Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería y Tecnologías Avanzadas

Práctica 2

Profesor

Iliac Huerta Trujillo

Unidad de Aprendizaje

Programación Orientada a Objetos

Presenta

Ramírez López Emilio	2022640230
Sánchez Díaz Nadya	2022640236
Velázquez Osorio Samara Ishtar	2022640188

04 de noviembre de 2022

Índice

Índice de figuras

1. Planteamiento del problema

Se requiere de una calculadora capaz de realizar 4 operaciones, suma, resta, multiplicación y transpuesta de matrices bidimencionales de cualquier tamaño.

El usuario deberá proporcionar el tamaño de la matriz y siempre que esta no sea mayor a 4x4 el usuario será quien la inicialice, en caso contrario se hará de forma aleatoria.

2. Justificación

El cálculo de matrices de manera manual es un poco tediosa, sobre todo cuando se trata de matrices muy grandes. El programa busca facilitar este procedimiento mediante el uso de selectores para elegir el cálculo a realizar.

Debido a que el cálculo de matrices tiene reglas que no se pueden romper, por ejemplo que sean de las mismas dimensiones en una suma o resta y que al realizar una multiplicación, el número de columnas de la primera matriz sea igual al número de filas de la segunda matriz; el programa tendrá incorporado que en una suma o resta ambas matrices bases tendrán las mismas dimensiones, mientras que en la multiplicación el usuario podrá especificar las dimensiones y, si no se cumple con la regla, avisar que la operación no se puede realizar.

3. Marco teórico

En matemáticas, una matriz es un arreglo bidimensional de números, estas se usan para múltiples aplicaciones y sirven, en particular, para representar los coeficientes de los sistemas de ecuaciones lineales o para representar las aplicaciones lineales; en este último caso las matrices desempeñan el mismo papel que los datos de un vector para las aplicaciones lineales.

Pueden sumarse, multiplicarse y descomponerse de varias formas, lo que también las hace un concepto clave en el campo del álgebra lineal.

De una manera muy resumida, la historia de las matrices se puede ver en el siguiente cuadro:

Año	Acontecimiento
200 a.C.	En china los matemáticos usan series de numeros
1848 d.C.	J.J.Sylvester introduce el termino "Matriz"
1858 d.C.	Cayley publica "Memorias sobre la teoría de matrices"
1878 d.C.	Frobenius demuestra resultados fundamentales en algebra matricial
1925 d.C.	Werner Heisenber utiliza la teoría matricial en la mecánica cuantica

3.1. Suma y resta

La unión de dos o más matrices solo puede hacerse si dichas matrices tienen la misma dimensión. Cada elemento de las matrices puede sumarse con los elementos que coincidan en posición en diferentes matrices.

En el caso de restar dos o más matrices se sigue el mismo procedimiento que usamos para sumar dos o más matrices.

En otras palabras, cuando sumamos o restamos matrices nos vamos a fijar en:

1. Las matrices compartan la misma dimensión.

2. Sumar o restar los elementos con la misma posición en matrices distintas.

$$\text{Sumar: } \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} + \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} a+e & b+f \\ c+g & d+h \end{pmatrix}$$

$$\text{Restar: } \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} - \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} a-e & b-f \\ c-g & d-h \end{pmatrix}$$

Como hemos dicho, primero comprovamos que sean matrices de igual dimensión. En este caso, son dos matrices 2×2. A continuación, sumamos los elementos que tienen las mismas coordenadas.

3.2. Multiplicación

Generalmente, la multiplicación de matrices cumple la propiedad no conmutativa, es decir, importa el orden de los elementos durante la multiplicación. Existen casos llamados matrices conmutativas que sí cumplen la propiedad.

Sean **RyX** dos matrices no conmutativas, implica que:

$$\mathrm{RX} \neq \mathrm{XR}$$

Sean **R** 'y **X** 'dos matrices conmutativas, implica que:

$$\mathrm{RX} = \mathrm{XR}$$

Para multiplicar dos matrices necesitamos que el número de columnas de la primera matriz sea igual al número de filas de la segunda matriz.

$$T_{(2x2)} \cdot F_{(2x2)} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} (a \cdot e + b \cdot g) & (a \cdot f + b \cdot h) \\ (c \cdot e + d \cdot g) & (c \cdot f + d \cdot h) \end{pmatrix}$$

El orden de multiplicación sería tomar la primera fila de la matriz T, multiplicarla por la primera columna de la matriz F y sumar sus elementos.

Podemos multiplicar una matriz por un escalar z cualquiera. En este caso z = 2.

$$2 \cdot \begin{pmatrix} 1 & 0 & 5 \\ 2 & -7 & 9 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 10 \\ 4 & -14 & 18 \end{pmatrix}$$

Cada elemento de la matriz queda multiplicado por el escalar $z = 2$.

3.3. Transpuesta

La transpuesta de una matriz $\mathbf{A}(n \times p)$ es una matriz B $(p \times n)$, obtenida mediante intercambio de filas y columnas, es decir, el primer renglón de **A** es la primera columna de su trnaspuesta, el segundo renglón de **A** es la segunda columnade su transpuesta, de forma que en símbolos puede expresarse mediante:

$$\text{bij=aji} \quad i = 1, 2, \ldots, p; \quad j = 1, 2, \ldots, n$$

En general, a la matriz transpuesta de **A** la denominaremos $\mathbf{A'oA^t}$.

Ejemplos

$$\mathbf{A} = \begin{bmatrix} 2 & 3 & 4 \\ 6 & -5 & 3 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 1 & 4 & 6 \\ 2 & 3 & 11 \\ -7 & 4 & 8 \end{bmatrix}$$

$$\mathbf{A}' = \begin{bmatrix} 2 & 6 \\ 3 & -5 \\ 4 & 3 \end{bmatrix} \qquad \mathbf{B}' = \begin{bmatrix} 1 & 2 & -7 \\ 4 & 3 & 4 \\ 6 & 11 & 8 \end{bmatrix}$$

4. Propuesta de solución

Creamos un programa que le permite al usuario elegir que operación con matrices desea realizar, o si ya no quiere realizar alguna operación, en cuyo caso se cerrará la calculadora.

Cuando el usuario elija suma o resta, se le pedira que ingrese las dimensiones de las matrices una sola vez, pues ambas deberán ser de igual tamaño, si elije multiplicación el usuario podrá decidir el tamaño de las dos matrices, la operación se ejecutará siempre y cuando el número de columnas de la primer matriz sea el mismo de las filas de la segunda; y si elije calcular la transpuesta, solo se le pedirá ingresar las dimensiones de una matriz.

En todos los casos, si las dimensiones proporcionadas por el usuario son mayores a 4x4 no se le dará la oportunidad al usuario de ingresar los datos de las matrices y se generarán matrices aleatorias.

5. Análisis y diseño

Al analizar el problema podemos ver que necesitamos usar la estructura switch para determinar que operación se realizará junto con un do-while que nos permite mostrar la calculadora al menos una vez y volver a hacerlo hasta que se nos pida finalizar la calculadora. Cada case en el switch será una operación diferente.

A continuación se presentan diagramas que ayudarán a comprender de manera gráfica nuestra propuesta de solución.

1. *Diagrama de actividades:* Es un tipo de diagrama dentro del lenguaje unificado de modelado (UML). El cual nos permitirá visualizar de manera general el flujo de actividades dentro de nuestro programa.

2. *Diagrama de flujo:* Representa la esquematización gráfica de un algoritmo, el cual muestra los pasos a seguir para la solución de un problema.

