

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

Unidad didáctica 3071

Contenidos

1.1	Definición	4
1.1.1	Características de los arreglos.....	4
1.2	Vectores.....	5
1.2.1	Declaración y dimensionamiento de un vector	6
1.2.2	Ejercicios.....	7
1.2.3	Respuestas.....	8
1.2.4	Dimensionamiento de usuario.....	9
1.2.5	Manejo de un arreglo.....	9
1.2.6	Llenado de un arreglo	12
1.2.7	Muestra de un arreglo.....	13
1.2.8	Modificación de un arreglo	17
1.2.9	Llenado de usuario	19
1.2.10	Llenado por azar (Random).....	22
1.2.11	Ejercicios.....	25
1.2.12	Respuestas.....	26
1.2.13	Ejemplos	27
1.3	Matrices.....	66
1.3.1	Declaración y dimensionamiento de una matriz.....	66
1.3.2	Ejercicios.....	68
1.3.3	Respuestas	69
1.3.4	Dimensionamiento por usuario	70
1.3.5	Manejo de una matriz	71
1.3.6	Llenado de una matriz.....	73
1.3.7	Muestra de una matriz.....	80
1.3.8	Modificación de una matriz	83
1.3.9	Llenado de usuario	87
1.3.10	Llenado por azar (Random).....	90

Capítulo IV.

Arreglos unidimensionales y multidimensionales



Objetivo general

Adquirir conocimientos sobre las características y manejo de programación de los arreglos unidimensionales y multidimensionales.



Objetivos específicos

Conocer la definición y las características de los arreglos unidimensionales y multidimensionales.

Comprender el funcionamiento del ciclo Para (For) aplicado a los arreglos de una y dos dimensiones.



Objetivos específicos

Implementar soluciones programadas para la resolución de ejercicios con arreglos de una y dos dimensiones.

1.1 Definición

Antes de definir un arreglo, recordaremos la conceptualización de variable. De forma muy resumida, una variable es un espacio reservado en memoria RAM en el cual se almacena un dato para el algoritmo. La definición de arreglo es muy similar a la de una variable, ambos son espacios en memoria RAM, pero la diferencia es que el arreglo es una estructura que puede almacenar más de un dato para el algoritmo. Podríamos decir que un arreglo es la unión de dos o más variables y que se ubican en una misma estructura.

1.1.1 Características de los arreglos

En cuanto a sus características los arreglos, al igual que las variables, tienen nombre y tipo de dato, y sus normas para esas características son las mismas a las estipuladas en el capítulo anterior.

Un detalle importante es que todos los datos que almacena un arreglo son del mismo tipo, por ejemplo, si declaramos un arreglo de tipo entero este solo podrá almacenar valores enteros en cada uno de sus espacios, de la misma forma se trabaja si se declara de tipo real, carácter o lógico (booleano).

Además del nombre y tipo de dato, los arreglos tienen dimensión; esta es básicamente el tamaño (cantidad de espacios) que el arreglo tendrá y se debe definir **antes de ser utilizado** en el algoritmo, el tamaño del arreglo no se podrá cambiar durante la ejecución del algoritmo. Esta característica también establece una clasificación para los arreglos, los de **una** dimensión son llamados **vectores** y los de **dos** dimensiones se denominan **matrices**.

Variable

3

Solo almacena un dato.
Por ejemplo, el número 3.

Vector

1	13	38	5	10
---	----	----	---	----

Almacena varios datos en una misma estructura.

El vector de la imagen puede almacenar cinco datos del mismo tipo, por ejemplo, cinco números enteros.

El vector tiene solo una dimensión.

Matriz

15	65.1	32	25.9	20.2
13.6	9	42	12.73	32
10	6	7.1	25	2

Almacena varios datos en una misma estructura.

La matriz de la imagen puede almacenar quince datos del mismo tipo, por ejemplo, quince números reales. Asimismo, el arreglo del ejemplo tiene tres filas y cinco columnas, es decir dos dimensiones.

1.2 Vectores

Los vectores son arreglos unidimensionales, tienen una sola dimensión, y se pueden representar como se muestra a continuación:

1	13	38	5	10
---	----	----	---	----

Vector de 5 posiciones (celdas).

Notemos que en cada celda del vector se pueden almacenar valores por separado, eso sí, todos son del **mismo tipo de dato**. Éstos se manejan de forma individual, por lo que si se desea modificar o mostrar se debe usar el nombre del vector al que pertenecen y el número de celda donde está ubicado el dato.

1.2.1 Declaración y dimensionamiento de un vector

Al igual que las variables, los vectores deben ser declarados, la declaración de un vector se realiza con la siguiente estructura:

Definir **Nombre del vector** Como **tipo de dato**;

Donde **Nombre del vector** será el identificador que se le otorgará al arreglo para su manejo en el algoritmo; y **tipo de dato** será la definición de los valores que podrá almacenar el arreglo, ya sea datos de tipo real, enteros, caracteres o lógicos.

Si quisiéramos declarar un vector que almacene datos numéricos de tipo real, por ejemplo, salarios, lo haríamos así:

Definir **VecSalarios** Como **Real**;

Luego de declarar un arreglo, se debe **dimensionar**, esto se hace para especificar el tamaño máximo (cantidad de celdas) que tendrá el vector. La dimensión del vector se hace de la siguiente forma:

Dimension VecSalarios (5);

El dimensionamiento de un arreglo siempre será un número **entero**, ya que no puede existir un arreglo con, por ejemplo, 5.7 celdas; o tiene 5 o tiene 6 celdas. Por ende, la **cantidad de posiciones** que posee un vector siempre será un **entero**. En el siguiente ejemplo se evidencia una declaración y dimensionamiento de un vector de 10 posiciones:

```
1 Algoritmo DeclaracionDimensionamiento
2
3     // Declaración de variables
4     Definir Vector Como Entero;
5
6     // Dimensión del vector
7     Dimension Vector(10);
8
9 FinAlgoritmo
```

1.2.2 Ejercicios

Declare y dimensione los vectores especificados, considere el tipo de dato y la dimensión.

1. Vector de 9 posiciones que almacena salarios de los funcionarios de un banco.

Declaración:

Dimensión:

2. Arreglo unidimensional de 6 celdas que almacena la cantidad de estudiantes de cada nivel de una escuela primaria.

Declaración:

Dimensión:

3. Vector de 25 posiciones que almacena los nombres de los empleados de una fábrica.

Declaración:

Dimensión:

4. Vector de 5 posiciones que almacena datos de falso o verdadero para un juego.

Declaración:

Dimensión:

5. Arreglo unidimensional de 2500 celdas que almacena los promedios ponderados y los estudiantes de un colegio.

Declaración:

Dimensión:

1.2.3 Respuestas

Las siguientes respuestas son representativas, ya que los nombres de los vectores pueden variar, lo que se debe tomar en consideración es que deben cumplir con las normas para identificar arreglos.

1. Vector de 9 posiciones que almacena salarios de los funcionarios de un banco.
Declaración: Definir SalariosFuncionarios Como Real;
Dimensión: Dimension SalariosFuncionarios(9);
2. Arreglo unidimensional de 6 celdas que almacena la cantidad de estudiantes de cada nivel de una escuela primaria.
Declaración: Definir CantidadEstudiantes Como Entero;
Dimensión: Dimension CantidadEstudiantes (6);
3. Vector de 25 posiciones que almacena los nombres de los empleados de una fábrica.
Declaración: Definir NombreEmpleados Como Caracter;
Dimensión: Dimension NombreEmpleados (25);
4. Vector de 5 posiciones que almacena si un personaje está dañado o no (datos de falso o verdadero) para un juego.
Declaración: Definir PersonajeDaniado Como Logico;
Dimensión: Dimension PersonajeDaniado(5);
5. Arreglo unidimensional de 2500 celdas que almacena los promedios ponderados y los estudiantes de un colegio.
Declaración: Definir PromediosPonderados Como Real;
Dimensión: Dimension PromediosPonderados (2500);

1.2.4 Dimensionamiento de usuario

Un arreglo puede ser dimensionado tanto por la persona que programa el algoritmo como por el usuario que emplea el sistema. La forma que estudiamos anteriormente es un dimensionamiento por programación, ahora veremos un dimensionamiento por el usuario.

En el dimensionamiento por usuario, el arreglo se declara y, antes de dimensionarlo, se consulta la cantidad de celdas que se desean, la respuesta del usuario se almacena en una variable de tipo entero y con esta variable se hace el dimensionamiento. Veamos un ejemplo:

```
1  Algoritmo DimensionamientoporUsuario
2
3      // Declaración de variables
4      Definir Vector Como Entero;
5      Definir TamanoVector Como Entero;
6
7      // Inicialización de variables
8      TamanoVector=0;
9
10     Escribir "Digite el tamaño que desea para el vector.";
11     Leer TamanoVector;
12
13     // Dimensionamiento del vector
14     Dimension Vector (TamanoVector);
15
16 FinAlgoritmo
```

En el código anterior, se muestra únicamente la declaración y dimensionamiento de un vector, notemos que antes de dimensionar el arreglo se solicita y lee el tamaño, o lo que es igual la cantidad de celdas. Luego, en la instrucción **Dimension**, se utiliza la variable **TamanoVector** en lugar de un número entero específico.

La diferencia de que el usuario defina el tamaño del arreglo en comparación con el dimensionamiento desde la programación es que en cada ejecución del algoritmo se puede usar un tamaño de vector diferente. Ahora, es importante aclarar que este tamaño no cambia durante la ejecución, es decir, si el usuario define un tamaño de 20 se trabajará con ese tamaño hasta que finalice la ejecución del algoritmo. Ya en otra ejecución el usuario podrá especificar el mismo u otro tamaño para el arreglo.

1.2.5 Manejo de un arreglo

Para el manejo de un arreglo siempre se debe considerar la celda donde se desea gestionar un dato, ya sea almacenarlo, modificarlo o mostrarlo. Regresemos por un momento a las

variables, para almacenar un dato en una variable, se otorga dicho dato por medio de un operador de asignación (=) de la siguiente forma:

Numero=5;

Donde **Numero** es una variable a la que se le está asignando el valor **cinco**, por ende, Numero vale, de ahí en adelante, cinco.

Ahora, si tratamos de hacer lo mismo con el vector debemos hacerlo indicando la celda donde se desea almacenar el dato, para eso utilizamos un elemento llamado índice. El **índice** es una **variable de tipo entero** que funcionará como **indicador de la celda** donde se desea trabajar.

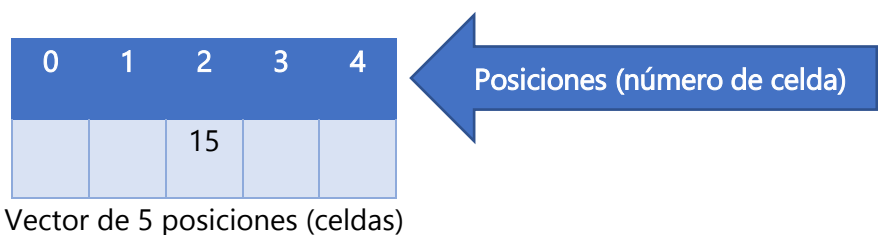
Siempre debemos emplear el **índice** para almacenar un dato en un vector, por ende, la siguiente instrucción es **errónea**:

Vector=15;

Ya que no se indica una celda donde guardar el valor 15, si quisiéramos colocar ese dato (el 15) en la celda número dos (2) del vector lo debemos hacer de la siguiente forma:

Vector(2) = 15;

Gráficamente, se vería así:



Al analizar el vector anterior, podemos apreciar una característica fundamental de los arreglos, la numeración de las celdas. En todos los arreglos, las celdas están identificadas con números enteros, esta identificación inicia con el número cero (0). Debemos tener claro la relación entre la dimensión del arreglo y la numeración de las celdas, si un arreglo es de 10 posiciones, la numeración de sus celdas irá de cero (0) a nueve (9).

0	1	2	3	4	5	6	7	8	9
1	13	38	5	10	5	6	9	0	15

Vector de 10 posiciones con la numeración de sus celdas

Regresando a nuestro vector de 5 posiciones:

0	1	2	3	4
		15		

¿Qué pasaría si deseamos llenarlo de ceros? Es decir, que en cada posición se almacene dicho número. Una forma de hacer esto es asignando en cada posición el cero de la siguiente forma:

Vector(0)=0;

Vector(1)=0;

Vector(2)=0;

Vector(3)=0;

Vector(4)=0;

Luego de las instrucciones anteriores tendríamos el vector lleno:

0	1	2	3	4
0	0	0	0	0

El **método anterior es ineficiente** y no se utiliza tal cual, lo que se debe hacer es utilizar ciclos y una variable para identificar el índice de la celda.

De ahora en adelante no utilizaremos un número determinado para representar el número de celda, sino que utilizaremos la **variable de tipo entero** llamada **Celda**. Llenemos el vector de números cinco, pero utilizaremos la variable Celda. Si lo hacemos con el método ineficiente tendremos algo así:

```
Celda=0;  
  
Vector(Celda)=5;  
  
Celda=1;  
  
Vector(Celda)=5;  
  
Celda=2;  
  
Vector(Celda)=5;  
  
Celda=3;  
  
Vector(Celda)=5;  
  
Celda=4;  
  
Vector(Celda)=5;
```

Como podemos ver, ahora el método ineficiente, **lo es aún más**, pero nos muestra algo interesante, al utilizar una variable en lugar de un número determinado para el campo que representa a la posición del vector, podemos **modificar el valor de dicha variable** y se **modificará también el índice** que hace referencia a la **posición**.

1.2.6 Llenado de un arreglo

Para llenar un arreglo de mejor forma, utilizaremos un ciclo para hacer que el método de llenado del vector sea eficiente, lo haremos con el **ciclo Para (For)** de la siguiente forma:

```
Para Celda=0 hasta 4 con Paso 1 Hacer  
  
    Vector(Celda)=5;  
  
Fin Para
```

Al ejecutar las instrucciones anteriores, podemos descubrir que, cada vez que se ingresa al ciclo, la variable Celda modificará su valor. La primera vez que se ingresa al Para, Celda tiene un valor de cero (0) y se ejecuta la instrucción `Vector(Celda)=5`; por ende, se almacenará un cinco en la posición cero del vector. En el segundo ingreso al ciclo, Celda vale 1 y se vuelve a ejecutar la asignación en el vector; todo este proceso de ingresos y asignaciones se repite tres veces más (para un total de 5). El ciclo finaliza cuando Celda tenga un valor de 5, ya que va de cero a cuatro (0 a 4), con ese valor (5) no entrará al ciclo.

Si hacemos el método anterior en un algoritmo completo tendríamos lo siguiente:

```
1  Algoritmo LlenadoVector
2      //Declaración de variables
3      Definir Vector Como Entero;
4      Definir Celda Como Entero;
5
6      //Dimensión del vector
7      Dimension Vector(5);
8
9      //Inicialización de variables
10     Celda=0;
11
12     //Llenado del vector
13     Para Celda=0 Hasta 4 Con Paso 1 Hacer
14         Vector(Celda)=5;
15     FinPara
16
17 FinAlgoritmo
```

1.2.7 Muestra de un arreglo

Ya sabemos cómo llenar un vector, ahora, para mostrarlo utilizaremos el mismo método de llenado, pero con un pequeño cambio dentro del ciclo Para; analicemos el siguiente fragmento de código:

```
Para Celda=0 Hasta 4 Con Paso 1 Hacer
    Escribir Vector(Celda);
FinPara
```

Como podemos ver, tanto para el código de llenado y como para el de muestra se utiliza un ciclo **Para**, la diferencia es que cambiamos la instrucción de asignación `Vector(Celda)=5` por la instrucción `Escribir Vector(Celda)`. Un error muy común es tratar de mostrar el vector como

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

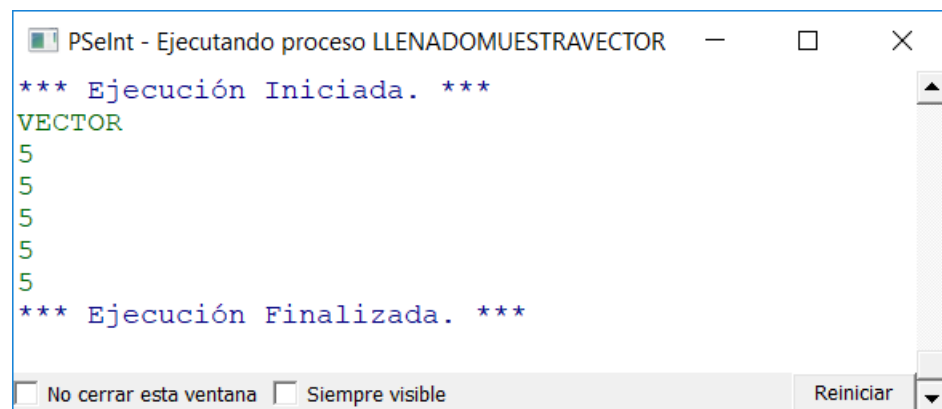
si fuese una variable, de esta forma: **Escribir Vector**; lo cual el sistema lo tomará como un error.

Debemos tener claro que, si un vector se **llena posición por posición**, se debe **mostrar posición por posición** y más adelante veremos que si desea **modificar** el contenido de un vector se debe hacer también **posición por posición**.

Veamos el pseudocódigo completo de llenado y muestra de un vector:

```
1  Algoritmo LlenadoMuestraVector
2      //Declaración de variables
3      Definir Vector Como Entero;
4      Definir Celda Como Entero;
5
6      //Dimensión del vector
7      Dimension Vector(5);
8
9      //Inicialización de variables
10     Celda=0;
11
12     //Llenado del vector
13     Para Celda=0 Hasta 4 Con Paso 1 Hacer
14         Vector(Celda)=5;
15     FinPara
16
17     //Muestra del vector
18     Escribir "VECTOR";
19     Para Celda=0 Hasta 4 Con Paso 1 Hacer
20         Escribir Vector(Celda);
21     FinPara
22
23 FinAlgoritmo
```

Esto mostraría la siguiente salida en pantalla:



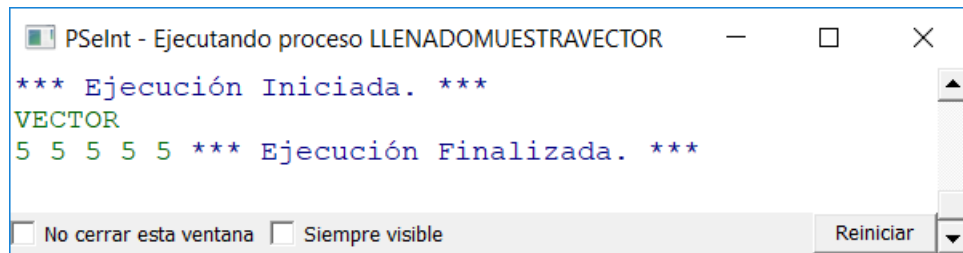
```
PSeInt - Ejecutando proceso LLENADOMUESTRAVECTOR
*** Ejecución Iniciada. ***
VECTOR
5
5
5
5
5
*** Ejecución Finalizada. ***
☐ No cerrar esta ventana ☐ Siempre visible [Reiniciar]
```

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

Para mostrar el vector de forma horizontal podemos utilizar la instrucción **Sin Saltar** y un espacio en blanco entre comillas, en el **Escribir**, de la siguiente forma:

Escribir Vector(Celda), " ", Sin Saltar;

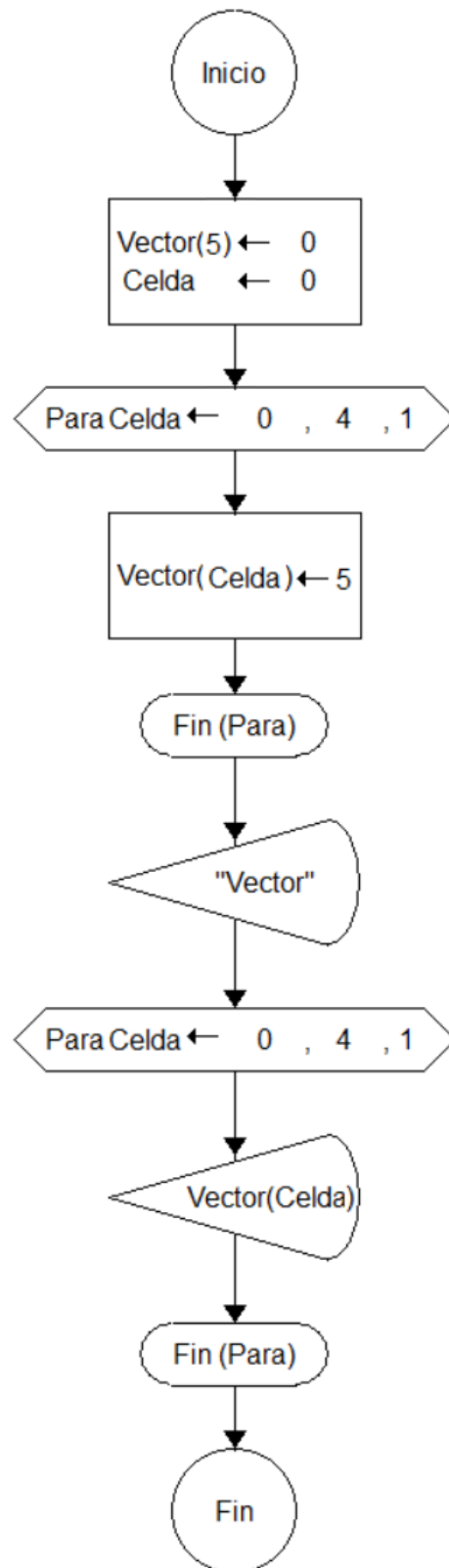
La salida sería así:



```
*** Ejecución Iniciada. ***
VECTOR
5 5 5 5 5 *** Ejecución Finalizada. ***
```

☐ No cerrar esta ventana ☐ Siempre visible Reiniciar

El diagrama de flujo del código anterior quedaría de la siguiente forma:



1.2.8 Modificación de un arreglo

Al igual que para llenar y mostrar un vector, si se desea modificar sus valores se debe hacer celda por celda, analicemos el ejemplo anterior, pero cada número 5 lo aumentaremos en 1, es decir, el vector quedará lleno de números seis, esto lo haremos en un ciclo luego de la muestra del vector y, al final, mostraremos de nuevo el vector, pero esta vez la salida será diferente, el código que agregaremos es el siguiente:

```
Para Celda=0 Hasta 4 Con Paso 1 Hacer
    Vector(Celda)= Vector(Celda) + 1;
FinPara
```

El fragmento anterior, almacena en la posición Celda del vector la suma de lo que contiene esa misma posición, que en el ejemplo es un cinco, más un uno, por lo que la posición tendrá guardado un seis. La versión en pseudocódigo es la siguiente:

```
1  Algoritmo ModificaVector
2
3      // Declaración de variables
4      Definir Vector Como Entero;
5      Definir Celda Como Entero;
6
7      // Dimensión del vector
8      Dimension Vector(5);
9
10     // Inicialización de variables
11     Celda = 0;
12
13     // Llenado del vector
14     Para Celda=0 Hasta 4 Hacer
15     |   Vector(Celda)=5;
16     FinPara
17
18     // Muestra del vector
19     Escribir "VECTOR";
20     Para Celda=0 Hasta 4 Hacer
21     |   Escribir Vector(Celda), ' ', Sin Saltar;
22     FinPara
23
24     // Modifica el vector sumando 1 al contenido de cada posición
25     Para Celda=0 Hasta 4 Hacer
26     |   Vector(Celda)=Vector(Celda) + 1;
27     FinPara
28
29     // Muestra del vector nuevamente
30     Escribir "";
31     Escribir "VECTOR MODIFICADO";
32     Para Celda=0 Hasta 4 Hacer
33     |   Escribir Vector(Celda), ' ', Sin Saltar;
34     FinPara
35
36 FinAlgoritmo
```

