

# **ESTRUCTURA DE DATOS: ARREGLOS**

## **1. Introduccion**

## **2. Arreglos**

- Concepto
- Caracteristicas

## **3. Arreglos Unidimensionales**

## **4. Arreglos Bidimensionales**

## **5. Ventajas del uso de arreglos**

## **6. Ejemplo**

# ESTRUCTURA DE DATOS: Arreglos

## 1. Introducción (I)

Cuando hablamos de *Estructuras de Datos* hacemos referencia a un conjunto de datos que poseen el mismo nombre, que pueden ser caracterizados por su organización y por las operaciones que se definen en ella.

### DATOS SIMPLES

Estandar --> Entero – Real – Caracteres - Logico

### ESTRUCTURA DE DATOS

**Estatica --> Vectores y Matrices, Registro, Archivos**

Dinamica --> Lineales: Listas, Pilas Colas  
--> No lineales: Arboles y Grafos

# ESTRUCTURA DE DATOS: Arreglos

## 1. Introducción (II)

Las **estructuras de datos estáticas** son aquellas en las que el tamaño ocupado en memoria se define antes que el programa se ejecute y el mismo no puede ser modificado durante la ejecución.

Los tipos de datos que vimos hasta ahora son *datos simples* cuya característica común es que cada variable representa a un elemento; en cambio los tipos de *datos estructurados* tienen como particularidad que con un nombre o identificador se puede representar múltiples datos individuales y a su vez cada uno de estos puede ser referenciado independientemente.

# ESTRUCTURA DE DATOS: Arreglos

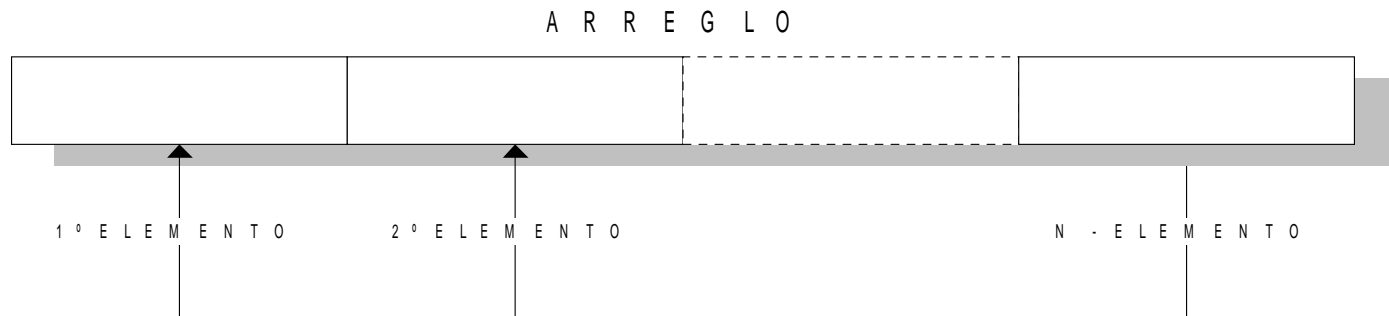
## 2. Arreglos: Concepto

*Arreglos se define como una colección finita, homogénea y ordenada de elementos.*

Finita: Todo arreglo tiene un límite, es decir, debe determinarse cual será el número máximo de elementos que podrán formar parte del arreglo.

Homogénea: Todos los elementos de un arreglo son del mismo tipo o naturaleza (todos enteros, todos booleanos, etc.- ), pero nunca una combinación de distintos tipos.

Ordenada: Se debe determinar cual es el primer elemento, el segundo, el tercero..... y el enésimo elemento.



# ESTRUCTURA DE DATOS: Arreglos

## 2. Arreglos: Características

Si un arreglo tiene la característica de que puede almacenar a N elementos del mismo tipo, deberá tener la posibilidad de permitir seleccionar a cada uno de ellos. Así se distinguen dos partes en los arreglos.

