Practica 2

1.13

Generado por Doxygen 1.9.5

Capítulo 1

Indice de archivos

1.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

Capítulo 2

Documentación de archivos

2.1. Referencia del Archivo C:/Users/ikerf/Desktop/Upiita/3er Semestre/POO/p2NES/Practica2Doxy.cpp

```
#include <iostream>
#include <chrono>
#include <random>
```

Funciones

- void menu ()

  Muestra un menu con las operaciones que puede realizar la calculadora de matrices y solicita escoger una.
- void leerMatriz (int, int, int)
- void randomMatriz (int, int, int, unsigned int)
- void imprimirMatriz (int, int, int)
- void sumaMatriz ()

  Permite sumar 2 matrices con dimensiones iguales.
- void restaMatriz ()

  Permite restar 2 matrices con dimensiones iguales.
- void multMatriz ()

  Permite multiplicar 2 matrices. El número de columnas de la primer matriz debe ser el mismo numero de filas de la segunda.
- void transMatriz ()

  transMatriz Saca la transpuestade una matriz.
- int main ()

  Función principal del programa. Permite elegir que operación realizar. Son 4 opciones, suma, resta, multiplicación, entre 2 matrices, y transpuesta de una matriz; y la opción de salir, cualquier otra no se acepta.
- void leerMatriz (int $**$M, int fila, int columna)

  Permite ingresar los datos de la matriz. El usuario podra ir ingresando los datos de la matiz por filas.
- void randomMatriz (int $**$M, int fila, int columna, unsigned int semilla)

  Genera una matriz con números aleatorios. Se creará una matriz con la cantidad de filas y columnas solicitada con números aleatorios.
- void imprimirMatriz (int $**$M, int fila, int columna)

  Permite mostrar en pantalla una matriz ya creada.

2.1.1. Descripción detallada

Versión

1.13

Fecha

26/10/2022

Autor

Ramírez López Emilio, Sánchez Díaz Nadya, Velázquez Osorio Samara Ishtar

Definición en el archivo Practica2Doxy.cpp.

2.1.2. Documentación de las funciones

2.1.2.1. imprimirMatriz() [1/2]

```
void imprimirMatriz (
            int ** M,
            int fila,
            int columna )
```

Permite mostrar en pantalla una matriz ya creada.

Parámetros

***M*	Doble apuntador que apunta al arreglo de filas reservadas de la matriz a escribir
fila	El numero de filas que tiene la matriz.
columna	El numero de columnas que tiene la matriz.

Definición en la línea 131 del archivo Practica2Doxy.cpp.

2.1.2.2. imprimirMatriz() [2/2]

```
void imprimirMatriz (
            int ,
            int ,
            int  )
```

2.1.2.3. leerMatriz() [1/2]

```
void leerMatriz (
            int ** M,
            int fila,
            int columna )
```

Permite ingresar los datos de la matriz. El usuario podra ir ingresando los datos de la matiz por filas.

Parámetros

***M*	Doble apuntador que apunta al arreglo de filas reservadas de la matriz a escribir
fila	El numero de filas que va a tener la matriz.
columna	El numero de columnas que va a tener la matriz.

Definición en la línea 94 del archivo Practica2Doxy.cpp.

2.1.2.4. leerMatriz() [2/2]

```
void leerMatriz (
            int ,
            int ,
            int  )
```

2.1.2.5. main()

```
int main ( )
```

Función principal del programa. Permite elegir que operación realizar. Son 4 opciones, suma, resta, multiplicación, entre 2 matrices, y transpuesta de una matriz; y la opción de salir, cualquier otra no se acepta.

Ver también

sumaMatriz, restaMatriz, multMatriz y transMatriz.

int opcion Guarda el número de la opción seleccionada

Definición en la línea 29 del archivo Practica2Doxy.cpp.

2.1.2.6. menu()

```
void menu ( )
```

Muestra un menu con las operaciones que puede realizar la calculadora de matrices y solicita escoger una.

Definición en la línea 71 del archivo Practica2Doxy.cpp.

2.1.2.7. multMatriz()

```
void multMatriz ( )
```

Permite multiplicar 2 matrices. El número de columnas de la primer matriz debe ser el mismo numero de filas de la segunda.

Ver también

randomMatriz, leerMatriz, imprimirMatriz.

< int filaA Guarda el número de filas que tendrá la matriz A

< int filaB Guarda el número de filas que tendrá la matriz B

< int columnaA Guarda el número de columnas que tendrá la matriz A

< int columnaB Guarda el número de columnas que tendrá la matriz B

Definición en la línea 276 del archivo Practica2Doxy.cpp.