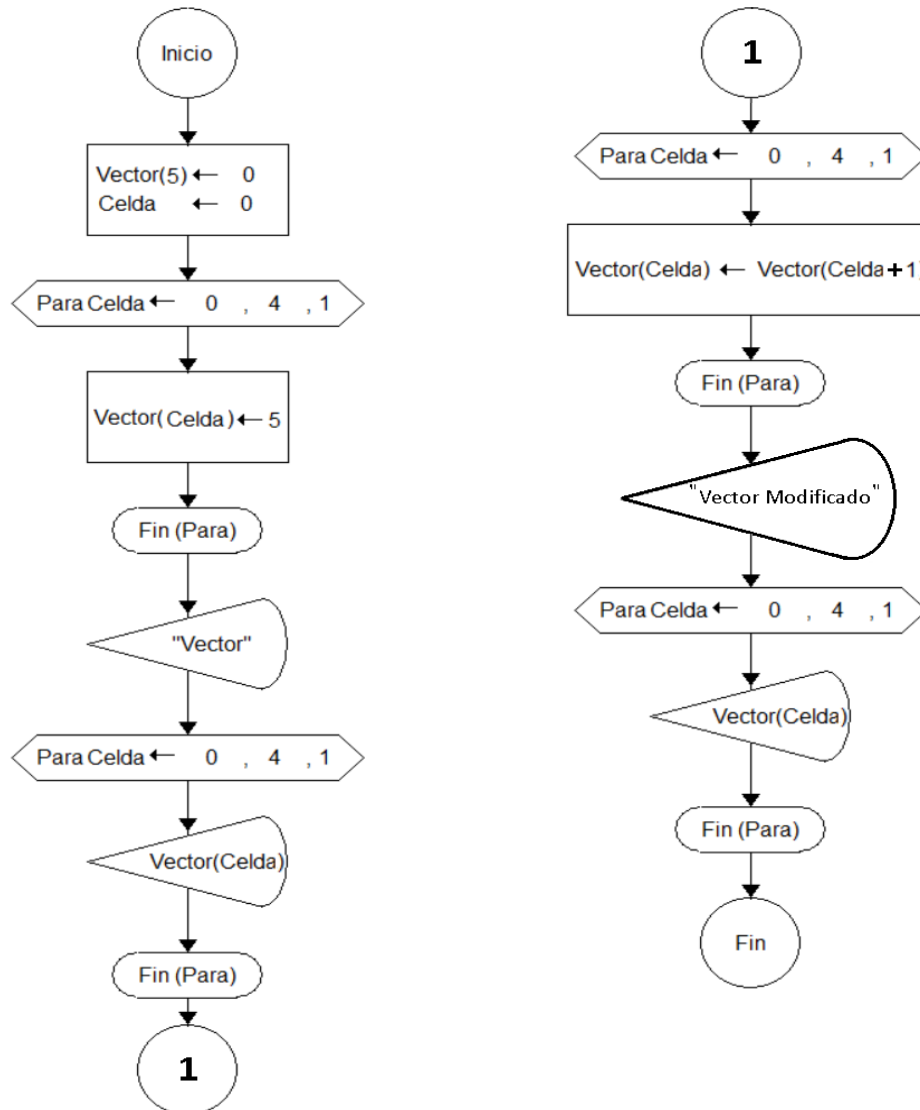
La salida del código anterior es la siguiente:

```
PSeInt - Ejecutando proceso MODIFICAVECTOR

*** Ejecución Iniciada. ***
VECTOR
5 5 5 5 5
VECTOR MODIFICADO
6 6 6 6 6 *** Ejecución Finalizada. ***

☐ No cerrar esta ventana ☐ Siempre visible Reiniciar
```

El diagrama de flujo queda de la siguiente forma:



1.2.9 Llenado de usuario

Ya hemos analizado el llenado de un vector de forma automática, pero **¿cómo haríamos si deseamos que el vector lo llene el usuario?**, la modificación se debe hacer en el ciclo Para de llenado, el código sería el siguiente:

// Llenado del vector

Escribir "Digite los datos para el vector";

Para Celda=0 Hasta 4 Hacer

Leer Vector(Celda);

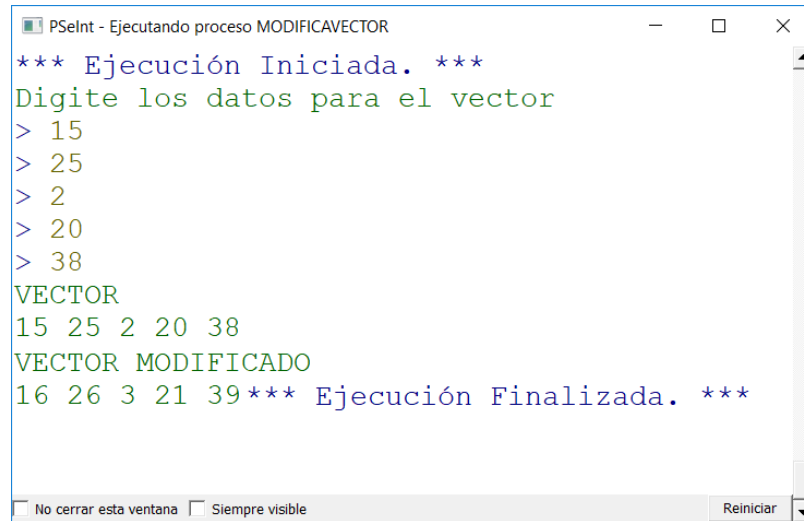
FinPara

La primera modificación fue un **mensaje**, antes del ciclo Para, **solicitando al usuario** que digitara los datos para el vector; la otra modificación es la instrucción **Leer**, dentro del Para, con el Leer se le brinda la oportunidad al usuario de digitar cualquier número entero para llenar el vector. Tomemos el último ejemplo para ver esta modificación en pseudocódigo:

```
1  Algoritmo ModificaVector
2
3      // Declaración de variables
4      Definir Vector Como Entero;
5      Definir Celda Como Entero;
6
7      // Dimensión del vector
8      Dimension Vector(5);
9
10     // Inicialización de variables
11     Celda = 0;
12
13     // Llenado del vector
14     Escribir "Digite los datos para el vector";
15     Para Celda=0 Hasta 4 Hacer
16     |     Leer Vector(Celda);
17     FinPara
18
19     // Muestra del vector
20     Escribir "VECTOR";
21     Para Celda=0 Hasta 4 Hacer
22     |     Escribir Vector(Celda), ' ', Sin Saltar;
23     FinPara
24
25     // Modifica el vector sumando 1 al contenido de cada posición
26     Para Celda=0 Hasta 4 Hacer
27     |     Vector(Celda)=Vector(Celda) + 1;
28     FinPara
29
30     // Muestra del vector nuevamente
31     Escribir "VECTOR MODIFICADO";
32     Para Celda=0 Hasta 4 Hacer
33     |     Escribir Vector(Celda), ' ', Sin Saltar;
34     FinPara
35
36 FinAlgoritmo
```

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

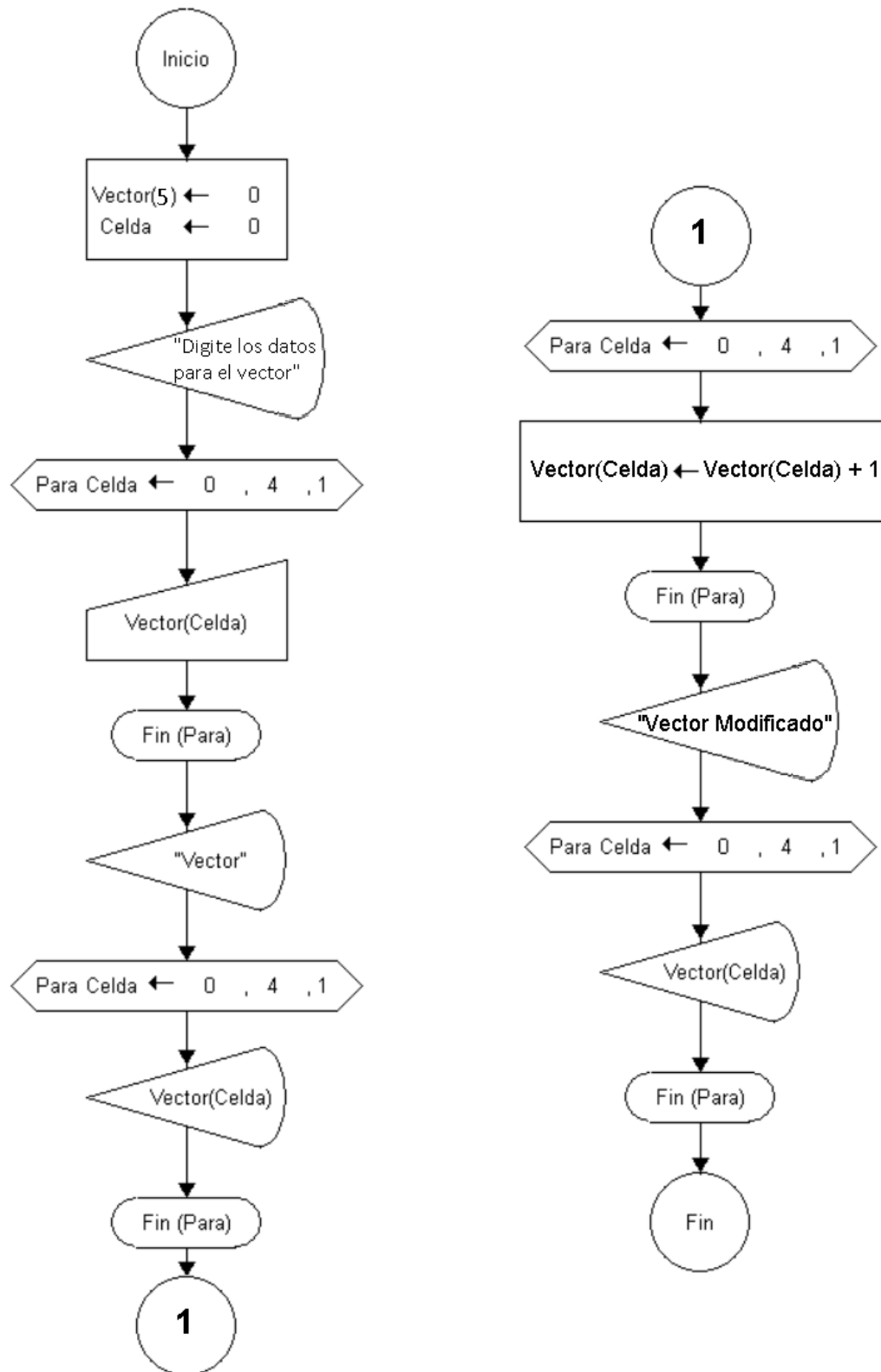
El código generaría la siguiente salida en pantalla:



```
*** Ejecución Iniciada. ***
Digite los datos para el vector
> 15
> 25
> 2
> 20
> 38
VECTOR
15 25 2 20 38
VECTOR MODIFICADO
16 26 3 21 39*** Ejecución Finalizada. ***
```

Notemos que a los valores que digitó el usuario se le suma un uno, y finalmente, se muestra el vector modificado.

El diagrama de flujo del código anterior es el siguiente:



1.2.10 Llenado por azar (Random)

Analicemos una última forma de llenar un vector, lo haremos con números **al azar**, esto quiere decir que los números los dará el sistema **aleatoriamente** y dentro de un rango que debemos definir. Anteriormente, el arreglo pudo llenarse con números al azar por el usuario, pero en este caso lo hará el sistema, lo que hace mucho más rápido el llenado del arreglo.

Estudiemos primero el funcionamiento de la **función azar**, esta tiene la siguiente estructura: **azar(N)**; donde N es la cantidad de números con que se dispondrá para elegir un valor aleatorio iniciando desde cero. Por ejemplo: **azar(5)**; seleccionará un número aleatorio entre 0 y 4 (0, 1, 2, 3, 4); **azar(10)** seleccionará un valor entre 0 y 9.

El número generado por la función **azar** debe almacenarse en una variable o, como es nuestro caso, en una posición del arreglo, por lo que se debe hacer de la siguiente forma:

Variable= azar(5);

Si lo hacemos en un arreglo, debemos asignar el valor de la función **azar** a una posición del vector, esto se realiza dentro de un ciclo, la instrucción específica de llenado sería así:

Vector(Celda)=azar(15);

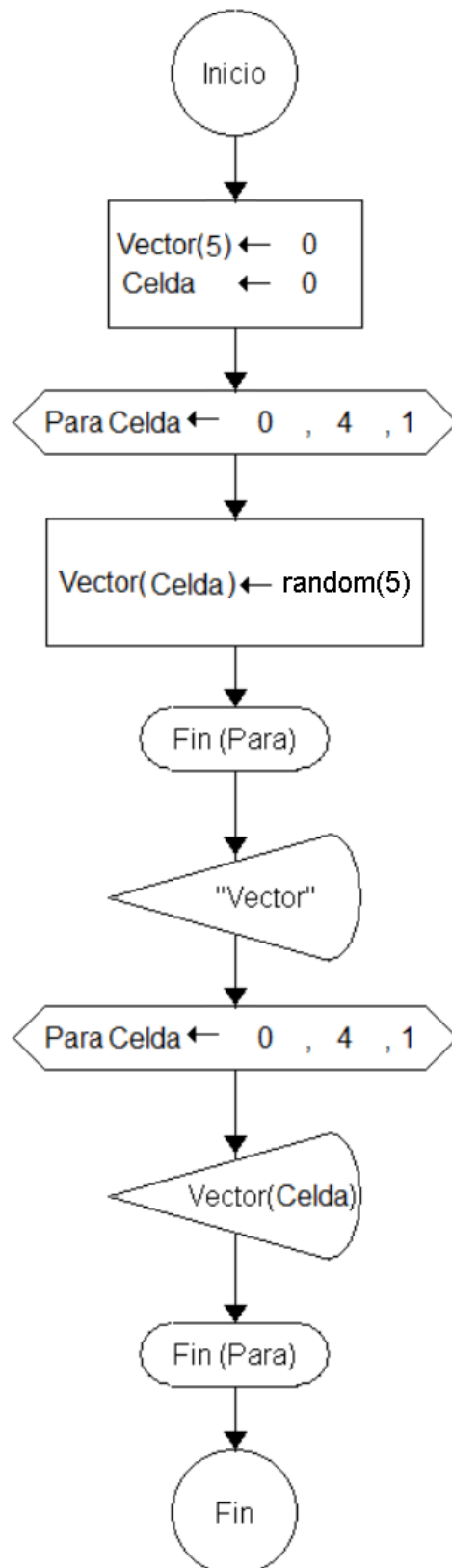
Veamos un ejemplo de llenado y muestra de un vector de 5 posiciones, cuyos valores son aleatorios, el código sería el siguiente:

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

```
1 Algoritmo LlenadoMuestraVectorAzar
2
3     // Declaración de variables
4     Definir Vector Como Entero;
5     Definir Celda Como Entero;
6
7     // Dimensión del vector
8     Dimension Vector(5);
9
10    // Inicialización de variables
11    Celda = 0;
12
13    // Llenado del vector con números aleatorios del 0 al 4
14    Para Celda=0 Hasta 4 Hacer
15    |   Vector(Celda)=azar(5);
16    FinPara
17
18    // Muestra del vector
19    Escribir "VECTOR";
20    Para Celda=0 Hasta 4 Hacer
21    |   Escribir Vector(Celda), ' ', Sin Saltar;
22    FinPara
23
24 FinAlgoritmo
```

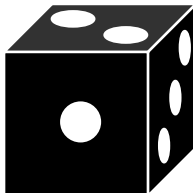
En el código, podemos ver que el cambio se dio en la línea 15, donde se emplea la función azar para generar un valor del cero al cuatro (0-4) y se asigna a una celda del vector, este proceso se repite por cada posición del arreglo.

El diagrama de flujo del código anterior es el presentado a continuación:



Notemos que en el programa DFD, empleado para crear el diagrama de flujo, la función para números aleatorios se llama **Random**, pero su funcionamiento es el mismo que la llamada **azar**.

Hasta el momento nuestro rango con la función azar ha ido desde cero (0) hasta un número menos indicado entre paréntesis, pero si deseamos variar el rango podemos hacerlo de forma muy sencilla, tomemos como ejemplo un dado.



Al analizar el rango de un dado este tiene números que van del 1 al 6, por ende, debemos modificar nuestra función azar. Si hacemos lo siguiente: `azar(6)`, esto nos generará un rango de cero a cinco (0-5), lo cual no nos funcionaría para un dado. La solución es sumar una unidad al número generado por la función azar, es decir:

`azar(6)+1.`

Con esta modificación si el azar genera un cero le sumará uno (0+1); y si genera un cinco el número final será seis (5+1).

Como podemos ver en el ejemplo anterior, a la función azar se le puede aplicar operaciones aritméticas por lo que los rangos que se pueden crear son muy variados.

1.2.11 Ejercicios

Para finalizar nuestra comprensión de la función azar (random), realice las siguientes funciones azar de acuerdo con el rango dado, por ejemplo:

Rango: 0-20.

Función: `azar(21);`

1. Rango: 2-15.

Función:

2. Rango: 0-101.

Función:

3. Rango: 10-99.

Función:

4. Rango: 200-500.

Función:

5. Rango: 4-22.

Función:

1.2.12 Respuestas

1. Rango: 2-15.

Función: $\text{azar}(14)+2$;

2. Rango: 0-101.

Función: $\text{azar}(102)$;

3. Rango: 10-99.

Función: $\text{azar}(90)+10$;

4. Rango: 200-506.

Función: $\text{azar}(307)+200$;

5. Rango: 4-22.

Función: $\text{azar}(19)+4$;

Si tomamos el ejemplo del llenado del vector usando `azar` (Random), pero lo modificamos para que se llene con números en un rango de 50 a 100 tendremos el siguiente código:

```
1 Algoritmo LlenadoMuestraVectorAzarRangoEsp
2
3     // Declaración de variables
4     Definir Vector Como Entero;
5     Definir Celda Como Entero;
6
7     // Dimensión del vector
8     Dimension Vector(5);
9
10    // Inicialización de variables
11    Celda = 0;
12
13    // Llenado del vector con números aleatorios del 50 al 100
14    Para Celda=0 Hasta 4 Hacer
15    |   Vector(Celda)=azar(51)+50;
16    FinPara
17
18    // Muestra del vector
19    Escribir "VECTOR";
20    Para Celda=0 Hasta 4 Hacer
21    |   Escribir Vector(Celda), ' ', Sin Saltar;
22    FinPara
23
24 FinAlgoritmo
```

Veamos que la única línea que cambia es la 15, y de ahí lo que modificamos fue el rango y suma a la función azar. En un diagrama de flujo el cambio sería el mismo que en el código.

Ya se han analizado diversos procesos de los arreglos: su declaración, dimensionamiento, llenado, muestra, modificación; y las estructuras que se emplearon para esas acciones son las mismas que para cualquier trabajo con arreglos, es decir, siempre se utilizarán los ciclos. Veamos ahora algunos ejemplos un poco más complejos, pero que conservan esas estructuras básicas.

1.2.13 Ejemplos

Ejemplo 1

El siguiente código y diagrama de flujo llena un vector de 50 posiciones con números aleatorios del 1 al 75. Luego, muestra en pantalla la cantidad de números pares, impares, múltiplos de 3 y múltiplos de 5 que hay en el vector, finalmente muestra el vector completo.

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

```
1  Proceso CuentaNumerosVector
2
3      //Declaración de variables
4      Definir Vector,Celda,CantidadPar,CantidadImpar,CantidadMultTres,CantidadMultCinco Como Entero;
5
6      //Dimensionamiento del vector
7      Dimension Vector[50];
8
9      //Inicialización de variables
10     Celda=0;
11     CantidadPar=0;
12     CantidadImpar=0;
13     CantidadMultTres=0;
14     CantidadMultCinco=0;
15
16     //Ciclo para llenar el vector
17     Para Celda<-0 Hasta 49 Con Paso 1 Hacer
18         Vector[Celda]=azar(75)+1;
19     FinPara
20
21     //Ciclo para determinar la cantidad de números pares, impares, múltiplos de 3 y múltiplos de 5
22     Para Celda<-0 Hasta 49 Con Paso 1 Hacer
23         Si Vector[Celda] mod 2 =0 entonces
24             CantidadPar=CantidadPar+1;
25         Sino
26             CantidadImpar=CantidadImpar+1;
27         FinSi
28         Si Vector[Celda] mod 3=0 entonces
29             CantidadMultTres=CantidadMultTres+1;
30         FinSi
31         Si Vector[Celda] mod 5=0 entonces
32             CantidadMultCinco=CantidadMultCinco+1;
33         FinSi
34     FinPara
35
36     //Muestra de resultados
37     Limpiar Pantalla;
38     Escribir "REPORTE";
39     Escribir "Cantidad números pares: ",CantidadPar;
40     Escribir "Cantidad números impares: ",CantidadImpar;
41     Escribir "Cantidad números múltiplos de 3: ",CantidadMultTres;
42     Escribir "Cantidad números múltiplos de 5: ",CantidadMultCinco;
43
44     //Ciclo para mostrar el vector
45     Escribir "";
46     Escribir "VECTOR: " Sin Saltar;
47     Para Celda<-0 Hasta 49 Con Paso 1 Hacer
48         Escribir Vector[Celda]," " Sin Saltar;
49     FinPara
50
51 FinProceso
```

Analicemos algunos bloques de códigos:

Analicemos el código:

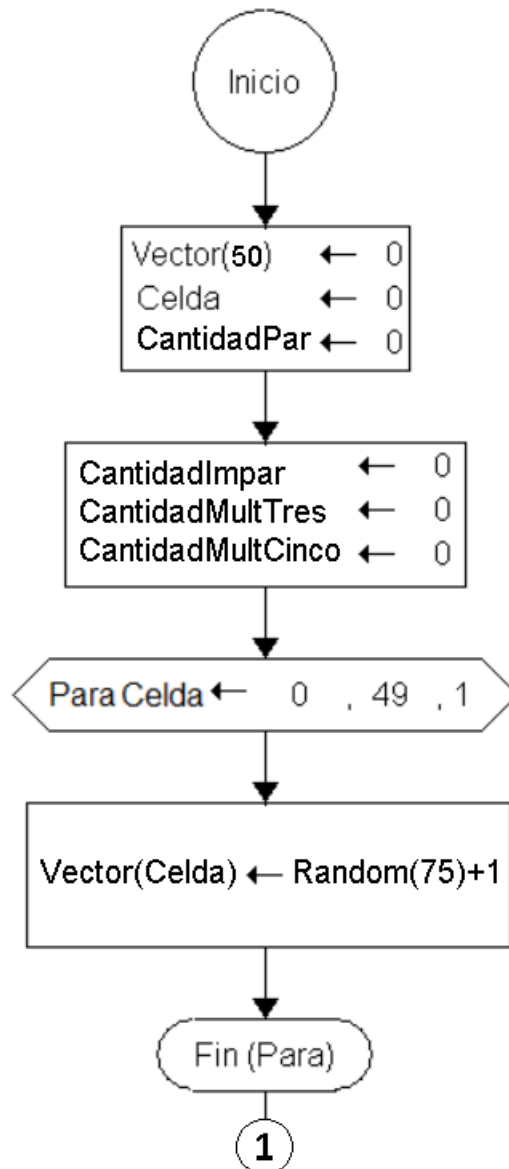
- ✓ Líneas 1-20: en este rango de instrucciones no hay nada nuevo con respecto a lo que ya hemos estudiado. Hay un inicio de algoritmo, declaración de variables, , dimensionamiento de arreglos, inicialización de variables y un ciclo para llenar el vector con números aleatorios.
- ✓ Líneas 22-34: en este bloque se podemos apreciar algo **nuevo**, el uso de **estructuras de decisión (Si) dentro de un ciclo Para**.

```
21 //Ciclo para determinar la cantidad de números pares, impares, multiplos de 3 y múltiplos de 5
22 Para Celda<-0 Hasta 49 Con Paso 1 Hacer
23   Si Vector[Celda] mod 2 =0 entonces
24     CantidadPar=CantidadPar+1;
25   Sino
26     CantidadImpar=CantidadImpar+1;
27   FinSi
28   Si Vector[Celda] mod 3=0 entonces
29     CantidadMultTres=CantidadMultTres+1;
30   FinSi
31   Si Vector[Celda] mod 5=0 entonces
32     CantidadMultCinco=CantidadMultCinco+1;
33   FinSi
34 FinPara
```