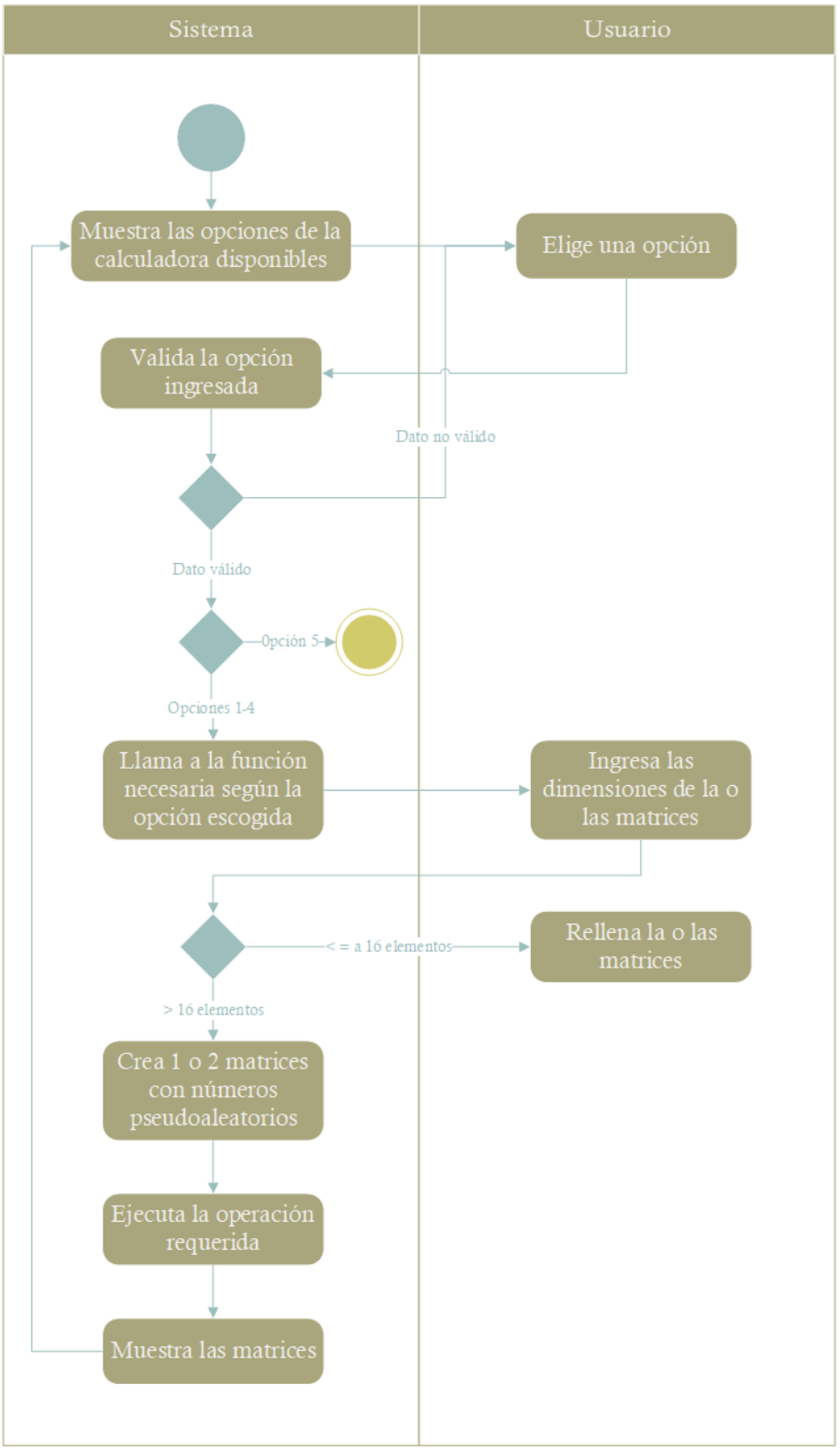

Figura 1: Diagrama actividades

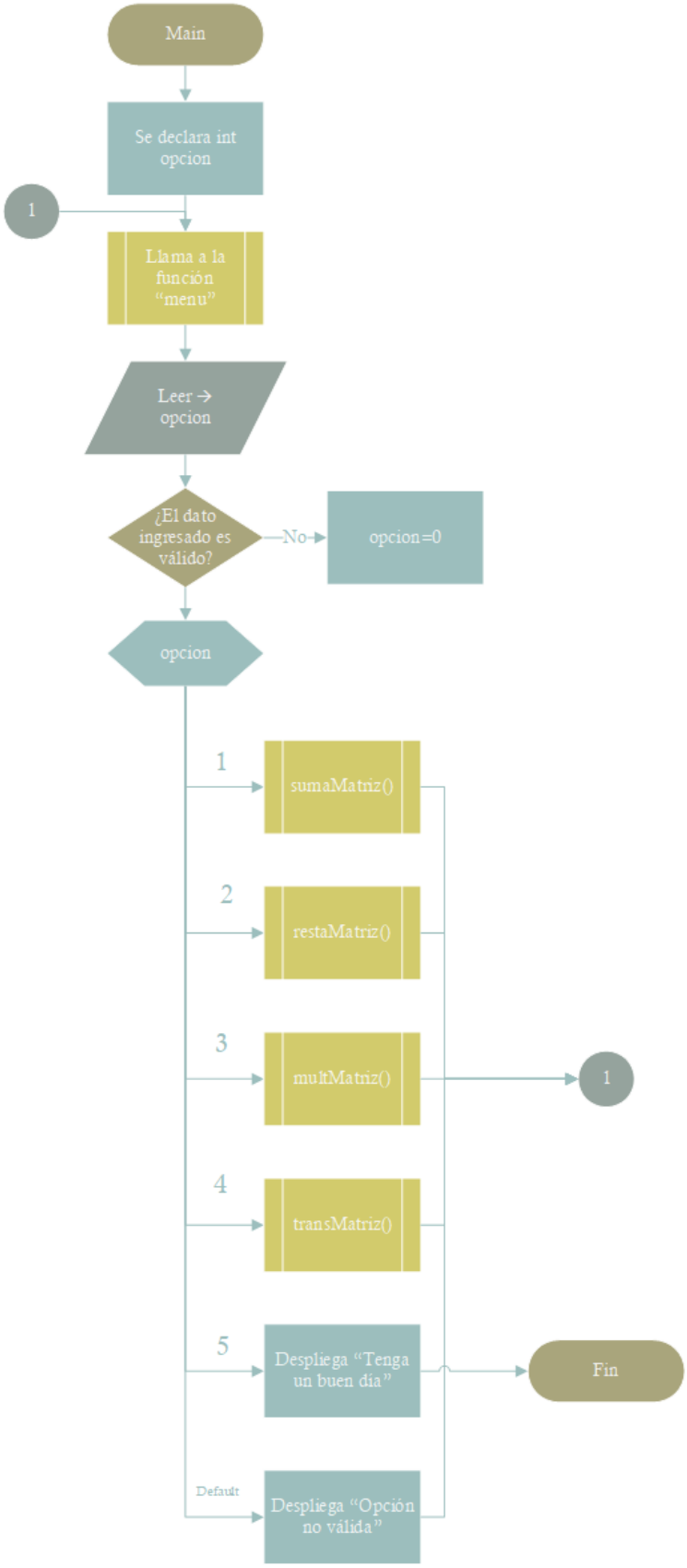

Figura 2: Diagrama función main

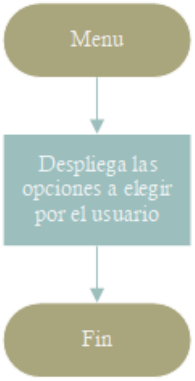

Figura 3: Diagrama función menu

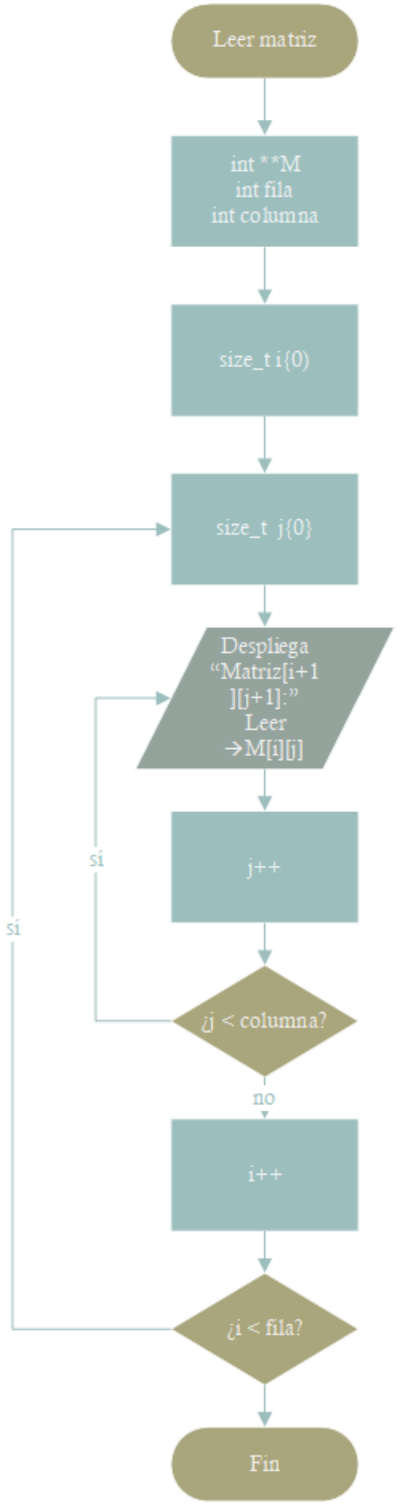

Figura 4: Diagrama función leerMatriz

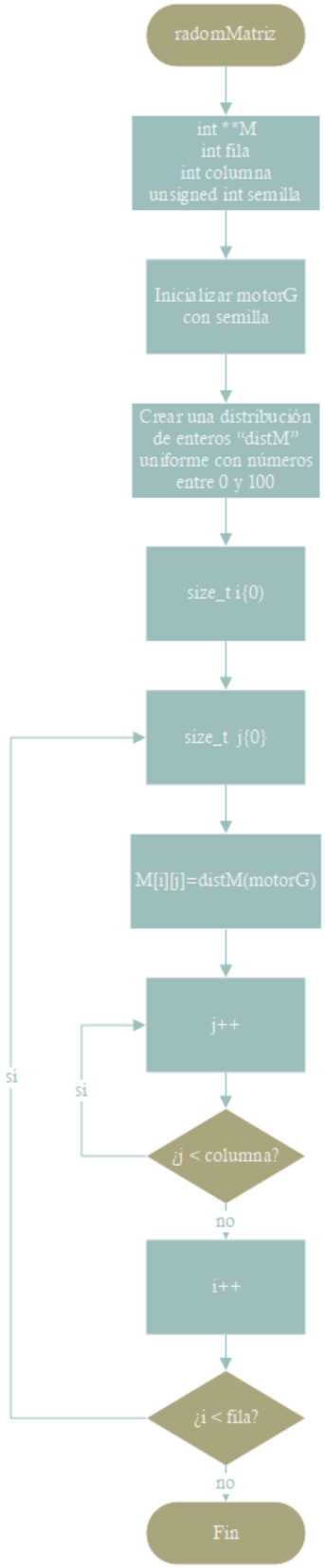

Figura 5: Diagrama función randomMatriz

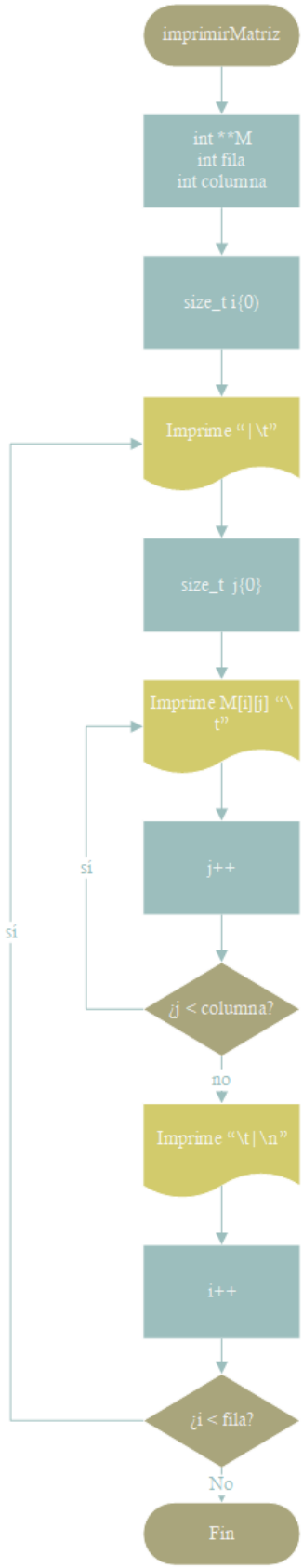

Figura 6: Diagrama función imprimirMatriz

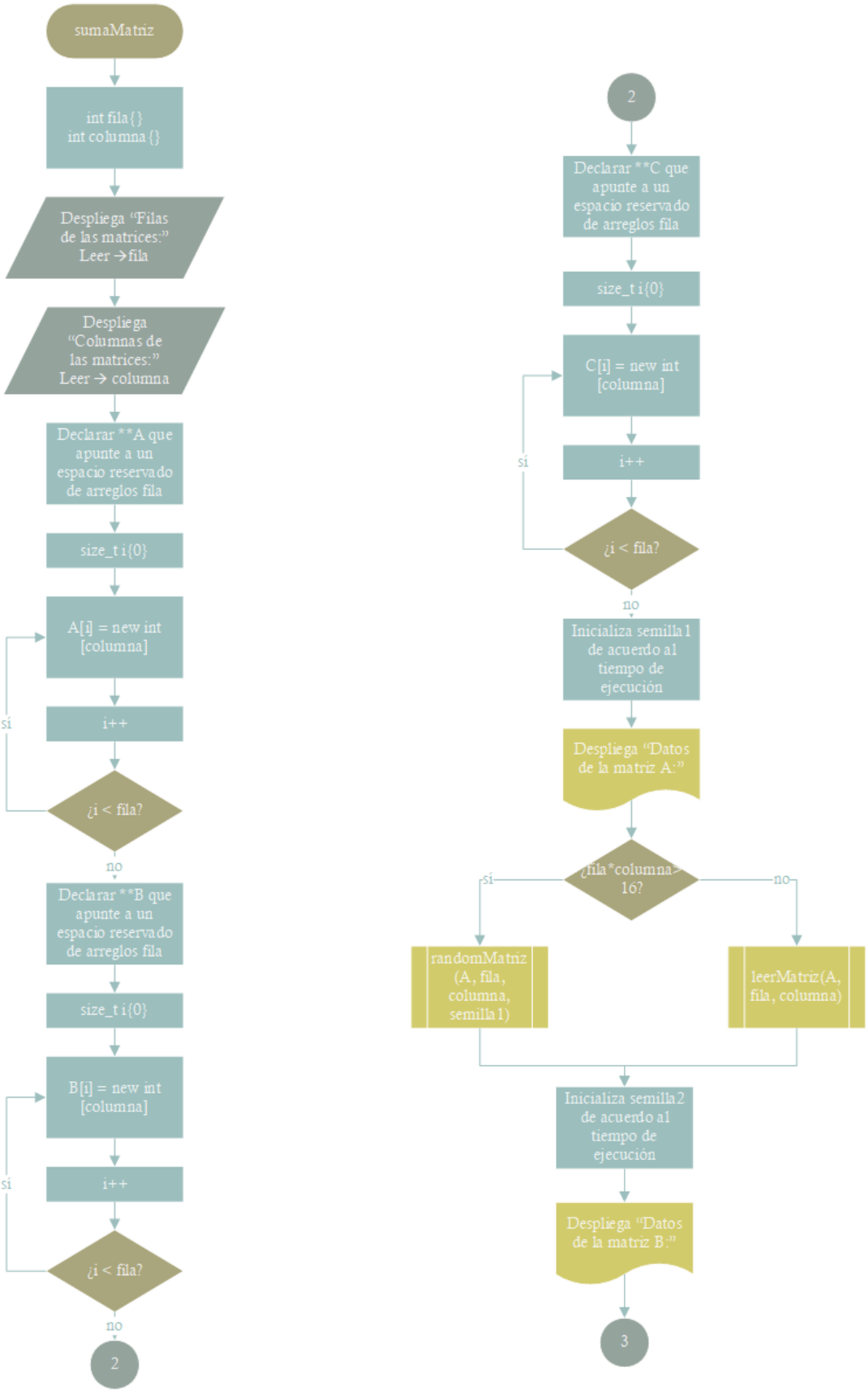

