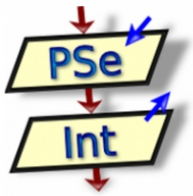
CURSO DE PROGRAMACIÓN FULL STACK

# ARREGLOS CON PSEINT




EGG

# GUÍA DE ARREGLOS

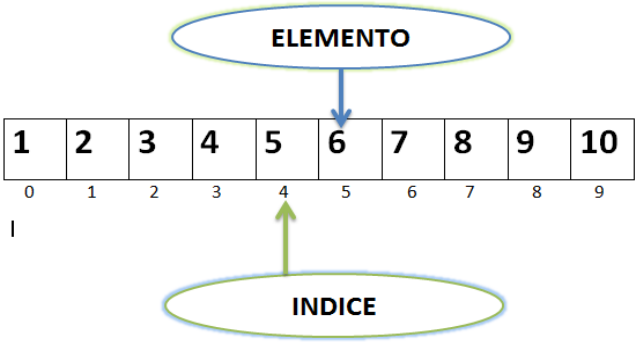
## ARREGLOS

En guías previas la manera de manipular datos era a través de variables, las variables nos dejan manejar de a un dato a la vez, pero si necesitáramos manejar varios datos juntos en un mismo lugar, usaríamos los arreglos.
Un *array* o arreglo (matriz o vector) es un conjunto *finito* y *ordenado* de elementos *homogéneos*. La propiedad "ordenado" significa que el elemento primero, segundo, tercero, ..., enésimo de un arreglo puede ser identificado. Los elementos de un arreglo son homogéneos, es decir, del mismo tipo de datos. Un arreglo puede estar compuesto de todos sus elementos de tipo cadena, otro puede tener todos sus elementos de tipo entero, etc, pero no puede ser de datos distintos. Los arreglos también pueden utilizarse en expresiones lógicas si necesitásemos comprobar varios elementos a la vez, o si necesitásemos saber si un elemento de nuestro arreglo, existe dentro del arreglo.

## ARREGLOS UNIDIMENSIONALES: VECTORES

El tipo más simple de arreglo es el arreglo unidimensional o vector. Un vector es un arreglo de n elementos, uno detrás de otro, que posee las siguientes características:

- *Se identifica por un único nombre de variable.*
- *Sus elementos se almacenan en posiciones del vector y cada a posición le corresponde un subíndice.*
- *Se puede acceder a cada uno de sus elementos a través del subíndice de forma ordenada o en forma aleatoria.*
- *Su tamaño es finito, esto significa que una vez definido su tamaño, este no puede cambiar. El tamaño es la cantidad de elementos que puede guardar nuestro vector.*




## SUBÍNDICE

- El subíndice es el número entero que identifica cada elemento dentro del vector, sin importar el tipo de dato que posea.
- Un vector de tamaño N posee N subíndices que se suceden de forma creciente y monótona. Ejemplo: 0 – 1 – 2 – 3 - 4 - 5 – 6 – N

- El valor inicial del primer subíndice depende del lenguaje; la mayoría de los modernos inician con el cero, por lo tanto, en PSeInt comenzarán en cero y los posibles valores de los subíndices irán desde 0 hasta N-1.

## DECLARACIÓN

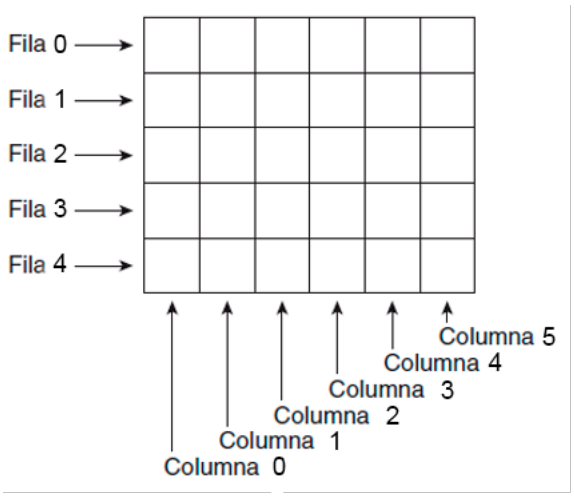
```
Definir nombre_vector como Tipo_de_Dato
Dimension nombre_vector(tamaño)
```

Donde Tipo_De_Dato se corresponde con cualquiera de los tipos de datos simples vistos previamente: entero, real, cadena, lógico.

La declaración Dimension nos sirve para darle el tamaño a nuestro vector, que recordemos, no puede cambiar una vez declarado. El tamaño va a ser siempre un numero entero o una variable entera, el tamaño no puede ser un numero con decimales.
El tamaño nos sirve para declarar cuantos elementos va a poder guardar nuestro vector. Si decimos que nuestro vector va a guardar 5 elementos, no puede guardar 6 o nos producirá un error.