Los **componentes o elementos** (valores que se almacenan en c/u de las casillas)

Los **índices** (Permiten hacer referencia a los componentes)

El número total de componentes (NTC) es igual al límite superior (LS) menos límite inferior (LI) mas 1

$$NTC = LS - LI + 1$$

El tipo de índice puede ser cualquier tipo ordinal (carácter, entero, enumerado)

El tipo de los componentes puede ser cualquiera (entero, real, cadena de caracteres, registro, etc.)

Se utilizan ( ) para indicar el índice de un arreglo. Entre los ( ) se debe escribir un valor ordinal (puede ser una variable, una constante o una expresión que dé como resultado un valor ordinal)

# ESTRUCTURA DE DATOS: Arreglos

## 2. Arreglos: Características

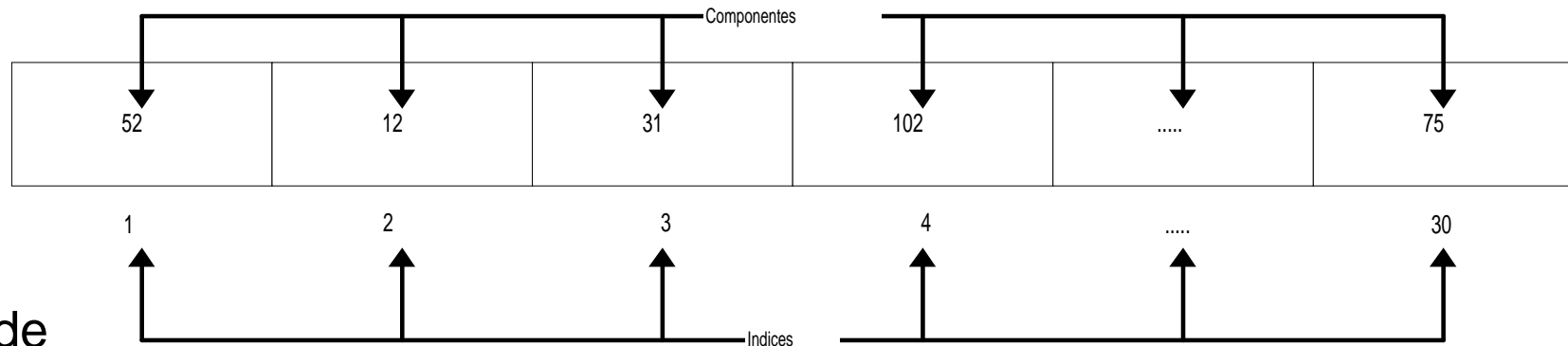
Ejemplo

Sea **V** un arreglo de 30 elementos enteros con índices enteros.

$V = (52, 12, 31, 102, \dots, 75)$

$V(50) = v(1), v(2), v(3), \dots, v(30),$

Su representación grafica será



Donde

$$NTC = (30 - 1 + 1) = 30$$

Cada componente del arreglo **V** será un número entero, y podrá accederse por medio de un índice que será un valor comprendido entre 1 y 30.

De esta manera, por ejemplo:

$V(1)$  hace referencia al elemento de la posición 1 cuyo valor es 52

$V(2)$  hace referencia al elemento de la posición 2 cuyo valor es 12

# ESTRUCTURA DE DATOS: Arreglos

## 2. Arreglos: Características

En cuanto a las **dimensiones** los arreglos pueden ser:

Unidimensional o vector: un solo índice

Bidimensional o matriz: dos índices

Multidimensional: mas de dos índices

### **Diferencia con registros**

Las dos diferencias sustanciales entre arreglos y registro son:

1) Un arreglo puede almacenar N elementos del mismo tipo, mientras que un registro puede almacenar N elementos de distintos tipos que se llaman campos.

2) Los componentes de un arreglo se acceden por medio de índices, mientras que en un registro los campos se acceden por medio de su nombre, el cual es único.

# ESTRUCTURA DE DATOS: Arreglos

## 3. Arreglos Unidimensionales: Vectores

### Operaciones

Podemos clasificar a las operaciones en las que intervienen arreglos de la siguiente manera:

- Lectura / escritura

- Recorrido

- Asignación

- Actualización (Añadir, eliminar, insertar)

- Ordenación

- Búsqueda

Como los arreglos son datos estructurados, muchas de estas operaciones no pueden llevarse a cabo de manera global, sino que se debe trabajar sobre cada componente. Seguidamente se analizará cada una de estas operaciones, excepto las dos últimas (ordenación y búsqueda) que serán estudiadas en capítulos posteriores.