Figura 7: Diagrama función sumaMatriz

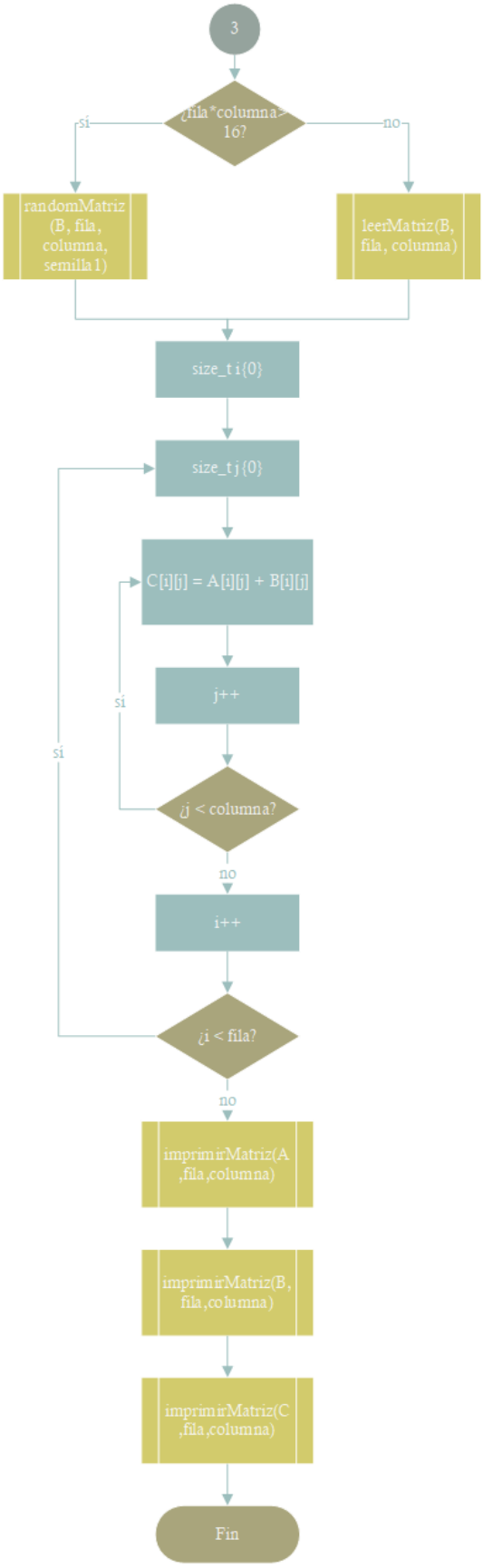

Figura 8: Diagrama función sumaMatriz continuación

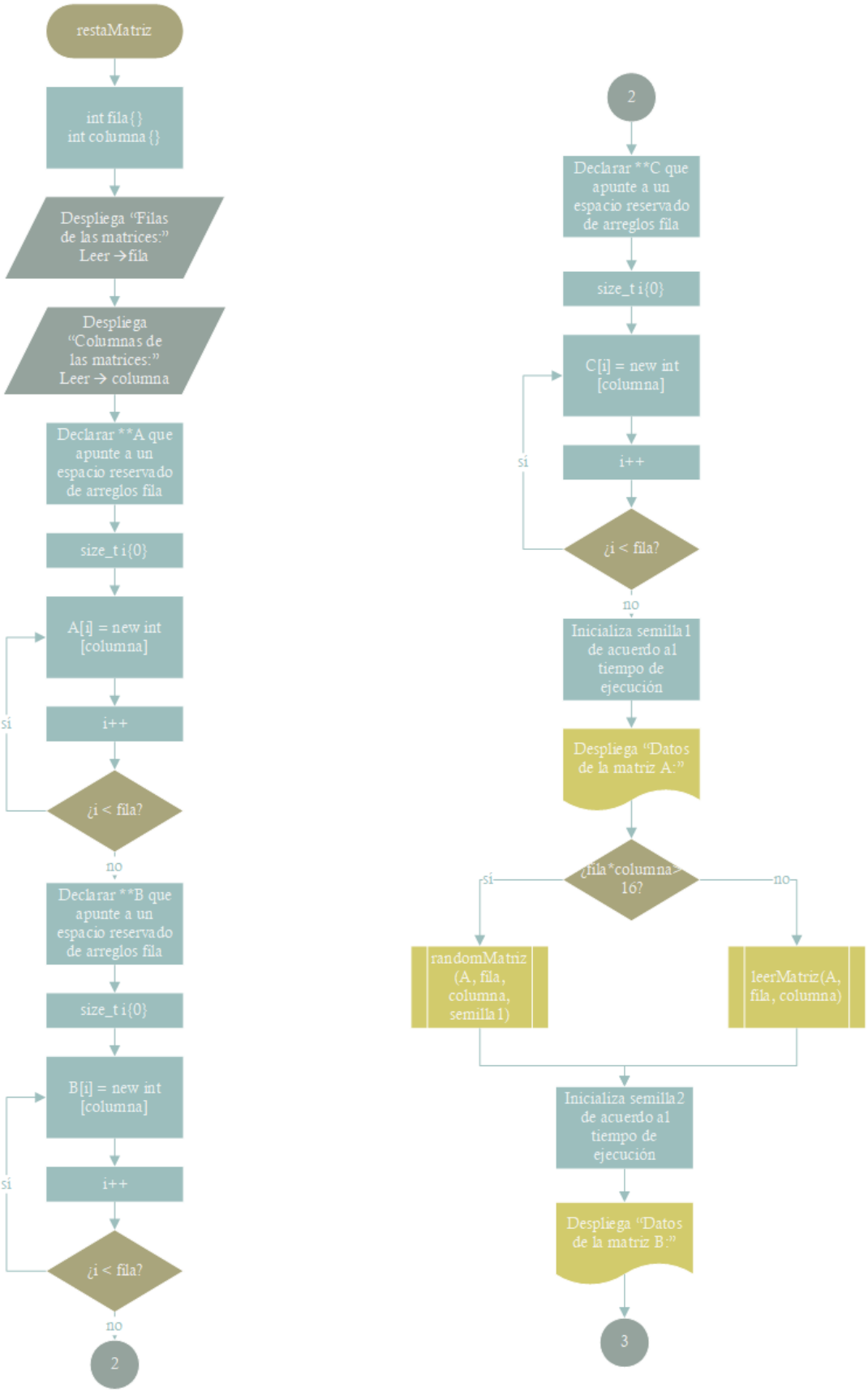

Figura 9: Diagrama función restaMatriz

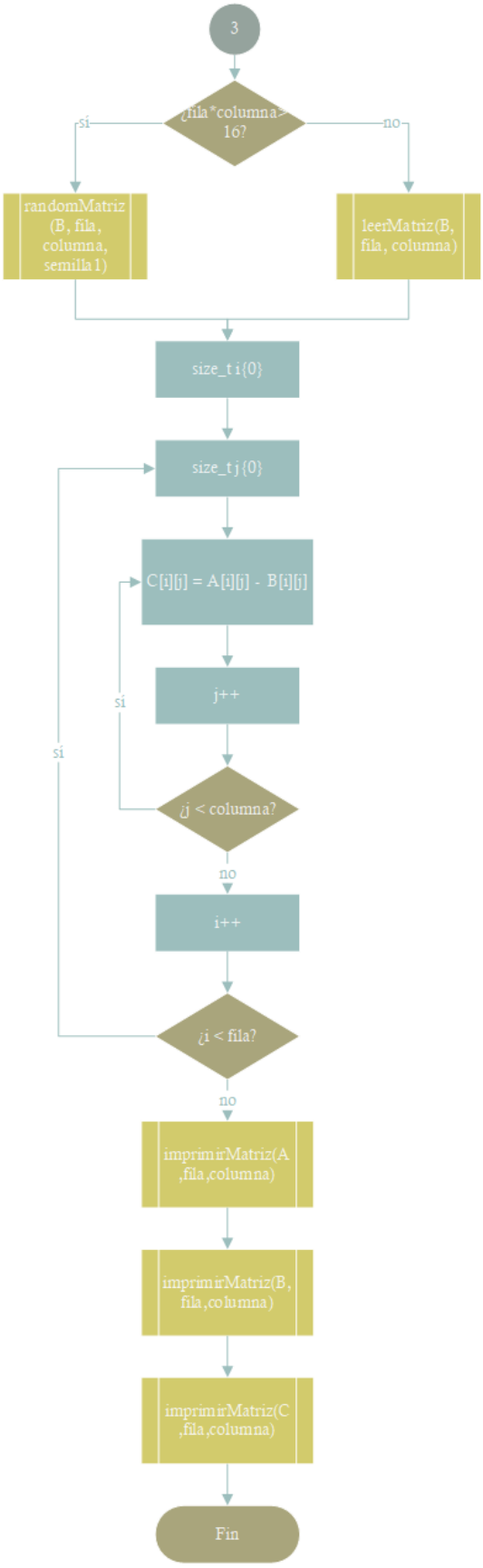

Figura 10: Diagrama función restaMatriz continuación

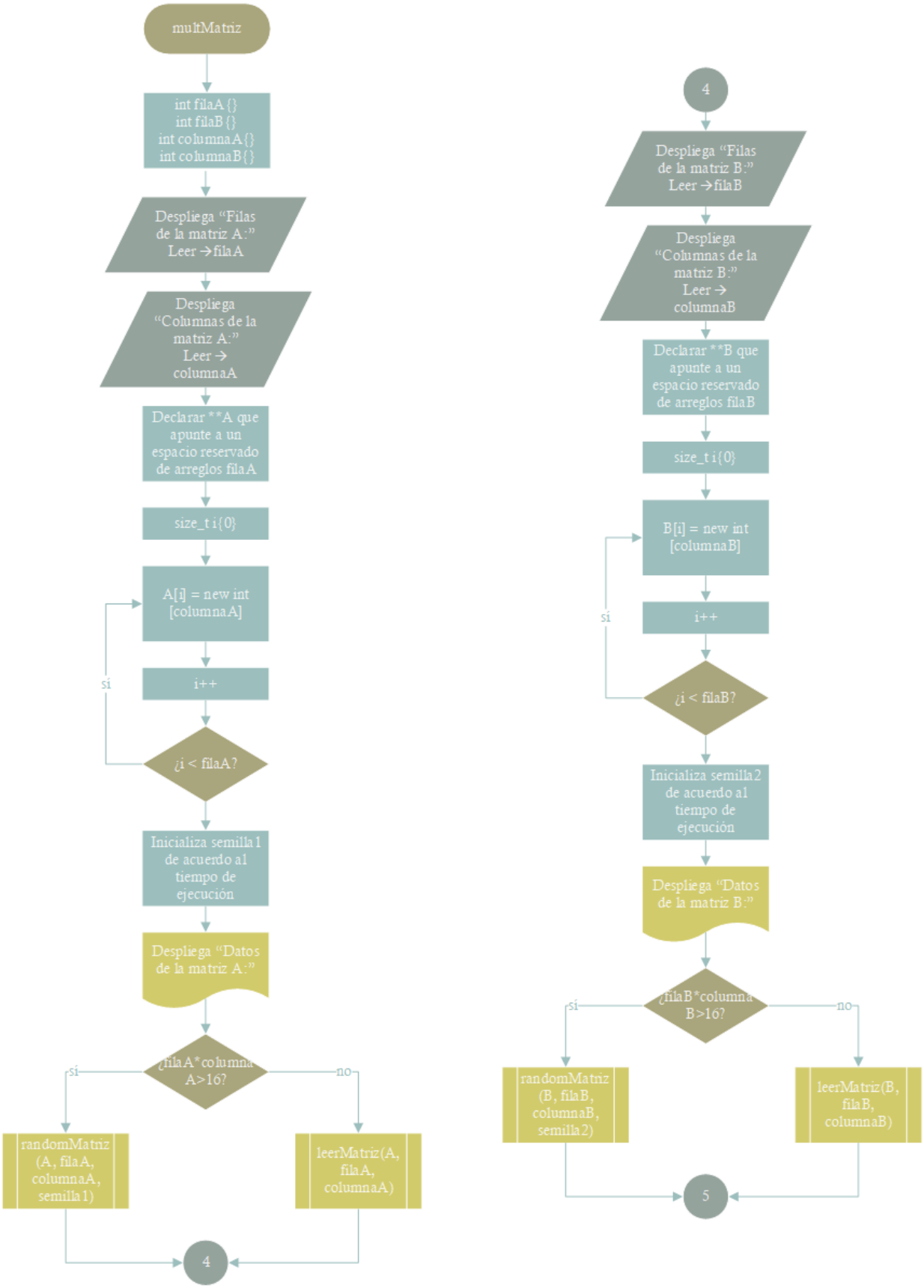

Figura 11: Diagrama función multMatriz

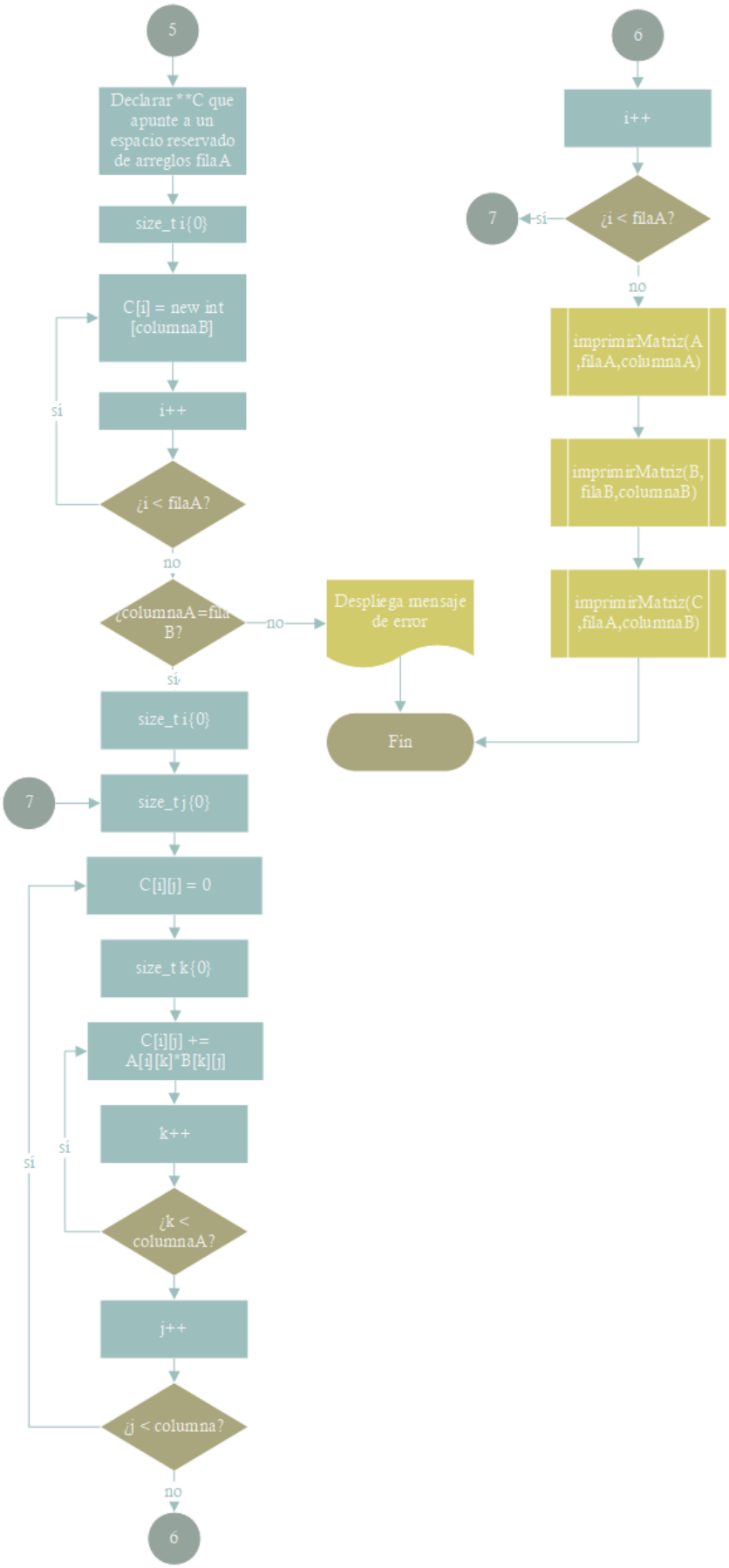

Figura 12: Diagrama función multMatriz continuación