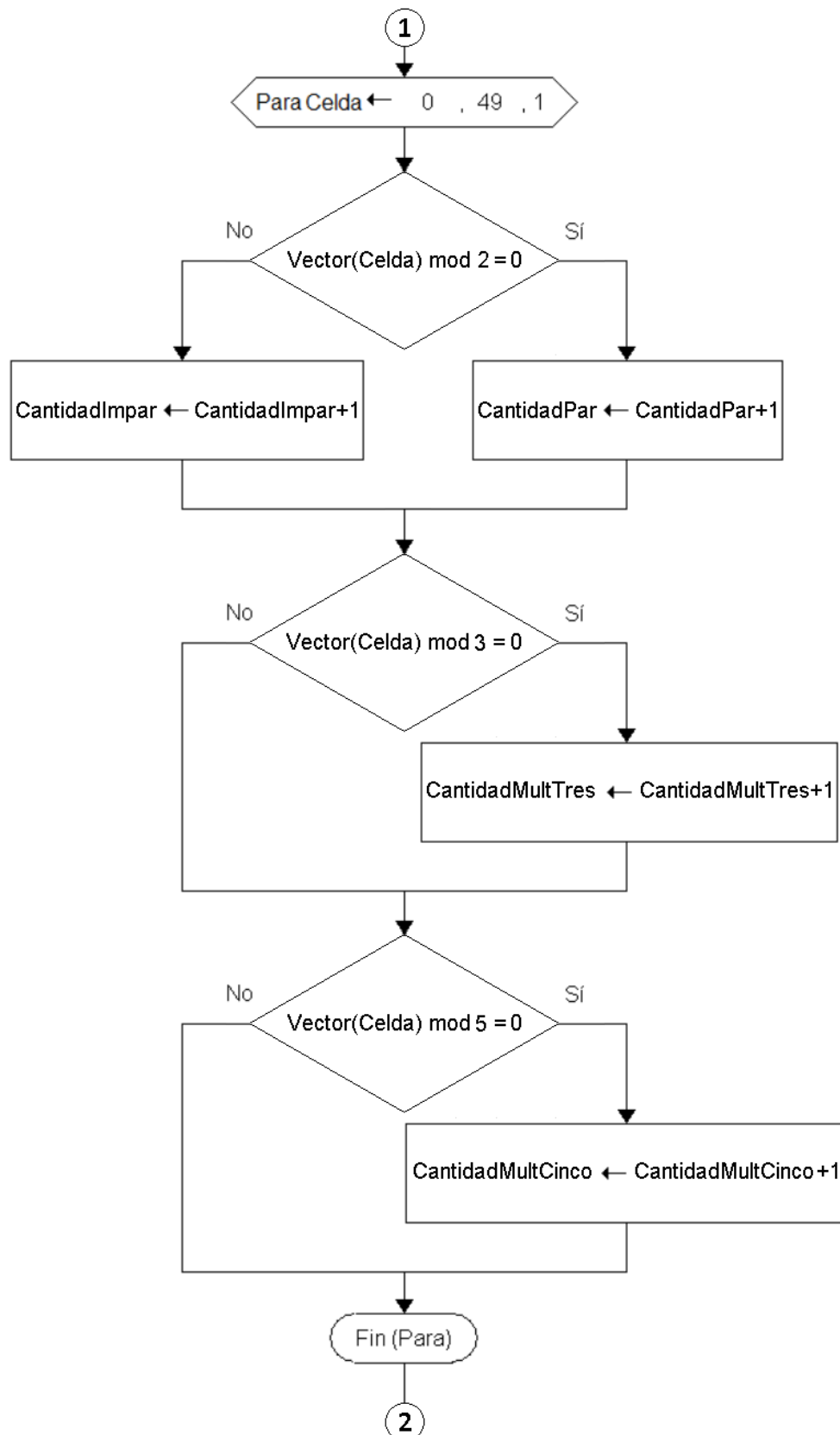
Lo que hacemos en estas instrucciones es evaluar cada celda del vector para determinar si es múltiplo de dos, si no lo es, si es múltiplo de tres o de cinco. El **Para** nos sirve para recorrer el vector, posición por posición, y se evalúa en cada celda si el valor que contiene, (el **Si** lo usamos para comparar si al hacer la división (**con el Mod**) del contenido de una celda del vector entre el número dos, su resultado (**residuo**) es igual a cero, en otras palabras, determinamos si **Vector[Celda]**,] **es divisible entre dos, tres y cinco**. Dependiendo del valor del residuo se le suma una unidad a la variable que irá almacenando el total de números respectivos. y, dependiendo del valor del residuo, se le suma una unidad a los contadores (CantidadPar, CantidadImpar, CantidadMultTres y CantidadMultCinco).

El diagrama de flujo presenta la misma lógica del código y su diseño es el siguiente:

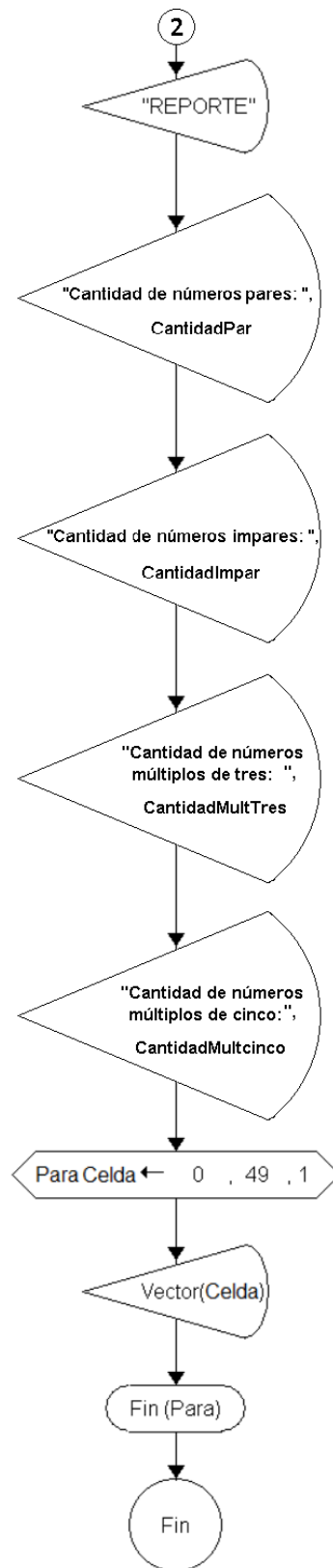
I parte



II Parte



III Parte



Ejemplo2

El segundo ejemplo le solicita al usuario los nombres de cinco atletas y sus respectivas distancias en la competencia de lanzamiento de martillo. Los nombres los debe almacenar en un vector llamado VecNombres y las distancias en otro arreglo llamado VecDistancias. Luego de introducir los datos, se debe mostrar el siguiente menú:

1. Atleta con la distancia más baja.
2. Atleta con la distancia más alta.
3. Promedio de las distancias.
4. Modificar datos.
5. Salir.

La funcionalidad de cada opción del menú es la siguiente:

1. Atleta con la distancia más baja: muestra el nombre y la distancia del atleta con la menor distancia de todas.
2. Atleta con la distancia más alta: muestra el nombre y la distancia del atleta con la mayor distancia de todas.
3. Promedio de las distancias: debe mostrar el promedio de las distancias reportadas.
4. Modificar datos: debe consultar al usuario cuál nombre de atleta desea cambiar, una vez localizado el dato debe preguntar el nuevo nombre y la nueva distancia, ambos valores deben quedar actualizados en los arreglos, al encontrar y cambiar los datos debe mostrar el siguiente mensaje "Datos cambiados con éxito."
De no encontrar el nombre buscado al inicio, debe enviar el siguiente mensaje: "Nombre NO encontrado", y consulta lo siguiente: ¿Desea buscar de nuevo con otro nombre? S/N, de acuerdo con la respuesta (S o N) busca de nuevo o sale al menú, asuma que el usuario digitará solo S o N como respuesta a esta pregunta.
El nombre que digite el usuario puede buscarlo en mayúscula o minúscula, esto no debe ser un impedimento para encontrar el dato, para esto investigue la función **Mayusculas** propia del programa.
5. Salir: esta opción muestra el siguiente mensaje "**Saliendo...por favor espere**" espera de 1 a 3 segundos y sale del algoritmo, para ese tiempo utilice la función Espera y el dato de 2 a 5 segundos lo determina con un azar.

Considere lo siguiente:

- ✓ El nombre y los dos apellidos se almacenan en una sola celda.
- ✓ Asuma que no habrá nombres ni distancias repetidas.
- ✓ La distancia puede tener metros y centímetros, al ser una competencia de élite, no se admiten distancias inferiores a 75.5 metros, debe validar este detalle de forma que si se digita una distancia menor se solicite nuevamente hasta que sea válida.

- ✓ Una vez que se muestran los resultados en las opciones 1 a la 4 del menú se debe mostrar un mensaje que diga "Presione cualquier tecla para regresar al menú", al presionar una tecla se debe mostrar de nuevo el menú en una pantalla limpia.
- ✓ Debe validar la opción que digita el usuario para el menú, es decir, si digita una opción que no está dentro del menú se le debe indicar mediante un mensaje de error y se le debe solicitar de nuevo hasta que digite una opción correcta.
- ✓ Para este ejemplo se implementarán las siguientes funciones propias del programa:
 - **Limpiar Pantalla:** borra lo que está en pantalla.
 - **Esperar Tecla:** detiene (pausa) la ejecución del algoritmo hasta que se presione una tecla.
 - **Esperar X segundos:** detiene (pausa) la ejecución del algoritmo hasta que se cumplan los segundos especificados en X.
 - **Mayusculas:** convierte una cadena de caracteres a mayúsculas.

A continuación, analizaremos el código, debido a su extensión se dividirá en siete partes.

I Parte: Declaraciones, dimensionamientos inicialización y llenado de arreglos

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

```
1 Algoritmo ControlLanzamientoMartillo
2
3 //Declaración de variables
4 Definir VecNombres, NombreBuscado, OpcionUsuario como Caracter;
5 Definir VecDistancias, MenorDistancia, MayorDistancia, SumatoriaDistancias, PromedioDistancias como Real;
6 Definir Celda, CeldaAux, OpcionMenu Como Entero;
7 Definir NombreEncontrado Como Logico;
8
9 //Dimensionamiento de arreglos
10 Dimension VecNombres(5);
11 Dimension VecDistancias(5);
12
13 //Inicialización de variables
14 NombreEncontrado=Falso;
15 NombreBuscado="X";
16 OpcionUsuario="X";
17 OpcionMenu=0;
18 MenorDistancia=0;
19 MayorDistancia=0;
20 SumatoriaDistancias=0;
21 PromedioDistancias=0;
22 Celda=0;
23 CeldaAux=0;
24
25 //Llenado de vectores
26 Para Celda=0 Hasta 4 Con Paso 1 Hacer
27     Limpiar Pantalla;
28     Escribir "Digite los datos de cada atleta. Presione la tecla Enter al final de cada dato.";
29     Escribir "DATOS ATLETA";
30     Escribir "";
31     Escribir "Nombre atleta ",Celda+1," ": ", Sin Saltar;
32     Leer VecNombres(Celda);
33
34     //Pasa el nombre a mayúsculas y lo almacena en la misma posición del vector
35     VecNombres(Celda)=Mayusculas(VecNombres(Celda));
36
37     //Validación de la distancia
38     Repetir
39         Escribir "Distancia atleta ",Celda+1," ": ", Sin Saltar;
40         Leer VecDistancias(Celda);
41         Si VecDistancias(Celda)<75.5 Entonces
42             Escribir "No se puede ingresar una distancia menor a 75.5 metros.";
43             Escribir "Digite la distancia de nuevo.";
44         FinSi
45     Hasta Que VecDistancias(Celda)>=75.5
46 FinPara
47
```

En esta primera parte del código, se **declaran las variables**, podemos notar que usamos cuatro tipos diferentes de datos, a saber: enteros, reales, cadena y lógicos. Luego tenemos la **dimensión de los arreglos** uno para almacenar los nombres y otro para las distancias. Posterior al dimensionamiento, encontramos la **inicialización** de variables.

La última parte de esta sección del código es el **llenado de los arreglos**, primero se procesa el nombre del primer atleta y luego la distancia de su lanzamiento. Para el nombre utilizamos la función **Mayusculas**, esto lo hacemos para que todos los nombres que se ingresen al arreglo se almacenen en mayúsculas, aunque se digiten en minúscula.

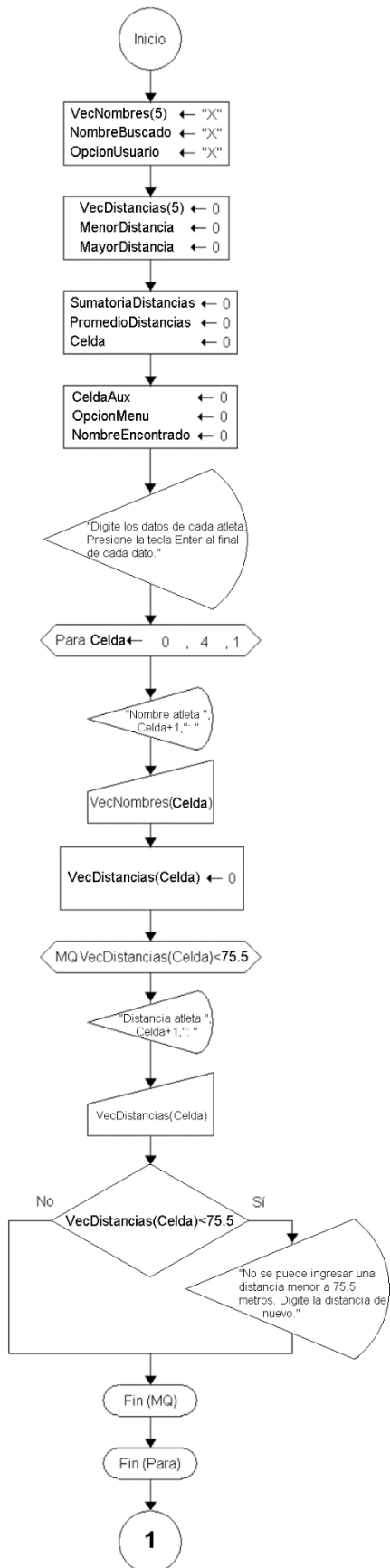
Primero asignamos a una celda el nombre del atleta que digitó el usuario, luego usamos la función **Mayusculas** y entre paréntesis le enviamos el dato (nombre) que está en la celda del vector, la función retorna el mismo texto, pero con todos sus caracteres en mayúscula y se asigna este dato a la misma celda donde estaba el texto original. Por ejemplo, si se digita el

nombre Xinia, convertimos ese texto a mayúsculas, es decir, XINIA, y lo asignamos a la celda que veníamos trabajando, por lo que al final en el arreglo solo tendremos el nombre en mayúscula. Esto nos será muy útil en la búsqueda de nombres más adelante.

Una vez llena la primera celda del vector de nombres, llenamos la primera celda del vector para las distancias, en este punto **validamos la distancia** reportada por medio de un **condicional y un ciclo**. Cada distancia se evalúa **si es mayor o igual a 75.5**, si la condición se cumple sale del ciclo, de lo contrario se repite nuevamente la solicitud del dato, esto se hace hasta que la distancia cumpla con la condición estipulada en el **Hasta que**.

El diagrama de flujo equivalente a la primera parte del pseudocódigo es el siguiente:

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.



II Parte: Menú

En la segunda parte del código analizaremos el menú y cada una de las opciones del Segun por aparte.

Ciclo repetir con el menú

```
46
47 //Muestra menú
48 Repetir
49
50 //Valida la opción del menú digitada por el usuario
51 //Hasta que se digite una opción válida se sale del ciclo
52 Repetir
53     Limpiar Pantalla;
54     Escribir "Seleccione alguna de las siguientes opciones:";
55     Escribir "1. Atleta con la distancia más baja.";
56     Escribir "2. Atleta con la distancia más alta.";
57     Escribir "3. Promedio de las distancias.";
58     Escribir "4. Modificar datos.";
59     Escribir "5. Salir.";
60     Leer OpcionMenu;
61     Si OpcionMenu<>1 Y OpcionMenu<>2 Y OpcionMenu<>3 Y OpcionMenu<>4 Y OpcionMenu<>5 Entonces
62         Limpiar Pantalla;
63         Escribir "Opción: ",OpcionMenu;
64         Escribir "Error. Debe digitar una de las opciones del menú.";
65         Escribir "Presione cualquier tecla para regresar al menú.";
66         Esperar Tecla;
67     FinSi
68     Hasta Que OpcionMenu=1 O OpcionMenu=2 O OpcionMenu=3 O OpcionMenu=4 O OpcionMenu=5
69
70
```

Luego de llenar los arreglos se habilita un ciclo Repetir con todo lo que se debe hacer en el menú, desde mostrarlo hasta ejecutar las opciones que elija el usuario.

La muestra del menú se hace dentro de otro ciclo Repetir, con la idea de validar la opción que se digite. Como podemos ver, primero se muestran las instrucciones y opciones del menú. Luego de leer el dato digitado por el usuario, se utiliza una estructura Si donde evalúa si el valor digitado es diferente a las opciones del menú; en caso de que lo sea, se muestran los mensajes respectivos. Luego se evalúan las condiciones del Hasta que y como el valor no es igual a ninguna de las opciones del menú el ciclo se repite, y se muestra el menú de nuevo.

Si el valor que se digitó fue una de las opciones del menú, entonces no se ingresa al Si, y se cumple con alguna de las opciones del menú, por lo que se sale de este y se continúa con la siguiente instrucción.

Es importante aclarar que el valor con el que termina la variable OpcionMenu luego del ciclo está validado, es decir, que es una de las opciones del menú, por lo que podemos usarlo sin problema en la estructura Segun.

A continuación, se mostrarán los fragmentos del código referente a cada opción del menú, debemos notar que todas esas opciones están dentro de una estructura Segun y que la misma, a su vez, está dentro del Repetir.

III Parte: Primera opción del menú (dentro del ciclo Repetir que inició en la línea 48)

```
69 //Ejecuta la opción seleccionada por el usuario
70 Segun OpcionMenu Hacer
71 1:
72     MenorDistancia=VecDistancias(0); //Le asigna a MenorDistancia el valor de la primera celda del vector
73     CeldaAux=0; //Se limpia la variable para una nueva evaluación
74     Limpiar Pantalla;
75
76     //Compara cada valor del arreglo con la variable MenorDistancia
77     Para Celda=0 Hasta 4 Con Paso 1 Hacer
78         Si VecDistancias(Celda)<MenorDistancia Entonces
79             MenorDistancia=VecDistancias(Celda);
80             CeldaAux=Celda;
81         FinSi
82     FinPara
83
84     //Muestra el resultado
85     Escribir "REPORTE";
86     Escribir "Atleta con la distancia más baja.";
87     Escribir "Nombre: ",VecNombres(CeldaAux);
88     Escribir "Distancia: ",VecDistancias(CeldaAux);
89     Escribir "";
90
91     Escribir "Presione cualquier tecla para regresar al menú.";
92     Esperar Tecla;
93
94
```

En la opción uno, lo primero que hacemos es asignarle a una variable el valor contenido en la primera celda del vector, esto es para evitar errores por el valor inicial (el que asignamos en la inicialización) de la variable.

El valor inicial de la variable MenorDistancia, en la línea 18, fue cero. Ahora, el vector está lleno de valores superiores a cero (recordemos que las distancias reportadas por el usuario deben ser iguales o mayores a 75.5) si no hiciéramos la línea 73, al comparar esta variable (con valor cero) con cada una de las distancias obtendríamos que la distancia menor es de cero, lo cual es un error; de ahí la importancia de la línea 73.

El segundo bloque de instrucciones es un ciclo Para con una estructura Si adentro, el ciclo va de cero a cuatro (utilizando la variable Celda). En cada ingreso ejecuta la condición Si, la cual compara el contenido de la celda con la variable MenorDistancia, en caso de que la condición sea verdadera asigna a MenorDistancia el valor que tiene esa celda, ya que el valor de la celda es menor al valor de la variable, lo que nos indica que tenemos un nuevo valor menor.

La instrucción de la línea 81 almacena en una variable llamada CeldaAux el valor de la variable Celda, esto es para saber dónde en qué posición está la menor distancia.

El último bloque de código dentro de la opción uno es para mostrar los resultados de la revisión de distancias, notemos que se utiliza la variable CeldaAux para visualizar la posición exacta donde se encuentra la menor distancia y a cuál atleta pertenece, recordemos que el nombre del atleta y su distancia están en vectores diferentes, pero en las mismas posiciones.

IV Parte: Segunda opción del menú (dentro del ciclo Repetir que inició en la línea 48)

```
94
95
96      2: MayorDistancia=VecDistancias(0); //Le asigna a MayorDistancia el valor de la primera celda del vector
97      CeldaAux=0;
98      Limpiar Pantalla;
99
100      //Compara cada valor del arreglo con la variable MayorDistancia
101      Para Celda=0 Hasta 4 Con Paso 1 Hacer
102          Si VecDistancias(Celda)>MayorDistancia Entonces
103              MayorDistancia=VecDistancias(Celda);
104              CeldaAux=Celda;
105          FinSi
106      FinPara
107
108      //Muestra el resultado
109      Escribir "REPORTE";
110      Escribir "Atleta con la distancia más alta.";
111      Escribir "Nombre: ",VecNombres(CeldaAux);
112      Escribir "Distancia: ",VecDistancias(CeldaAux);
113      Escribir "";
114
115      Escribir "Presione cualquier tecla para regresar al menú.";
116      Esperar Tecla;
117
```

En la opción dos, se hace lo mismo que en la opción uno, pero cambiamos la variable MenorDistancia por MayorDistancia y el signo relacional de la línea 102. La lógica y su orden son los mismos que para la primera opción.

