

República Bolivariana de Venezuela  
Ministerio de Educación Superior  
Universidad Tecnológica del Centro



## ARREGLOS INTRODUCCIÓN A LA INFORMÁTICA

«Él es quien perdona todas tus maldades  
el que sana tus dolencias, el que rescata  
del hoyo tu vida, el que te corona de  
favores y misericordia, el que sacia de bien  
tu boca de modo que te rejuvenezcas  
como el águila.»  
Salmo 103:3

Profesor : DELY M. GIL A.  
VALENCIA, Nov .2007

### ARREGLOS (ARRAY)

Un array es una estructura de datos en la que se almacena una colección de datos del mismo tipo (por ejemplo las notas de los alumnos). Al tipo se le llama *tipo base* del arreglo. Los datos individuales se llaman *elementos del arreglo*.

### TIPOS DE ARREGLOS

Los arreglos pueden ser:

- Unidimensionales, también llamados Vectores o listas
- Bidimensionales, denominados tablas o matrices.
- Multidimensionales, con tres o más dimensiones.

### CARACTERÍSTICAS DE LOS ARREGLOS

Un arreglo se caracteriza por :

1. Almacenar los elementos del arreglo en posiciones de memoria continua
2. Tener un único nombre de variable que representa a todos los elementos (Notas), y éstos a su vez se diferencian por un índice o subíndice.  
Notas[0], ..., Notas[n-1] {Lenguaje C}  
Notas[1]..Notas[n] {Pascal}
3. Acceso directo o aleatorio a los elementos individuales del arreglo.

## FORMATO DE DECLARACIÓN DE UN ARRAY

*nombre\_array* : arreglo [*tipo subíndice*] de *tipo*

*nombre\_array* identificador válido

*tipo subíndice* puede ser de tipo ordinal:

*entero*, *carácter* pero no **REAL**.

Existe un elemento por cada valor del tipo Subíndice.

*tipo* describe el tipo de cada elemento del vector  
 todos los elementos de un vector son del mismo tipo

## DECLARACIÓN DE UN ARRAY

Las variables de tipo **arreglo** se declaran en la sección **Constante**, o **Variables**.

*CONSTANTE*

arr\_Num : arreglo [1..2] de entero = (-5.5);

Declaración en Variables:

*VARIABLES*

nombres : arreglo[1..30] de string[30];

calif : arreglo[1..30] de real;

numero : arreglo[0..100] de 1..100;

## SUBÍNDICE DE UN ARRAY

El subíndice o índice de un arreglo debe ser de tipo simple:  
 entero, carácter NO REAL. Ejemplo :

1..10 entero

'C'..'N' char

Variables

Conjunto = arreglo['A'..'Z'] de entero;

Alumnos = arreglo [1..10] de real;

## TIPO DE DATO DE UN ARREGLO

El tipo de dato de un array puede ser cualquier dato válido,  
 Ejemplos:

a) Variables

Estado : array[1..100] de logico;

ArrayReal : Arreglo[1..15] de real;

NumeroDeArray : Arreglo[0..100] de 1..1999;

Linea : Arreglo[1..80] de Caracter;

## ARREGLO UNIDIMENSIONAL

Un arreglo de una dimensión (vector o lista) es un tipo de datos estructurado compuesto de un número de elementos finitos, tamaño fijo y elementos homogéneos.

*Finito* indica que hay un último elemento, *tamaño fijo* significa que el tamaño del arreglo debe ser conocido en tiempo de compilación, *homogéneo* significa que todos son del mismo tipo.

