FUNDAMENTOS DE PROGRAMACIÓN

¿QUÉ ES ALGORITMO?

La palabra algoritmo se deriva de la traducción al latín de la palabra árabe alkhowarizmi, nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX.

Es un método para resolver un problema mediante una serie de pasos definidos, precisos y finitos.

TIPOS DE ALGORITMOS

Existen dos tipos y son llamados así por su naturaleza:

Cualitativos: Son aquellos en los que se describen los pasos utilizando palabras.

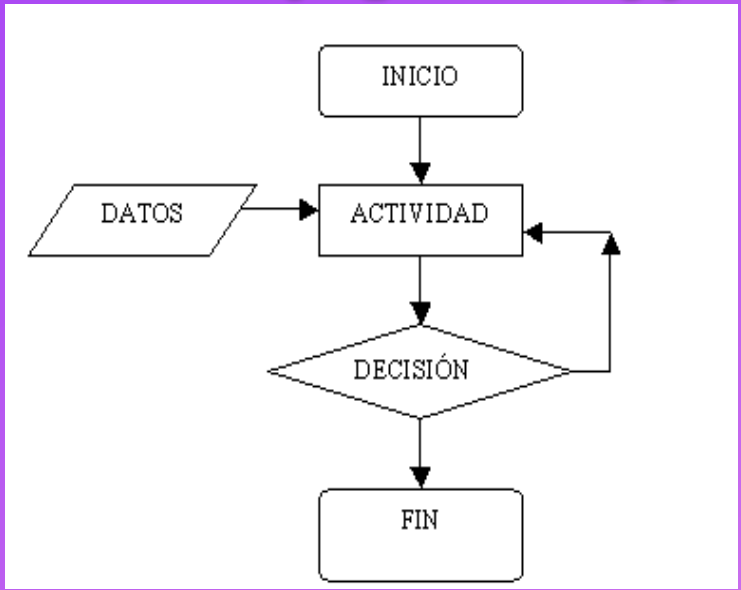
Cuantitativos: Son aquellos en los que se utilizan cálculos numéricos para definir los pasos del proceso.

LENGUAJES ALGORITMICOS

Un Lenguaje algorítmico es una serie de símbolos y reglas que se utilizan para describir de manera explícita un proceso.

TIPOS DE LENGUAJE

Gráficos: *(diagrama de flujo)*



No Gráficos: *(pseudocodigo).*

```
1   Subrutina principal()
2   Inicio
3   leer(decimal)
4   leer(entero)
5   Si ( base>1 && base<11 ) entonces
6     residuo = 0
7     resultado = 0
8     dividendo = decimal
9     final = 0
10    multi = 1
11    mientrasque (dividendo >= base) haga
12       residuo= dividendo MOD base
13       dividendo = dividendo / base – residuo / base
14       final = final + residuo * multi
15       multi = multi * 10
16          FMQ
17    Final = final + dividendo * multi
18    Escribir("El          numero          "+decimal+"en
19    base"+base+"es: "+final)
20  Sino
21          Escribir("Error")
22  FinSi
23  Fin_subrutina
```

Linea 3: decimal=5
Linea 4: base=2
Linea 5: 2>1 && 2<11
Linea 6: residuo=0
Linea 7: resultado=0
Linea 8: dividendo=5
Linea 9: final=0
Linea 10: multi=1
Linea 11: 5>=2
Linea 12: residuo=1
Linea 13: dividendo=2.5-0.5=2
Linea 14: final=1
Linea 15: multi=10
Linea 11: 2>=2
Linea 12: residuo=0
Linea 13: dividendo=1
Linea 14: final=1
Linea 15: multi=100
Linea 11: 1>=2
Linea 17: Final=1+1*100=101

Linea18: "El número 5 en base 2 es 101"

DIAGRAMA DE FLUJO

Un diagrama de flujo es la representación gráfica de las operaciones que realiza un algoritmo.

Los diagramas de flujo deben dibujarse de arriba hacia abajo y leerse de derecha a izquierda.