V Parte: Tercera opción del menú (dentro del ciclo Repetir que inició en la línea 48)

```
117
118
119      3: Limpiar Pantalla;
120      //Limpia las variables para el cálculo de promedio con datos diferentes a los originales
121      SumatoriaDistancias=0;
122      PromedioDistancias=0;
123
124      //Suma los valores de todas las celdas del vector
125      Para Celda=0 Hasta 4 Con Paso 1 Hacer
126          SumatoriaDistancias=SumatoriaDistancias+VecDistancias(Celda);
127      FinPara
128
129      //Calcula el promedio
130      PromedioDistancias=SumatoriaDistancias/5;
131
132      //Muestra el resultado
133      Escribir "REPORTE";
134      Escribir "Promedio de distancias: ",PromedioDistancias;
135      Escribir "";
136      Escribir "Presione cualquier tecla para regresar al menú.";
137      Esperar Tecla;
138
```

La opción tres, inicia con una asignación a las variables SumatoriaDistancias y PromedioDistancias de cero, luego se inicia un ciclo Para, donde se suman todas las distancias de los atletas. Esta sumatoria se hace mediante la técnica de acumulador, es decir, se almacena en la variable SumatoriaDistancias el valor de ella misma más el valor de la primera celda del vector, en la segunda entrada al ciclo se hace lo mismo, pero con la diferencia que el valor de la variable SumatoriaDistancias ya contiene el valor de la primera celda, por lo que se suma este valor más el de la segunda celda, así sucesivamente hasta terminar de recorrer el arreglo.

Posterior a la suma de las distancias, se calcula el promedio de éstas y, finalmente, se muestra el resultado.

VI Parte: Cuarta opción del menú (dentro del ciclo Repetir que inició en la línea 48)

```
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184

4:
Repetir
    Limpiar Pantalla;
    NombreEncontrado=Falso;

    //Solicita y lee el nombre del atleta a buscar en el vector
    Escribir "DATOS DEL ATLETA";
    Escribir "Digite el nombre del atleta a buscar.";
    Leer NombreBuscado;
    NombreBuscado=Mayusculas(NombreBuscado); //Pasa el nombre a buscar a mayúsculas
    Para Celda=0 Hasta 4 Con Paso 1 Hacer
        Si NombreBuscado=VecNombres(Celda) Entonces
            Limpiar Pantalla;
            Escribir "Nombre encontrado.";
            Escribir "Digite los nuevos datos.";
            Escribir "Nombre: ",Sin Saltar;
            Leer VecNombres(Celda);

            //Pasa el nombre a mayúsculas y lo almacena en la misma posición del vector
            VecNombres(Celda)=Mayusculas(VecNombres(Celda));

            //Valida la distancia
            Repetir
                Escribir "Distancia: ", Sin Saltar;
                Leer VecDistancias(Celda);
                Si VecDistancias(Celda)<75.5 Entonces
                    Escribir "No se puede ingresar una distancia menor a 75.5 metros.";
                    Escribir "Digite la distancia de nuevo.";
                FinSi
            Hasta Que VecDistancias(Celda)>=75.5

            Escribir "Cambio efectuado con éxito.";
            Celda=5;
            NombreEncontrado=Verdadero;
        FinSi
    FinPara
    Si NombreEncontrado=Falso Entonces
        Escribir "Nombre NO encontrado.";
        Escribir "¿Desea buscar otro nombre? S/N";
        Leer OpcionUsuario;
    FinSi
    Hasta Que OpcionUsuario="N" O OpcionUsuario="n" O NombreEncontrado=Verdadero
    Escribir "";
    Escribir "Presione cualquier tecla para regresar al menú.";
    Esperar Tecla;
```

En esta opción, tiene un ciclo Repetir para todo su proceso, dentro de este se inicia con una asignación a la variable de tipo lógico (verdadero o falso) llamada NombreEncontrado de falso. Luego, solicita y lee un nombre a buscar y pasa ese nombre a mayúsculas.

Posterior al paso anterior, se inicia un ciclo Para donde se evalúa si el nombre digitado por el usuario es igual a alguno de los nombres que se encuentran en el vector (esto se hace celda por celda). En caso de que la condición del Si se cumpla, se solicita el nombre nuevo y se almacena en el vector de nombres (esto se hace en la misma posición donde se encontró el nombre buscado) para terminar la gestión de este primer dato, se pasa el nuevo nombre a mayúsculas.

Luego de procesar el nombre, se ejecuta la solicitud, lectura y validación de la nueva distancia, este bloque de código es igual al utilizado en el llenado inicial del vector. En la parte final de ese Si, se muestra un mensaje al usuario y se asigna un valor de cinco a la

variable Celda, esto sirve para salir del ciclo Para de una vez y no seguir ejecutándolo, además se asigna un valor de verdadero a la variable NombreEncontrado.

Fuera del ciclo Para se evalúa si la variable NombreEncontrado es falso; si lo es, se muestra un mensaje al usuario y se consulta si se desea buscar otro nombre. El ciclo Repetir finaliza con una condición que evalúa si la variable OpcionUsuario tiene un valor de "N" o "n" o si la variable NombreEncontrado es verdadera. Si alguna de estas condiciones se cumple quiere decir que el usuario no desea buscar más nombres o que el nombre fue encontrado, en cualquiera de esos casos el ciclo finaliza.

VII Parte: Opción De Otro Modo del menú (dentro del ciclo Repetir que inició en la línea 48)

```
184
185         De Otro Modo:
186             Escribir "Saliendo...por favor espere.";
187             Esperar (azar(3)+1) Segundos;
188         FinSegun
189     Hasta Que OpcionMenu=5
190
191 FinAlgoritmo
```

Esta opción se ejecuta si ninguna de las cuatro anteriores lo hizo, esto quiere decir que el usuario digitó la opción cinco del menú y desea salir de la aplicación, por ende, se le muestra un mensaje y se espera de uno a tres segundos para salir de la estructura Según, la cual tiene su fin en la línea 188. También, podemos notar que el ciclo Repetir de la línea 48 tiene su fin en el **Hasta que** de la línea 189.

Ejemplo 3

Este algoritmo debe ordenar o invertir un vector. El usuario determina lo que desea hacer mediante el siguiente menú:

Digite una de las siguientes opciones para el vector:

1. Llenar.
2. Ordenar.
3. Invertir.
4. Salir.

Explicación de las opciones del menú

1. Llenar: llena el vector con números aleatorios del 10 al 99, pero los números deben ser de tipo real, es decir, pueden tener o no decimales; realice algún cálculo para que

cada número tenga un dígito decimal del 0 al 9, esto debe ser de forma aleatoria también.

2. Ordenar: ordena el arreglo de menor a mayor (ascendentemente). Debe mostrar el vector original y el ordenado. No puede utilizar otro vector, por lo que se debe modificar el vector que se llenó al inicio.
3. Invertir: invierte el orden de los elementos del vector de la siguiente forma: la primera celda del vector original será la última del vector invertido, la segunda será la penúltima y así sucesivamente. Debe mostrar el vector original y el ordenado. No puede utilizar otro vector.
4. Salir: muestra un mensaje de despedida y sale del programa luego de un segundo.

Considere los siguientes puntos:

- ✓ El tamaño del vector lo define el usuario, el tamaño debe ser mayor o igual a 5 y menor o igual a 50, esto debe validarlo.
- ✓ Debe validar la opción que digite el usuario en el menú. Si digita una opción incorrecta se le debe solicitar de nuevo hasta que digite un dato válido.
- ✓ Debe validar que, si el vector no se ha llenado, se debe indicar al usuario en caso de elegir la opción dos o tres del menú, mostrando el siguiente mensaje: **"Vector vacío, debe llenar el vector."** Luego del mensaje se le indica lo siguiente **"Presione cualquier tecla para volver al menú."**
- ✓ Luego de mostrar los resultados de ordenar o invertir el vector, debe mostrar el siguiente mensaje **"Presione cualquier tecla para volver al menú."**
- ✓ Para el mensaje de presionar cualquier tecla debe utilizar la función **Esperar tecla**.

A continuación, analizaremos el código del ejemplo anterior. Al igual que en el segundo ejemplo, haremos el análisis del código por partes.

I Parte: Declaraciones, inicializaciones y dimensionamiento con validación

```
1  Algoritmo OrdenaInvierteVector
2
3      //Declaraciones
4      Definir Celda, TamanoVector, OpcionMenu Como Entero;
5      Definir VectorNumeros, CeldaTemporal Como Real;
6      Definir VectorLleno Como Logico;
7
8      //Inicializaciones
9      Celda=0;
10     CeldaTemporal=0;
11     TamanoVector=0;
12     OpcionMenu=0;
13     VectorLleno=Falso;
14
15     //Valida el tamaño del vector y lo dimensiona
16     Repetir
17         Escribir "Digite el tamaño del vector.";
18         Leer TamanoVector;
19         Si TamanoVector<5 O TamanoVector>50 Entonces
20             Limpiar Pantalla;
21             Escribir "Error. Debe digitar un tamaño entre 5 y 50.";
22         SiNo
23             Dimension VectorNumeros(TamanoVector);
24         FinSi
25     Hasta Que TamanoVector>=5 Y TamanoVector<=50
26
```

En la primera sección del código, se efectúa la **declaración de variables** con tipos de datos enteros, reales y lógicos. Posteriormente, se **inician** las variables y se procede con la **validación del tamaño** del arreglo.

En la validación, se emplea un ciclo **Repetir** en el cual se solicita y leer el tamaño del vector, luego se revisa que el valor digitado esté entre **5 y 50**, si ese valor es menor a 5 o mayor a 50 se emite un mensaje de error; pero si el tamaño propuesto está dentro del rango (5 a 50) se dimensiona el arreglo y se sale del ciclo.

II Parte: Menú y validación.

```
26
27 Repetir
28
29 //Muestra del menú y Validación de la opción digitada
30 Repetir
31     Limpiar Pantalla;
32     Escribir "Digite una de las siguientes opciones para el vector:";
33     Escribir "1. Llenar.";
34     Escribir "2. Ordenar.";
35     Escribir "3. Invertir.";
36     Escribir "4. Salir.";
37     Leer OpcionMenu;
38
39     Si OpcionMenu<>1 Y OpcionMenu<>2 Y OpcionMenu<>3 Y OpcionMenu<>4 Entonces
40         Escribir "Error. Debe digitar una de las opciones del menú.";
41         Escribir "Presione cualquier tecla para volver al menú.";
42         Esperar Tecla;
43     FinSi
44
45 Hasta Que OpcionMenu=1 O OpcionMenu=2 O OpcionMenu=3 O OpcionMenu=4
46
```

Esta parte del código inicia con un ciclo **Repetir** (línea 27), que tendrá dentro de sí toda la estructura del menú y todo el desarrollo de cada una de las opciones que el usuario elija.

La segunda parte, detalla únicamente la impresión del menú y la validación de la opción que digite el usuario. Como podemos apreciar, en la **línea 30**, inicia un ciclo **Repetir**, donde primero se muestra el menú y se lee un dato; luego, se implementa un **Si** para comparar lo digitado con las opciones del menú, si ese dato es **diferente a 1, 2, 3 y 4** se muestra un mensaje de error. El ciclo **finalizará** cuando la variable **OpcionMenu** sea igual a 1, 2, 3 o 4.

III Parte: Opción uno del menú (dentro del ciclo Repetir iniciado en la línea 27)

```
46
47 Segun OpcionMenu Hacer
48
49     1:
50         //Llena vector
51         Escribir "Llenando el vector...";
52         Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
53             VectorNumeros(Celda)=azar(90)+10;
54             VectorNumeros(Celda)=VectorNumeros(Celda)+azar(10)/10;
55         FinPara
56         VectorLleno=Verdadero;
57         Escribir "Vector lleno.";
58         Escribir "Presione cualquier tecla para volver al menú.";
59         Esperar Tecla;
60
```

La primera opción del menú se emplea para llenar el arreglo, para esto se emplea un ciclo **Para**, pero con una pequeña variante en el dato del **Hasta** que no habíamos utilizado antes. En la **línea 52**, el **Hasta** del ciclo **Para** utiliza la variable **TamanoVector** con una resta (**TamanoVector-1**), esto obedece a que **TamanoVector** lo determina el usuario pero al usarlo como límite debemos **quitarle una unidad** porque el **conteo** de las celdas **inicia en cero**.

Por ejemplo, si el usuario determinó que el tamaño del vector es **diez**, significa que **TamañoVector es igual a diez** y que el arreglo tiene **diez celdas**, pero la numeración de estas va de **cero a nueve** por lo que usamos **TamañoVector-1** para que nuestro límite sea **9**. Esta instrucción (**TamañoVector-1**) es una **operación aritmética**, cuyo resultado siempre será, para este ejemplo, **9** por eso se utiliza como límite, pero es importante aclarar que el valor de **TamañoVector** no se está cambiando ya que no se utiliza un operador de asignación (=).

Ya con el detalle del límite claro, analicemos el llenado del vector en cuanto a su contenido. El arreglo debe tener números aleatorios del **10 al 99**, pero que pueden tener o no decimales, esto nos brinda la siguiente situación, podemos generar los **números al azar** en el rango establecido sin problema, pero los números serán únicamente de **tipo entero**, por lo que **no tienen decimales**.

La solución es bastante simple; primero, en la **línea 53**, asignamos un número **aleatorio** (del 10 al 99) a una **celda del vector**. Luego, en la **línea 54**, asignamos a la **misma celda** la **suma** del contenido de **esa celda** más un **número aleatorio del 0 al 9**, pero **dividido** entre **10**.

Veamos un ejemplo, supongamos que el número **aleatorio inicial** (10 al 99) es **15**, luego generamos otro número **al azar**, pero del **cero al nueve**, supongamos que es **7**, ese número lo dividimos entre diez (**7/10**) por lo que nos daría **0.7**, finalmente ese **0.7** lo sumamos al contenido de la celda, es decir, **15**, por lo que el **15 se convierte en un 15.7**.

De esa forma nuestro arreglo tendrá números aleatorios del 10 al 99, pero con decimales, también puede que no los tenga, ya que el segundo azar puede darnos un cero, por lo que se dividiría cero entre diez y el resultado se suma al número aleatorio de la celda.

Luego de llenar el vector, se cambia el valor de la variable booleana (lógica) **VectorLleno** a **Verdadero**, esta variable nos indicará, como es el caso, que el **vector tiene datos**, si el valor de esta variable es **Falso** entonces significa que el **vector aún no tiene valores**. Esto nos servirá para determinar si **ingresamos a las opciones dos y tres** del menú, ya que no podríamos **ordenar o invertir** el arreglo si **no tiene valores** en su interior.

IV Parte: Opción dos del menú (dentro del ciclo Repetir iniciado en la línea 27)

```
60
61
62      2: Si VectorLleno=Verdadero Entonces
63          //Muestra vector original
64          Escribir "VECTOR ORIGINAL";
65          Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
66              Escribir VectorNumeros(Celda)," ",Sin Saltar;
67          FinPara
68
69          //Ordena vector de menor a mayor
70          Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71              Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72                  Si VectorNumeros(CeldaAux) < VectorNumeros(Celda) Entonces
73                      CeldaTemporal=VectorNumeros(CeldaAux);
74                      VectorNumeros(CeldaAux)=VectorNumeros(Celda);
75                      VectorNumeros(Celda)=CeldaTemporal;
76                  FinSi
77              FinPara
78          FinPara
79
80          //Muestra vector ordenado
81          Escribir"";
82          Escribir "VECTOR ORDENADO";
83          Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
84              Escribir VectorNumeros(Celda)," ",Sin Saltar;
85          FinPara
86          Escribir "";
87          Escribir "Presione cualquier tecla para volver al menú.";
88          Esperar Tecla;
89      Sino
90          Limpiar Pantalla;
91          Escribir "Vector vacío, debe llenar el vector.";
92          Escribir "Presione cualquier tecla para volver al menú.";
93          Esperar Tecla;
94      FinSi
95
```

En esta opción, se valida primero si el vector está vacío o no, lo que se utiliza es la variable de tipo lógico **VectorLleno**, si esta variable es igual a verdadero quiere decir que el vector ya tiene datos y, por ende, se podrán ejecutar las instrucciones dentro del Si, de lo contrario (**Sino de la línea 89**) se le muestra al usuario un mensaje indicando que el vector está vacío y que debe llenarlo primero.

Dentro del Si, lo primero que se hace es mostrar el vector original, para esto se utiliza un ciclo **Para** que va de la línea **65 a la 67**. Notemos que, aparte del contenido de una celda, se emplea un espacio en blanco (entre comillas) y la instrucción Sin Saltar, esto tiene como efecto que se impriman los datos del vector con espacio entre cada elemento y en línea horizontal.

Luego de la muestra del vector, se procede a ordenar sus elementos de menor a mayor, para esto emplearemos **dos ciclos Para**, uno dentro de otro; dos variables para identificar celdas (**Celda** y **CeldaAux**); una **estructura Si** dentro del ciclo Para interno y una variable para almacenar de forma temporal un elemento del vector (**CeldaTemporal**).

Lo que haremos es comparar el primer elemento del vector con cada uno de los otros elementos; si alguno de los dos es menor al otro entonces los cambiaremos de posición, veamos en detalle la lógica utilizada con un ejemplo y siguiendo el bloque de código de las líneas 70 a la 78.

```

70  Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71      Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72          Si VectorNumeros(CeldaAux) < VectorNumeros(Celda) Entonces
73              CeldaTemporal=VectorNumeros(CeldaAux);
74              VectorNumeros(CeldaAux)=VectorNumeros(Celda);
75              VectorNumeros(Celda)=CeldaTemporal;
76          FinSi
77      FinPara
78  FinPara
    
```

Tomemos el siguiente vector como ejemplo:

13.0	11.9	20.2	74.5	65.6
------	------	------	------	------

El tamaño del vector es de **cinco**, por lo que la variable **TamanoVector** tiene ese valor, pero notemos que en los ciclos **Para** empleamos **TamanoVector-1** ya que la numeración de las celdas va de **cero a cuatro**, si no hacemos esa resta en el límite del ciclo se llegaría hasta la celda numerada con cinco, lo cual es un **error** ya que no existe; es claro que tenemos una **quinta celda**, pero la misma está **identificada con el número cuatro**.

Número de celda	0	1	2	3	4
Contenido	13.0	11.9	20.2	74.5	65.6

Ahora, analicemos el orden de ejecución de los ciclos **Para anidados**; cuando tenemos un **Para** dentro de otro, se ingresa primero al **Para externo** (**Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer**), luego se ingresa al **Para interno** (**Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer**), este ciclo interno tiene la particularidad que su variable de control (**CeldaAux**) se inicializa con el mismo valor de la variable **Celda**, del ciclo externo. Esto hace que el ciclo interno se ejecute una vez menos cada vez que se reinicia.

La **primera vez** que se pasa del **Para externo al interno**, este último se **ejecuta cinco veces**; la **segunda vez** que se hace ese mismo paso de un ciclo a otro, el interno se hace **cuatro veces**; la **tercera vez** se realizan **tres ejecuciones**; la **cuarta vez** que se pasa de un ciclo a otro las **ejecuciones del interno son dos**; mientras que la **última (quinta) vez** que se hace el cambio, el ciclo interno **se ejecuta una vez**.

Una vez aclarado cómo se ejecutan los ciclos Para anidados de nuestro ejemplo y los detalles del proceso de ordenamiento, analizaremos los ingresos a los ciclos con el fin de entender el proceso lógico.

Lo primero que debemos notar es que cada vez que se ingresa al ciclo Para interno se ejecuta un **Si** que compara si el contenido de la celda del **vector** identificada con la variable **CeldaAux** es **menor** al contenido de la celda del **vector** identificada con la variable **Celda**, dependiendo del cumplimiento de la condición del Si, se ejecutan ciertas acciones.

Ejecución 1 de 5 del Para externo

Ejecución 1 de 5 del Para interno

Estado al inicio del ciclo	Estado al final del ciclo	
Celda: 0.	Celda: 0.	
CeldaAux: 0.	CeldaAux: 0.	
VectorNumeros: 13.0 11.9 20.2 74.5 65.6	VectorNumeros: 13.0 11.9 20.2 74.5 65.6	
CeldaTemporal: 0	CeldaTemporal: 0	

Código del Si (línea 72)

```

70      Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71          Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72              Si VectorNumeros(CeldaAux) < VectorNumeros(Celda) Entonces
73                  CeldaTemporal=VectorNumeros(CeldaAux);
74                  VectorNumeros(CeldaAux)=VectorNumeros(Celda);
75                  VectorNumeros(Celda)=CeldaTemporal;
76              FinSi
77          FinPara
78      FinPara
  
```

Se evalúa si el contenido de la celda del vector identificada con CeldaAux es menor al contenido de la celda identificada con Celda, en esta primera ejecución del ciclo interno ambas variables tienen el mismo valor (cero), por lo que la condición la podemos ver de la siguiente forma:

Si VectorNumeros(0) < VectorNumeros(0) Entonces...

Si pasamos la expresión a los valores que tiene el vector tendríamos lo siguiente:

Si 13.0 < 13.0 Entonces...

Esta expresión nos da como resultado un valor de **Falso**, por ende, no se ingresa al Si y no se ejecutan las instrucciones de las líneas 73 a 75.

Luego se repite de nuevo el ciclo Para interno, pero el valor de **CeldaAux** aumenta en una **unidad**. Ahora veamos la segunda vez que se ingresa al ciclo Para interno.

Ejecución 1 de 5 del Para externo

Ejecución 2 de 5 del Para interno

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 0.	Celda: 0.
CeldaAux: 1.	CeldaAux: 1.
VectorNumeros: 13.0 11.9 20.2 74.5 65.6	VectorNumeros: 11.9 13.0 20.2 74.5 65.6
CeldaTemporal: 0	CeldaTemporal: 11.9

Código del Si (línea 72)

```

70      Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71          Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72              Si VectorNumeros(CeldaAux) < VectorNumeros(Celda) Entonces
73                  CeldaTemporal=VectorNumeros(CeldaAux) ;
74                  VectorNumeros(CeldaAux)=VectorNumeros(Celda) ;
75                  VectorNumeros(Celda)=CeldaTemporal ;
76              FinSi
77          FinPara
78      FinPara

```

Se realiza la misma comparación, pero observemos que la numeración de las celdas ya no es la misma, debido a que **CeldaAux vale uno**, la expresión es la siguiente:

Si VectorNumeros(1) < VectorNumeros(0) Entonces...

Si pasamos la expresión a los valores que tiene el vector tendríamos lo siguiente:

Si 11.9 < 13.0 Entonces...

En este caso el valor de la expresión es verdadero, entonces se ejecutarán las instrucciones de las líneas 73, 74 y 75.

Analicemos la línea 73, a la variable **CeldaTemporal** se le asigna el valor contenido en la **celda uno del vector**, es decir, se le asigna el **11.9**. Es importante aclarar que en este paso **no se eliminan valores**, por lo que el 11.9 está tanto en la celda uno del vector como en la variable CeldaTemporal.

En la línea 74, se le asigna al **vector en la celda uno** lo que tiene el **mismo vector en la celda cero**, es decir, se suplanta el **11.9 por el 13.0**. En este paso el vector queda con el 13.0 en la celda cero y en la celda uno.

En la línea 75, se asigna a la **celda cero del vector** lo que contiene la variable **CeldaTemporal**, por ende, el **13.0 se sustituye por el 11.9**.

Ejecución 1 de 5 del Para externo

Ejecución 3 de 5 del Para interno

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 0.	Celda: 0.
CeldaAux: 2.	CeldaAux: 2.
VectorNumeros: 11.9 13.0 20.2 74.5 65.6	VectorNumeros: 11.9 13.0 20.2 74.5 65.6
CeldaTemporal: 11.9	CeldaTemporal: 11.9

Código del Si (línea 72)

```

70      Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71          Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72              Si VectorNumeros(CeldaAux) < VectorNumeros(Celda) Entonces
73                  CeldaTemporal=VectorNumeros(CeldaAux);
74                  VectorNumeros(CeldaAux)=VectorNumeros(Celda);
75                  VectorNumeros(Celda)=CeldaTemporal;
76              FinSi
77          FinPara
78      FinPara

```

Se realiza la misma comparación, de nuevo, prestemos atención a la numeración de las celdas:

Si VectorNumeros(2) < VectorNumeros(0) Entonces...