# ESTRUCTURA DE DATOS: Arreglos

## 3. Arreglos Unidimensionales: Vectores

### Operaciones: Lectura / escritura

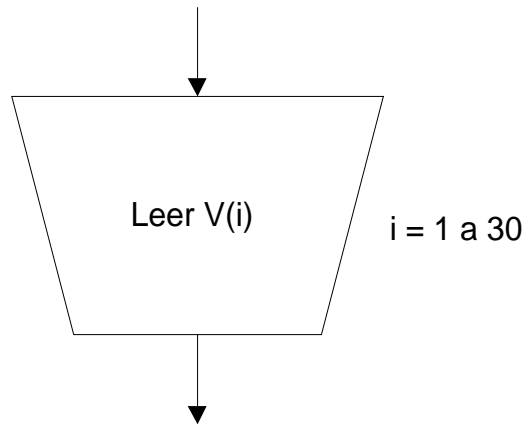
El proceso de lectura /escritura de un arreglo se realiza de la siguiente manera:

Leer  $V(i)$       *Lee todo el arreglo*

Escribir  $V(i)$    *Escribe todo el arreglo*

Leer  $V(3)$       *Lee el elemento 3 del arreglo*

La notacion grafica correspondiente



# ESTRUCTURA DE DATOS: Arreglos

## 3. Arreglos Unidimensionales: Vectores

### Operaciones: Recorrido

Recorrer un vector significa acceder a todos y a cada uno de sus elementos desde el principio hasta el final o viceversa.

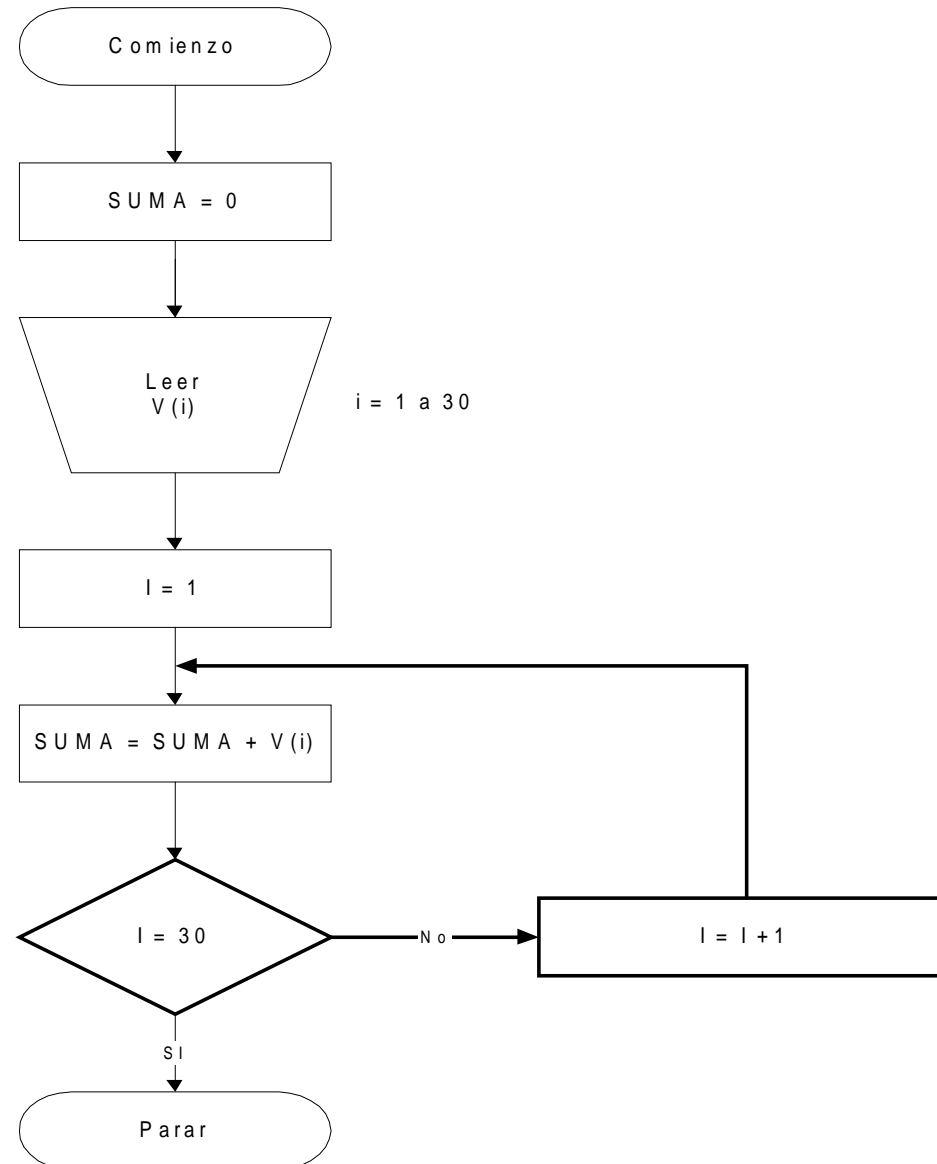
Se puede acceder a los elementos de un vector para introducir datos (leer) en él o bien para ver su contenido (escribir). A la operación de acceder a todos los elementos para efectuar una acción determinada se denomina *recorrido* del vector. Esta operación se realiza usando estructuras repetitivas, cuya variable de control  $i$ , se utiliza como subíndice del vector (por ejemplo  $V(i)$ ). El incremento del contador del bucle producirá el tratamiento sucesivo de los elementos del vector.