## OPERACIONES CON ARREGLOS

1. Lectura de un vector  
Repita para i:=0 hasta 100-1 hacer  
    Leer(Notas[i]);
- 2.- Escritura de un Vector  
Repita para i:= 1 hasta Numero-1 hacer  
    Escribir(Notas[i]);
- 4.- Búsqueda
  - Lineal (Secuencial) {No ordenados}
  - Binaria para arreglos ordenados
- 5.- Ordenamiento
  - Inserción, Selección, Burbuja o Intercambio, Shell, QuickSort (Ordenación rápida), Ordenación por Fusión o Mezcla
- 6.- Inserción y eliminación

## Vector\_edades;

{El siguiente programa captura 20 edades}

Constante

MaxPersonas = 10;

Var

edades : arreglo [1..MaxPersonas] de entero;

//En lenguaje C se comienza desde 0.

I : entero;

/\*\*\*\*\*\*Programa Principal\*\*\*\*\*\*/

Inicio

{lectura de arreglo}

Repita para i:=1 hasta MaxPersonas hacer

    Inicio

        Escribir('Edad de la ',i,' persona : ');

        Leer(edades[i]);

    Fin; //RP

**//CALCULE EL PROMEDIO DE LAS EDADES**

**Genere un vector de tamaño 20 con números entre 0 y 10. Almacene en un vector A todos los números negativos y en un vector B todos los positivos o iguales a cero**

```
NEGATIVOS_POSITIVOS;
```

```
/*El siguiente programa genera un vector de 10 elementos
entre 0 y 10 y almacene los valores positivos y negativos
En vectores diferentes*/
```

```
Constante
```

```
Elementos = 10;
```

```
Var
```

```
V : arreglo [1..Elementos] de entero;
```

```
i : entero;
```

```
Inicio
```

```
i=1;
```

```
j=1;
```

```
k=1
```

```
Mientras (i<=Elementos) hacer
```

```
Inicio
```

```
V[i] = random(11) ; //Rango entre [0..10];
```

```
Si (v[i]<0)
```

```
Inicio
```

```
a[j]=v[i] ;
```

```
j=j+1 ;
```

```
Fin
```

```
Sino
```

```
Inicio
```

```
b[k]=v[i] ;
```

```
k=k+1;
```

```
Fin
```

```
i=i+1;
```

```
Fin Mientras;
```

```
Fin.
```

**Genere un vector de tamaño 10 con números reales leídos desde teclado. Calcule el promedio e indique cuantos elementos del vector son mayores que el promedio y cuantos menores o iguales.**

```
MAY_MEN_PROMEDIO;
```

```
Limite = 10;
```

```
Variables
```

```
V : Array[1..Limite] de real;
```

```
Identidad : logico;
```

```
I,contmay,contmen : entero;
```

```
Inicio
```

```
Contmay :=0;
```

```
Contmen :=0;
```

```
Suma :=0;
```

```
Repita para i:= 1 hasta Limite hacer
```

```
Inicio
```

```
Escribir('Elemento ',i,' del vector: ');
```

```
Leer(V[i]);
```

```
Suma := Suma + V[i];
```

```
Fin; //RP
```

```
Promedio: =suma/Limite
```

```
// Contar mayores y menores
```

```
i=1;
```

```
Mientras (i<Limite) hacer
```

```
Inicio
```

```
Si (V[i] > promedio)
```

```
contmay=contmay+1
```

```
Sino
```

```
contmen=contmen + 1;
```

```
i=i+1
```

```
Fin Mientras;
```

```
Escribir (' La cantidad de elementos mayores que el
promedio son',contmay);
```

```
Escribir mostrar (' La cantidad de elementos menores o
iguales al promedio son',contmen);
```

```
Fin.
```

```

Identidad;
{Programa que verifica si dos vectores son iguales.
Usar una variable lógica.}
Constante
  Limite = 5;
Variables
  M,N      : Arreglo[1..Limite] de enteros;
  Identidad : logico;
  I        : entero

Inicio
  Repita para i:= 1 hasta Limite hacer
    Inicio
      M[i]:=Random(41)-20; {Rango entre [-20..20]}
      N[i]:=Random(41)-20;
    Fin RP;
  i:=1;
  While (I<Limite) y (A[I] = B[I]) hacer
    I:= I+1;
  Si A[I] = B[I] entonces
    Identidad := Verdadero
  Sino
    Identidad = Falso;

  Según Identidad sea
    Verdadero : Escribir ('Los arreglos son iguales');
    Falso     : Escribir ('Los arreglos no son iguales');
  Fin; //Según Sea
Fin.