Si pasamos la expresión a los valores que tiene el vector tendríamos lo siguiente:

Si 20.2 < 11.9 Entonces...

El valor de la expresión es Falso, por lo que no se ejecutan las líneas 73, 74 y 75.

Ejecución 1 de 5 del Para externo

Ejecución 4 de 5 del Para interno

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 0.	Celda: 0.
CeldaAux: 3.	CeldaAux: 3.
VectorNumeros: 11.9 13.0 20.2 74.5 65.6	VectorNumeros: 11.9 13.0 20.2 74.5 65.6
CeldaTemporal: 11.9	CeldaTemporal: 11.9

Código del Si (línea 72)

```

70      Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71          Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72              Si VectorNumeros(CeldaAux) < VectorNumeros(Celda) Entonces
73                  CeldaTemporal=VectorNumeros(CeldaAux) ;
74                  VectorNumeros(CeldaAux)=VectorNumeros(Celda) ;
75                  VectorNumeros(Celda)=CeldaTemporal;
76              FinSi
77          FinPara
78      FinPara

```

Se realiza la misma comparación, pero no olvidemos la nueva numeración:

Si VectorNumeros(3) < VectorNumeros(0) Entonces...

Si pasamos la expresión a los valores que tiene el vector tendríamos lo siguiente:

Si 74.5 < 11.9 Entonces...

El valor de la expresión es Falso, por lo que las líneas 73, 74 y 75 no se ejecutan.

Ejecución 1 de 5 del Para externo

Ejecución 5 de 5 del Para interno

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 0.	Celda: 0.
CeldaAux: 4.	CeldaAux: 4.
VectorNumeros: 11.9 13.0 20.2 74.5 65.6	VectorNumeros: 11.9 13.0 20.2 74.5 65.6
CeldaTemporal: 11.9	CeldaTemporal: 11.9

Código del Si (línea 72)

```

70      Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71          Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72              Si VectorNumeros(CeldaAux) < VectorNumeros(Celda) Entonces
73                  CeldaTemporal=VectorNumeros(CeldaAux) ;
74                  VectorNumeros(CeldaAux)=VectorNumeros(Celda) ;
75                  VectorNumeros(Celda)=CeldaTemporal;
76              FinSi
77          FinPara
78      FinPara

```

Se realiza la misma comparación, pero no olvidemos la nueva numeración:

Si VectorNumeros(4) < VectorNumeros(0) Entonces...

Si pasamos la expresión a los valores que tiene el vector tendríamos lo siguiente:

Si 65.6 < 11.9 Entonces...

El valor de la expresión es Falso, por lo que las líneas 73, 74 y 75 no se ejecutan.

La variable **CeldaAux** incrementa su valor a 5, por lo que **no se ingresa más al ciclo interno** y nos debemos **regresar al Para externo**.

Notemos que el vector, luego de estas ejecuciones ya tiene en su primera posición (celda cero) el menor valor (11.9) ya que se comparó con todos los demás números del vector y ninguno cumplió con la condición del Si. Lo que haremos ahora es hacer de nuevo las comparaciones, pero empezando en la segunda celda del vector.

Luego de pasar del Para interno al externo, la variable Celda aumenta su valor y pasa de tener un cero a un uno.

Ejecución 2 de 5 del Para externo

Ejecuciones 1 a la 4 del Para interno

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 1.	Celda: 1.
CeldaAux: 1.	CeldaAux: 4.
VectorNumeros: 11.9 13.0 20.2 74.5 65.6	VectorNumeros: 11.9 13.0 20.2 74.5 65.6
CeldaTemporal: 11.9	CeldaTemporal: 11.9

Código del Si (línea 72)

```

70      |         |         |         | Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71      |         |         |         |     Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72      |         |         |         |         Si VectorNumeros(CeldaAux) < VectorNumeros(Celda) Entonces
73      |         |         |         |             CeldaTemporal=VectorNumeros(CeldaAux);
74      |         |         |         |             VectorNumeros(CeldaAux)=VectorNumeros(Celda);
75      |         |         |         |             VectorNumeros(Celda)=CeldaTemporal;
76      |         |         |         |         FinSi
77      |         |         |         |     FinPara
78      |         |         |         | FinPara

```

En esta ejecución debemos tener claro que las comparaciones se hacen a partir de la **segunda celda del vector** y las variables **CeldaAux** y **Celda** inician ambas con el **mismo valor**, un uno.

Es decir, se **comparará el 13.0 con el mismo, con el 20.2, el 74.5 y el 65.6**; y **en todos los casos** no se ingresará al Si porque la **condición dará como resultado un Falso**. Esto se repite en todas las ejecuciones del ciclo interno hasta que se termina, sale del mismo y **reingresa al Para externo por tercera vez**.

Ejecución 3 de 5 del Para externo

Ejecuciones 1 a la 3 del Para interno

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 2.	Celda: 2.
CeldaAux: 2.	CeldaAux: 4.
VectorNumeros: 11.9 13.0 20.2 74.5 65.6	VectorNumeros: 11.9 13.0 20.2 74.5 65.6
CeldaTemporal: 11.9	CeldaTemporal: 11.9

Código del Si (línea 72)

```

70      Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71      Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72          Si VectorNumeros(CeldaAux) < VectorNumeros(Celda) Entonces
73              CeldaTemporal=VectorNumeros(CeldaAux);
74              VectorNumeros(CeldaAux)=VectorNumeros(Celda);
75              VectorNumeros(Celda)=CeldaTemporal;
76          FinSi
77      FinPara
78  FinPara

```

Esta tercera vez que se reinicia el Para interno, las comparaciones se hacen a partir de la **tercera celda del vector** y las variables **CeldaAux** y **Celda** inician ambas con el **mismo valor**, un dos.

En este caso, se **comparará el 20.2 con el mismo, con el 74.5 y el 65.6**. Nuevamente, **en todos los casos** no se ingresará al Si porque la **condición dará como resultado un Falso**.

Ya en este punto podemos notar que nuestro vector está casi ordenado ya que en sus primeras tres posiciones tiene elementos en orden ascendente: 11.9 13.0 20.2; solo faltaría pasar el 74.5 en lugar del 65.6; de esta forma pasamos al **cuarto ingreso al Para externo**.

Ejecución 4 de 5 del Para externo

Ejecución 1 de 2 del Para interno

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 3.	Celda: 3.
CeldaAux: 3.	CeldaAux: 3.
VectorNumeros: 11.9 13.0 20.2 74.5 65.6	VectorNumeros: 11.9 13.0 20.2 74.5 65.6
CeldaTemporal: 11.9	CeldaTemporal: 11.9

Código del Si (línea 72)

```

70      Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71          Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72              Si VectorNumeros(CeldaAux) < VectorNumeros(Celda) Entonces
73                  CeldaTemporal=VectorNumeros(CeldaAux);
74                  VectorNumeros(CeldaAux)=VectorNumeros(Celda);
75                  VectorNumeros(Celda)=CeldaTemporal;
76              FinSi
77          FinPara
78      FinPara

```

En esta ejecución, las variables **CeldaAux** y **Celda** inician con un valor de **tres** y se comparará lo contenido en la cuarta celda del vector con ella misma y con la quinta. Recordemos que este ciclo interno se ejecutará únicamente dos veces. La comparación será la siguiente:

Si VectorNumeros(3) < VectorNumeros(3) Entonces...

Si pasamos la expresión a los valores que tiene el vector tendríamos lo siguiente:

Si 74.5 < 74.5 Entonces...

El valor de la expresión es **Falso**, por lo que las **líneas 73, 74 y 75 no se ejecutan**. Ahora pasamos a la segunda ejecución del Para interno.

Ejecución 4 de 5 del Para externo

Ejecución 2 de 2 del Para interno

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 3.	Celda: 3.
CeldaAux: 4.	CeldaAux: 4.
VectorNumeros: 11.9 13.0 20.2 74.5 65.6	VectorNumeros: 11.9 13.0 20.2 65.6 74.5
CeldaTemporal: 65.6	CeldaTemporal: 65.6

Código del Si (línea 72)

```

70      Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71          Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72              Si VectorNumeros(CeldaAux) < VectorNumeros(Celda) Entonces
73                  CeldaTemporal=VectorNumeros(CeldaAux);
74                  VectorNumeros(CeldaAux)=VectorNumeros(Celda);
75                  VectorNumeros(Celda)=CeldaTemporal;
76              FinSi
77          FinPara
78      FinPara

```

En esta segunda ejecución del Para interno la variable **CeldaAux** tiene un valor de **cuatro**, la comparación será la siguiente:

Si VectorNumeros(4) < VectorNumeros(3) Entonces...

Si pasamos la expresión a los valores que tiene el vector tendríamos lo siguiente:

Si 65.6 < 74.5 Entonces...

En este caso el valor de la expresión es **verdadero**, entonces se ejecutarán las instrucciones de las líneas **73, 74 y 75**.

En la línea 73, a la variable **CeldaTemporal** se le asigna el valor contenido en la **celda cuatro del vector**, es decir, se le asigna el **65.6**.

En la línea 74, se le asigna al **vector en la celda cuatro** lo que tiene el **mismo vector en la celda tres**, es decir, se suplanta el **65.6 por el 74.5**.

En la línea 75, se asigna a la **celda tres del vector** lo que contiene la variable **CeldaTemporal**, por ende, el **74.5 se sustituye por el 65.6**.

Ya en esta ejecución nuestro vector quedó ordenado, sin embargo, nos falta un ingreso más al Para externo, el cual analizamos a continuación.

Ejecución 5 de 5 del Para externo

Ejecución 1 de 1 del Para interno

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 4.	Celda: 4.
CeldaAux: 4.	CeldaAux: 4.
VectorNumeros: 11.9 13.0 20.2 65.6 74.5	VectorNumeros: 11.9 13.0 20.2 65.6 74.5
CeldaTemporal: 65.6	CeldaTemporal: 65.6

Código del Si (línea 72)

```

70      Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
71          Para CeldaAux=Celda Hasta TamanoVector-1 Con Paso 1 Hacer
72              Si VectorNumeros (CeldaAux) < VectorNumeros (Celda) Entonces
73                  CeldaTemporal=VectorNumeros (CeldaAux) ;
74                  VectorNumeros (CeldaAux)=VectorNumeros (Celda) ;
75                  VectorNumeros (Celda)=CeldaTemporal;
76              FinSi
77          FinPara
78      FinPara

```

En esta última ejecución, las variables **CeldaAux** y **Celda** inician en **cuatro**, y se **comparará únicamente la quinta celda del vector con ella misma**; es decir, el **74.5 con el 74.5**. En esta comparación el **resultado será falso**, dando como efecto que las líneas 73, 74 y 75 **no se ejecuten**.

La variable **CeldaAux** incrementa su valor a **cinco**, por lo que no ingresa más al ciclo interno, mientras que la variable **Celda** también **aumenta su valor a cinco**, por lo que tampoco ingresa

57

V Parte: Opción tres del menú y Otro Modo (dentro del ciclo Repetir iniciado en la línea 27)

```

95
96
97      3:
98      Si VectorLleno=Verdadero Entonces
99          //Muestra vector original
100         Escribir "VECTOR ORIGINAL";
101         Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
102             Escribir VectorNumeros (Celda) , " ",Sin Saltar;
103         FinPara
104
105         //Invierte el vector
106         Para Celda=0 Hasta ((TamanoVector-1)/2) Con Paso 1 Hacer
107             CeldaTemporal=VectorNumeros (TamanoVector-1-Celda) ;
108             VectorNumeros (TamanoVector-1-Celda)=VectorNumeros (Celda) ;
109             VectorNumeros (Celda)=CeldaTemporal;
110         FinPara
111
112         //Muestra vector invertido
113         Escribir"";
114         Escribir "VECTOR INVERTIDO";
115         Para Celda=0 Hasta TamanoVector-1 Con Paso 1 Hacer
116             Escribir VectorNumeros (Celda) , " ",Sin Saltar;
117         FinPara
118         Escribir " ";
119         Escribir "Presione cualquier tecla para volver al menú.";
120         Esperar Tecla;
121     Sino
122         Limpiar Pantalla;
123         Escribir "Vector vacío, debe llenar el vector.";
124         Escribir "Presione cualquier tecla para volver al menú.";
125         Esperar Tecla;
126     FinSi
127
128     De Otro Modo:
129         Limpiar Pantalla;
130         Escribir "Saliendo...";
131         Esperar 1 segundos;
132     FinSegun
133
134     Hasta Que OpcionMenu=4
135 FinAlgoritmo

```

En la opción tres, se invierte el orden del vector, es importante aclarar que se toma el vector original, pero si el usuario entró primero a la opción dos, o sea, ordenó el vector, será ese arreglo el que se tome para invertirlo.

Al igual que en la segunda opción, primero se valida si el vector está lleno o vacío, si el arreglo ya tiene elementos entonces se muestra en pantalla el contenido de cada celda; esto se hace mediante el ciclo Para que inicia en la línea 100.

Posterior a la muestra del vector; se invierte su orden; es decir, la última celda pasará a ser la primera, la penúltima será la segunda y así sucesivamente hasta cambiar todos los elementos. La lógica de esta solución está basada en que se utiliza una **variable** para almacenar de **forma temporal** un elemento del vector para luego pasarlo a otra posición;

otro detalle importante es que el cambio de elementos llegará hasta la mitad del arreglo. La inversión se realiza en un solo ciclo **Para**, el cual abarca desde la **línea 105 a la 109** y funciona de la siguiente forma.

```

104      //Invierte el vector
105      Para Celda=0 Hasta ((TamanoVector-1)/2) Con Paso 1 Hacer
106          CeldaTemporal=VectorNumeros (TamanoVector-1-Celda) ;
107          VectorNumeros (TamanoVector-1-Celda)=VectorNumeros (Celda) ;
108          VectorNumeros (Celda)=CeldaTemporal;
109      FinPara
    
```

En la línea 105, se establece el inicio del Para, ahí debemos notar un detalle importante con el límite del ciclo, al tamaño del vector se le resta una unidad y luego se divide entre dos, esto sirve para que el flujo lógico llegue hasta la mitad del arreglo, por ejemplo, si el vector es de seis (**tamaño par**) posiciones:

45.6	76.7	54.6	25.9	19.8	15.1
------	------	------	------	------	------

Se hace la operación $((\text{TamanoVector}-1)/2)$, el resultado es **2.5**, esto hará que la variable **Celda llegue hasta la posición dos** del arreglo, ya que luego del dos su valor sube a tres, pero la condición es hasta 2.5 por lo que ya no entraría al ciclo. La secuencia de orden sería la siguiente:

Vector antes del cambio	45.6	76.7	54.6	25.9	19.8	15.1
	↑					↑
Posiciones por cambiar	0	1	2	3	4	5
Vector posterior al cambio	15.1	76.7	54.6	25.9	19.8	45.6

Vector antes del cambio	15.1	76.7	54.6	25.9	19.8	45.6
Posiciones por cambiar	0	↑ 1	2	3	↑ 4	5
Vector posterior al cambio	15.1	19.8	54.6	25.9	76.7	45.6

Vector antes del cambio	15.1	19.8	54.6	25.9	76.7	45.6
Posiciones por cambiar	0	1	↑ 2	↑ 3	4	5
Vector posterior al cambio	15.1	19.8	25.9	54.6	76.7	45.6

Si el vector es de un **tamaño impar**, por ejemplo **cinco**, la operación $((\text{TamañoVector}-1)/2)$ tendría como resultado un **dos** y la variable Celda llegaría también al valor **dos** antes de salir del ciclo, la secuencia de cambios sería la siguiente:

Vector antes del cambio	76.7	54.6	25.9	19.8	15.1
Posiciones por cambiar	↑ 0	1	2	3	↑ 4
Vector posterior al cambio	15.1	54.6	25.9	19.8	76.7

Vector antes del cambio	15.1	54.6	25.9	19.8	76.7
Posiciones por cambiar	0	↑ 1	2	↑ 3	4
Vector posterior al cambio	15.1	19.8	25.9	54.6	76.7

Vector antes del cambio	15.1	19.8	25.9	54.6	76.7
Posiciones por cambiar	0	1	↑ 2	3	4
Vector posterior al cambio	15.1	19.8	25.9	54.6	76.7

En el último cambio, la posición es la misma por lo que el cambio se hace en la misma celda, es decir, cambia el **25.9** por él mismo.

Ahora, tomaremos el vector de seis posiciones para analizar la lógica del código, el arreglo es el siguiente:

45.6	76.7	54.6	25.9	19.8	15.1
------	------	------	------	------	------

Ejecución 1 de 3

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 0.	Celda: 0.
TamanoVector-1-Celda: 5.	TamanoVector-1-Celda: 5.
VectorNumeros: 45.6 76.7 54.6 25.9 19.8 15.1	VectorNumeros: 15.1 76.7 54.6 25.9 19.8 46.6
CeldaTemporal: 0 (podría contener otro dato)	CeldaTemporal: 15.1

Código del Para (línea 105)

```

104      //Invierte el vector
105      Para Celda=0 Hasta ((TamanoVector-1)/2) Con Paso 1 Hacer
106          CeldaTemporal=VectorNumeros (TamanoVector-1-Celda) ;
107          VectorNumeros (TamanoVector-1-Celda)=VectorNumeros (Celda) ;
108          VectorNumeros (Celda)=CeldaTemporal;
109      FinPara

```

En esta primera ejecución, en la línea 106, se le asigna a la variable **CeldaTemporal** el contenido de la celda representada por la instrucción:

VectorNumeros(TamanoVector-1-Celda)

Esto sería **VectorNumeros(6-1-0)**, es decir, **VectorNumeros(5)**, el contenido de esa celda es **15.1**, por lo que **CeldaTemporal** tendría ese valor.

En la línea 107 se le asigna al vector en la posición **cinco (TamanoVector-1-Celda)**, lo que tiene el mismo vector en la posición **Celda**, es decir, la cero **VectorNumeros(0)**, el contenido de esa celda es **45.6**, podemos notar que este paso es el que cambia el valor de la primera posición y lo copia en la última posición del vector. Recordemos que el **último valor** lo habíamos **copiado en CeldaTemporal**, por lo que **no lo hemos perdido**.

Finalmente, en la línea 108, se asigna a **VectorNumeros(Celda)**, es decir **VectorNumeros(0)**, el contenido de **CeldaTemporal**, por lo que se asigna en esa **posición el 15.1**.

Ejecución 2 de 3

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 1.	Celda: 1.
TamanoVector-1-Celda: 4.	TamanoVector-1-Celda: 4.
VectorNumeros: 15.1 76.7 54.6 25.9 19.8 46.6	VectorNumeros: 15.1 19.8 54.6 25.9 76.7 46.6

CeldaTemporal: 15.1	CeldaTemporal: 19.8
---------------------	---------------------

Código del Para (línea 105)

```

104      //Invierte el vector
105      Para Celda=0 Hasta ((TamanoVector-1)/2) Con Paso 1 Hacer
106          CeldaTemporal=VectorNumeros (TamanoVector-1-Celda) ;
107          VectorNumeros (TamanoVector-1-Celda)=VectorNumeros (Celda) ;
108          VectorNumeros (Celda)=CeldaTemporal;
109      FinPara

```

En la segunda ejecución, en la línea 106, se le asigna a la variable **CeldaTemporal** el contenido de la celda representada por la instrucción:

VectorNumeros(TamanoVector-1-Celda)

Esto sería **VectorNumeros(6-1-1)**, es decir, **VectorNumeros(4)**, el contenido de esa celda es **19.8**, por lo que **CeldaTemporal** tendría ese valor.

En la línea **107** se le asigna al vector en la posición **cuatro (TamanoVector-1-Celda)**, lo que tiene el mismo vector en la posición **Celda**, es decir, la uno **VectorNumeros(1)**, el contenido de esa celda es **76.7**.

En la línea **108**, se asigna a **VectorNumeros(Celda)**, es decir **VectorNumeros(1)**, el contenido de **CeldaTemporal**, por lo que se asigna en esa posición el **19.8**.

Ejecución 3 de 3

Estado al inicio del ciclo	Estado al final del ciclo
Celda: 2.	Celda: 2.
TamanoVector-1-Celda: 3.	TamanoVector-1-Celda: 3.
VectorNumeros: 15.1 19.8 54.6 25.9 76.7 46.6	VectorNumeros: 15.1 19.8 25.9 54.6 76.7 46.6
CeldaTemporal: 25.9	CeldaTemporal: 25.9

Código del Para (línea 105)

```

104      //Invierte el vector
105      Para Celda=0 Hasta ((TamanoVector-1)/2) Con Paso 1 Hacer
106          CeldaTemporal=VectorNumeros (TamanoVector-1-Celda) ;
107          VectorNumeros (TamanoVector-1-Celda)=VectorNumeros (Celda) ;
108          VectorNumeros (Celda)=CeldaTemporal;
109      FinPara

```

En la última ejecución del ciclo, se le asigna a la variable **CeldaTemporal** el contenido de la celda representada por la instrucción:

VectorNumeros(TamanoVector-1-Celda)

Esto sería **VectorNumeros(6-1-2)**, es decir, **VectorNumeros(3)**, el contenido de esa celda es **25.9**, por lo que **CeldaTemporal** tendría ese valor.

En la línea **107** se le asigna al vector en la posición **tres (TamañoVector-1-Celda)**, lo que tiene el mismo vector en la posición **Celda**, es decir, la dos **VectorNumeros(2)**, el contenido de esa celda es **54.6**.

En la línea **108**, se asigna a **VectorNumeros(Celda)**, es decir **VectorNumeros(2)**, el contenido de **CeldaTemporal**, por lo que se asigna en esa posición el **25.9**.

Al final de esta ejecución, la variable **Celda** aumenta su valor a tres, por lo que no se repite más el ciclo y sale del mismo; nuestro vector pasó de esto:

45.6	76.7	54.6	25.9	19.8	15.1
------	------	------	------	------	------

A esto:

15.1	19.8	25.9	54.6	76.7	45.6
------	------	------	------	------	------

Con el arreglo invertido, se muestra el nuevo orden, por medio de un ciclo Para igual al que se empleó para mostrar el vector original.

```
111      //Muestra vector invertido
112      Escribir"";
113      Escribir "VECTOR INVERTIDO";
114      Para Celda=0 Hasta TamañoVector-1 Con Paso 1 Hacer
115      |   Escribir VectorNumeros(Celda), " ",Sin Saltar;
116      FinPara
117      Escribir "";
118      Escribir "Presione cualquier tecla para volver al menú.";
119      Esperar Tecla;
120      Sino
121      |   Limpiar Pantalla;
122      |   Escribir "Vector vacío, debe llenar el vector.";
123      |   Escribir "Presione cualquier tecla para volver al menú.";
124      |   Esperar Tecla;
125      FinSi
126
127      De Otro Modo:
128      |   Limpiar Pantalla;
129      |   Escribir "Saliendo...";
130      |   Esperar 1 segundos;
131      FinSegun
132
133      Hasta Que OpcionMenu=4
134
135      FinAlgoritmo
```

Lo que sigue luego de la muestra, es el **Sino** del Si de validación de la línea **97**, para saber si el vector estaba lleno.

En la línea **127**, se ejecuta la instrucción **De Otro Modo**, la cual se utiliza para **salir** del algoritmo, en la línea **131** se **cierra** la estructura **Según**, mientras que en la **133** se **cierra** el ciclo **Repetir** iniciado en la línea **27**.