Las principales aplicaciones de los diagramas de flujo son:

- Auxiliar en el desarrollo de un programa.
- Mostrar gráficamente la secuencia lógica de un programa.
- Proporcionar la documentación del programa.
- Utilizarse como una hoja de trabajo de referencia para la codificación de un programa

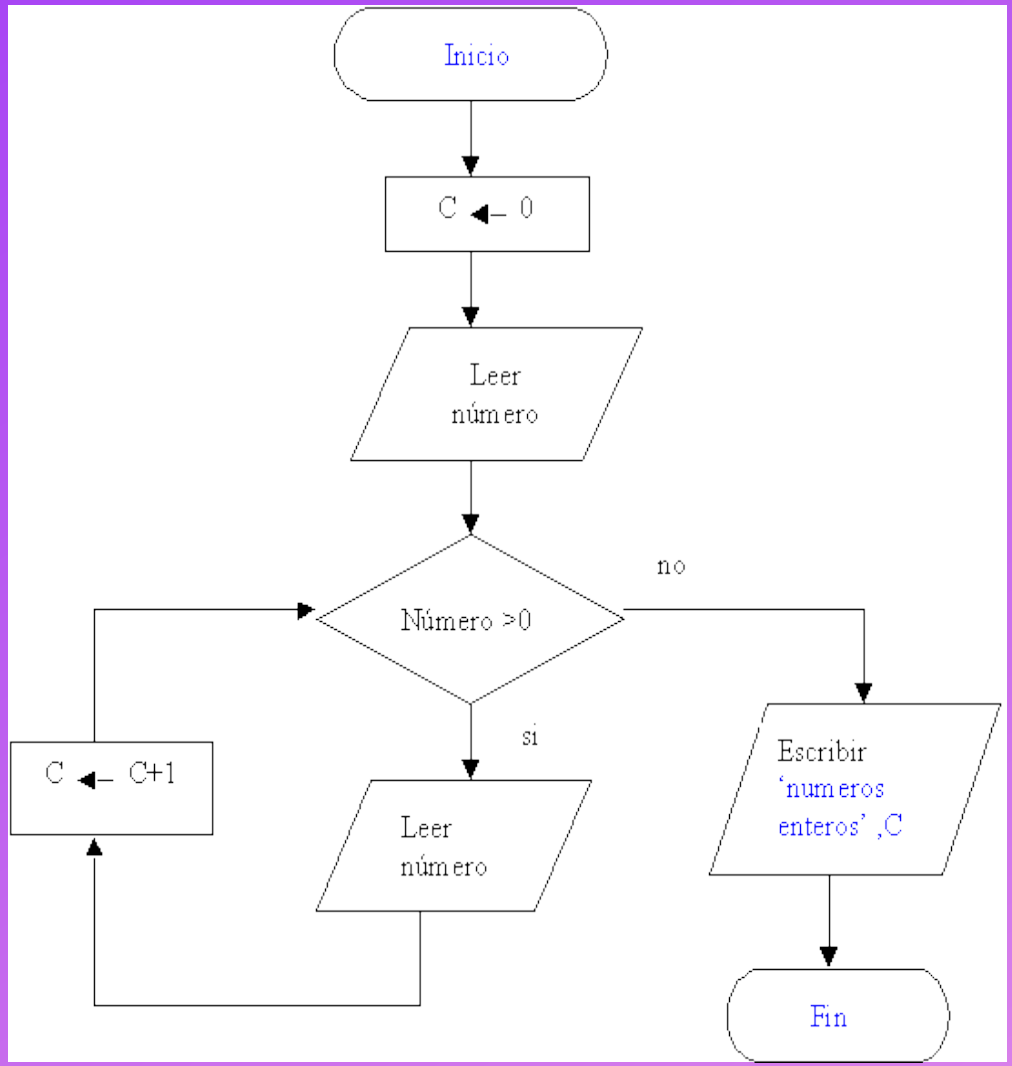
SIMBOLOS DE FLUJO

Para desarrollar adecuadamente un diagrama de flujo, hay que familiarizarse con un serie de símbolos, cada uno de los cuales representa una acción especifica. Estos son algunos:

SÍMBOLO	NOMBRE	FUNCIÓN
	INICIO O TERMINAL	Señala donde si inicio termina un diagrama, puede representar también una parada o interrupción programada que sea necesario realizar en un diagrama.
	DESICION	Indican cursos de acción alternos como resultado de una decisión de si o no.
	PROCESO	Cualquier tipo de operación que puede originar cambio de valor, formato o posición de la información almacenada.
	CONECTOR	Denota una salida hacia o una entrada de otra parte del diagrama de flujo o de otro diagrama de flujo que lo une al primero.
	ENTRADA O SALIDA	Representa funcione I/O, se utiliza para indicar información que ingresa o sale del diagrama de flujo
	LINEA DE FLUJO	Las lineas de flujo indican la secuencia en que se llevara a cabo las distintas acciones descritas en el diagrama de flujo.
	CONECTOR DE PAGINA	Indica la fuente o el destinode renglones que ingresan o salen del diagrama de flijo.