```

```

Suma_Vectores;
{El siguiente programa suma dos vectores }
Constante
  N      = 12;
Variables
  X ,Y   = Arreglo[1..N] de entero;

Inicio

  Repita para j:= 1 hasta N hacer
    Inicio
      Escribir('Elemento ',i,' del vector X: ');
      Leer(X[j])
      Escribir('Elemento ',i,' del vecto Yr: ');
      Leer(Y[j])
    Fin;
  Repita para j:= 1 hasta N hacer
    Z[I] := X[i] + Y[i];
  Repita para j:= 1 hasta N hacer
    Escribir(Z[j])
Fin.

```

**EJERCICIO:** Dado un vector con 10 elementos numéricos enteros ya almacenados, indique cuántos de ellos son múltiplos de 3

## ARREGLOS PARALELOS

Dos o más arreglos que utilizan el mismo subíndice para referirse a términos homólogos se llaman arreglos paralelos.

Basados en el programa anterior se tienen las edades de 'x' personas, para saber a que persona se refiere dicha edad se puede usar otro arreglo en forma paralela y asociarle los nombres de manera simultánea con las edades.

Nombres[1]	<b>Fidel</b>	Edades[1]	<b>82</b>
Nombres[2]	<b>Ely</b>	Edades[2]	<b>28</b>
	.		.
	.		.
	.		.
Nombres[10]	<b>Juan</b>	Edades[10]	<b>9</b>

Paralelo\_edades;  
{El siguiente programa captura 10 edades y nombres por medio de arreglos paralelos y los muestra ordenados en forma ascendente}

Const

MaxPersonas = 10;

Var

edades :arreglo [1..MaxPersonas] de entero;

nombres :arreglo [1..MaxPersonas] de cadena;

aux\_nom :cadena;

i,j,aux\_edad :entero;

begin

ClrScr;

{lectura de arreglos paralelos de manera simultánea}

Repita para i:=1 to MaxPersonas hacer

inicio

Escribir(i,'.- Nombre : ','Edad : ');

Leer(nombres[i]) ;

Escribir(i,'.- Edad : ');

Leer(edades[i])

fin;

//Calcular la mayor edad e indicar el nombre de la persona que tenga dicha edad.

### ORDENAMIENTO BURBUJA (BUBBLE)

Consiste en realizar pasadas sucesivas direccionando desde el primer elemento hasta el penúltimo, comparando cada uno de ellos con el siguiente. Esta versión del método va colocando en cada pasada el menor elemento de los tratados en la primera posición (burbujean), y el mayor al fondo del arreglo (hunden).

Gráficamente podemos observar lo siguiente:

**Vector**

23	19	45	15
----	----	----	----

**Pasada 1: i:= 1**

23	19	45	15
----	----	----	----

J=1

19	23	45	15
----	----	----	----

J=2

19	23	45	15
----	----	----	----

J=3

19	23	15	45
----	----	----	----

**Pasada 2: i:= 2**

19	23	15	45
----	----	----	----

J=1

19	23	15	45
----	----	----	----

J=2

19	15	23	45
----	----	----	----

**Pasada 3: i:= 3**

19	15	23	45
----	----	----	----

J=1

15	19	23	45
----	----	----	----

**Observar:**

- Se usan 3 pasadas para arreglos de 4 elementos, es decir, n-1 pasadas.
- **LAS PASADAS SON I := N-1**
- Para la pasada 1 (i=1) se realizan comparaciones (j=1,2,3)
- Para la pasada 2 (i=2) se realizan comparaciones (j=1,2)
- Para la pasada 3 (i=3) se realizan comparaciones (j=1)
- **LAS COMPARACIONES SON: 1..N-I**