Esta operación es muy utilizada en este tipo de estructuras de datos, dado que cuando se está en presencia de un vector, el acceso a toda la información se realiza recorriéndolo. En algunos casos se puede acceder a un determinado elemento o a varios de ellos con ciertas características sin necesidad de recorrer todo el arreglo, por ejemplo acceder solo al último elemento que sabemos a priori posee la suma de los elementos anteriores.

# ESTRUCTURA DE DATOS: Arreglos

## 3. Arreglos Unidimensionales: Vectores

### Operaciones: Recorrido



# ESTRUCTURA DE DATOS: Arreglos

## 3. Arreglos Unidimensionales: Vectores

### Operaciones: Actualización

Muchas veces resulta interesante que dado un arreglo, puedan añadirse nuevos elementos o eliminar o insertar componentes. Estas resultan las tres operaciones elementales que se pueden realizar en un arreglo: *añadir*, *eliminar* e *insertar* elementos.

Cuando se realiza una operación de ***añadir*** un nuevo elemento a continuación del último valor no nulo, la única condición necesaria para esta operación es comprobar que haya espacio para el nuevo elemento.

# ESTRUCTURA DE DATOS: Arreglos

## 3. Arreglos Unidimensionales: Vectores

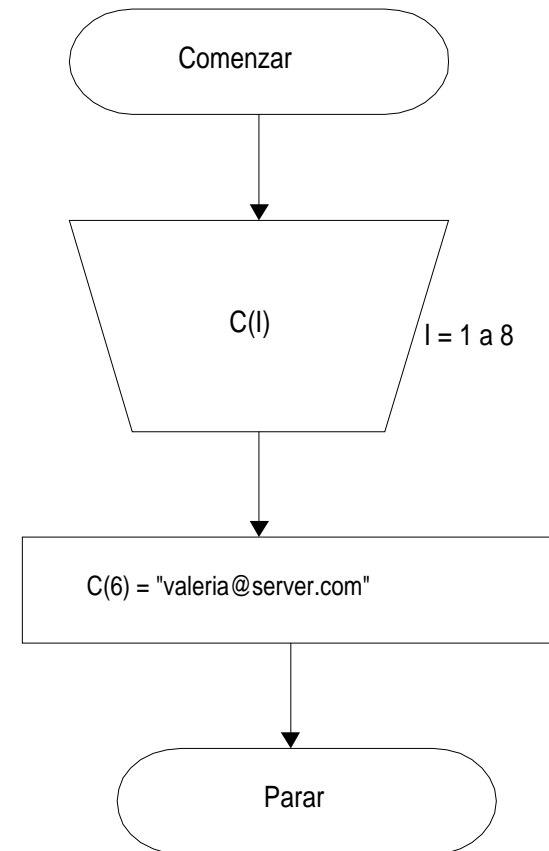
### Operaciones: Actualizacion

#### Ejemplo

Dado un vector C de 8 elementos que contiene una nómina de 5 direcciones de correo ordenadas alfabéticamente. Se desea añadir la dirección [valeria@server.com](mailto:valeria@server.com)

1	alicia@server.com
2	castor@server.com
3	daniel@server.com
4	marta@server.com
5	sonia@server.com
6	
7	
8	

1	alicia@server.com
2	castor@server.com
3	daniel@server.com
4	marta@server.com
5	sonia@server.com
6	valeria@server.com
7	
8	



