantidad de argumentos enviados por teclado
 * @param argv puntero a los argumentos enviados por teclado
 * @return devuelve 0 al finalizar el programa
 */
int main(int argc, char** argv)
{
    int s = 0;
    int seleccion = 0;
    int temporal = 0;
    int filasA;
    int columnasA;
    int filasB;
    int columnasB;
    int filasR;
    int columnasR;
    float** matrizA = NULL;
    float** matrizB = NULL;
```

```
float** matrizResultado = NULL;
int existeA = 0;
int existeB = 0;
int AB = 0;
float escalar = 0;
Bienvenido();
do{
    printf("\n\n");
    seleccion = Menu();
    switch (seleccion)
    {
    case 1:
        printf("Seleccione la matriz que desea crear\n\t Digite:\n\t\t1 para matriz A\n
        s = scanf("%d",&AB);
        if (AB == 1){
            printf("Digite el numero de Filas\n");
            int s = scanf("%d",&filasA);
            printf("Digite el numero de Columnas\n");
            s = scanf("%d",&columnasA);
            if ((filasA ==0) || (columnasA ==0)){
                printf("Error algun dato es cero");
            }else{
                matrizA = CrearMatriz(filasA, columnasA);
            }
            if (matrizA == NULL){
                printf("Error al crear la matriz A\n");
            }else{
                printf("Matriz A creada con exito\nLLene los campos de la matriz\n\n");
                existeA = 1;
                matrizA = LlenarMatriz(matrizA,filasA,columnasA);
            }
        }else if (AB == 2){
            printf("Digite el numero de Filas\n");
            int s = scanf("%d",&filasB);
            printf("Digite el numero de Columnas\n");
            s = scanf("%d",&columnasB);
            if ((filasB ==0) || (columnasB ==0)){
                printf("Error algun dato es cero");
            }else{
                matrizB = CrearMatriz(filasB, columnasB);
            }
            if (matrizB == NULL){
                printf("Error al crear la matriz A\n");
            }else{
                printf("Matriz B creada con exito\nLLene los campos de la matriz\n\n");
                existeB = 1;
                matrizB = LlenarMatriz(matrizB,filasB,columnasB);
            }
        }
        break;
    case 2:
        printf("Seleccione la matriz que desea Limpiar\n\t Digite:\n\t\t1 para matriz A
        s = scanf("%d",&AB);
        if ((AB == 1) && (existeA == 1)){
```

```
            matrizA = LimpiarMatriz(matrizA,filasA,columnasA);

    }else if ((AB == 2) && (existeB == 1)){
            matrizB = LimpiarMatriz(matrizB,filasB,columnasB);
    }
    break;
case 3:
    printf("Seleccione la matriz que desea Borrar\n\t Digite:\n\t\t1 para matriz A\
    s = scanf("%d",&AB);
    if ((AB == 1) && (existeA == 1)){
            free(matrizA);
            filasA = 0;
            columnasA = 0;
            existeA = 0;

    }else if ((AB == 2) && (existeB == 1)){
            free(matrizB);
            filasB = 0;
            columnasB = 0;
            existeB = 0;
    }
    break;
case 4:
    if((filasA == filasB) && (columnasA == columnasB) && (existeA ==1) && (existeB
        matrizResultado = SumarMatrices(matrizA,matrizB,filasA,columnasA);
        ImprimirMatriz(matrizResultado,filasA,columnasA);
        free(matrizResultado);
    }else{
        printf("Imposible sumar matrices de diferentes dimensiones");
    }
    break;
case 5:
    if((columnasA == filasB) && (existeA == 1) && (existeB == 1)){
        matrizResultado = MultiplicarMatrices(matrizA,matrizB,filasA,columnasA,fila
        ImprimirMatriz(matrizResultado,filasA,columnasB);
        free(matrizResultado);
    }else{
        printf("Error en la dimension, las matrices deben ser m*N y N*c");
    }
    break;
case 6:
    printf("Seleccione la matriz a la que desea sumar\n\t Digite:\n\t\t1 para matri
    s = scanf("%d",&AB);
    if ((AB == 1) && (existeA == 1)){
        printf("Digite el escalar\n");
        s = scanf("%f",&escalar);
        matrizA = SumarEscalar(matrizA,filasA,columnasA,escalar);
        ImprimirMatriz(matrizA,filasA,columnasA);
    }else if ((AB == 2) && (existeB == 1)){
        printf("Digite el escalar\n");
        s = scanf("%f",&escalar);
        matrizB = SumarEscalar(matrizB,filasB,columnasB,escalar);
        ImprimirMatriz(matrizB,filasB,columnasB);
    }
```

```
            break;
        case 7:
            printf("Seleccione la matriz a la que desea multiplicar\n\t Digite:\n\t\t1 para
            s = scanf("%d",&AB);
            if ((AB == 1) && (existeA == 1)){
                printf("Digite el escalar\n");
                s = scanf("%f",&escalar);
                matrizA = MultiplicarEscalar(matrizA,filasA,columnasA,escalar);
                ImprimirMatriz(matrizA,filasA,columnasA);
            }else if ((AB == 2) && (existeB == 1)){
                printf("Digite el escalar\n");
                s = scanf("%f",&escalar);
                matrizB = MultiplicarEscalar(matrizB,filasB,columnasB,escalar);
                ImprimirMatriz(matrizB,filasB,columnasB);
            }
            break;
        case 8:
            printf("Seleccione la matriz que desea Trasponer\n\t Digite:\n\t\t1 para matriz
            s = scanf("%d",&AB);
            if ((AB == 1) && (existeA == 1)){
                matrizA = Transpuesta(matrizA,filasA,columnasA);
                temporal = filasA;
                filasA = columnasA;
                columnasA = temporal;
                ImprimirMatriz(matrizA,filasA,columnasA);
            }else if ((AB == 2) && (existeB == 1)){
                matrizB = Transpuesta(matrizB,filasB,columnasB);
                temporal = filasB;
                filasB = columnasB;
                columnasB = temporal;
                ImprimirMatriz(matrizB,filasB,columnasB);
            }
            break;
        case 9:
            printf("Seleccione la matriz que desea imprimir\n\t Digite:\n\t\t1 para matriz
            s = scanf("%d",&AB);
            if ((AB == 1) && (existeA == 1)){
                ImprimirMatriz(matrizA,filasA,columnasA);
            }else if ((AB == 2) && (existeB == 1)){
                ImprimirMatriz(matrizB,filasB,columnasB);
            }
            break;
        default:
            seleccion = 0;
            break;
        }
    }while (seleccion != 0);
/*
    int* matriz = MCrearMatriz(1,1);
    /*matriz[0] = 0;
    matriz[1] = 1;
    matriz[2] = 1;
    matriz[3] = 1;
    printf("tamaño: %ld",sizeof(matriz));
```

```
    printf("%d\n",matriz[0]);
    printf("%d\n",matriz[3]);
*/
    return 0;
}
```

6. Conclusión

Se concluye que el uso de memoria dinámica es sumamente útil para trabajar los datos, aunque al principio el uso de esta puede resultar algo abstracto después de usarlo algún tiempo se puede aclarar mejor el uso de las misma y se logra entender la importancia que estos tienen