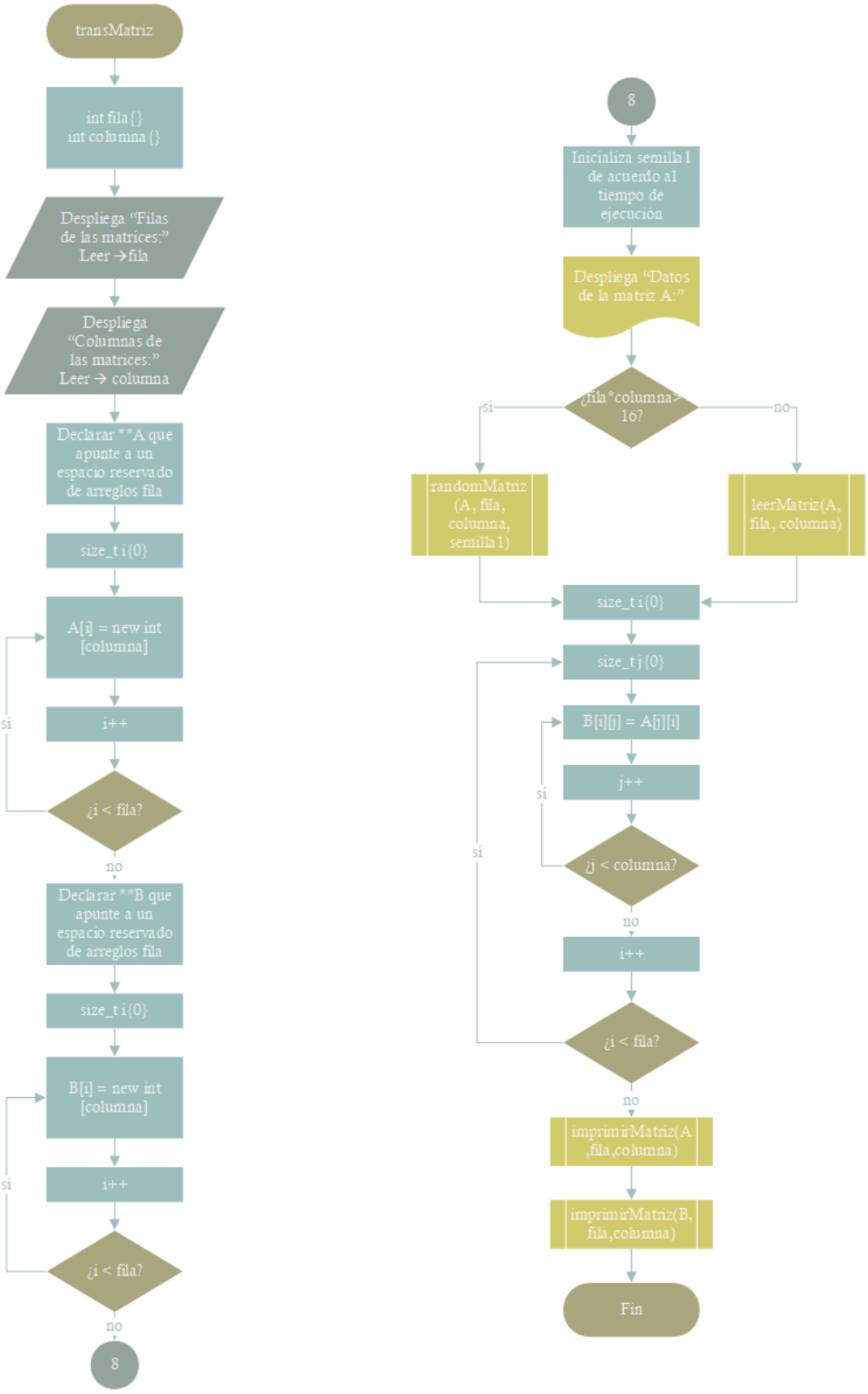

Figura 13: Diagrama función transMatriz

6. Desarrollo

A continuación se explica cada una de las funciones que conforman el programa

6.1. Main

La función principal que nos permite elegir mediante un ciclo do-while la operación que deseamos ejecutar después de haber mandado a llamar a la función "menu", se valida con un if, cin.fail y cin.bad que la opción ingresada sea valida y en caso de serlo, entra a un switch que manda a llamar a la función necesaria según la opción seleccionada.

```
29  int main(){
30
31      int opcion{}; /**int opcion Guarda el número de la opción seleccionada*/
32
33
34      do{
35          menu();
36          cin >> opcion;
37          if(cin.fail() || cin.bad()){
38              opcion = 0;
39              cin.clear();
40              cin.ignore();
41          }
42
43          switch(opcion){
44              case 1:
45                  sumaMatriz();
46                  break;
47              case 2:
48                  restaMatriz();
49                  break;
50              case 3:
51                  multMatriz();
52                  break;
53              case 4:
54                  transMatriz();
55                  break;
56              case 5:
57                  cout << "\n\nTENGA UN BUEN DIA\n\n";
58                  break;
59              default:
60                  cout << "\n\nOPCION NO VALIDA\n\n";
61                  break;
62          }
63      }while(opcion != 5);
64      return 0;
65  }
```