2.1.2.8. randomMatriz() [1/2]

```
void randomMatriz (
            int ** M,
            int fila,
            int columna,
            unsigned int semilla )
```

Genera una matriz con números aleatorios. Se creará una matriz con la cantidad de filas y columnas solicitada con números aleatorios.

Parámetros

***M*	Doble apuntador que apunta al arreglo de filas reservadas de la matriz a escribir
fila	El numero de filas que va a tener la matriz.
columna	El numero de columnas que va a tener la matriz.
semilla	A partir de la cual se generan los numeros aleatorios.

Definición en la línea 112 del archivo Practica2Doxy.cpp.

2.1.2.9. randomMatriz() [2/2]

```
void randomMatriz (
            int ,
            int ,
            int ,
            unsigned int  )
```

2.1.2.10. restaMatriz()

```
void restaMatriz ( )
```

Permite restar 2 matrices con dimensiones iguales.

Ver también

randomMatriz, leerMatriz, imprimirMatriz.

< int fila Guarda el número de filas que tendrá la matriz

< int columna Guarda el número de columnas que tendrá la matriz

Definición en la línea 210 del archivo Practica2Doxy.cpp.

2.1.2.11. sumaMatriz()

```
void sumaMatriz ( )
```

Permite sumar 2 matrices con dimensiones iguales.

Ver también

randomMatriz, leerMatriz, imprimirMatriz.

< int fila Guarda el número de filas que tendrá la matriz

< int columna Guarda el número de columnas que tendrá la matriz

Definición en la línea 145 del archivo Practica2Doxy.cpp.

2.1.2.12. transMatriz()

```
void transMatriz ( )
```

transMatriz Saca la transpuestade una matriz.

Ver también

randomMatriz, leerMatriz, imprimirMatriz.

< Guarda el número de filas que tendrá la matriz

< Guarda el número de columnas que tendrá la matriz

Definición en la línea 358 del archivo Practica2Doxy.cpp.

2.2. Practica2Doxy.cpp

Ir a la documentación de este archivo.

```
00001
00008 #include <iostream
00009 #include <chrono
00010 #include <random
00011
00012 using namespace std;
00013
00014 void menu();
00015 void leerMatriz(int, int, int);
00016 void randomMatriz(int, int, int, unsigned int);
00017 void imprimirMatriz(int, int, int);
00018 void sumaMatriz();
00019 void restaMatriz();
00020 void multMatriz();
00021 void transMatriz();
00022
00029 int main(){
00030
00031    int opcion{};
00034     do{
00035         menu();
00036         cin  opcion;
```

00037 if(cin.fail() || cin.bad()){

00038 opcion = 0;

00039 cin.clear();

00040 cin.ignore();

00041 }

00042

00043 switch(opcion){

00044 case 1:

00045 sumaMatriz();

00046 break;

00047 case 2:

00048 restaMatriz();

00049 break;

00050 case 3:

00051 multMatriz();

00052 break;

00053 case 4:

00054 transMatriz();

00055 break;

