**UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE**

**DATOS PERSONALES**

**Apellidos y nombres:** Morales Sosa Jeimy Marley

**Apellidos y nombres:** Orrico Emme Camilo Andrés

**Id**: L00418832

**Id**: L00418282

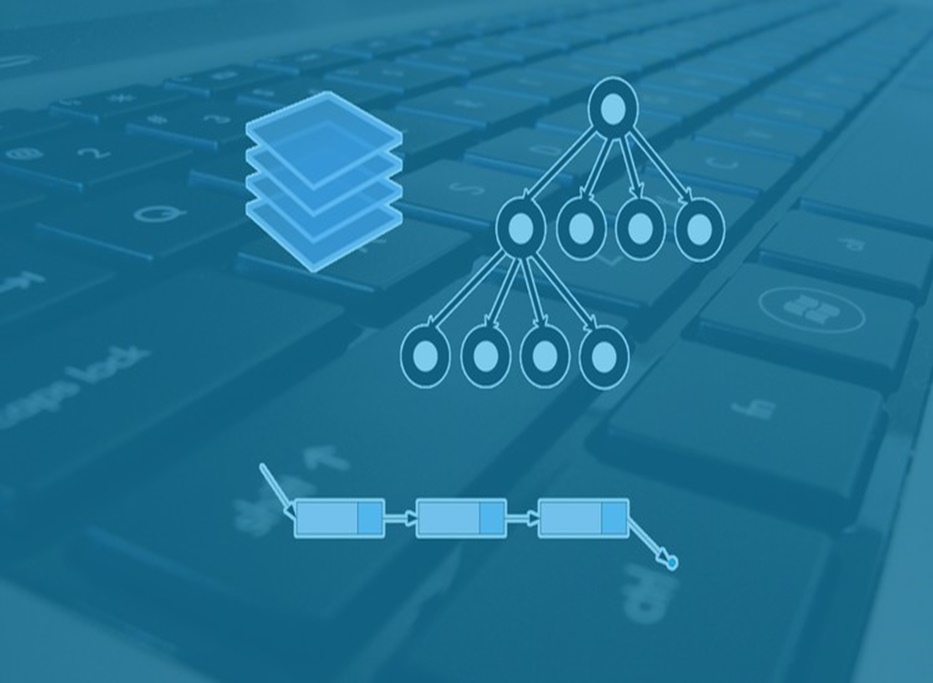
**ASIGNATURA**

Estructura de Datos

Nrc: 7999

**PROFESOR**

Ing. Edgar Fernando Solís Acosta



**Codificación**

**Enunciado**

Crear un programa que calcule la suma y la multiplicación de dos matrices con memoria dinámica utilizando recursividad y plantillas

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* UFA-ESPE

\* Autores: Jeimy Marley Morales Sosa & Camilo Andres Orrico Emme

\* Modificacion: Martes, 14 de noviembre

\* Purpose: Operaciones con Matrices

\*Nivel: Tercero NRC:7999

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#include "Matriz.cpp"

#define MAX 5

using namespace std;

int main(int argc, char\*\* argv){

Matriz <int>f;

Matriz <int>c;

Matriz <int>f1;

Matriz <int>c1;

Matriz <int>seleccionar;

int i,j,opc;

int \*\*matriz1 ,\*\*matriz2, \*\*resultado,\*\*resultado1,\*\*resultado2 ;

char datoEntero[10];

Matriz<int> funcion;

do{

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*OPERACIONES DE LAS MATRICES\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

cout<<"\t[1] MULTIPLICACION\n\t[2] SUMA\n\t[3] SUMA RECURSIVA\n\t[4] MULTIPLICACION RECURSIVA\n\t[0]Salir\n";

str\_cpy(datoEntero,ingresarDatosEnteros("Seleccione: "));

opc=(funcion\_atoi(datoEntero,strlen(datoEntero)));

seleccionar.setDato(opc);

if(seleccionar.getDato() ==0){

return 0;

}

}while(seleccionar.getDato()>4 || seleccionar.getDato()<1);

/////////////////////////////////////////////////////////////////////////////////////////

cout<<"\nIngreso de la primera matriz"<<endl;

str\_cpy(datoEntero,ingresarDatosEnteros("\nDigite el numero de filas : "));

f.setFilas(funcion\_atoi(datoEntero,strlen(datoEntero)));

str\_cpy(datoEntero,ingresarDatosEnteros("\nDigite el numero de columnas : "));

c.setColumnas(funcion\_atoi(datoEntero,strlen(datoEntero)));

//reservando memoria para la primera matriz

matriz1 = new int\*[MAX];

for ( i=0;i<f.getFilas();i++){

matriz1[i] = new int[MAX];

}

funcion.ingresarMatriz(matriz1,f.getFilas(),c.getColumnas());