Figura 14: Función main

6.2. Menu

Muestra al usuario las opciones disponibles para la calculadora de matrices:
Opción 1: Suma de matrices
Opción 2: Resta de matrices
Opción 3: Multiplicación de matrices
Opción 4: Transpuesta de Matriz
Opción 5: Finalizar calculadora

```
81  void menu(){
82      cout << "\n         CALCULADORA DE MATRICES"
83           << "\n/////////////////////////////////////////"
84           << "\n1. Suma de Matrices..... (Pulsa 1)"
85           << "\n2. Resta de Matrices..... (Pulsa 2)"
86           << "\n3. Multiplicacion de Matrices... (Pulsa 3)"
87           << "\n4. Transpuesta de Matriz..... (Pulsa 4)"
88           << "\n5. Finalizar Calculadora..... (Pulsa 5)"
89           << "\n/////////////////////////////////////////"
90           << "\nNOTA: LA MATRIZ SE GENERARA DE MANERA"
91           << "\n      ALEATORIA SI PUEDE CONTENER"
92           << "\n      MAS DE 16 ELEMENTOS"
93           << "\n      (CON UN RANGO 0-100)"
94           << "\n\nSELECCIONA UNA OPCION (1-5): ";
95  }
```

Figura 15: Función menu

6.3. leerMatriz

Permite al usuario ingresar los datos de la matriz elemento por elemento. Sus parámetros son el doble apuntador a la matriz dinámica reservada, lo que permitirá guardar la matriz en el espacio de memoria que le fue asignado, el número de filas y el número de columnas. Usando un ciclo for anidado se recorren las filas y en cada fila las columnas, para ingresar elemento por elemento.

```
104 void leerMatriz(int **M, int fila, int columna){
105     cout << "\nRellenar la matriz:\n";
106     for(size_t i{0}; i < fila; i++){
107         for(size_t j{0}; j < columna; j++){
108             cout << "Matriz[" << i+1 << "][" << j+1 << "]: ";
109             cin >> M[i][j];
110         }
111     }
112 }
```

Figura 16: Función leerMatriz

6.4. randomMatriz

Crea una matriz de números aleatorios según la cantidad de filas y columnas ingresada. Recibe como parámetros el doble apuntador a la matriz dinámica reservada, número de filas, número de columnas y la semilla para generar los números aleatorios.

Una vez generado el motor y declarado una distribución uniforme con datos de tipo entero entre 0 y 100, mediante el for y el for anidado se le va asignando un número pseudoaleatorio a cada elemento.

```
122  void randomMatriz(int **M, int fila, int columna, unsigned int semilla){
123      default_random_engine motorG(semilla);
124      uniform_int_distribution<int> distM(0,100);
125
126      cout << "\nMatriz:\n";
127      for(size_t i{0}; i < fila; i++){
128          for(size_t j{0}; j < columna; j++){
129              M[i][j] = distM(motorG);
130              cout << "Matriz[" << i+1 << "][" << j+1 << "]: " << M[i][j] << endl;
131          }
132      }
133  }
```

Figura 17: Función randomMatriz

6.5. imprimirMatriz

Va imprimiendo en pantalla elemento por elemento los valores guardados en la matriz dada usando dos ciclos for. Sus parámetros son el doble apuntador a la matriz dinámica reservada, el número de filas y el número de columnas (para los ciclos for).

```
141  void imprimirMatriz(int **M, int fila, int columna){
142      for(size_t i{0}; i < fila; i++){
143          cout << "|\t";
144          for(size_t j{0}; j < columna; j++){
145              cout << M[i][j] << "\t";
146          }
147          cout << "\t|\n";
148      }
149  }
```

Figura 18: Función imprimirMatriz

6.6. sumaMatriz y restaMatriz

Efectúa la suma de dos matrices de dimensiones iguales Se crean las variables que almacenarán el número de filas y columnas y se solicitan al usuario. A continuación, se crean los espacios reservados para las dos matrices que se van a sumar y la matriz del resultado:

1. Se crea un apuntador que apunta a la dirección del apuntador del arreglo de filas que reservamos con new. Este es quien nos permitirá acceder al arreglo bidimensional.

2. Usando un for, recorremos los elementos del arreglo del doble apuntador, es decir, recorremos las filas una por una.

3. Para cada iteración, es decir, para cada fila, reservamos un arreglo con el número de columnas dado por el usuario.

Una vez reservadas las tres matrices, se inicializan las semillas en tiempos de ejecución diferentes para evitar el primer elemento igual.

Después, con un if, si el número de elementos m*n es mayor a 16, se manda a llamar a la función randomMatriz", de otra forma, se llama a "leerMatriz"

Una vez generadas las matrices, se hace la suma con dos ciclos for que nos permiten sumar elemento por elemento y guardarlo en la nueva matriz C y finalmente se imprimen las matrices mandando a llamar a la función ïmprimirMatriz". Funciona casi igual para la resta de matrices, con la única diferencia de que se resta e lugar de sumar.

```
108  void sumaMatriz(){
109      int fila{};
110      int columna{};
111
112      cout << "\nLAS MATRICES TENDRAN LAS MISMAS DIMENSIONES\n"
113           << "\nDimensiones de las matrices:";
114      cout << "\nFilas de las matrices: ";
115      cin >> fila;
116      cout << "\nColumnas de las matrices: ";
117      cin >> columna;
118
119      int **A = new int*[fila];           //doble apuntador para hacer matriz A
120      for(size_t i{0}; i < fila; i++){    //Para cada fila del apuntador se hace
121          A[i] = new int[columna];        //las columnas necesarias
122      }
123
124      int **B = new int*[fila];           //doble apuntador para hacer matriz B
125      for(size_t i{0}; i < fila; i++){
126          B[i] = new int[columna];
127      }
128
129      int **C = new int*[fila];           //doble apuntador para hacer matriz C
130      for(size_t i{0}; i < fila; i++){
131          C[i] = new int[columna];
132      }
133
134      //Se inicializan las semillas en tiempos de ejecucion diferentes para evitar primer elemento igual
135      unsigned int semilla1 = chrono::steady_clock::now().time_since_epoch().count();
```

Figura 19: Función sumaMatriz

```
146      cout << "\nDatos de la matriz B: ";
147      if(fila*columna > 16){
148          randomMatriz(B, fila, columna, semilla2);
149      }else{
150          leerMatriz(B, fila, columna);
151      }
152
153      for(size_t i{0}; i < fila; i++){
154          for(size_t j{0}; j < columna; j++){
155              C[i][j] = A[i][j] + B[i][j];
156          }
157      }
158
159      cout << "\nMatriz A:\n";
160      imprimirMatriz(A, fila, columna);
161      cout << "\nMatriz B:\n";
162      imprimirMatriz(B, fila, columna);
163      cout << "\nSuma de las matrices A+B:\n";
164      imprimirMatriz(C, fila, columna);
165      system("PAUSE()");
166      cout << "\n" << endl;
167  }
```

Figura 20: Función sumaMatriz continuación

6.7. multMatriz

Permite multiplicar dos matrices cuya condición es que el número de columnas de la primer matriz sea igual al número de filas de la segunda.

Se crean cuatro variables para almacenar el número de columnas y filas de cada matriz y se solicita al usuario que las ingrese.

Las matrices dinámicas A y B se crean de la misma manera en que se hizo y explicó en sumaMatriz. Para la matriz C se utilizan el número de filas de A y el número de columnas de B.

Si el número de columnas de A es igual al número de filas de B se usan tres ciclos for, el más externo recorre el número de fila de A, el segundo recorre el número de columnas de B y va inicializando cada elemento en 0 para posteriormente entrar a otro for que recorra las columnas de A y a cada elemento Cij le asigne la suma del producto del elemento Aik y Bkj donde k es el parámetro que va aumentando el último ciclo for. Una vez terminados los ciclos for, se imprimen las tres matrices.