00056 case 5:

```
00057 cout « "\n\nTENGA UN BUEN DIA\n\n";

»00058 break;

»00059 default:

»00060 cout “ "\n\nOPCION NO VALIDA\n\n";

»”00061 break;

»”00062 }

»”00063 }while(opcion != 5);

»”00064 return 0;

»”00065 }

»”00066

»”00067

»”00071 void menu(){

»”00072 cout ‘ "\n CALCULADORA DE MATRICES"

»”’00073  "\n//////////////////////////////////////////"

»”’00074  "\n1. Suma de Matrices..... (Pulsa 1)"

»”’00075  "\n2. Resta de Matrices..... (Pulsa 2)"

»”’00076  "\n3. Multiplicacion de Matrices... (Pulsa 3)"

»”’00077  "\n4. Transpuesta de Matriz..... (Pulsa 4)"

»”’00078  "\n5. Finalizar Calculadora..... (Pulsa 5)"

»”’00079  "\n//////////////////////////////////////////"

»”’00080  "\nNOTA: LA MATRIZ SE GENERARA DE MANERA"

»”’00081  "\n ALEATORIA SI PUEDE CONTENER"

»”’00082  "\n MAS DE 16 ELEMENTOS"

»”’00083  "\n (CON UN RANGO 0-100)"

»”’00084  "\n\nSELECCIONA UNA OPCION (1-5): ";

»”’00085 }

»”’00086

»”’00094 void leerMatriz(int **M, int fila, int columna){

»”’00095 cout  "\nRellenar la matriz:\n";

»”’00096 for(size_t i{0}; i <fila; i++){
```

```
»"'00097 for(size_t j{0}; j <columna; j++){
»"'00098 cout  "Matriz["  i+1  "]["  j+1  "]: ";
»"'00099 cin  M[i][j];
»"'00100 }
»"'00101 }
»"'00102 }
»"'00103
»"'00112 void randomMatriz(int **M, int fila, int columna, unsigned int semilla){
»"'00113 default_random_engine motorG(semilla);
»"'00114 uniform_int_distribution<int>distM(0,100);
»"'00115
»"'00116 cout  "\nMatriz:\n";
»"'00117 for(size_t i{0}; i <fila; i++){
»"'00118 for(size_t j{0}; j <columna; j++){
»"'00119 M[i][j] = distM(motorG);
»"'00120 cout  "Matriz["  i+1  "]["  j+1  "]: "  M[i][j]  endl;
»"'00121 }
»"'00122 }
»"'00123 }
»"'00124
»"'00131 void imprimirMatriz(int **M, int fila, int columna){
»"'00132 for(size_t i{0}; i <fila; i++){
»"'00133 cout  "|\t";
»"'00134 for(size_t j{0}; j <columna; j++){
»"'00135 cout  M[i][j]  "\t";
»"'00136 }
»"'00137 cout  "\t|\n";
»"'00138 }
»"'00139 }
»"'00140
```

```
»"'00145 void sumaMatriz(){
»"'00146 int fila{};
»"'00147 int columna{};
»"'00149 cout  "\nLAS MATRICES TENDRAN LAS MISMAS DIMENSIONES\n"
»"'00150  "\nDimensiones de las matrices:";
»"'00151 cout  "\nFilas de las matrices: ";
»"'00152 cin  fila;
»"'00153 cout  "\nColumnas de las matrices: ";
»"'00154 cin  columna;
»"'00155
»"'00156 int **A = new int*[fila]; //doble apuntador para hacer matriz A
»"'00157 for(size_t i{0}; i <fila; i++){ //Para cada fila del apuntador se hace
»"'00158 A[i] = new int[columna]; //las columnas necesarias
»"'00159 }
»"'00160
»"'00161 int **B = new int*[fila]; //doble apuntador para hacer matriz B
»"'00162 for(size_t i{0}; i <fila; i++){
»"'00163 B[i] = new int[columna];
»"'00164 }
»"'00165
»"'00166 int **C = new int*[fila]; //doble apuntador para hacer matriz C
»"'00167 for(size_t i{0}; i <fila; i++){
»"'00168 C[i] = new int[columna];
»"'00169 }
»"'00170
»"'00171 //Se inicializan las semillas en tiempos de ejecucion diferentes para evitar primer elemento igual
»"'00172 unsigned int semilla1 = chrono::steady_clock::now().time_since_epoch().count();
»"'00173
»"'00174 cout  "\nDatos de la matriz A: ";
»"'00175 if(fila*columna >16){
```

»"'00176 randomMatriz(A, fila, columna, semilla1);

»"'00177 }else{

»"'00178 leerMatriz(A, fila, columna);

»"'00179 }

»"'00180

»"'00181 unsigned int semilla2 = chrono::steady_clock::now().time_since_epoch().count();

»"'00182

»"'00183 cout "\nDatos de la matriz B: ";

»"'00184 if(fila*columna >16){

»"'00185 randomMatriz(B, fila, columna, semilla2);

»"'00186 }else{

»"'00187 leerMatriz(B, fila, columna);

»"'00188 }

»"'00189

»"'00190 for(size_t i{0}; i <fila; i++){

»"'00191 for(size_t j{0}; j <columna; j++){

»"'00192 C[i][j] = A[i][j] + B[i][j];

»"'00193 }

»"'00194 }

»"'00195

»"'00196 cout "\nMatriz A:\n";

»"'00197 imprimirMatriz(A, fila, columna);

»"'00198 cout "\nMatriz B:\n";

»"'00199 imprimirMatriz(B, fila, columna);

»"'00200 cout "\nSuma de las matrices A+B:\n";

»"'00201 imprimirMatriz(C, fila, columna);

»"'00202 system("PAUSE()");

»"'00203 cout "\n" endl;