/////////////////////////////////////////////////////////////////////////////////////////

cout<<"\nIngreso de la segunda matriz"<<endl;

str\_cpy(datoEntero,ingresarDatosEnteros("\nDigite el numero de filas : "));

f1.setFilas(funcion\_atoi(datoEntero,strlen(datoEntero)));

str\_cpy(datoEntero,ingresarDatosEnteros("\nDigite el numero de columnas : "));

c1.setColumnas(funcion\_atoi(datoEntero,strlen(datoEntero)));

//reservando memoria para la segunda matriz

matriz2 = new int\*[MAX];

for (i=0;i<f1.getFilas();i++){

matriz2[i] =new int[MAX];

}

funcion.ingresarMatriz(matriz2,f1.getFilas(),c1.getColumnas());

funcion.imprimirMatriz(matriz1 ,f.getFilas(),c.getColumnas());

funcion.imprimirMatriz(matriz2 ,f1.getFilas(),c1.getColumnas());

/////////////////////////////////////////////////////////////////////////////////////////

switch (seleccionar.getDato()){

case 1:

resultado = new int\*[MAX];

for (i=0;i<f1.getFilas();i++){

resultado[i] =new int[MAX];

}

resultado1 = new int\*[MAX];

for (i=0;i<f1.getFilas();i++){

resultado1[i] =new int[MAX];

}

if(f.getFilas()==f1.getFilas() && c.getColumnas()==c1.getColumnas()){

funcion.multiplicarMatriz(matriz1, matriz2,resultado,resultado1,f.getFilas(), c.getColumnas(), f1.getFilas(), c1.getColumnas());

funcion.imprimirMatriz(resultado1,f.getFilas(),c1.getColumnas());

}else{

system("cls");

cout<<"No es posible calcular la multiplicacion"<<endl;

}

free(matriz1);

free(matriz2);

free(resultado);

free(resultado1);

break;

case 2:

if(f.getFilas()==f1.getFilas() && c.getColumnas()==c1.getColumnas()){

funcion.sumaMatriz(matriz1,f.getFilas(),c.getColumnas(),matriz2);

funcion.imprimirMatriz(matriz1 ,f.getFilas(),c.getColumnas());

}else{

system("cls");

cout<<"No es posible calcular la suma"<<endl;

}

free(matriz1);

free(matriz2);

break;

case 3:

resultado2 = new int\*[MAX];

for (i=0;i<f1.getFilas();i++){

resultado2[i] =new int[MAX];

}

if(f.getFilas()==f1.getFilas() && c.getColumnas()==c1.getColumnas()){

funcion.sumaMatrizRecursiva(matriz1, matriz2,resultado2, f.getFilas()-1, c.getColumnas()-1, funcion);

funcion.imprimirMatriz(resultado2 ,f.getFilas(),c.getColumnas());

}else{

cout<<"No es posible calcular la suma Recursiva"<<endl;

}

free(matriz1);

free(matriz2);

free(resultado2);

break;

case 4:

int suma = 0;

int pri = 0;

int seg = 0;

resultado2 = new int\*[MAX];

for (i=0;i<f1.getFilas();i++){

resultado2[i] =new int[MAX];

}

if(f.getFilas()==f1.getFilas() && c.getColumnas()==c1.getColumnas()){

funcion.multiplicacion(matriz1, matriz2,resultado2, f.getFilas(), c.getColumnas(), c1.getColumnas(),suma, pri, seg, 0, 0, 0, funcion);

funcion.imprimirMatriz(resultado2 ,f.getFilas(),c.getColumnas());

}else {

cout<<"No es posible calcular la multiplicacion Recursiva"<<endl;

}

free(matriz1);

free(matriz2);

free(resultado2);

break;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* UFA-ESPE

\* Autores: Jeimy Marley Morales Sosa

\* Modificacion: Martes, 14 de noviembre

\* Purpose: Operaciones con Matrices

\*Nivel: Tercero NRC:7999

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <conio.h>

#include <stdlib.h>

#include <time.h>

#include "Matriz.h"

#include "validar.cpp"

template<typename T>

void Matriz<T>::setMatriz(T m){

this->\*\*matriz=m;

}

template<typename T>

T Matriz<T>::getMatriz(){

return this->\*\*matriz;

}

template<typename T>

void Matriz<T>::setDato(T d){

this->dato=d;

}

template<typename T>

T Matriz<T>::getDato(){

return this->dato;

}

template<typename T>

void Matriz<T>::setFilas(T f){

this->filas=f;

}

template<typename T>

T Matriz<T>::getFilas()const{

return this->filas;

}

template<typename T>

void Matriz<T>::setColumnas(T c){

this->columnas=c;

}

template<typename T>

T Matriz<T>::getColumnas()const{

return this->columnas;

}

template<typename T>

void Matriz<T>::ingresarMatriz(T \*\*matriz, T filas, T columnas)

{

char datoEntero[10];

cout<<"\nIngrese los valores para la matriz"<<endl;

for(int i=0;i<filas;i++){

for(int j=0;j<columnas;j++){

do{

cout<<"\nDigite un numero: ["<<i<<" "<<j<<"]"<<": ";

str\_cpy(datoEntero,ingresarDatosEnteros(" "));