**ALGORITMO A**

```

{ordenación del ejercicio de edades}
Repita para i:=1 hasta MaxPersonas-1 hacer
  Inicio
  Repita para j:=1 hasta MaxPersonas-i hacer
    Inicio
    Si edades[j]>edades[j+1] entonces
      Inicio
      aux          :=edades[j];
      edades[j]    :=edades[j+1];
      edades[j+1] :=aux
      Fin;
    Fin;
  Escribir(edades[i]) //escritura del array
  Fin;

```

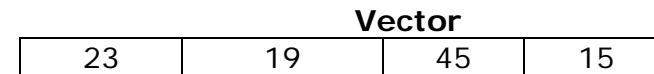
**ALGORITMO B**

```

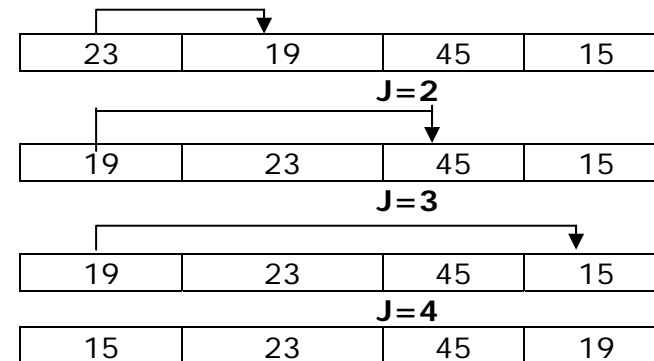
{ordenación del ejercicio de edades}
Repita para i:=1 hasta MaxPersonas-1 hacer
  Inicio
  Repita para j:=i+1 hasta MaxPersonas hacer
    Inicio
    Si edades[i]>edades[j] entonces
      Inicio
      aux          :=edades[i];
      edades[i]    :=edades[j];
      edades[j]    :=aux
      Fin;
    Fin;
  Escribir(edades[i]) //escritura del array
  Fin;

```

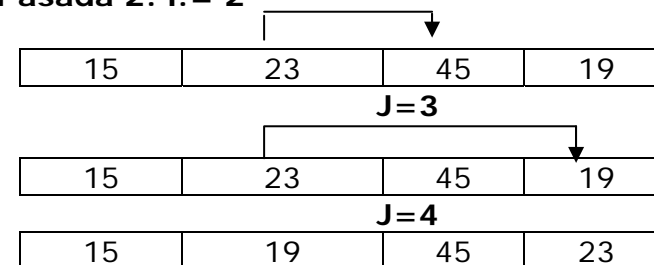
Gráficamente podemos observar lo siguiente:



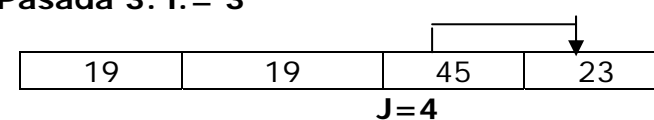
**Pasada 1: i:= 1**



**Pasada 2: i:= 2**



**Pasada 3: i:= 3**



**BÚSQUEDA LINEAL:** Recorre el arreglo hasta encontrar el elemento a buscar. Si encuentra el elemento devuelve la posición y la variable lógica en verdadero de lo contrario devuelve posición -1 y falso en la variable lógica.

```

Encontrado := falso;
Posicion := -1;
l := 1;
Mientras (i <= n) y encontrado = falso) hacer
  Inicio
    Si a[i] = elemento entonces
      Inicio
        Posicion := i;
        Encontrado := verdadero;
      Fin
    l := i+1;
  Fin.