EJEMPLO:

Contar los números enteros positivos introducidos por teclado. Se consideran dos variables enteras NUMERO y CONTADOR (contará el número de enteros positivos). Se supone que se leen números positivos y se detiene el bucle cuando se lee un número negativo o cero.



¿QUE ES PSEUDOCODIGO?

El Pseudocódigo es una notación algorítmica textual, que a pesar de su formalismo, puede llegar a ser muy parecido al lenguaje natural.

Se caracteriza por permitir la declaración de las variables y constantes que intervienen en el algoritmo. Para ello el pseudocódigo dispone de un conjunto de palabras reservadas que expresan tanto las acciones elementales como las diferentes estructuras lógicas del algoritmo.

```
ALGORITMO nombre_algoritmo;
CONSTANTES
/*Sección reservada para la declaración de datos constantes utilizados por el algoritmo*/
…
VARIABLES
/*Sección reservada para la declaración de datos variables utilizados por el algoritmo*/
…
INICIO /*Comienza el cuerpo principal de acciones del algoritmo*/
                ACCION_1;
                ACCION_2;
                ACCION_3;
                …
                ACCION_N;
FIN
```

En estas estructuras, las palabra que aparecen en negrita, son el conjunto de palabras reservadas. Un ejemplo de algoritmo descrito por pseudocódigo seria este:

También se pueden observar unas frases escritas entre símbolos **/*** ***/.** Estas frases son comentarios que a efectos del funcionamiento del algoritmo no tiene ninguna utilidad; se entiende que el algoritmo las ignora. Su única función es la de incorporar aclaraciones al código del algoritmo que sirvan para su comprensión.

Ejemplo:

Programa que calcula el área de un cuadrado a partir de un lado dado por teclado.

```
1 area_cuadrado
2 main /* también se puede llamar principal*/
3 /*VARIABLES*/
4 lado, área;
5 INICIO
6  Vizualizar;  «Introduce el lado del cuadrado»
7  Leer lado;
8  Area – lado x lado;
9  Vizualizar  «El área del cuadrado es», area;
10 FIN
```

- La numeración en la parte izquierda nos ayuda a verificar el numero de líneas del pseudocódigo.
- Cuando lo estemos simulando en un programa nos ayuda también a buscar fallas mas fácilmente por líneas.
- Los pseudocódigos son TOP DOWN, se escriben y se leen de arriba hacia abajo.

PROGRAMACION

En informática, la programación es un proceso por el cual se escribe (en un lenguaje de programación), se prueba, se depura y se mantiene el código fuente de un programa informático.

Un programa normalmente implementa (traduce a un lenguaje de programación concreto) un algoritmo.

Los programas suelen subdividirse en partes menores (módulos), de modo que la complejidad algorítmica de cada una de las partes sea menor que la del programa completo, lo cual ayuda al desarrollo del programa.

Se han propuesto diversas técnicas de programación cuyo objetivo es mejorar tanto el proceso de creación de software como su mantenimiento. Entre ellas, se pueden mencionar las siguientes:

- Programación estructurada
- Programación modular
- Programación orientada a objetos (POO)
- Programación declarativa

¿QUE ES ESTRUCTURA DE CONTROL?

En programación, una estructura de control permite controlar el flujo de la ejecución de instrucciones. Con estas estructuras, el programador puede determinar el orden en que se ejecutarán las instrucciones que están dentro de estas estructuras.

Los lenguajes de programación modernos tienen estructuras de control similares. Básicamente lo que varía entre las estructuras de control de los diferentes lenguajes es su sintaxis, cada lenguaje tiene una sintaxis propia para expresar la estructura.

Las estructuras de control se pueden agrupar en dos clase:

- Estructuras de selección, también denominadas condicionales.
- Estructuras de repetición , también conocidas como iterativas o bucles.

ESTRUCTURAS DE CONTROL SELECTIVA

En una estructura de selección/decisión, el algoritmo al ser ejecutado toma una decisión, ejecutar o no ciertas instrucciones si se cumplen o no ciertas condiciones. Las condiciones devuelven un valor, verdadero o falso, determinado así la secuencia a seguir.