# ESTRUCTURA DE DATOS: Arreglos

## 3. Arreglos Unidimensionales: Vectores

### Operaciones: Actualizacion

La operación de ***eliminar*** un elemento al final del arreglo no presenta ningún problema; en cambio, si el borrado se realiza en el interior del mismo esto provoca el efecto contrario al de *insertar*, el movimiento deberá ser hacia arriba (l-1) de los elementos inferiores a él para reorganizar el vector.

# ESTRUCTURA DE DATOS: Arreglos

## 4. Arreglos Bidimensionales: Matrices

Un arreglo de dos dimensiones, también denominada matriz, se define como una tabla de tablas, o vector de vectores, es decir, es aquella en la cual uno de sus elementos es, a su vez, una tabla unidimensional.

Podemos comparar una matriz con una hoja de papel cuadriculado en la que cada cuadrícula corresponderá a un elemento.

En este gráfico podemos observar que cada fila está dividida en varias columnas. Por lo tanto, para poder referenciar un elemento de la matriz, hay que especificar el nombre de la misma (igual que con los vectores) y, entre paréntesis, dos subíndices separados por coma; el primero indicará la fila en la que se encuentra el elemento y el segundo la columna.

	Columna 1	Columna 2	Columna 3	Columna 4
Fila 1				
Fila 2			12	
Fila 3				

# ESTRUCTURA DE DATOS: Arreglos

## 4. Arreglos Bidimensionales: Matrices

El caso anterior, representado en forma matricial sería

Donde la matriz llamada MAT tiene filas que varían de 1 a 3 y columnas que varían de 1 a 4, por lo tanto diremos que la matriz MAT tiene 3 x 4 elementos.

$I = 1 \dots 3$

$J = 1 \dots 4$

Si generalizamos el rango, resultaría:

$I = 1 \dots M$

$J = 1 \dots N$

	1	2	3	4
1				
2			MAT(I,J)	
3				

Y diremos que la matriz MAT tiene  $M \times N$  elementos. Existen  $N$  elementos en cada fila y  $M$  elementos en cada columna.

El resultado de multiplicar la cantidad de filas por cantidad de columnas es el **tamaño** de la matriz. En nuestro ejemplo anterior el tamaño es de 12 ( $3 \times 4$ ).