»"'00204 }

»"'00205

```
»"'00210 void restaMatriz(){
»"'00211 int fila{};
»"'00212 int columna{};
»"'00214 cout  "\nLAS MATRICES TENDRAN LAS MISMAS DIMENSIONES\n"
»"'00215  "\nDimensiones de las matrices:";
»"'00216 cout  "\nFilas de las matrices: ";
»"'00217 cin  fila;
»"'00218 cout  "\nColumnas de las matrices: ";
»"'00219 cin  columna;
»"'00220
»"'00221 int **A = new int*[fila]; //doble apuntador para hacer matriz A
»"'00222 for(size_t i{0}; i <fila; i++){ //Para cada fila del apuntador se hace
»"'00223 A[i] = new int[columna]; //las columnas necesarias
»"'00224 }
»"'00225
»"'00226 int **B = new int*[fila]; //doble apuntador para hacer matriz B
»"'00227 for(size_t i{0}; i <fila; i++){
»"'00228 B[i] = new int[columna];
»"'00229 }
»"'00230
»"'00231 int **C = new int*[fila]; //doble apuntador para hacer matriz C
»"'00232 for(size_t i{0}; i <fila; i++){
»"'00233 C[i] = new int[columna];
»"'00234 }
»"'00235
»"'00236 //Se inicializan las semillas en tiempos de ejecucion diferentes para evitar primer elemento igual
»"'00237 unsigned int semilla1 = chrono::steady_clock::now().time_since_epoch().count();
»"'00238
»"'00239 cout  "\nDatos de la matriz A: ";
»"'00240 if(fila*columna >16){
```

```
»"'00241 randomMatriz(A, fila, columna, semilla1);
»"'00242 }else{
»"'00243 leerMatriz(A, fila, columna);
»"'00244 }
»"'00245
»"'00246 unsigned int semilla2 = chrono::steady_clock::now().time_since_epoch().count();
»"'00247
»"'00248 cout  "\nDatos de la matriz B: ";
»"'00249 if(fila*columna >16){
»"'00250 randomMatriz(B, fila, columna, semilla2);
»"'00251 }else{
»"'00252 leerMatriz(B, fila, columna);
»"'00253 }
»"'00254
»"'00255 for(size_t i{0}; i <fila; i++){
»"'00256 for(size_t j{0}; j <columna; j++){
»"'00257 C[i][j] = A[i][j] - B[i][j];
»"'00258 }
»"'00259 }
»"'00260
»"'00261 cout  "\nMatriz A:\n";
»"'00262 imprimirMatriz(A, fila, columna);
»"'00263 cout  "\nMatriz B:\n";
»"'00264 imprimirMatriz(B, fila, columna);
»"'00265 cout  "\nResta de las matrices A-B:\n";
»"'00266 imprimirMatriz(C, fila, columna);
»"'00267 system("PAUSE()");
»"'00268 cout  "\n"  endl;
»"'00269 }
»"'00270
```

```
»"'00276 void multMatriz(){
»"'00277 int filaA{};
»"'00278 int filaB{};
»"'00279 int columnaA{};
»"'00280 int columnaB{};
»"'00282 cout  "\nEL NUMERO DE FILAS DE LA MATRIZ B DEBE SER"
»"'00283  "\nIGUAL AL NUMERO DE COLUMNAS DE LA MATRIZ A";
»"'00284 cout  "\nDimension de la matriz A:";
»"'00285 cout  "\nFilas de la matriz: ";
»"'00286 cin  filaA;
»"'00287 cout  "Columnas de la matriz: ";
»"'00288 cin  columnaA;
»"'00289
»"'00290 int **A = new int*[filaA]; //doble apuntador para hacer matriz A
»"'00291 for(size_t i{0}; i <filaA; i++){ //Para cada fila del apuntador se hace
»"'00292 A[i] = new int[columnaA]; //las columnas necesarias
»"'00293 }
»"'00294
»"'00295 //Se inicializan las semillas en tiempos de ejecucion diferentes para evitar primer elemento igual
»"'00296 unsigned int semilla1 = chrono::steady_clock::now().time_since_epoch().count();
»"'00297
»"'00298 cout  "\nDatos de la matriz A: ";
»"'00299 if(filaA*columnaA >16){
»"'00300 randomMatriz(A, filaA, columnaA, semilla1);
»"'00301 }else{
»"'00302 leerMatriz(A, filaA, columnaA);