Por lo general los lenguajes de programación disponen de dos estructuras de este tipo: estructura de decisión simple (if), y estructura de decisión múltiple (SWITCH)

- IF: significa SI (condicional) en español. Su funcionamiento es simple. Se evalúa una condición, si es verdadera ejecuta un código, si es falsa, ejecuta otro código (o continúa con la ejecución del programa).

```
if (condición)
ejecuta esto si la condición es verdadera
else
ejecuta esto si la condición es falsa
```

IF en pseudocódigo:

- Switch: permite elegir ejecutar diferentes códigos (conjunto de sentencias de programación) dependiendo de un valor.

```
switch (valor) {
case „1": ejecuta esto si valor es 1
break;
case „2": ejecuta esto si valor es 2
break;
case „3": ejecuta esto si valor es 3
break;
default: ejecuta esto si valor no es ninguno de los anteriores
break;
};
```

La forma general es:

```
letra='e';
switch(letra);
{
        case 'a':
        case 'A':
                printf(``Es la vocal a\n'');
                break;
        case 'e':
        case 'E':
                printf(``Es la vocal e\n'');
                break;
        case 'i':
        case 'I':
                printf(``Es la vocal i\n'');
                break;
        case 'o':
        case 'O':
                printf(``Es la vocal o\n'');
                break;
        case 'u':
        case 'U':
                printf(``Es la vocal u\n'');
                break;
        default: printf(``Es una consonante\n'');
}
```

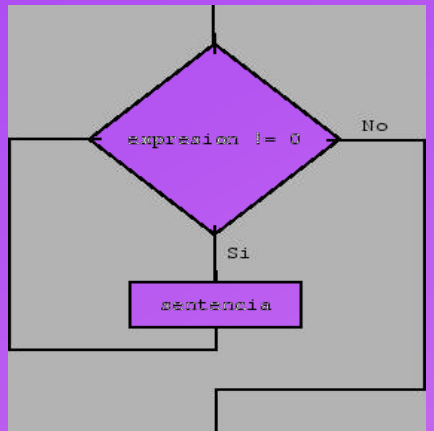
Un ejemplo de uso de esta sentencia es el siguiente fragmento de programa, que decide si imprime la vocal dada:

ESTRUCTURA DE CONTROL REPETITIVA

Ejecuta cero o más veces un grupo de instrucciones (bucle). El número de repeticiones está determinado por un número dado, o hasta que deje de cumplirse o se cumpla una condición.

Las estructuras de repetición más usuales en los lenguajes de programación suelen ser WHILE y FOR.

- **While:** Permite al programador especificar las veces que se repita una acción (una o más sentencias de programación) mientras una condición se mantenga verdadera. La forma del while en lenguaje C es:



- La sentencia se ejecutará mientras el valor de expresión sea verdadero.
- Primero se evalúa expresión
- Lo normal es que la sentencia incluya algún elemento que altere el valor de expresión proporcionando así la condición de salida del bucle.
- Si la sentencia es compuesta se encierra entre { }.

Un ejemplo de uso de esta sentencia es el siguiente fragmento de programa, que calcula la suma de los números del 1 al 100:

```
int suma, limite;

suma=1; limite=100;
while(limite>0)
{
    suma=suma+limite;
    limite--;
}
```

- For: Una de las estructuras de repetición empleada en la programación, de algoritmos para repetir un código (una o más sentencias de programación) dependiendo de un contador.

```
int contador;

for (contador =
1; contador<=10; contador++)
{
printf("Repetición numero %d
", contador);
};
```

Primero se crea la variable contador de tipo entero (será la variable de control en el for). Luego se ejecuta la estructura for iniciando la variable contador en 1. Luego se verifica que se cumple la condición contador<=10 y se ejecuta el bloque dentro de la estructura, o sea, imprime en pantalla Repetición número 1.

Luego la variable contador es incrementada en uno con la expresión contador++ y el ciclo se inicia otra vez. La variable contador ahora vale 2, por lo tanto se verifica la condición y se vuelve a ejecutar el código. Este proceso se ejecuta hasta que contador toma el número 11 y la condición se hace falsa y no ejecuta el bloque.

ARREGLOS (ARRAYS)

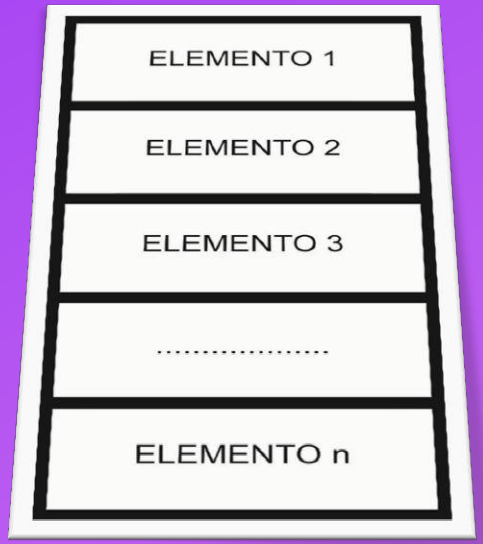
Un arreglo (array) es una colección de datos del mismo tipo, que se almacenan en posiciones consecutivas de memoria y reciben un nombre común. Para referirse a un determinado elemento de un array se deberá utilizar un índice, que especifique su posición relativa en el array. Un arreglo es una colección finita, homogénea y ordenada de elementos.