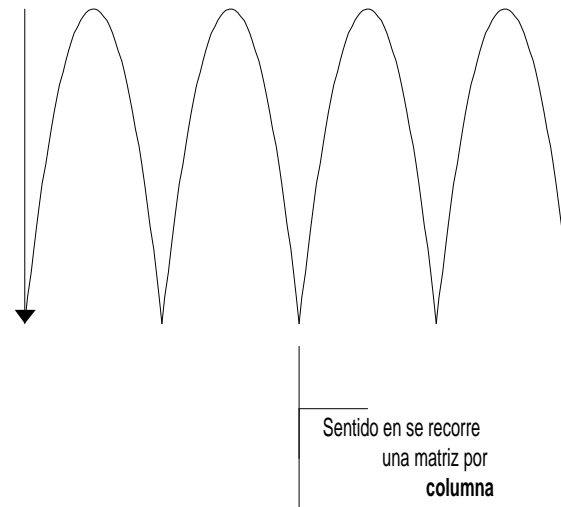
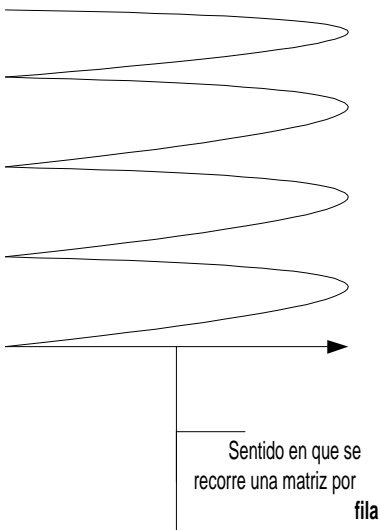
# ESTRUCTURA DE DATOS: Arreglos

## 4. Arreglos Bidimensionales: Matrices

### Recorrido de una matriz

Como vimos anteriormente, recorrer una tabla de dos dimensiones significa acceder a todos y a cada uno de sus elementos. Este proceso de recorrer la matriz se llevará a cabo mediante la estructura repetitiva anidada.

El recorrido de los elementos de la matriz se puede realizar *por fila* o *por columna*. Para recorrer por fila la matriz MAT se debe realizar dos estructuras repetitivas anidadas. En la primera de ellas (la mas externa) se realizan tres iteraciones para recorrer las 3 filas. En cada una de esas iteraciones, se realizará a su vez 4 iteraciones para recorrer los 4 elementos de cada fila (uno por cada columna).



# ESTRUCTURA DE DATOS: Arreglos

## 4. Arreglos Bidimensionales: Matrices

### Recorrido de una matriz

#### Ejemplo

Supongamos que tenemos una matriz que contiene de los doce meses del año las 4 temperaturas máximas de cada mes  $T(12,4)$  y se desea imprimir los datos.

	Temperatura Máxima 1	Temperatura Máxima 2	Temperatura Máxima 3	Temperatura Máxima 4
01 enero	30	31	33	30
02 febrero	29	31	30	30
03 marzo	22	24	24	23
04 abril	25	23	24	24
.....				
12 diciembre	28	26	29	30

Si deseamos imprimir los datos por mes (fila de la matriz) debemos recorrer la misma por fila de forma tal que por cada fila debemos recorrer las 4 columnas de la misma. Pero como la matriz tiene 12 filas, este proceso se repite 12 veces – uno por cada fila – y de esta manera formamos dos ciclos anidados. Uno mas externo – fila - que se repite 12 veces y uno mas interno – columna – que por cada fila se repite 4 veces.

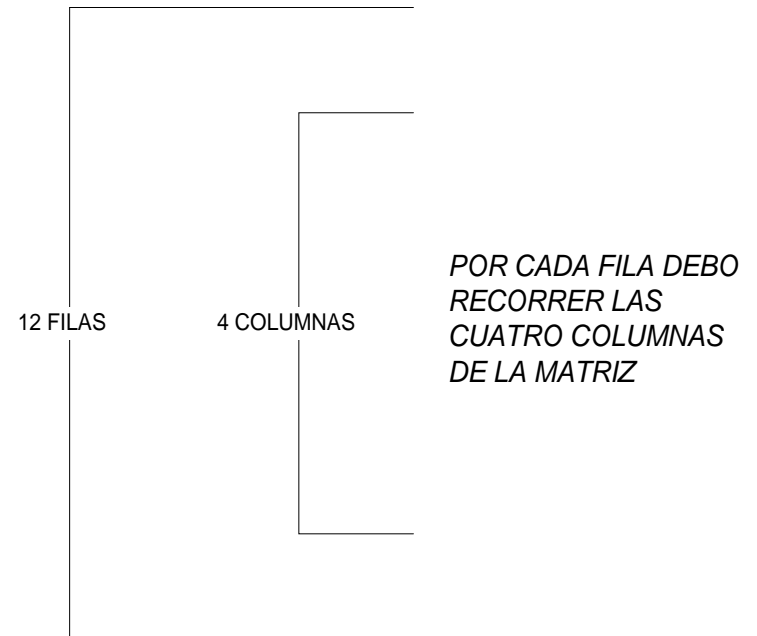
# ESTRUCTURA DE DATOS: Arreglos

## 4. Arreglos Bidimensionales: Matrices

### Recorrido de una matriz

Entonces, recorrer esta matriz para imprimirla consistirá en:

- Posicionarse en la primer fila ( $I=1$ ) y recorrer todas sus columnas (desde  $J=1$  hasta  $J=4$ ).
- Posicionarse en la segunda fila ( $I=2$ ) y volver a recorrer, nuevamente, todas sus columnas (desde  $J=1$  hasta  $J=4$ ).
- Repetir estas operaciones para cada valor de  $I$  hasta que se hayan realizado para la última fila, es decir para  $I=12$ .



# ESTRUCTURA DE DATOS: Arreglos

## 4. Arreglos Bidimensionales: Matrices

### Recorrido de una matriz

El Pseudocódigo y diagrama de flujo correspondientes para recorrer e imprimir la matriz  $T(12,4)$  sería:

Comenzar

Para  $I = 1$  a 12

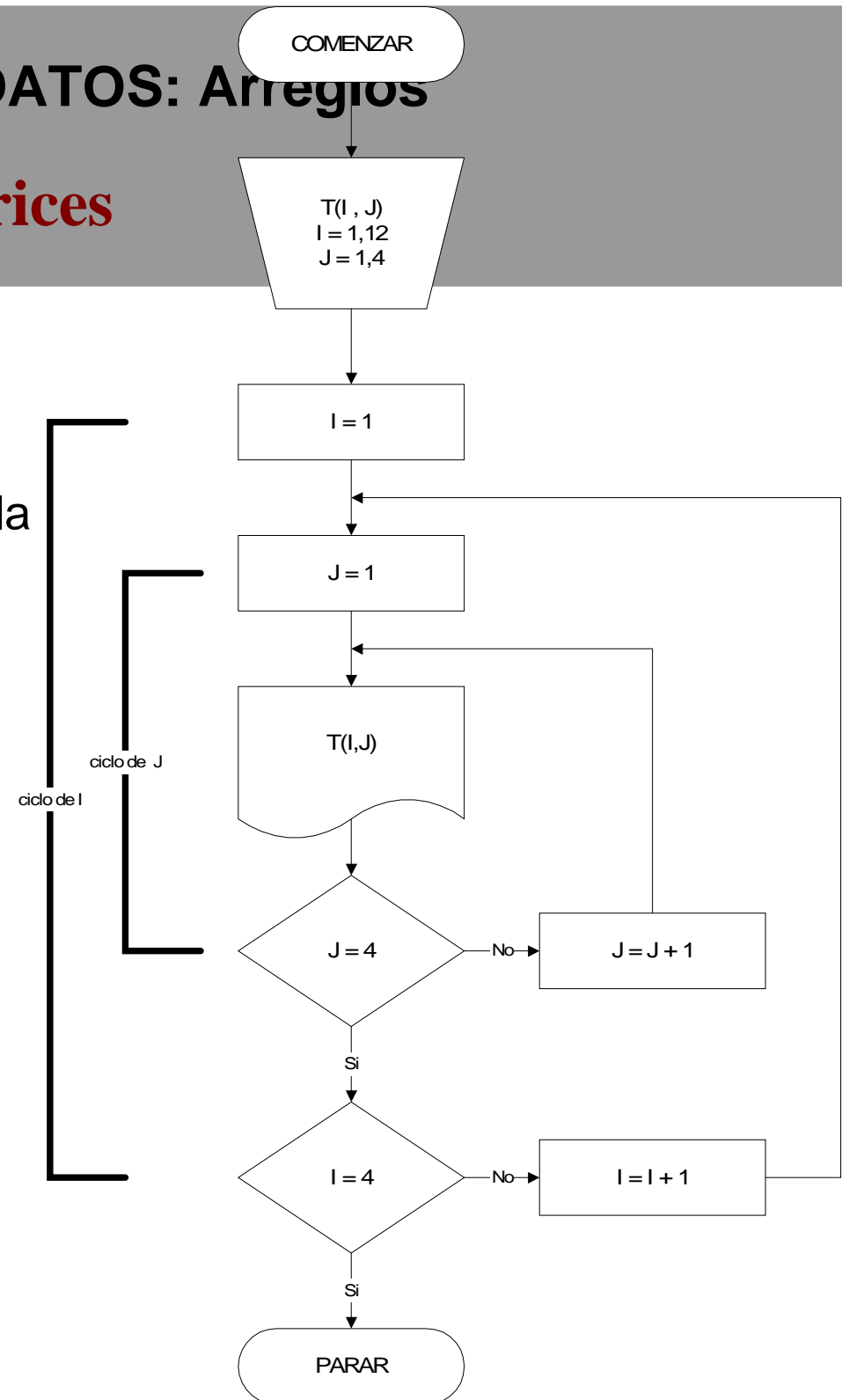
Para  $J = 1$  a 4

**Imprimir  $T(I,J)$**

Fin\_para

Fin\_para

Parar.



# ESTRUCTURA DE DATOS: Arreglos

## 5. Ventajas del uso de arreglos

Posibilita el uso de menor cantidad de nombres de variables, dado que con un solo nombre genérico se almacenan  $N$  datos, mientras que con el uso de datos simples necesitaríamos  $N$  nombres de variables distintas.

Menor tamaño del algoritmo cuando se utiliza gran cantidad de variables al mismo tiempo en memoria.

A partir de un archivo desordenado, los arreglos permiten generar los mismos datos con un cierto orden.

Se tiene todo el conjunto de datos en memoria sin necesidad de acceder al dispositivo.

La facilidad de acceso a la información, ya sea a un elemento en particular o a un grupo con ciertas condiciones; y por otro lado los distintos sentidos para el recorrido del mismo, pueden ser desde el inicio al fin o viceversa.

Los arreglos permiten el uso de parámetros en la definición del tamaño, con lo cual se logra mayor flexibilidad en los algoritmos