Ejemplo 4

Buscar un número en un vector y determinar cuántas veces aparece el número buscado en el arreglo. Considere los siguientes aspectos:

- El vector es de tamaño 100.
- Se debe llenar con números aleatorios del 100 al 500.
- El número por buscar lo digita el usuario, se debe validar que ese número esté entre 100 y 500, de lo contrario se le solicitará de nuevo hasta que digite un dato válido.
- Al final de la búsqueda se muestra un informe de búsqueda donde se indique si el número está o no en el arreglo y de estarlo debe mostrar cuántas veces aparece.
- Luego de la muestra del informe de búsqueda se debe preguntar al usuario si desea hacer otra búsqueda, si la respuesta es afirmativa, se solicita el número a buscar nuevamente, de lo contrario se termina el algoritmo.

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

```
1  Proceso BusquedaenVector
2
3      //Declaraciones de variables y vector
4      Definir Vector, Celda, NumeroBuscado, CantidadCoincidencias Como Entero;
5      Definir Opcionusuario Como Caracter;
6
7      //Dimensionamiento del arreglo
8      Dimension Vector(100);
9
10     //Inicializaciones de variables
11     Celda=0;
12     NumeroBuscado=0;
13     CantidadCoincidencias=0;
14     Opcionusuario="S";
15
16     //Llenado del vector
17     Para Celda<-0 Hasta 99 Con Paso 1 Hacer
18     |     Vector(Celda)=azar(401)+100;
19     FinPara
20
21     Mientras Opcionusuario="S" o Opcionusuario="s" Hacer
22     |
23     |     //Solicitud, lectura y validación del número a buscar
24     |     Repetir
25     |     |     Escribir "Digite un número a buscar en el vector.";
26     |     |     Leer NumeroBuscado;
27     |     |     Si NumeroBuscado<100 O NumeroBuscado>500 Entonces
28     |     |     |     Escribir "Error. El número a buscar debe estar entre 100 y 500.";
29     |     |     FinSi
30     |     Hasta Que NumeroBuscado>=100 Y NumeroBuscado<=500
31     |
32     |     //Búsqueda del número en el vector
33     |     Escribir "Búsqueda del número ",NumeroBuscado," en el vector.";
34     |     Para Celda<-0 Hasta 99 Con Paso 1 Hacer
35     |     |     Si NumeroBuscado=Vector(Celda) Entonces
36     |     |     |     CantidadCoincidencias=CantidadCoincidencias+1;
37     |     |     FinSi
38     |     FinPara
39     |
40     |     //Muestra de resultados
41     |     Limpiar Pantalla;
42     |     Escribir "*****INFORME DE BÚSQUEDA*****";
43     |     Si CantidadCoincidencias>0 Entonces
44     |     |     Escribir "El número ",NumeroBuscado," está ",CantidadCoincidencias," veces en el vector.";
45     |     Sino
46     |     |     Escribir "El número ",NumeroBuscado," NO está en el vector.";
47     |     FinSi
48     |     Escribir "";
49     |     Escribir "¿Desea hacer otra búsqueda? S/N";
50     |     Leer Opcionusuario;
51     FinMientras
52
53 FinProceso
```

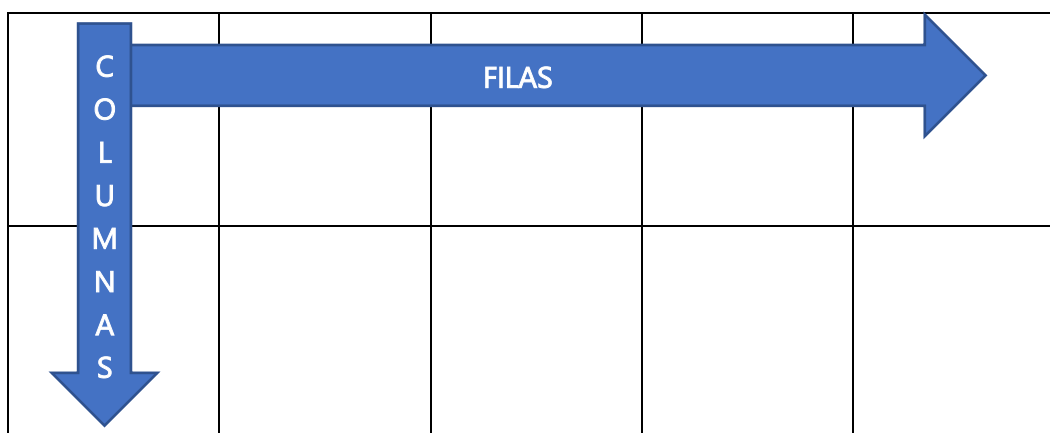
1.3 Matrices

Las matrices son arreglos multidimensionales, específicamente, de dos dimensiones, y se pueden representar como se muestra a continuación:

1	13	38	5	10
25	9	80	74	12

Matriz de 10 celdas o posiciones.

Las matrices se componen de filas y columnas, las filas son horizontales y las columnas verticales.



Identificación de filas y columnas en una matriz.

Las matrices, al igual que los vectores, almacenan datos del mismo tipo en todas sus celdas, y sus valores se trabajan por separado, es decir, podemos modificar un dato en una celda específica sin alterar otros.

Para utilizar un elemento de la matriz se debe hacer referencia al nombre de esta y a la posición específica donde se ubica el elemento.

1.3.1 Declaración y dimensionamiento de una matriz

La declaración de las matrices se hace de la misma forma que con los vectores o variables, empleando la siguiente estructura:

Definir **Nombre de la matriz** Como **tipo de dato**;

Donde **Nombre de la matriz** será el identificador que se le otorgará al arreglo para su manejo en el algoritmo; y **tipo de dato** será la definición de los valores que podrá almacenar el arreglo, ya sea datos de tipo real, enteros, caracteres o lógicos.

Si quisiéramos declarar una matriz que almacene datos numéricos de tipo entero, por ejemplo, cantidad de estudiantes, lo haríamos así:

Definir **MatCantidadEstudiantes** Como **Entero**;

Luego de declarar un arreglo, se debe **dimensionar**, esto se hace para definir el tamaño máximo (cantidad de celdas) que tendrá la matriz. La dimensión de una matriz debe especificar cuántas filas y columnas, **siempre el primer número serán las filas y el segundo las columnas**, veamos el siguiente ejemplo:

Dimension MatCantidadEstudiantes (2,5);

La igual que con los vectores, el dimensionamiento de una matriz siempre serán números **enteros**, ya que no puede existir una matriz con, por ejemplo, 5.7 filas o 2.2columnas. En el ejemplo anterior de dimensionamiento, la matriz es de 2 filas y 5 columnas.

Es importante aclarar que una matriz tendrá las filas y columnas que se necesiten para resolver la situación dada, y que puede ser cuadrada (con la misma cantidad de filas y columnas) o rectangular (ya sea con más filas que columnas o viceversa). En el siguiente ejemplo se evidencia una declaración y dimensionamiento de una matriz de 5 filas y 5 columnas y luego otra de 7 filas y 3 columnas:

```
1 Algoritmo DeclaracionDimensionamientoMatriz
2
3   // Declaración de variables
4   Definir Matriz1, Matriz2 Como Entero;
5
6   // Dimensión de las matrices
7   Dimension Matriz1(5,5);
8   Dimension Matriz2(7,3);
9
10 FinAlgoritmo
```

1.3.2 Ejercicios

Declare y dimensione las matrices especificadas, considere el tipo de dato y la dimensión.

1. Matriz de 50 posiciones (10 filas 5 columnas) que almacena mediante una marca ("S" o "N") los días que tienen teletrabajo 10 funcionarios de un banco.

Declaración:

Dimensión:

2. Arreglo multidimensional cuadrado de 10 filas que almacena el peso de los pacientes de un hospital.

Declaración:

Dimensión:

3. Matriz de 20 posiciones que almacena los nombres de los empleados de una fábrica, organizados por 5 columnas que representan las áreas para las que laboran.

Declaración:

Dimensión:

4. Matriz de 6 posiciones (2 columnas) que almacena datos de falso o verdadero.

Declaración:

Dimensión:

5. Arreglo multidimensional cuadrado de 1000 celdas que almacena temperaturas en centígrados para un proyecto de investigación.

Declaración:

Dimensión:

1.3.3 Respuestas

Las siguientes respuestas son representativas, ya que los nombres de las matrices pueden variar, lo que se debe tomar en consideración es que deben cumplir con las normas para identificar arreglos.

1. Matriz de 50 posiciones (10 filas 5 columnas) que almacena mediante una marca ("S" o "N") los días que tienen teletrabajo 10 funcionarios de un banco.

Declaración: Definir DiasTeletrabajo Como Carácter;

Dimensión: Dimension DiasTeletrabajo(10,5);

2. Arreglo multidimensional cuadrado de 10 filas que almacena el peso de los pacientes de un hospital.

Declaración: Definir PesosPacientes Como Real;

Dimensión: Dimension PesosPacientes(10,10);

3. Matriz de 20 posiciones que almacena los nombres de los empleados de una fábrica, organizados por 5 columnas que representan las áreas para las que laboran.

Declaración: Definir NombresEmpleados Como Carácter;

Dimensión: Dimension NombresEmpleados (4,5);

4. Matriz de 6 posiciones (2 columnas) que almacena datos de falso o verdadero.

Declaración: Definir DatosBooleanos Como Logico;

Dimensión: Dimension DatosBooleanos(3,2);

5. Arreglo multidimensional cuadrado de 10000 celdas que almacena temperaturas en centígrados para un proyecto de investigación.

Declaración: Definir Temperaturas Como Real;

Dimensión: Dimension Temperaturas (100,100);

1.3.4 Dimensionamiento por usuario

Las matrices también pueden ser dimensionadas por el usuario, el procedimiento es similar al utilizado para un vector, pero se deben definir dos dimensiones en vez de una.

Recordemos que primero se declara el arreglo y, antes de dimensionarlo, se consulta la cantidad de filas y columnas que tendrá la matriz, las respuestas del usuario se almacenan en variables de tipo entero y con esas variables se hace el dimensionamiento, debemos aclarar que si la matriz es cuadrada no sería necesario solicitar ambas dimensiones, sino solo una, ya que se asume que tiene la misma cantidad de filas y de columnas. Veamos un ejemplo:

```
1  Algoritmo DimensionamientoMatrizCuadUsuario
2
3      // Declaración de variables
4      Definir Matriz Como Real;
5      Definir TamanoMatriz Como Entero;
6
7      // Inicialización de variables
8      TamanoMatriz=0;
9
10     Escribir "Digite la cantidad de filas que tendrá la matriz.";
11     //También podría ser: Escribir "Digite el tamaño de la matriz.";
12     Leer TamanoMatriz;
13
14     // Dimensionamiento de la matriz
15     Dimension Matriz(TamanoMatriz,TamanoMatriz);
16
17 FinAlgoritmo
```

En el código anterior, se muestra la declaración y dimensionamiento de una matriz cuadrada, notemos que solo se solicita un dato para el tamaño y ese se repite para definir tanto las filas como las columnas.

Ahora, veamos otro ejemplo, pero con una matriz rectangular:

```
1 Algoritmo DimensionamientoMatrizRectUsuario
2
3     // Declaración de variables
4     Definir Matriz Como Real;
5     Definir CantidadFilas, CantidadColumnas Como Entero;
6
7     // Inicialización de variables
8     CantidadFilas=0;
9     CantidadColumnas=0;
10
11     Escribir "Digite la cantidad de filas que tendrá la matriz.";
12     Leer CantidadFilas;
13     Escribir "Digite la cantidad de columnas que tendrá la matriz.";
14     Leer CantidadColumnas;
15
16     // Dimensionamiento de la matriz
17     Dimension Matriz (CantidadFilas,CantidadColumnas) ;
18
19 FinAlgoritmo
```

Podemos notar que para una matriz rectangular se deben solicitar las filas y las columnas por aparte, y de la misma forma se deben usar en el dimensionamiento.

1.3.5 Manejo de una matriz

Para el manejo de una matriz siempre se debe considerar la fila y la columna donde se desea gestionar un dato, ya sea almacenarlo, modificarlo o mostrarlo. Si deseamos agregar un valor a una celda de la matriz se debe hacer con la siguiente estructura:

Matriz (Fila,Columna)= Valor;

Recordemos que las celdas inician su numeración en cero (0), por lo que una matriz de tres filas y cuatro columnas tendría un rango de cero a dos (en filas) y de cero a tres (en columnas), tanto la variable que representa las filas como la que representa las columnas funcionan como los índices de la matriz.

Siempre debemos emplear los **índices** para almacenar un dato en una matriz, por ende, la siguiente instrucción es **errónea**:

Matriz=25;

Ya que no se indica una celda donde guardar el valor 25, si quisiéramos colocar ese dato en la matriz debemos especificar la fila y la columna (separadas por coma) de la siguiente forma:

Matriz (2,1) = 25;

Gráficamente, se vería así:

Columnas	0	1	2	3
<u>Filas</u>				
<u>0</u>				
<u>1</u>				
<u>2</u>		25		

Matriz de 12 celdas (3 filas y 4 columnas)

Al igual que con los vectores, las matrices se operan mediante ciclos, pero con la diferencia de que son dos ciclos (anidados) y no solo uno. La razón de este cambio es muy sencilla, para los vectores solo se requiere manejar un índice para referenciar una celda, mientras que las matrices necesitan dos índices, uno para las filas y otro para las columnas.

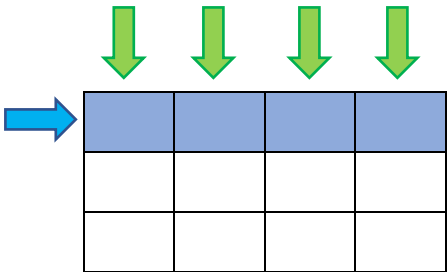
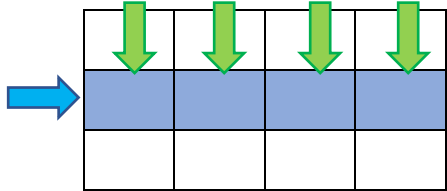
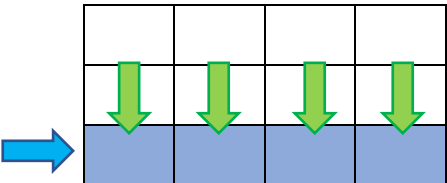
Recordemos que al tener ciclos anidados su ejecución se hace de forma paulatina, es decir, se ejecuta una primera vez el ciclo externo y, al ingresar al interno, se ejecuta en su totalidad. En la segunda ejecución del ciclo externo se vuelve a ejecutar el ciclo interno en su totalidad, esto se repite hasta que el ciclo externo haya completado todas sus repeticiones.

Por ejemplo, si tenemos dos ciclos, uno externo que se ejecutará 3 veces y otro interno que lo hará 4 veces, en la primera ejecución del ciclo externo se procesará cuatro veces el ciclo interno, en la segunda ejecución del ciclo externo se vuelve a procesar el ciclo interno otras cuatro veces. Este comportamiento se hace hasta que el ciclo externo cumpla con sus tres ejecuciones; para este ejemplo en concreto, el ciclo externo se ejecuta tres veces, pero el interno lo hace en doce ocasiones, ya que, por cada procesamiento del ciclo externo, el ciclo interno se ejecuta cuatro veces ($3 \times 4 = 12$).

Normalmente, para trabajar con matrices utilizaremos dos ciclos Para anidados, ya sea para llenarla, mostrarla o modificarla, un ciclo llevará el control de las filas mientras que el otro lo hará con las columnas.

1.3.6 Llenado de una matriz

El llenado de una matriz puede hacerse por filas o por columnas, para esta explicación haremos el primero (por filas). La mecánica de ejecución es muy sencilla, primero nos posicionamos en la primera fila e iremos avanzando en las columnas desde la primera hasta llegar a la última, luego nos pasamos a la segunda fila e iniciamos el proceso nuevamente con las columnas desde la primera hasta la última esto se repite hasta llegar a la última fila y última columna.

1	Nos posicionamos en la primera fila (→) y luego avanzamos desde la primera hasta la última columna (↓).	
2	Luego avanzamos a la segunda fila y repetimos el proceso con las columnas, desde la primera hasta la última.	
3	Por último, pasamos a la tercera fila y se reitera el proceso con las columnas.	

Para trabajar con un ejemplo específico usaremos una matriz de tres filas y cuatro columnas y la llenaremos de ceros (0), el orden de llenado será por filas, es decir, como se mostró anteriormente; el algoritmo es el siguiente:

```
1  Algoritmo LlenadoMatriz
2
3      //Declaración de variables
4      Definir Matriz Como Entero;
5      Definir Fila, Columna Como Entero;
6
7      //Dimensión de la matriz
8      Dimension Matriz(3,4);
9
10     //Inicialización de variables
11     Fila=0;
12     Columna=0;
13
14     //Llenado de la matriz
15     Para Fila=0 Hasta 2 Con Paso 1 Hacer
16     |
17     |     Para Columna=0 Hasta 3 Con Paso 1 Hacer
18     | |     Matriz(Fila,Columna)=0;
19     | |     FinPara
20     |
21     FinPara
22
23 FinAlgoritmo
```

Como podemos observar, en la línea 15 se inicia con el llenado del arreglo, el primer **Para (externo)** se encarga del control de las **filas**, cuya numeración va de **cero a dos** (son tres filas). En la línea 17 se inicia con el **Para (interno)** de las **columnas**, con numeración de **cero a tres** (son cuatro columnas) dentro de este se ubica la instrucción que **asigna a cada celda de la matriz el número cero**.

La **primera** vez que se ingresa al **Para externo**, el valor de **Fila** se **inicializa en cero**, una vez dentro de este ciclo se **ingresa al Para interno**, donde el valor de **Fila** sigue siendo **cero** y la variable **Columna** se **inicializa en cero**. En el **Para interno** tenemos la instrucción de **asignación** de valor **Matriz(Fila,Columna)=0;** esto lo podemos interpretar como **Matriz(0,0)=0;** que es igual que decir a la matriz llamada Matriz en la posición 0,0 asigne un cero.

Columna	0	1	2	3
Fila				
0	0			
1				
2				

Luego de la asignación de cero en la celda 0,0, regresamos a la línea 17, se incrementa el valor de Columna a uno y se ingresa de nuevo al ciclo; ahí se ejecuta de nuevo la instrucción **Matriz(Fila,Columna)=0**; pero que ahora es **Matriz(0,1)=0**; por lo que el cero se asignará en la siguiente celda:

Columna	0	1	2	3
Fila				
0	0	0		
1				
2				

Nuevamente nos devolvemos a la línea 17, donde se incrementa el valor de Columna, ahora vale dos, por lo que al ejecutar la instrucción **Matriz(Fila,Columna)=0**; tendremos **Matriz(0,2)=0**; por ende:

Columna	0	1	2	3
Fila				
0	0	0	0	
1				
2				

Regresamos a la línea 17 y Columna aumenta a tres, ingresamos al ciclo y ejecutamos **Matriz(Fila,Columna)=0;** o lo que es lo mismo **Matriz(0,3)=0;** por lo que nuestra matriz varía así:

Columna	0	1	2	3
Fila				
0	0	0	0	0
1				
2				

Luego de esa asignación se regresa a la línea 17, Columna aumenta su valor a cuatro, por lo que no ingresa al Para, pasamos al fin de ese ciclo (línea 19) y regresamos al Para externo, Fila aumenta su valor a uno (recordemos que estaba en cero) e ingresamos de nuevo al Para interno, en ese momento Columna se vuelve a inicializar en cero, por ende entramos a ejecutar la instrucción **Matriz(Fila,Columna)=0;** pero con los nuevos valores, es decir **Matriz(1,0)=0;** la matriz se vería así:

Columna	0	1	2	3
Fila				
0	0	0	0	0
1	0			
2				

Las ejecuciones del ciclo Para interno se repiten hasta que Columna sea cuatro nuevamente y Fila mantiene su valor en uno, la matriz evolucionaría de las siguientes formas:

Columna	0	1	2	3
Fila				
0	0	0	0	0
1	0	0		
2				

Columna	0	1	2	3
Fila				
0	0	0	0	0
1	0	0	0	
2				

Columna	0	1	2	3
Fila				
0	0	0	0	0
1	0	0	0	0
2				

Al terminar la última ejecución del Para interno, la variable Columna aumenta a cuatro y se regresa al Para externo, se aumenta el valor de Fila en uno, ahora vale dos, por lo que entrará una última vez al ciclo. Luego, se ejecuta nuevamente el ciclo Para interno

empezando con la asignación de cero a Columna y se repetirá el ciclo hasta que Columna tenga un valor de cuatro; la secuencia de la matriz sería la siguiente:

Columna	0	1	2	3
Fila				
0	0	0	0	0
1	0	0	0	0
2	0			

Columna	0	1	2	3
Fila				
0	0	0	0	0
1	0	0	0	0
2	0	0		

Columna	0	1	2	3
Fila				
0	0	0	0	0
1	0	0	0	0
2	0	0	0	

Columna	0	1	2	3
Fila				
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0

Una vez que se ha completado el ciclo interno se pasa al externo, donde Fila aumenta su valor a 3, por ende, no ingresa al ciclo, terminando así esa parte del algoritmo y dejando la matriz llena de ceros.

La ejecución de este código tiene el siguiente orden:

Ciclo externo	Ejecución 1							
	Ciclo interno							
	Ejecución 1		Ejecución 2		Ejecución 3		Ejecución 4	
	Fila 0	Columna 0	Fila 0	Columna 1	Fila 0	Columna 2	Fila 0	Columna 3
Ciclo externo	Ejecución 2							
	Ciclo interno							
	Ejecución 1		Ejecución 2		Ejecución 3		Ejecución 4	
	Fila 1	Columna 0	Fila 0	Columna 1	Fila 0	Columna 2	Fila 0	Columna 3
Ciclo externo	Ejecución 3							
	Ciclo interno							
	Ejecución 1		Ejecución 2		Ejecución 3		Ejecución 4	
	Fila 2	Columna 0	Fila 0	Columna 1	Fila 0	Columna 2	Fila 0	Columna 3

1.3.7 Muestra de una matriz

Ya tenemos claro como llenar una matriz, ahora analizaremos como mostrarla. Si para llenar una matriz necesitamos dos ciclos Para anidados, para visualizarla emplearemos los mismos ciclos, pero con una pequeña diferencia en la instrucción del ciclo interno; analicemos el siguiente fragmento de código:

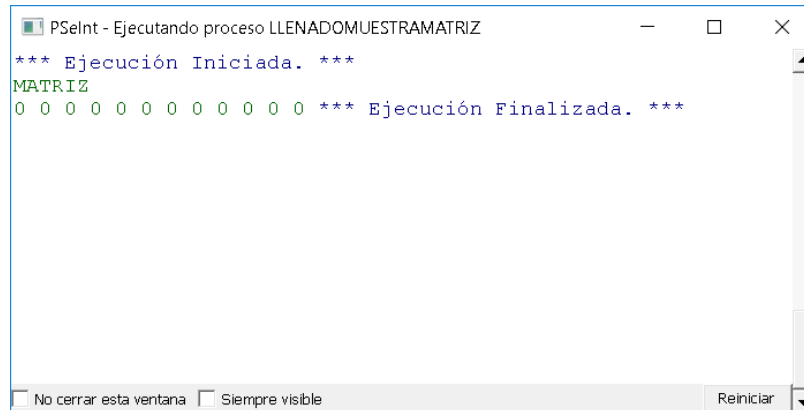
```
Para Fila=0 Hasta 2 Con Paso 1 Hacer
    Para Columna=0 Hasta 3 Con Paso 1 Hacer
        Escribir Matriz(Fila,Columna);
    FinPara
FinPara
```

Como podemos ver, tanto para el código de llenado y como para el de muestra se utilizan ciclos Para anidados, la diferencia es que cambiamos la instrucción de asignación **Matriz(Fila,Columna)=0** por la instrucción **Escribir Matriz(Fila,Columna)**. Recordemos que para trabajar con una matriz se debe hacer celda por celda, por lo que si deseamos mostrar todos los elementos que contiene un arreglo multidimensional debemos hacerlo de esa forma.