- *Finita*: Todo arreglo tiene un límite; es decir, debe determinarse cuál será el número máximo de elementos que podrán formar parte del arreglo.
- *Homogénea*: Todos los elementos del arreglo deben ser del mismo tipo.
- *Ordenada*: Se puede determinar cuál es el primer elemento, el segundo, el tercero,.... y el n-ésimo elementó.

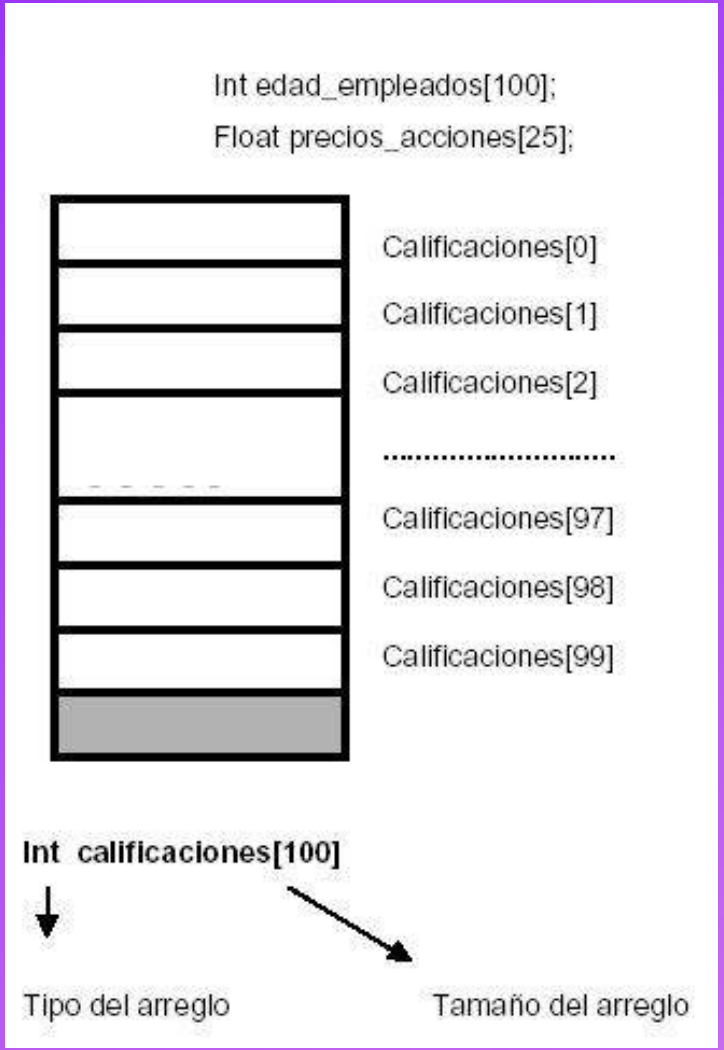
Los arreglos se clasifican de acuerdo con el número de dimensiones que tienen. Así se tienen los:

- Unidimensionales (vectores)
- Bidimensionales (tablas o matrices)
- Multidimensionales (tres o más dimensiones)