```
program Ejemplo 1
    {Ingresa un parametro M y un vector.
    Informa la suma de sus elementos positivos e imprime el
    vector}
```

```
uses Crt ;
```

```
var i, j, suma_positivos, m : integer;
    vector : array [1..100] of real;
    tecla : char;
```

```
begin
    clrscr;
    writeln('Ingresa cantidad de elementosdel vector < 100):');
    readln(m);
```

```
{carga el vector por pantalla}
```

```
    for i:=1 to m do
        begin
            write(i,'i) '); readln(vector[i])
        end;
```

```
{recorre el vector}
```

```
    for I := 1 to m do
        begin
            if vector[i] > 0 then
                suma_positivos:=suma_positivos+1
        end;
```

```
{Imprime el vector y los valores hallados}
```

```
writeln('Lista el Vector:');
```

```
for i:=1 to m do
    write (vector[i]:6:2);
```

```
writeln('');
```

```
writeln('Cantidad de positivos: ',suma_positivos:6);
```

```
tecla:=readkey
```

```
end.
```

```
program Ejemplo 2
  {Ingresar una matriz A(10,8),
  calcular e informar la suma de sus elementos}
```

```
Uses WinCrt;
```

```
var
  i,j,suma:integer;
  A:array[1..3,1..2]of integer;
```

```
begin
  clrscr;

  writeln('Ingresa los elementos de la matriz (números enteros)');

  {carga la matriz y suma los elementos}
  for i:= 1 to 3 do
    begin
      for j:= 1 to 2 do
        begin
          write('Ingresa el elemento de la posición ',i,', ',j,', ': ');
          readln(A[i,j]);
          suma:=suma+A[i,j];
        end;
      end;

  Writeln('La suma de los elementos de la matriz es: ',suma);
end.
```