Veamos el pseudocódigo completo de llenado y muestra de una matriz:

```
1  Algoritmo LlenadoMuestraMatriz
2
3      // Declaración de variables
4      Definir Matriz Como Entero;
5      Definir Fila, Columna Como Entero;
6
7      // Dimensión de la matriz
8      Dimension Matriz{3,4};
9
10     // Inicialización de variables
11     Fila = 0;
12     Columna = 0;
13
14     // Llenado de la matriz
15     Para Fila=0 Hasta 2 Con Paso 1 Hacer
16     |
17     |     Para Columna=0 Hasta 3 Con Paso 1 Hacer
18     | |     Matriz(Fila,Columna)=0;
19     | |     FinPara
20     |
21     FinPara
22
23     // Muestra de la matriz
24     Escribir "MATRIZ";
25     Para Fila=0 Hasta 2 Con Paso 1 Hacer
26     |
27     |     Para Columna=0 Hasta 3 Con Paso 1 Hacer
28     | |     Escribir Matriz(Fila, Columna), ' ', Sin Saltar;
29     | |     FinPara
30     |
31     FinPara
32
33 FinAlgoritmo
```


Esto mostraría la siguiente salida en pantalla:

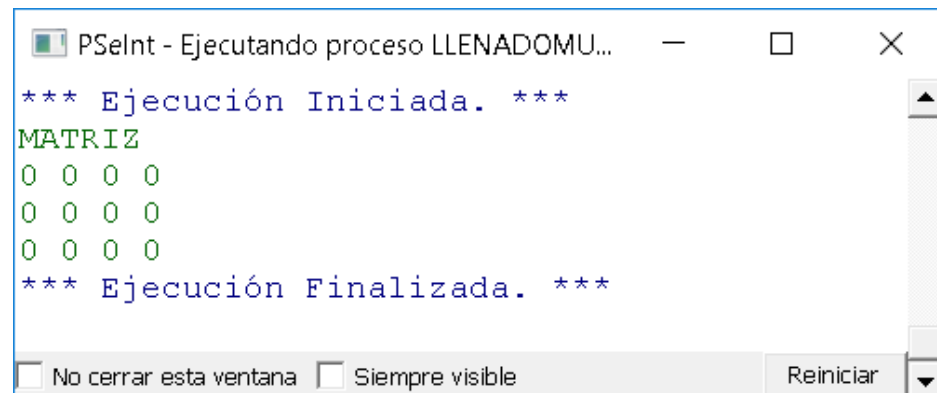


```
PSeInt - Ejecutando proceso LLENADOMUESTRAMATRIZ
*** Ejecución Iniciada. ***
MATRIZ
0 0 0 0 0 0 0 0 0 0 0 0 *** Ejecución Finalizada. ***
No cerrar esta ventana  Siempre visible  Reiniciar
```

Como podemos ver la muestra de la matriz se hace de forma horizontal, pero podríamos darle un aspecto más cercano a la representación gráfica de una matriz, una cuadrícula. Para lograrlo insertaremos la instrucción Escribir ""; en la línea 30 del código, ese fragmento del algoritmo quedaría de la siguiente forma:

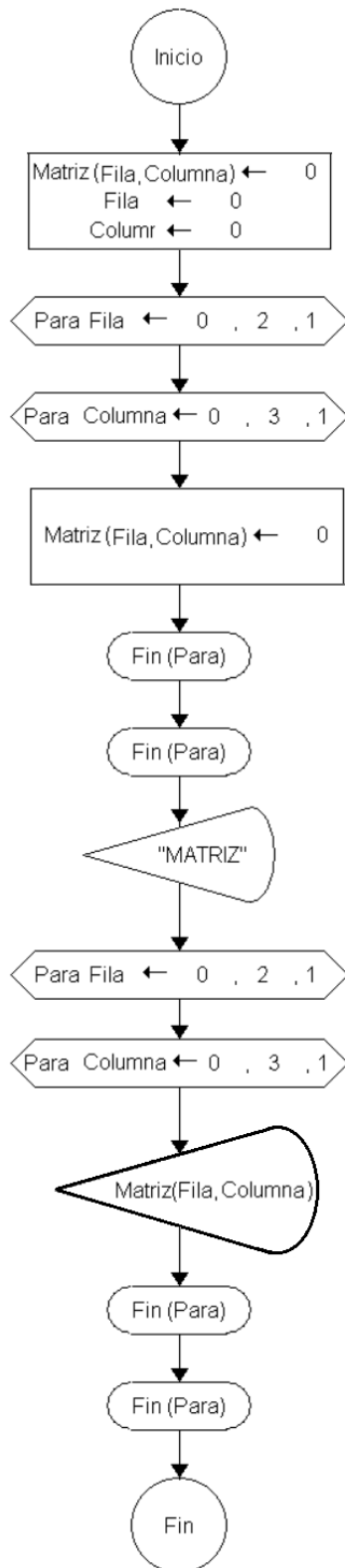
```
23      // Muestra de la matriz
24      Escribir "MATRIZ";
25      Para Fila=0 Hasta 2 Con Paso 1 Hacer
26      |
27      |      Para Columna=0 Hasta 3 Con Paso 1 Hacer
28      |      |      Escribir Matriz(Fila, Columna), ' ', Sin Saltar;
29      |      FinPara
30      |      Escribir " ";
31      FinPara
32
33  FinAlgoritmo
```

La salida cambiaría a la siguiente:



```
PSeInt - Ejecutando proceso LLENADOMU...
*** Ejecución Iniciada. ***
MATRIZ
0 0 0 0
0 0 0 0
0 0 0 0
*** Ejecución Finalizada. ***
No cerrar esta ventana  Siempre visible  Reiniciar
```

El diagrama de flujo del código anterior quedaría de la siguiente forma:



Nota: en el DFD las salidas en pantalla se muestran en ventanas separadas por lo que la matriz se muestra celda por celda y no de forma cuadrada.

1.3.8 Modificación de una matriz

Si se desea modificar los valores de una matriz se debe hacer celda por celda, analicemos el ejemplo de llenado de la matriz, pero la llenaremos de números uno y a cada número uno le sumaremos un número al azar de 1 a 9, es decir, la matriz quedará llena de números que irán en ese rango (1-9). Esto lo haremos en dos ciclos Para anidados luego de la muestra del arreglo y, al final, mostraremos de nuevo la matriz, pero esta vez la salida será diferente, el código que agregaremos es el siguiente:

Para Fila=0 Hasta 2 Con Paso 1 Hacer

Para Columna=0 Hasta 3 Con Paso 1 Hacer

Matriz(Fila,Columna)= Matriz(Fila,Columna) + azar(9)+1;

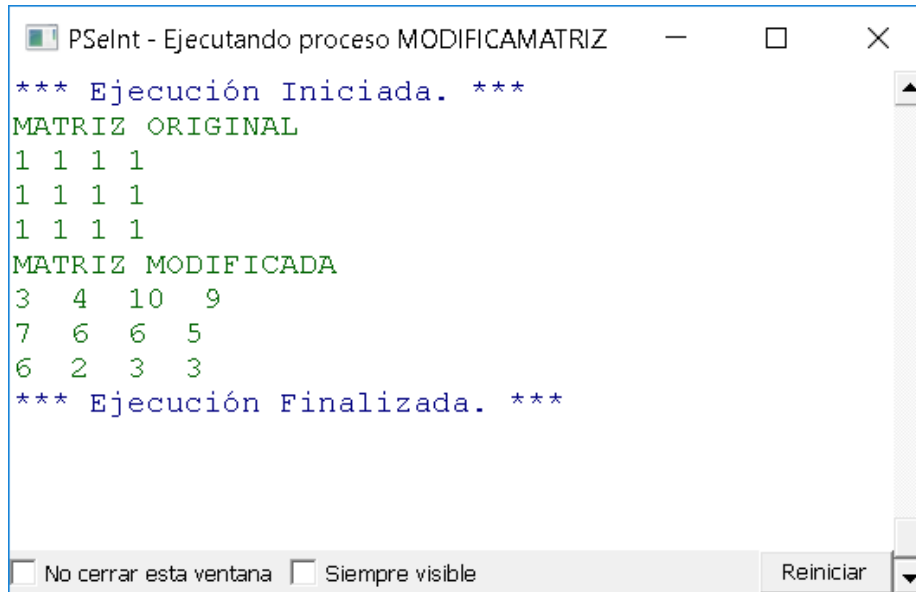
FinPara

FinPara

El fragmento anterior, almacena en la posición Fila,Columna de la matriz la suma de lo que contiene esa misma posición, que en el ejemplo es un uno, más un número aleatorio de 1 a 9, por lo que la posición tendrá guardado un número diferente al uno actual. La versión en pseudocódigo es la siguiente:

```
1  Algoritmo ModificaMatriz
2
3      // Declaración de variables
4      Definir Matriz Como Entero;
5      Definir Fila, Columna Como Entero;
6
7      // Dimensión de la matriz
8      Dimension Matriz(3,4);
9
10     // Inicialización de variables
11     Fila = 0;
12     Columna = 0;
13
14     // Llenado de la matriz
15     Para Fila=0 Hasta 2 Con Paso 1 Hacer
16     |
17     |   Para Columna=0 Hasta 3 Con Paso 1 Hacer
18     |   |   Matriz{Fila,Columna}=1;
19     |   FinPara
20     FinPara
21
22     // Muestra de la matriz
23     Escribir "MATRIZ ORIGINAL";
24     Para Fila=0 Hasta 2 Con Paso 1 Hacer
25     |
26     |   Para Columna=0 Hasta 3 Con Paso 1 Hacer
27     |   |   Escribir Matriz{Fila, Columna}, " ", Sin Saltar;
28     |   FinPara
29     |
30     |   Escribir "";
31     FinPara
32
33     // Modificación de la matriz
34     Para Fila=0 Hasta 2 Con Paso 1 Hacer
35     |
36     |   Para Columna=0 Hasta 3 Con Paso 1 Hacer
37     |   |   Matriz{Fila, Columna} = Matriz{Fila, Columna} + azar(9)+1;
38     |   FinPara
39     FinPara
40
41     // Muestra de la matriz
42     Escribir "MATRIZ MODIFICADA";
43     Para Fila=0 Hasta 2 Con Paso 1 Hacer
44     |
45     |   Para Columna=0 Hasta 3 Con Paso 1 Hacer
46     |   |   Escribir Matriz{Fila, Columna}, " ", Sin Saltar;
47     |   FinPara
48     |
49     |   Escribir "";
50     FinPara
51
52     FinAlgoritmo
```

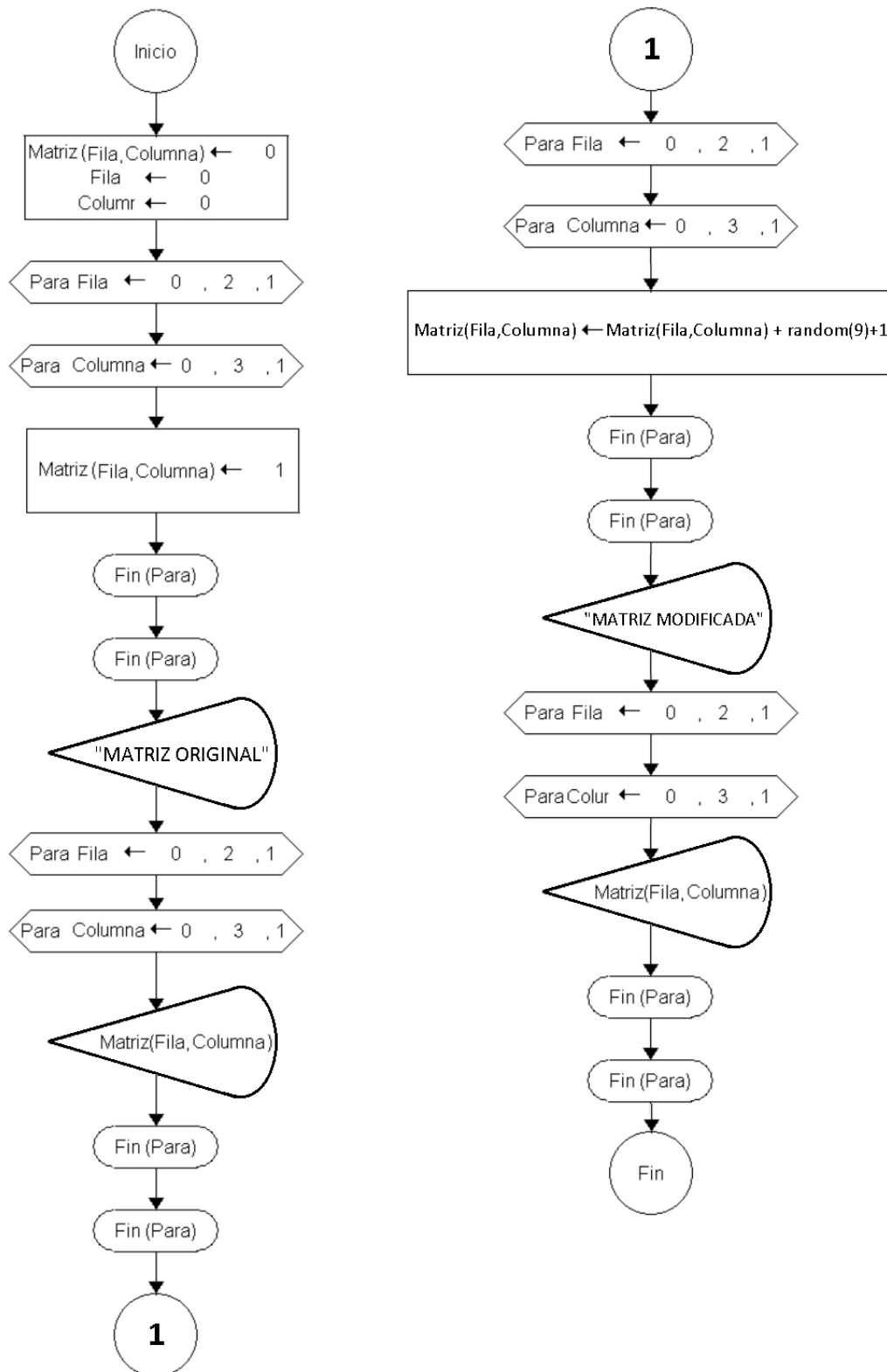
La salida del código anterior es la siguiente:



```
*** Ejecución Iniciada. ***  
MATRIZ ORIGINAL  
1 1 1 1  
1 1 1 1  
1 1 1 1  
MATRIZ MODIFICADA  
3 4 10 9  
7 6 6 5  
6 2 3 3  
*** Ejecución Finalizada. ***
```

☐ No cerrar esta ventana ☐ Siempre visible Reiniciar

El diagrama de flujo queda de la siguiente forma:



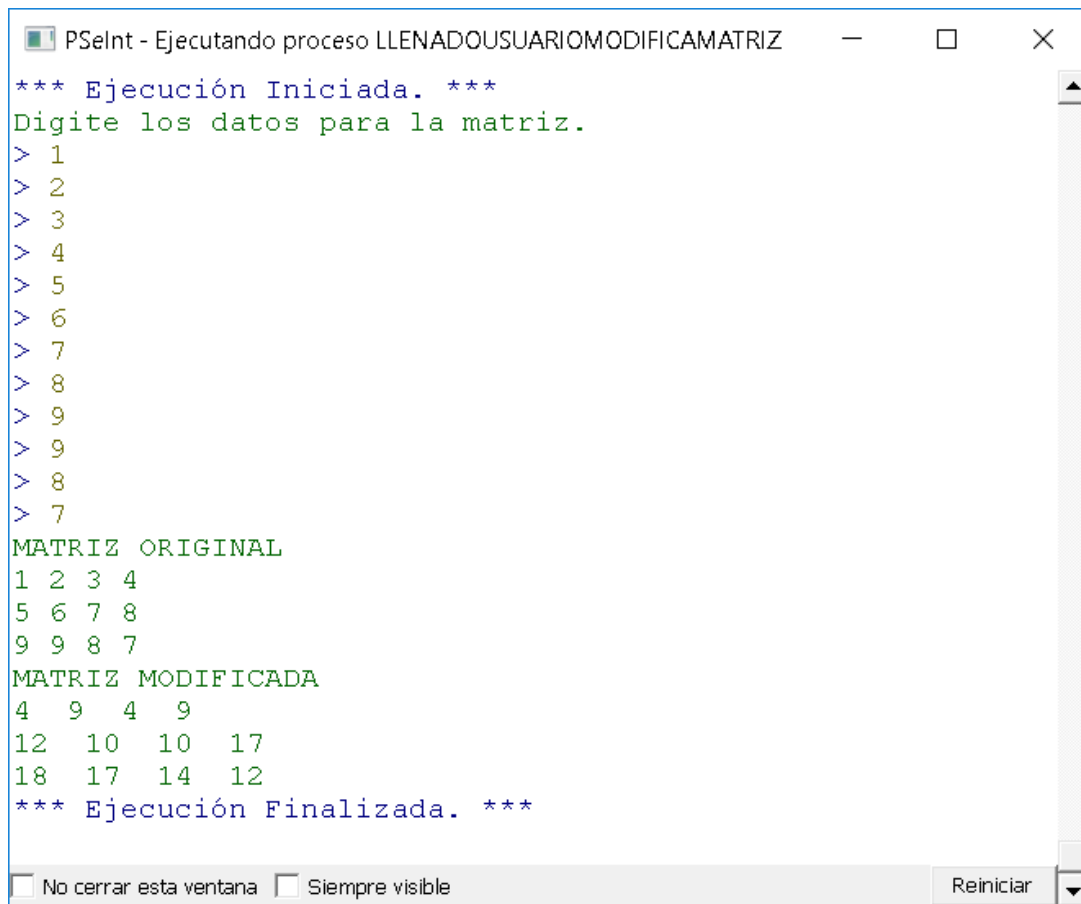
1.3.9 Llenado de usuario

Ya hemos analizado el llenado de una matriz de forma automática, ahora veamos cómo se hace cuándo es el usuario quien debe llenar el arreglo. La mecánica es similar a la empleada para los vectores, la diferencia es que se usan dos ciclos en vez de uno y la modificación se debe hacer en el ciclo Para interno. Tomaremos el código de modificación con números aleatorios visto anteriormente, pero el llenado lo hará el usuario, el código es el siguiente:

```
1  Algoritmo LlenadoUsuarioModificaMatriz
2
3  // Declaración de variables
4  Definir Matriz Como Entero;
5  Definir Fila, Columna Como Entero;
6
7  // Dimensión de la matriz
8  Dimension Matriz(3,4);
9
10 // Inicialización de variables
11 Fila = 0;
12 Columna = 0;
13
14 // Llenado de la matriz
15 Escribir "Digite los datos para la matriz.";
16 Para Fila=0 Hasta 2 Con Paso 1 Hacer
17     Para Columna=0 Hasta 3 Con Paso 1 Hacer
18         Leer Matriz(Fila,Columna);
19     FinPara
20 FinPara
21
22 // Muestra de la matriz
23 Escribir "MATRIZ ORIGINAL";
24 Para Fila=0 Hasta 2 Con Paso 1 Hacer
25     Para Columna=0 Hasta 3 Con Paso 1 Hacer
26         Escribir Matriz(Fila, Columna)," ", Sin Saltar;
27     FinPara
28     Escribir "";
29 FinPara
30
31 // Modificación de la matriz
32 Para Fila=0 Hasta 2 Con Paso 1 Hacer
33     Para Columna=0 Hasta 3 Con Paso 1 Hacer
34         Matriz(Fila, Columna) = Matriz(Fila, Columna) + azar(9)+1;
35     FinPara
36 FinPara
37
38 // Muestra de la matriz
39 Escribir "MATRIZ MODIFICADA";
40 Para Fila=0 Hasta 2 Con Paso 1 Hacer
41     Para Columna=0 Hasta 3 Con Paso 1 Hacer
42         Escribir Matriz(Fila, Columna)," ", Sin Saltar;
43     FinPara
44     Escribir "";
45 FinPara
46
47 FinAlgoritmo
```

La primera modificación fue un **mensaje**, antes del ciclo Para externo (línea 15), **solicitando al usuario** la digitación de los datos para la matriz; la otra modificación (línea 19) es la instrucción **Leer**, dentro del Para interno, con el Leer se le brinda la oportunidad al usuario de digitar cualquier número entero para llenar el arreglo.

El código generaría la siguiente salida en pantalla:

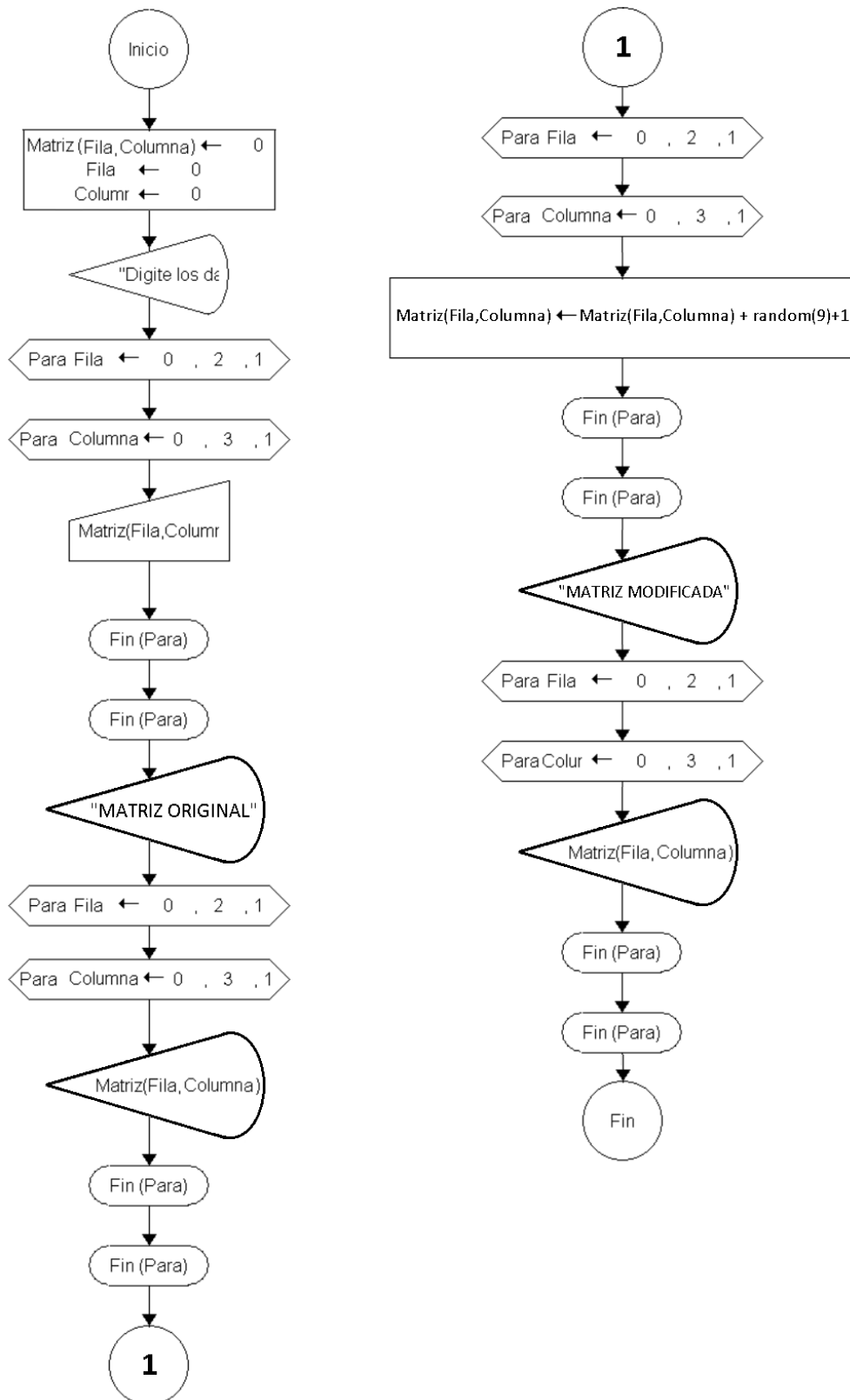


```
*** Ejecución Iniciada. ***
Digite los datos para la matriz.
> 1
> 2
> 3
> 4
> 5
> 6
> 7
> 8
> 9
> 9
> 8
> 7
MATRIZ ORIGINAL
1 2 3 4
5 6 7 8
9 9 8 7
MATRIZ MODIFICADA
4 9 4 9
12 10 10 17
18 17 14 12
*** Ejecución Finalizada. ***
```

☐ No cerrar esta ventana ☐ Siempre visible Reiniciar

Los doce números que aparecen luego del mensaje “Digite los datos para la matriz”, son los que digitó el usuario. Luego podemos observar la muestra de la matriz original, posterior a esta muestra se visualiza la matriz modificada, es decir, los números que digitó el usuario, pero con la suma de un número aleatorio de 1 a 9.