ARREGLOS UNIDIMENSIONLES



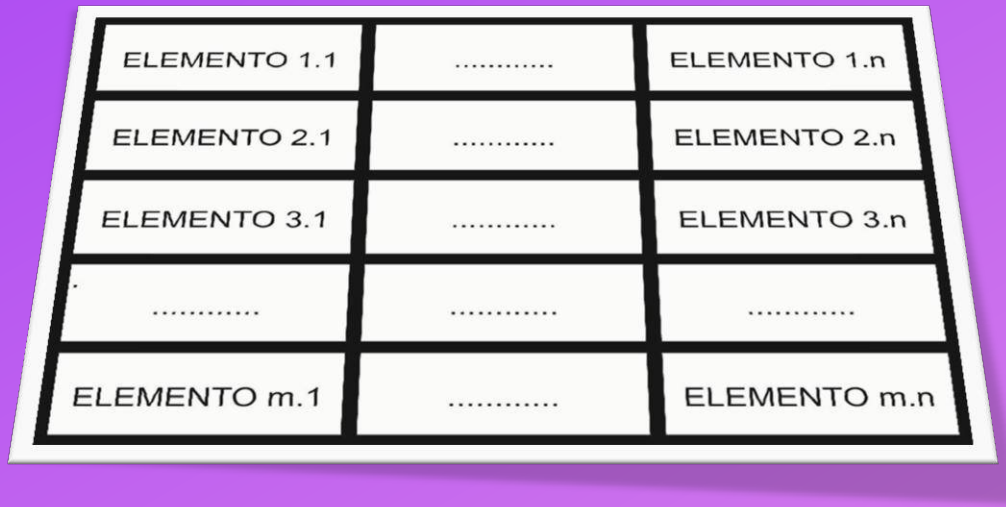
Están formados por un conjunto de elementos de un mismo tipo de datos que se almacenan bajo un mismo nombre, y se diferencian por la posición que tiene cada elemento dentro del arreglo de datos. Al declarar un arreglo, se debe inicializar sus elementos antes de utilizarlos. Para declarar un arreglo tiene que indicar su tipo, un nombre único y la cantidad de elementos que va a contener.



Para acceder a valores específicos del arreglo, use un valor de índice que apunte al elemento deseado. Por ejemplo, para acceder al primer elemento del arreglo calificaciones debe utilizar el valor de índice 0 (calificaciones[0]).

ARRELOS BIDIMENSIONALES

Es un conjunto de datos homogéneo, finito y ordenado, donde se hace referencia a cada elemento por medio de dos índices. El primero se utiliza para los renglones (filas) y el segundo para las columnas. También puede definirse como un arreglo de arreglos. Internamente en memoria se reservan MxN posiciones consecutivas para almacenar todos los elementos del arreglo.



Declaración de una matríz

Id_arreglo=arreglo[liminf..limsup,liminfc..limsupc] de tipo

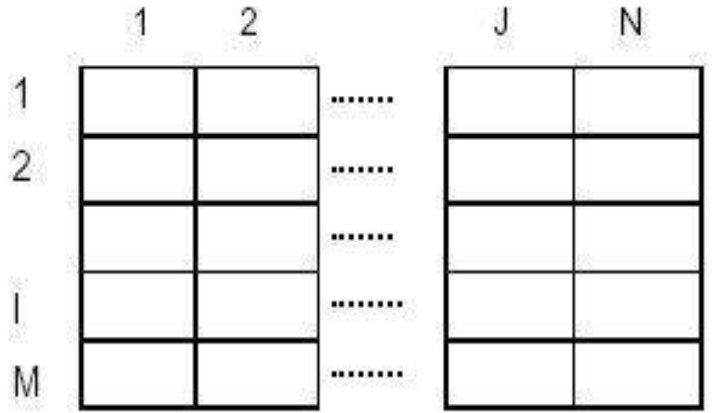
Filas

columnas

Por ejemplo: $1 \leq I \leq M$

$1 \leq J \leq N$

Se representa en forma de una tabla:



Pseudocódigo para el recorrido por filas:

```
Const
    M=valor1
    N= valor2
Tipo
    Array[1..M,1..N] de real:matriz
Var
     Matriz:A
              Desde i = 1 hasta M hacer
                       Desde j = 1 hasta N hacer
                   Escribir (A[i,j])
                            Fin_desde
                   Fin_desde
```

El recorrido por columnas se hace de manera similar, invirtiendo el sentido de los índices.

```
Desde j = 1 hasta N hacer
       Desde i = 1 hasta M hacer
                Escribir (A[i,j])
          Fin_desde
Fin_desde
```

WEBGRAFÍA Y LICENCIA

- Textos tomados, corregidos y modificados de diferentes páginas de Internet, tutoriales y documentos.
- Este documento se encuentra bajo Licencia Creative Commons 2.5 Argentina (BY-NC-SA), por la cual se permite su exhibición, distribución, copia y posibilita hacer obras derivadas a partir de la misma, siempre y cuando se cite la autoría del Prof. Matías E. García y sólo podrá distribuir la obra derivada resultante bajo una licencia idéntica a ésta.
- Autor:

Matías E. García

Prof. & Tec. en Informática Aplicada
www.profmatiasgarcia.com.ar
info@profmatiasgarcia.com.ar