```

**BÚSQUEDA BINARIA:** Antes de ejecutar esta búsqueda es necesario que el vector esté ordenado.

```

Primero := 1 ; Ultimo := n; encontrado := falso;
Mientras (Primero <= Ultimo ) y (encontrado = falso) hacer
  Inicio
    Central := (primero+Ultimo) div 2;
    Si elemento = A[Central] ent encontrado := verdadero
  Sino
    Si Elemento > A[Central] ent
      Primero := Central + 1;
    sino
      Ultimo := Central - 1
  Fin
Si (encontrado = falso ) entonces Binaria := -1
Sino Binaria := Central;

```

### UNIÓN DE VECTORES

Pasos a seguir:

- ❖ Recorrer el primer arreglo (A)
- ❖ Buscar que no esté en el vector de Unión (C) para ser insertado en C.
- ❖ Recorrer el segundo arreglo (b)
- ❖ Buscar que no esté en el vector de Unión (C) para ser insertado en C.

10	20	30	40	50	A		
2	10	4	10	50	B		
10	20	30	40	50	2	4	C

### INTERSECCIÓN DE VECTORES

Pasos a seguir:

- ❖ Recorrer el primer arreglo (A)
- ❖ Buscar el elemento en (B)
- ❖ Si el elemento está en B
  - Buscar que no esté en C para ser insertado.

10	20	30	40	50	A
2	10	4	10	50	B
10	50	C			

## INSERCIÓN DE ELEMENTOS EN VECTORES

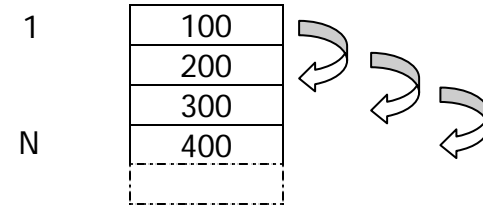
Requiere de cuatro acciones:

- Verificar si hay espacio en el vectos.
- Desplazar los elementos para liberar el espacio para el valor a insertar
- Asignar el valor en la posición
- Modificar la dimensión del arreglo

Casos de Inserción:

- Al inicio del arreglo
- Entre elementos del arreglo
- Al final del arreglo

## INSERCIÓN AL INICIO DEL ARREGLO (POS = 1)



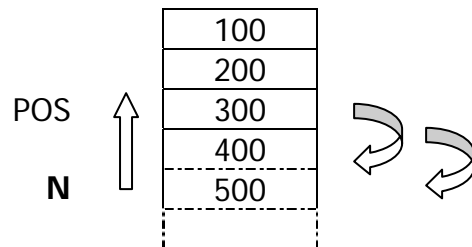
Para  $i := N$  hasta 1 hacer

$\text{Vector}[i+1] := \text{Vector}[i];$

$\text{Vector}[1] := \text{Valor}$

$N := N + 1;$

## INSERCIÓN ENTRE ELEMENTOS ( $1 > \text{POS} < N$ )



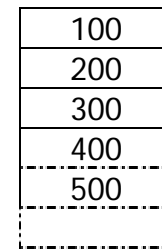
Para  $i := N$  hasta POS hacer

$\text{Vector}[i+1] := \text{Vector}[i];$

$\text{Vector}[\text{POS}] := \text{Valor}$

$N := N + 1;$

## INSERCIÓN AL FINAL DEL ARREGLO (POS = N+1)



$\text{Vector}[\text{POS}] := \text{Valor}$

$N := \text{Inc}(N);$

## ELIMINACIÓN DE ELEMENTOS EN VECTORES

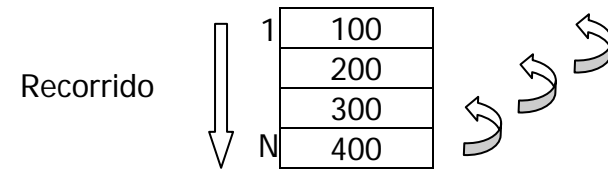
Requiere de dos acciones:

- Desplazar los elementos para eliminar el valor
- Modificar la dimensión del arreglo

Casos de Eliminación:

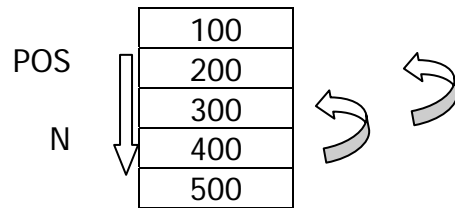
- El primer elemento del arreglo (POS=1)
- Entre elementos del arreglo
- Al final del arreglo (POS =N)

## ELIMINACIÓN AL INICIO DEL ARREGLO (POS = 1)



Para  $i := 2$  hasta  $N-1$  hacer  
 Vector[i] := Vector[i+1];  
 N := N-1;

## ELIMINACIÓN ENTRE ELEMENTOS ( $1 > POS < N$ )



Para  $i := POS$  hasta  $N-1$  hacer  
 Vector[i] := Vector[i+1];  
 N := N - 1;

## ELIMINACIÓN AL FINAL DEL ARREGLO (POS = N+1)

100
200
300
400
500

N := N-1;

## ARREGLO BIDIMENSIONAL

Un arreglo bidimensional (tabla o matriz) es un arreglo con dos índices, al igual que los vectores que deben ser ordinales o tipo subrango.

		Columnas		
		1	2	3
Filas	1	A[1,1]	A[1,2]	A[1,3]
	2			
	3			
	4	A[4,1]	A[4,2]	A[4,3]

Para localizar o almacenar un valor en el arreglo se deben especificar dos posiciones (dos subíndices), uno para la fila y otro para la columna.

## FORMATO DE ARREGLO BIDIMENSIONALES

*identificador* = array [*índice1*] [*índice 2*] de tipo de elemento

### MANIPULACIÓN DE TABLA

Recorrido por fila

Para Fila := 1 hasta 3 hacer

Para Columna:= 1 hasta 4 hacer

Leer(A[*fila*][*columna*]);

Recorrido por Columnas

Para Columna:= 1 hasta 4 hacer

Para Fila:= 1 hasta 3 hacer

Leer(A[*columna*][*fila*]);

## EJEMPLO ARRAY BIDIMENSIONAL

Supóngase que se desea almacenar las calificaciones de 5 alumnos obtenidas en 3 exámenes. En este caso se usará un arreglo bidimensional (tabla o matriz) de 5 filas y 4 columnas en la cual se almacenará las calificaciones de 3 exámenes en 3 columnas y la cuarta columna se utilizará para almacenar su promedio respectivo, además de un arreglo unidimensional (vector) donde en forma paralela se almacenarán los nombres de los alumnos de la siguiente forma:

		Exam 1	Exam 2	Exam3	Promedio
Alumno[1]	<b>Fidel</b>	<b>10</b>	<b>10</b>	<b>10</b>	<b>10</b>
Alumno[2]	<b>Ely</b>	<b>20</b>	<b>20</b>	<b>20</b>	<b>20</b>
Alumno[3]	<b>Luis</b>	<b>15</b>	<b>15</b>	<b>15</b>	<b>15</b>
Alumno[4]	<b>Juan</b>	<b>05</b>	<b>05</b>	<b>05</b>	<b>05</b>

```
Matriz_Vector;
{El siguiente programa captura las calificaciones de 5 alumnos en 3
exámenes, y despliega en pantalla los promedios }
Constante
  MaxAlumno = 5;
  MaxExamen = 4; {Columna 4 almacena el promedio}
Var
  Alumno      :arreglo[1..MaxAlumno]      de cadena;
  examen      :arreglo[1..MaxAlumno][1..MaxExamen] de real;
  aux_examen  :arreglo[1..MaxExamen]      de real;
{reserva 20 posiciones de memoria de datos reales :
5 filas por 4 columnas}
  promedio   :real;
  aux_alumno  :cadena;
  i,j       :entero;

Inicio
  {lectura de arrays paralelos de manera simultánea}
  Write('Nombre');
  Write('Examen1 Examen2 Examen3 Promedio');
  Para i:=1 hasta MaxAlumno hacer
    Inicio
      leer(alumno[i]); {lectura de vector}
      promedio:=0;
      Para j:=1 hasta MaxExamen-1 hacer
        Inicio
          leer(examen[i][j]); {lectura de matriz}
          promedio:=promedio+examen[i,j];
        fin;
      examen[i,j+1]:=promedio/3;
    Fin;
```

Calcule lo siguiente:

- 1.- El promedio por cada asignatura.
- 2.- La mayor nota en cada asignatura. Debe mostrar el nombre del alumno.
- 3.- El alumno de mayor promedio.