En caso de no cumplirse la condición, se manda un mensaje de error.

```
230  void multMatriz(){
231      int filaA{};
232      int filaB{};
233      int columnaA{};
234      int columnaB{};
235
236      cout << "\nEL NUMERO DE FILAS DE LA MATRIZ B DEBE SER"
237           << "\nIGUAL AL NUMERO DE COLUMNAS DE LA MATRIZ A";
238      cout << "\nDimension de la matriz A:";
239      cout << "\nFilas de la matriz: ";
240      cin >> filaA;
241      cout << "Columnas de la matriz: ";
242      cin >> columnaA;
243
244      int **A = new int*[filaA];          //doble apuntador para hacer matriz A
245      for(size_t i{0}; i < filaA; i++){   //Para cada fila del apuntador se hace
246          A[i] = new int[columnaA];       //las columnas necesarias
247      }
248
249      //Se inicializan las semillas en tiempos de ejecucion diferentes para evitar primer elemento igual
250      unsigned int semilla1 = chrono::steady_clock::now().time_since_epoch().count();
251
252      cout << "\nDatos de la matriz A: ";
253      if(filaA*columnaA > 16){
254          randomMatriz(A, filaA, columnaA, semilla1);
255      }else{
256          leerMatriz(A, filaA, columnaA);
257      }
258
259      cout << "\nDimension de la matriz B:";
260      cout << "\nFilas de la matriz: ";
261      cin >> filaB;
262      cout << "Columnas de la matriz: ";
263      cin >> columnaB;
264
265      int **B = new int*[filaB];          //doble apuntador para hacer matriz B
266      for(size_t i{0}; i < filaB; i++){
267          B[i] = new int[columnaB];
268      }
```

Figura 21: Función multMatriz

```
269
270     unsigned int semilla2 = chrono::steady_clock::now().time_since_epoch().count();
271
272     cout << "\nDatos de la matriz B: ";
273     if(filaB*columnaB > 16){
274         randomMatriz(B, filaB, columnaB, semilla2);
275     }else{
276         leerMatriz(B, filaB, columnaB);
277     }
278
279     int **C = new int*[filaA];          //doble apuntador para hacer matriz C
280     for(size_t i{0}; i < filaA; i++){   //La matriz C debe tener el numero de filas
281         C[i] = new int[columnaB];       //de la matriz A y las columnas de la matriz B
282     }
283
284     if(columnaA == filaB){
285         for(size_t i{0}; i < filaA; i++){
286             for(size_t j{0}; j < columnaB; j++){
287                 C[i][j] = 0;
288                 for(size_t k{0}; k < columnaA; k++){
289                     C[i][j] += A[i][k] * B[k][j];
290                 }
291             }
292         }
293         cout << "\nMatriz A:\n";
294         imprimirMatriz(A, filaA, columnaA);
295         cout << "\nMatriz B:\n";
296         imprimirMatriz(B, filaB, columnaB);
297         cout << "\nMultiplicacion de las matrices A*B:\n";
298         imprimirMatriz(C, filaA, columnaB);
299     }else{
300         cout << "\n         NO SE PUEDEN MULTIPLICAR"
301              << "\n   EL NUMERO DE COLUMNAS DE LA MATRIZ A NO"
302              << "\nCOINCIDE CON EL NUMERO DE FILAS DE LA MATRIZ B";
303     }
304     system("PAUSE()");
305     cout << "\n" << endl;
306 }
```

Figura 22: Función multMatriz continuación

6.8. transMatriz

Calcula la transpuesta de una matriz dada. Inicia declarando las variables donde se almacenaran las filas y columnas escogidas para después solicitarlas al usuario. Una vez hecho, se crean los espacios para las matrices A (la que el usuario ingresa) y B (la transpuesta) del mismo modo que en las demás funciones donde se necesitaron, se genera una semilla y dependiendo del número de elementos de la matriz dada, se llama a randomMatriz.º "leerMatriz".

Para armar la transpuesta, se usan dos ciclos for para ir recorriendo filas y columnas y en cada iteración el elemento Bij corresponda a su respectivo Aji. Por último se imprimen las dos matrices.

```
308  void transMatriz(){
309      int fila{};
310      int columna{};
311
312      cout << "\nDimensiones de las matrices:";
313      cout << "\nFilas de las matrices: ";
314      cin >> fila;
315      cout << "\nColumnas de las matrices: ";
316      cin >> columna;
317
318      int **A = new int*[fila];           //doble apuntador para hacer matriz A
319      for(size_t i{0}; i < fila; i++){    //Para cada fila del apuntador se hace
320          A[i] = new int[columna];        //las columnas necesarias
321      }
322
323      int **B = new int*[columna];           //doble apuntador para hacer matriz B
324      for(size_t i{0}; i < columna; i++){
325          B[i] = new int[fila];
326      }
327
328      unsigned int semilla1 = chrono::steady_clock::now().time_since_epoch().count();
329
330      cout << "\nDatos de la matriz A: ";
331      if(fila*columna > 16){
332          randomMatriz(A, fila, columna, semilla1);
333      }else{
334          leerMatriz(A, fila, columna);
335      }
336
337      for(size_t i{0}; i < columna; i++){
338          for(size_t j{0}; j < fila; j++){
339              B[i][j] = A[j][i];
340          }
341      }
```

Figura 23: Función transMatriz

```
342
343      cout << "\nMatriz:\n";
344      imprimirMatriz(A, fila, columna);
345      cout << "\nTranspuesta:\n";
346      imprimirMatriz(B, columna, fila);
347      system("PAUSE()");
348      cout << "\n" << endl;
349  }
```

Figura 24: Función transMatriz continuación

7. Conclusiones

Durante la realización de esta práctica se aplicaron los conocimientos acerca de funciones, apuntadores, generador de números aleatorios y reserva dinámica de memoria. Cada parte del programa se dividió en diferentes funciones para una mejor lectura y evitar escribir el mismo código varias veces.

Hubieron complicaciones durante el desarrollo de como generar la matriz de números aleatorios debido a que en un inicio no supimos como ibamos a mandar los datos necesarios para que funcionara la función. Una vez superada esa parte, apareció otro problema en el cual el primer elemento de ambas matrices generadas de manera aleatoria eran iguales o tenían una diferencia de una unidad. Esto se comprobó al usar la función de restaMatriz en la cual el primer elemento de la matriz resultante tenía como resultado 1, 0 o -1. La solución fue generar las semillas en lineas de código no continuas. Primero generabamos una semilla, la usamos y luego generamos la segunda semilla para usarla en la segunda matriz.

También al momento de realizar de la matriz base para ser rellenada se presento el problema de como generar que tenga los elementos necesarios o solicitados. Al ser un espacio que nunca iba a ser constante, se optó por el uso de reserva dinamica de memoria mediante un doble apuntador. Primero se generaba una columna que tuviera la cantidad solicitada de filas seguido de la creación de cada fila con la cantidad solicitada de columnas.

Al realizar la función para obtener la transpuesta de la matriz se nos complicaba hacer que el programa se ejecutara adecuadamente. Resultó ser que la lectura y asignación de los valores no se había hecho correctamente. Lo que en un inicio se había hecho para la matriz A, se tenía que hacer de manera inversa en la matriz B.

Al finalizar cada cálculo solicitado, se presenta una pausa para que el usuario pueda ver el resultado antes de volver a mostrar el menú. Una vez que el usuario ha terminado sus cálculos, indica que quiere finalizar el programa.

8. Documentación

Se anexa a los archivos entregados el PDF generado por doxygen.