»"'00303 }
»"'00304
»"'00305 cout  "\nDimension de la matriz B:";
»"'00306 cout  "\nFilas de la matriz: ";
```

```
»"'00307 cin  filaB;
»"'00308 cout  "Columnas de la matriz: ";
»"'00309 cin  columnaB;
»"'00310
»"'00311 int **B = new int*[filaB]; //doble apuntador para hacer matriz B
»"'00312 for(size_t i{0}; i <filaB; i++){
»"'00313 B[i] = new int[columnaB];
»"'00314 }
»"'00315
»"'00316 unsigned int semilla2 = chrono::steady_clock::now().time_since_epoch().count();
»"'00317
»"'00318 cout  "\nDatos de la matriz B: ";
»"'00319 if(filaB*columnaB >16){
»"'00320 randomMatriz(B, filaB, columnaB, semilla2);
»"'00321 }else{
»"'00322 leerMatriz(B, filaB, columnaB);
»"'00323 }
»"'00324
»"'00325 int **C = new int*[filaA]; //doble apuntador para hacer matriz C
»"'00326 for(size_t i{0}; i <filaA; i++){ //La matriz C debe tener el numero de filas
»"'00327 C[i] = new int[columnaB]; //de la matriz A y las columnas de la matriz B
»"'00328 }
»"'00329
»"'00330 if(columnaA == filaB){
»"'00331 for(size_t i{0}; i <filaA; i++){
»"'00332 for(size_t j{0}; j <columnaB; j++){
»"'00333 C[i][j] = 0;
»"'00334 for(size_t k{0}; k <columnaA; k++){
»"'00335 C[i][j] += A[i][k] * B[k][j];
»"'00336 }
```

```
»"'00337 }
»"'00338 }
»"'00339 cout  "\nMatriz A:\n";
»"'00340 imprimirMatriz(A, filaA, columnaA);
»"'00341 cout  "\nMatriz B:\n";
»"'00342 imprimirMatriz(B, filaB, columnaB);
»"'00343 cout  "\nMultiplicacion de las matrices A*B:\n";
»"'00344 imprimirMatriz(C, filaA, columnaB);
»"'00345 }else{
»"'00346 cout  "\n NO SE PUEDEN MULTIPLICAR"
»"'00347  "\n EL NUMERO DE COLUMNAS DE LA MATRIZ A NO"
»"'00348  "\nCOINCIDE CON EL NUMERO DE FILAS DE LA MATRIZ B";
»"'00349 }
»"'00350 system("PAUSE()");
»"'00351 cout  "\n"  endl;
»"'00352 }
»"'00353
»"'00358 void transMatriz(){
»"'00359 int fila{};
»"'00360 int columna{};
»"'00362 cout  "\nDimensiones de las matrices:";
»"'00363 cout  "\nFilas de las matrices: ";
»"'00364 cin  fila;
»"'00365 cout  "\nColumnas de las matrices: ";
»"'00366 cin  columna;
»"'00367
»"'00368 int **A = new int*[fila]; //doble apuntador para hacer matriz A
»"'00369 for(size_t i{0}; i <fila; i++){ //Para cada fila del apuntador se hace
»"'00370 A[i] = new int[columna]; //las columnas necesarias
»"'00371 }
```

```
»"'00372

»"'00373 int **B = new int*[columna]; //doble apuntador para hacer matriz B

»"'00374 for(size_t i{0}; i <columna; i++){

»"'00375 B[i] = new int[fila];

»"'00376 }

»"'00377

»"'00378 unsigned int semilla1 = chrono::steady_clock::now().time_since_epoch().count();

»"'00379

»"'00380 cout  "\nDatos de la matriz A: ";

»"'00381 if(fila*columna >16){

»"'00382 randomMatriz(A, fila, columna, semilla1);

»"'00383 }else{

»"'00384 leerMatriz(A, fila, columna);

»"'00385 }

»"'00386

»"'00387 for(size_t i{0}; i <columna; i++){

»"'00388 for(size_t j{0}; j <fila; j++){

»"'00389 B[i][j] = A[j][i];

»"'00390 }

»"'00391 }

»"'00392

»"'00393 cout  "\nMatriz:\n";

»"'00394 imprimirMatriz(A, fila, columna);

»"'00395 cout  "\nTranspuesta:\n";

»"'00396 imprimirMatriz(B, columna, fila);

»"'00397 system("PAUSE()");

»"'00398 cout  "\n"  endl;

»"'00399 }
```

»”’Índice alfabético