El diagrama de flujo del código anterior es el siguiente:



1.3.10 Llenado por azar (Random)

El llenado aleatorio de una matriz utiliza la función azar y mantiene la misma mecánica de ciclos, es decir, dos ciclos Para anidados.

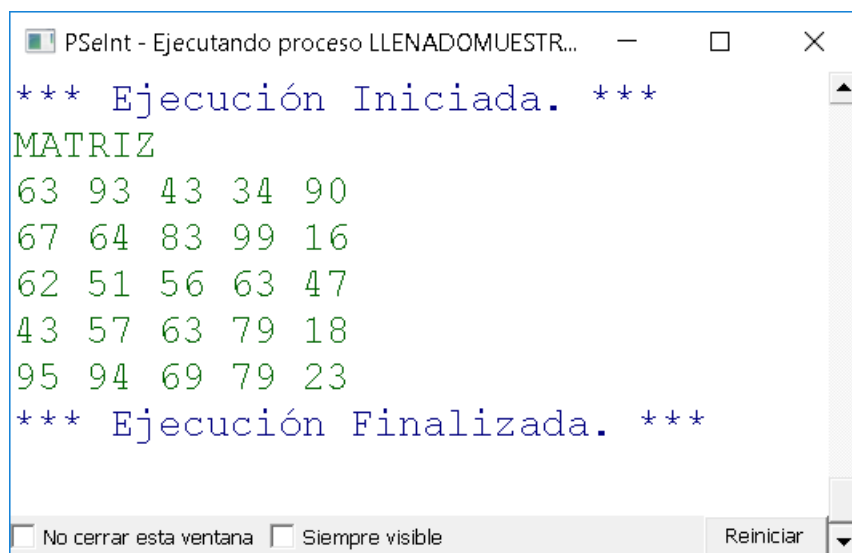
El funcionamiento del azar es el mismo que empleamos para el llenado de los vectores, nada más que, en esta ocasión, la asignación se hace a una matriz y no a un vector.

Veamos un ejemplo de llenado y muestra de una matriz cuadrada de 25 celdas, cuyos valores son aleatorios en un rango de 15 a 99, el código sería el siguiente:

```
1  Algoritmo LlenadoMuestraMatrizAzar
2
3      // Declaración de variables
4      Definir Matriz, Fila, Columna Como Entero;
5
6      // Dimensión de la matriz
7      Dimension Matriz(5,5);
8
9      // Inicialización de variables
10     Fila = 0;
11     Columna = 0;
12
13     // Llenado de la matriz
14     Para Fila=0 Hasta 4 Con Paso 1 Hacer
15     |
16         Para Columna=0 Hasta 4 Con Paso 1 Hacer
17         |     Matriz(Fila,Columna)= azar(85)+15;
18         FinPara
19     FinPara
20
21     // Muestra de la matriz
22     Escribir "MATRIZ";
23     Para Fila=0 Hasta 4 Con Paso 1 Hacer
24     |
25         Para Columna=0 Hasta 4 Con Paso 1 Hacer
26         |     Escribir Matriz(Fila, Columna), ' ', Sin Saltar;
27         FinPara
28         Escribir "";
29     FinPara
30
31 FinAlgoritmo
```

En el código, podemos ver que el cambio se dio en la línea 17, donde se emplea la función azar para generar un valor del 0 al 84 y luego se le suma un 15 al número generado, esto hace que el rango sea de 15 al 99; posterior a esa operación se asigna ese número a una celda específica de la matriz, al estar dentro de ciclos ese proceso se repite hasta cumplir con las condiciones de los Para y la matriz quedará llena.

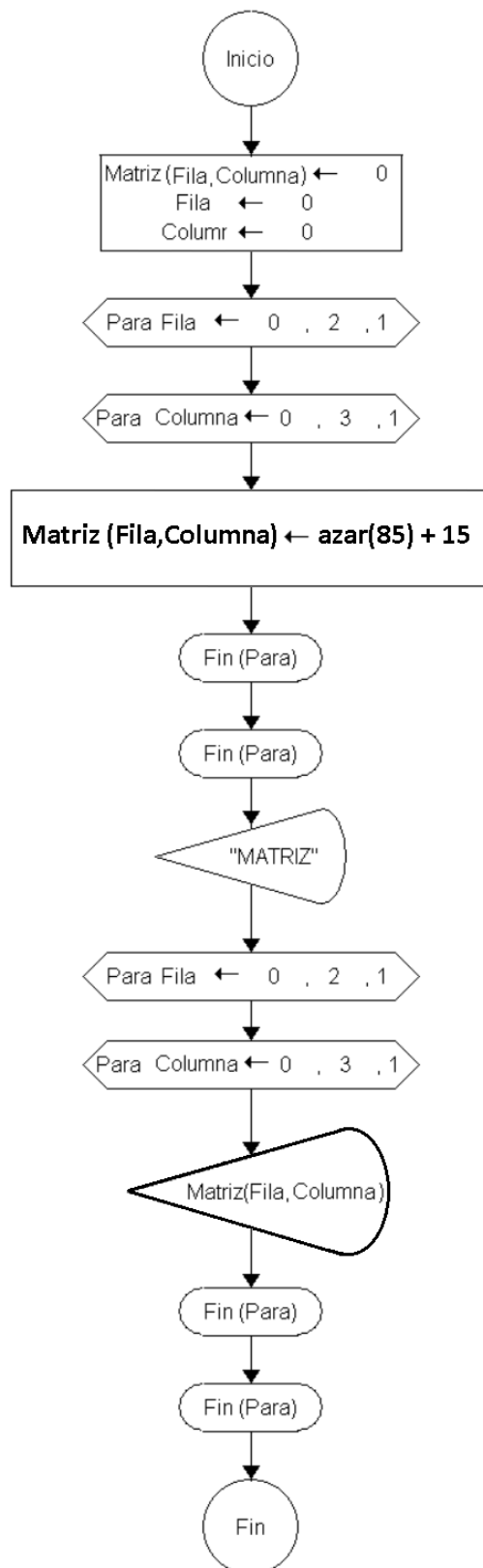
La salida de ese código es la siguiente, tomemos en cuenta que al ser números aleatorios los resultados pueden variar, pero la cantidad de elementos y el rango son los mismos:



```
*** Ejecución Iniciada. ***
MATRIZ
63 93 43 34 90
67 64 83 99 16
62 51 56 63 47
43 57 63 79 18
95 94 69 79 23
*** Ejecución Finalizada. ***
```

☐ No cerrar esta ventana ☐ Siempre visible Reiniciar

El diagrama de flujo del código anterior es el presentado a continuación:



Ya hemos analizado diversos procesos de los arreglos, veamos ahora algunos ejemplos un poco más complejos, pero que conservan el uso de las estructuras básicas.

Ejemplo 1

Buscar un número en una matriz y determinar cuántas veces aparece el número buscado en el arreglo. Considere los siguientes aspectos:

- La matriz es de 3 filas y 4 columnas.
- La matriz la llena el usuario con números enteros.
- Debe solicitar el número a buscar al usuario.
- Debe mostrar la matriz en forma cuadrada.
- Si el número está en la matriz debe mostrar cuántas veces está en el arreglo y si no lo está debe indicarlo también.

A continuación, analizaremos el código del ejemplo anterior:

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

```
1  Proceso BusquedaenMatriz
2
3      //Declaraciones de variables y matriz
4      Definir Fila, Columna Como Entero;
5      Definir Matriz Como Entero;
6      Definir NumeroBuscado Como Entero;
7      Definir CantidadCoincidencias Como Entero;
8
9      //Dimensionamiento de la matriz: 3 filas y 4 columnas
10     Dimension Matriz(3,4);
11
12     //Inicializaciones de variables
13     Fila=0;
14     Columna=0;
15     NumeroBuscado=0;
16     CantidadCoincidencias=0;
17
18     //Llenado de la matriz
19     Escribir "Digite los valores para la matriz.";
20     Para Fila<-0 Hasta 2 Con Paso 1 Hacer
21         Para Columna<-0 Hasta 3 Con Paso 1 Hacer
22             Escribir "Posición (",Fila,",",Columna,")": " ",Sin Saltar;
23             Leer Matriz(Fila,Columna);
24         FinPara
25     FinPara
26
27     //Solicitud y lectura de un número a buscar en la matriz
28     Escribir "Digite un número a buscar en la matriz.";
29     Leer NumeroBuscado;
30
31     //Búsqueda del número en la matriz
32     Para Fila<-0 Hasta 2 Con Paso 1 Hacer
33         Para Columna<-0 Hasta 3 Con Paso 1 Hacer
34             Si NumeroBuscado=Matriz(Fila,Columna) Entonces
35                 CantidadCoincidencias=CantidadCoincidencias+1;
36             FinSi
37         FinPara
38     FinPara
39
40     //Muestra de la matriz
41     Limpiar Pantalla;
42     Escribir "Valores de la matriz";
43     Para Fila<-0 Hasta 2 Con Paso 1 Hacer
44         Para Columna<-0 Hasta 3 Con Paso 1 Hacer
45             Escribir Matriz(Fila,Columna)," ",Sin Saltar;
46         FinPara
47     Escribir "";
48     FinPara
49
50     //Muestra de resultado de la búsqueda
51     Escribir "";
52     Escribir "Resultado.";
53     Si CantidadCoincidencias>0 Entonces
54         Escribir "El número ",NumeroBuscado," está ",CantidadCoincidencias," veces en la matriz.";
55     Sino
56         Escribir "El número ",NumeroBuscado," NO está en la matriz.";
57     FinSi
58
59     FinProceso
```

Para el ejemplo declaramos las variables en las líneas cuatro a las siete, luego dimensionamos la matriz y finalmente hacemos las inicializaciones.

De la línea 19 a la 25 llenamos la matriz con los números que digite el usuario y en las líneas 27 y 28 solicitamos y leemos el número a buscar en el arreglo.

La **búsqueda** del número digitado por el usuario en la matriz la hacemos en las **líneas 32 a la 38**, notemos que se usan **dos ciclos Para anidados** y en el Para interno implementamos un **Si para comparar** el contenido de la variable **NumeroBuscado** con una **celda específica** de la matriz (**Si NumeroBuscado=Matriz(Fila,Columna)**), si la condición tiene un valor de **verdadero** entonces se **incrementa** en uno la variable **CantidadCoincidencias**, esto se repite hasta que se abarquen todas las celdas de la matriz.

Una vez finalizados esos dos ciclos anidados se emplean otros dos para **mostrar la matriz**, **líneas 42 a la 48**, veamos que se utilizan las instrucciones **Sin Saltar**, en el Escribir dentro del Para interno y un **Escribir** fuera de ese mismo ciclo interno.

Finalmente, en las **líneas 51 a 57** se muestran los resultados de la búsqueda, en un **Si** evaluamos el valor de la variable **CantidadCoincidencias**, si la misma tiene un valor **superior a cero** significa que el número buscado se encontraba, al menos una vez, en la matriz; de lo contrario (**Sino**) se determina que el número no estaba en el arreglo.

Ejemplo 2

Determinar la sumatoria, el promedio, el número mayor y el número menor de una matriz cuadrada. Considere los siguientes aspectos:

1. La matriz tendrá una dimensión máxima de 10 filas y 10 columnas, pero el usuario podrá definir la dimensión de la matriz para cada ejecución del algoritmo.
2. La dimensión que digite el usuario para la matriz debe estar validada, la misma no puede ser menor de 1 ni superior a 10.
3. La matriz se debe llenar con números aleatorios del 100 al 500.
4. Al final, debe mostrar un informe con los siguientes datos de la matriz:
 - a. Sumatoria.
 - b. Promedio.
 - c. Número mayor.
 - d. Número menor.

Este documento forma parte de una unidad didáctica que está en desarrollo en la Universidad Estatal a Distancia de Costa Rica. Su contenido se encuentra bajo la ley de Propiedad Intelectual y cualquier mención a este debe ser indicado como obra en proceso.

El código completo del algoritmo es el siguiente:

```
1  Proceso PromedioMatriz
2
3  /***DECLARACIONES***/
4  Definir Matriz,Fila,Columna Como Entero;
5  Definir Sumatoria,NumMayor,NumMenor,Casillas,TamanoMatriz Como Entero;
6  Definir Promedio Como Real;
7
8  /***DIMENSIONAMIENTO DE LA MATRIZ***/
9  Dimension Matriz(10,10);
10
11 /***INICIALIZACIÓN DE VARIABLES***/
12 Promedio=0;
13 Sumatoria=0;
14 Casillas=0;
15 TamanoMatriz=0;
16
17 /***DIMENSIONAMIENTO DE LA MATRIZ POR EL USUARIO***/
18 Escribir "Bienvenid@";
19 Repetir
20     Escribir "Digite el tamaño de la matriz.";
21     Leer TamanoMatriz;
22     Si TamanoMatriz<1 O TamanoMatriz>10 Entonces
23         Limpiar Pantalla;
24         Escribir "Error. El tamaño de la matriz debe estar entre 1 y 10.";
25     FinSi
26 Hasta Que TamanoMatriz>=1 Y TamanoMatriz<=10
27
28
29 /***LLENA LA MATRIZ Y CALCULA LA SUMATORIA EN EL MISMO CICLO***/
30 Para Fila=0 Hasta TamanoMatriz-1 Con Paso 1 Hacer
31     Para Columna<=0 Hasta TamanoMatriz-1 Con Paso 1 Hacer
32         Matriz[Fila,Columna]= azar(401)+100;
33         Sumatoria=Sumatoria+Matriz[Fila,Columna];
34         Casillas=Casillas+1;
35     FinPara
36 FinPara
37
38 /***CÁLCULO DEL PROMEDIO***/
39 Promedio=Sumatoria/Casillas;
40
41
42
43 NumMayor=Matriz[0,0];
44 NumMenor=Matriz[0,0];
45
46 /***MUESTRA LA MATRIZ Y EL PROMEDIO***/
47 /***DETERMINA NÚMERO MAYOR Y MENOR DE LA MATRIZ***/
48 Limpiar Pantalla;
49 Escribir "MATRIZ";
50 Para Fila=0 hasta TamanoMatriz-1 con paso 1 Hacer
51     Para Columna=0 hasta TamanoMatriz-1 con paso 1 Hacer
52
53         Escribir Matriz[Fila,Columna], " " Sin Saltar;
54
55         Si Matriz[Fila,Columna]>NumMayor Entonces
56             NumMayor<-Matriz[Fila,Columna];
57         FinSi
58
59         Si Matriz[Fila,Columna]<NumMenor Entonces
60             NumMenor=Matriz[Fila,Columna];
61         FinSi
62
63     FinPara
64
65     Escribir " ";
66
67 FinPara
68
69 Escribir "";
70 Escribir "-----";
71 Escribir "INFORME";
72 Escribir "La suma de toda la matriz es de:", Sumatoria;
73 Escribir "El promedio de toda la matriz es de:", Promedio;
74 Escribir "El número mayor es: ", NumMayor;
75 Escribir "El número menor es: ", NumMenor;
76 Escribir "-----";
77 Escribir " ";
78 FinProceso
```


Debido a la extensión del código se hará la explicación por secciones.

I Parte: Declaraciones, dimensionamiento e inicializaciones

```
1  Proceso PromedioMatriz
2
3      /**DECLARACIONES**/
4      Definir Matriz,Fila,Columna Como Entero;
5      Definir Sumatoria,NumMayor,NumMenor,Casillas,TamanoMatriz Como Entero;
6      Definir Promedio Como Real;
7
8      /**DIMENSIONAMIENTO DE LA MATRIZ**/
9      Dimension Matriz(10,10);
10
11     /**INICIALIZACIÓN DE VARIABLES**/
12     Promedio=0;
13     Sumatoria=0;
14     Casillas=0;
15     TamanoMatriz=0;
```

En esta primera sección del código se hacen las declaraciones de las líneas 4 a la 6. Luego, en la línea 9, hacemos el dimensionamiento general del arreglo, recordemos que en la situación a resolver nos dicen que el usuario define el tamaño de la matriz, pero que esta será de máximo 10 filas y 10 columnas, es decir, el usuario no puede digitar una dimensión que esté fuera del rango 1–10.

Por último, de las líneas 12 a la 15, se efectúan las inicializaciones de las variables.

II Parte: Dimensionamiento hecho por el usuario y validación del tamaño de la matriz

```
16
17     /**DIMENSIONAMIENTO DE LA MATRIZ POR EL USUARIO**/
18     Escribir "Bienvenid@";
19     Repetir
20     |   Escribir "Digite el tamaño de la matriz.";
21     |   Leer TamanoMatriz;
22     |   Si TamanoMatriz<1 O TamanoMatriz>10 Entonces
23     |   |   Limpiar Pantalla;
24     |   |   Escribir "Error. El tamaño de la matriz debe estar entre 1 y 10.";
25     |   FinSi
26     Hasta Que TamanoMatriz>=1 Y TamanoMatriz<=10
```

En la segunda parte el código, se solicita, lee y valida el tamaño que el usuario definirá para la matriz. Dentro de un ciclo **Repetir** se solicita y lee el tamaño, luego, y mediante un **Si**, evaluamos si el **valor digitado** por el usuario es **menor a uno o mayor a diez**, si lo es mostramos un **mensaje de error** y el ciclo se repite; pero si el valor digitado está dentro del **rango de 1 a 10**, entonces el ciclo termina y se continúa con el resto del algoritmo.

Lo que tenemos una vez fuera del ciclo es una **variable** con un valor **entre 1 y 10** que servirá de **límite** de los ciclos **Para** que usaremos en el manejo de la matriz.

III Parte: Llenado de matriz, sumatoria y cálculo del promedio

```
27
28
29      /**LLENA LA MATRIZ Y CALCULA LA SUMATORIA EN EL MISMO CICLO**/
30      Para Fila=0 Hasta TamanoMatriz-1 Con Paso 1 Hacer
31          Para Columna<-0 Hasta TamanoMatriz-1 Con Paso 1 Hacer
32              Matriz[Fila,Columna]= azar(401)+100;
33              Sumatoria=Sumatoria+Matriz[Fila,Columna];
34              Casillas=Casillas+1;
35          FinPara
36      FinPara
37
38      /**CÁLCULO DEL PROMEDIO**/
39      Promedio=Sumatoria/Casillas;
40
41
42
43      NumMayor=Matriz[0,0];
44      NumMenor=Matriz[0,0];
45
```

En la tercera parte del código, nos enfocaremos en el llenado de la matriz y los cálculos solicitados. En los ciclos de llenado, notemos que el límite llega hasta `TamanoMatriz-1`, esto es debido a que el conteo de las celdas inicia en cero y debe llegar hasta el tamaño que digitó el usuario, pero si no hacemos la resta (`TamanoMatriz-1`) entonces contaremos una celda más de la que realmente debería tener el arreglo.

Por ejemplo, si el usuario determinó que el tamaño de la matriz fuese de 5 entonces el conteo iría de cero a cinco, con lo que estaría procesando seis celdas (0, 1, 2, 3, 4, 5) para evitar que esto pase, restamos una unidad al límite establecido por el usuario, por ende tendría un conteo de cero a cuatro, por lo que procesaríamos las cinco celdas deseadas (0, 1, 2, 3, 4).

Una vez aclarado el detalle del límite, veamos el ciclo Para interno; en la línea 32 se asigna a la matriz un número aleatorio del 100 al 500. En la línea 33 se acumula el contenido de la celda en la variable Sumatoria, esto más adelante nos servirá para el promedio. Finalmente, en la línea 34, se incrementa el contador de casillas, esto también lo utilizaremos para el cálculo del promedio.

Al salir de los ciclos Para anidados, se calcula el promedio, línea 39 y se asignan a las variables NumMayor y NumMenor el valor de la posición 0,0 de la matriz.

IV Parte: Determina número mayor y menor, y muestra de resultados

```
46  /***MUESTRA LA MATRIZ Y EL PROMEDIO***/
47  /***DETERMINA NÚMERO MAYOR Y MENOR DE LA MATRIZ***/
48  Limpiar Pantalla;
49  Escribir "MATRIZ";
50  Para Fila=0 hasta TamanoMatriz-1 con paso 1 Hacer
51      Para Columna=0 hasta TamanoMatriz-1 con paso 1 Hacer
52          Escribir Matriz[Fila,Columna], " " Sin Saltar;
53          Si Matriz[Fila,Columna]>NumMayor Entonces
54              NumMayor<-Matriz[Fila,Columna];
55          FinSi
56          Si Matriz[Fila,Columna]<NumMenor Entonces
57              NumMenor=Matriz[Fila,Columna];
58          FinSi
59      FinPara
60      Escribir " ";
61  FinPara
62
63  Escribir "";
64  Escribir "-----";
65  Escribir "INFORME";
66  Escribir "La suma de toda la matriz es de:", Sumatoria;
67  Escribir "El promedio de toda la matriz es de:", Promedio;
68  Escribir "El número mayor es: ", NumMayor;
69  Escribir "El número menor es: ", NumMenor;
70  Escribir "-----";
71  Escribir " ";
72  FinProceso
```

En la última parte del código, se determina el **número mayor y menor del arreglo** y se muestran los resultados. En los ciclos Para anidados que inician en la **línea 50**, primero se **muestra** una celda de la **matriz** (línea 53).

Luego de esa muestra, se emplean dos **Si** por separado, en el primero se evalúa si el contenido de la **celda** actual de la matriz es **mayor** al valor de la variable **NumMayor**, de ser así se hace la **sustitución** del valor actual de NumMayor por el valor de la celda de la matriz, ya que ese contenido de la celda sería el nuevo número mayor.

En el **segundo Si**, se evalúa si el contenido de la **celda** actual es **menor** que el valor de **NumMenor**, si esto es verdadero, entonces se hace la sustitución respectiva.

Al finalizar esos ciclos, tendremos la matriz impresa en pantalla y habremos determinado el número mayor y menor del arreglo. Finalmente, de las líneas **69 a la 77**, se muestran los **resultados**.