### MANEJO DE MATRICES

Diagonal Principal ( $i = j$ )

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

Para  $i := 1$  hasta  $N$  hacer  
     $A[i,i] := 0;$

### MANEJO DE MATRICES

Diagonal Secundaria ( $i+j = N+1$ )

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

Para  $i := 1$  hasta  $N$  hacer  
    Inicio  
         $j := N - i + 1;$   
         $A[i,i] := 0;$   
    Fin;

Para  $i := 1$  hasta  $N$  hacer  
    Si  $(i+j = N+1)$  ent  
         $A[i,i] := 0;$

### MANEJO DE MATRICES

Diagonal Principal Superior ( $i < j$ )

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

Para  $r i := 1$  hasta  $N-1$  hacer  
    Para  $j := i+1$  hasta  $N$  hacer  
         $A1[i,j] := 0;$

Para  $i := 1$  hasta  $N$  hacer  
    Para  $j := 1$  hasta  $N$  hacer  
        Si  $(i < j)$  ent  
             $A1[i,j] := 0;$

### MANEJO DE MATRICES

Diagonal Secundaria Superior ( $i+j < N+1$ )

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

Para  $i := 1$  hasta  $N-1$  hacer  
    Para  $j := 1$  hasta  $N-i$  hacer  
         $A1[i,j] := 0;$

Para  $i := 1$  hasta  $N$  hacer  
    Para  $j := 1$  hasta  $N$  hacer  
        Si  $(i+j < N+1)$  ent  
             $A1[i,j] := 0;$

**MANEJO DE MATRICES**

Diagonal Principal Inferior ( $i > j$ )

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

Para i:= 2 hasta N hacer  
 Para j:=1 hasta i-1 hacer  
 A1[i,j]:= 0;

Para i:= 1 hasta Dim hacer  
 Para j:= 1 hasta Dim hacer  
 Si (i>j) ent  
 A1[i,j]:= 0;

**MANEJO DE MATRICES**

Diagonal Secundaria Inferior ( $i+j > N+1$ )

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

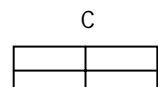
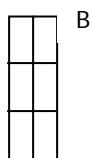
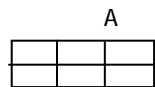
Para i:= 2 hasta N hacer  
 Para j:= N-i+2 hasta N hacer  
 A1[i,j]:= 0;

Para i:= 1 hasta N hacer  
 Para j:= 1 hasta N hacer  
 Si (i+j > N +1) ent  
 A1[i,j]:= 0;

**MANEJO DE MATRICES**

Producto de Matrices

Para i:= 1 hasta N hacer  
 Para j:= 1 hasta N hacer  
 inicio  
 prod[i,j] := 0;  
 Para k:= 1 hasta N hacer  
 prod[i,j]:= prod[i,j] + a[i,k]\*b[k,j];  
 fin;



FA=FC ;CB=CC

For i:= 1 to FA  
 For j:= 1 to CB  
 .....  
 For k:= 1 to CA

**MATRIZ SIMÉTRICA**

Una matriz A se dice que es simétrica si  
 $A(i,j) = A(j,i)$  para todo i,j dentro  
 de los límites de las matriz. (MXM)

**MATRIZ ASIMÉTRICA**

$$A(i,j) = - A(j,i)$$

**MATRIZ TRASPUESTA**

Una matriz M se dice que es traspuesta (MT) si  
 $M(i,j) = MT(j,i)$  para todo i,j dentro  
 de los límites de las matriz. (MXN)