\*(\*(matriz+i)+j)=funcion\_atoi(datoEntero,strlen(datoEntero));

//cin>>\*(\*(matriz+i)+j);

}while (\*(\*(matriz+i)+j)==NULL);

}

}

}

template<typename T>

void Matriz<T>::imprimirMatriz(T \*\*matriz, T filas, T columnas)

{

// cout<<"\nTEMPLATE\n "<<endl;

cout<<"\nLa matriz es:\n "<<endl;

for (int i=0;i<filas;i++){

for(int j =0;j<columnas;j++){

cout<<\*(\*(matriz+i)+j)<<" ";

//printf(" %p",&\*(\*(matriz+i)+j));

}

cout<<"\n";

}

}

template<typename T>

T Matriz<T>::sumaMatrizRecursiva(T \*\*matriz1, T \*\*matriz2,T \*\*resultado, T f, T c,Matriz funcion){

if((f==0)&&(c==0)){//caso base solo se va a terminar cuando 0 0

\*(\*(resultado+f)+c)=\*(\*(matriz1+f)+c) + \*(\*(matriz2+f)+c);

return \*(\*(resultado+f)+c);

}else{

if((f>=0)&&(c>=0)){

\*(\*(resultado+f)+c)=\*(\*(matriz1+f)+c) + \*(\*(matriz2+f)+(c));

c--;

return funcion.sumaMatrizRecursiva(matriz1, matriz2,resultado, f, c,funcion);

}

c=sizeof(matriz1[0])/sizeof(matriz1[0][0]);

f--;

return funcion.sumaMatrizRecursiva(matriz1, matriz2,resultado, f, c,funcion);

}

}

template<typename T>

void Matriz<T>::multiplicarMatriz(T \*\*matrizA, T \*\*matrizB,T \*\*resultado ,T \*\*resultado1,T fA, T cA, T fB, T cB)

{ int suma=0;

//misno columnas A= Filas B

//Matriz resultante filas A con Columnas B

Matriz funcion;

int k;

cout<<"\nLa multiplicacion de las matrices es \n "<<endl;

for (int i=0; i<fA;i++){

for(int j=0; j<cB;j++ ){

suma=0;

for( k=0; k<cA;k++){

(\*(\*(resultado+i)+k))= ((\*(\*(matrizA+i)+k)) \* (\*(\*(matrizB+k)+j)));

suma=suma+(\*(\*(resultado+i)+k));

\*(\*(resultado1+i)+j)=suma;//memoria

}

cout<<suma<<" | ";

}printf("\n");

}

}

template<typename T>

void Matriz<T>::sumaMatriz(T \*\*matriz1, T filas, T columnas, T \*\*matriz2)

{

cout<<"\nLa suma de las matrices es "<<endl;

for (int i=0;i<filas;i++){

for(int j=0;j<columnas;j++){

\*(\*(matriz1+i)+j) += \*(\*(matriz2+i)+j);

}

}

}

template<typename T>

void Matriz<T>::multiplicacion(T \*\*matrizA,T \*\*matrizB, T \*\*matrizResultado, T f, T c, T c1,T suma, T primero, T segundo, T i, T k, T j, Matriz funcion) {

if (k == c1) {

(\*(\*(matrizResultado + i) + j)) = suma;

j++;

k = 0;

suma = 0;

if (j == c) {

i++;

j = 0;

}

}

if (i == f) return;

primero = \*(\*(matrizA + i) + k);

segundo = \*(\*(matrizB + k) + j);

suma += primero \* segundo;

k++;

funcion.multiplicacion(matrizA, matrizB, matrizResultado, f, c, c1, suma, primero, segundo, i, k, j,funcion);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* UFA-ESPE

\* Autores: Jeimy Marley Morales Sosa

\* Modificacion: Martes, 14 de noviembre

\* Purpose: Operaciones con Matrices

\*Nivel: Tercero NRC:7999

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define \_\_Matriz\_Matriz\_h

template<typename T>

class Matriz{

public:

void setMatriz(T m);

T getMatriz() ;

void setFilas(T f);

T getFilas() const;

void setColumnas(T c);

T getColumnas() const;

void setDato(T m);

T getDato() ;

void ingresarMatriz(T \*\*m, T f, T c);

void sumaMatriz(T \*\*m1 , T f, T c, T \*\*m2);

void multiplicarMatriz(T \*\*matrizA, T \*\*matrizB,T \*\*resultado,T \*\*resultado1, T fA, T cA, T fB, T cB);

void imprimirMatriz(T \*\*m , T f, T c);

T sumaMatrizRecursiva(T \*\*matriz1, T \*\*matriz2,T \*\*resultado, T f, T, Matriz funcion);

void multiplicacion(T \*\*matrizA,T \*\*matrizB, T \*\*matrizResultado, T f, T c, T c1,T suma, T primero, T segundo, T i, T k, T j, Matriz funcion);

private:

T \*\*matriz;

T filas;

T columnas;

T dato;

};