## ARREGLOS BIDIMENSIONALES: MATRICES

Una *matriz* se puede considerar como un vector de vectores. Una matriz es un conjunto de elementos, todos del mismo tipo, en el cual el orden de los componentes es significativo y en el que se necesita especificar dos *subíndices* para poder identificar cada elemento del arreglo. Si se visualiza un arreglo unidimensional, se puede considerar como una columna de datos; un arreglo bidimensional o matriz es un grupo de filas y columnas:




En una matriz un subíndice no es suficiente para especificar un elemento, se referencian con dos subíndices: el primer subíndice se refiere a la fila y el segundo subíndice se refiere a la columna. Por lo tanto, una matriz se considera que tiene dos dimensiones (una dimensión por cada subíndice) y necesita un valor para cada subíndice para poder identificar un elemento individual. En notación estándar, normalmente el primer subíndice se refiere a la fila del arreglo, mientras que el segundo subíndice se refiere a la columna.

## DECLARACIÓN

```
Definir nombre_matriz como Tipo_de_Dato
Dimension nombre_matriz(tamañoFila,tamañoColumna)
```

Donde Tipo_De_Dato se corresponde con cualquiera de los tipos de datos simples vistos previamente: entero, real, cadena, lógico.

La declaración Dimension nos sirve para darle el tamaño a nuestra matriz, que recordemos, no puede cambiar una vez declarada. A diferencia de los vectores, vamos a tener que darle un tamaño a las filas y un tamaño a las columnas, separadas por coma.
A la hora de definir el tamaño de una matriz, no es necesario que sean matrices cuadradas. Las matrices cuadradas son, cuando tienen el mismo tamaño tanto para las filas que para las columnas. Pero podemos crear matrices que tengan valores distintos en filas y columnas.

## ARREGLOS MULTIDIMENSIONALES

Un arreglo puede ser definido de tres dimensiones, cuatro dimensiones, hasta de n-dimensiones. Los conceptos de rango de subíndices y número de elementos se pueden ampliar directamente desde arreglo s de una y dos dimensiones a estos arreglos de orden más alto. En general, un arreglo de n-dimensiones requiere que los valores de los n subíndices puedan ser especificados a fin de identificar un elemento individual del arreglo. Si cada componente de un arreglo tiene *n subíndices,* el arreglo se dice que es sólo de *n-dimensiones*.

## DECLARACIÓN

```
Definir nombre_arreglo como Tipo_de_Dato
Dimension nombre_arreglo(tamañoDim1,tamañoDim2,…, tamañoDimN)
```

## ASIGNAR ELEMENTOS A UN ARREGLO

### VECTORES

Cuando queremos ingresar un elemento en nuestro arreglo vamos a tener que elegir el subíndice en el que lo queremos guardar. Una vez que tenemos el subíndice decidido tenemos que invocar nuestro vector por su nombre y entre paréntesis el subíndice en el que lo queremos guardar. Después, pondremos el signo de igual (que es el operador de asignación) seguido del elemento a guardar.
El elemento a guardar debe coincidir con el tipo de dato de nuestro arreglo, si nuestro arreglo es de tipo entero, solo podemos guardar números enteros. También sucede algo parecido con el subíndice, no podemos llamar un subíndice que no existe, recordemos que los subíndices dependen del tamaño de nuestro arreglo. Entonces si tenemos un arreglo de tamaño 5, no podemos llamar el subíndice 6 porque no existe.

Ejemplo:

```
nombre_arreglo(0) = 4
```

Esta forma de asignación implica asignar todos los valores de nuestro arreglo de uno en uno, esto va a conllevar un trabajo bastante grande dependiendo del tamaño de nuestro arreglo.

Entonces, para poder asignar varios valores a nuestro arreglo y no hacerlo de uno en uno usamos un bucle Para. El bucle Para, al poder asignarle un valor inicial y un valor final a una variable, podemos adaptarlo fácilmente a nuestros arreglos. Ya que, pondríamos el valor inicial de nuestro arreglo y su valor final en las respectivas partes del Para. Nosotros, usaríamos la variable creada en el Para, y la pasaríamos a nuestro arreglo para representar todos los subíndices del arreglo, de esa manera, recorriendo todas las posiciones de nuestro arreglo, asignándole a cada posición un elemento.

```
Para i<-0 Hasta 4 Con Paso 1 Hacer
      nombre_arreglo(i) = 4
Fin Para
```

Nuestra variable i pasara por todos los subíndices de nuestro arreglo, ya que ira desde 0 hasta 4. Recordemos que los arreglos arrancan de 0, entonces, debemos calcular que, si el tamaño que le definimos al arreglo es de 5, necesitamos que nuestro Para vaya de 0 a 4

### MATRICES

Cuando queremos asignar un elemento a un arreglo bidimensional o matriz, vamos a necesitar pasarle dos subíndices, uno para las filas y otro para las columnas.

```
nombre_matriz(0)(0) = 10
```

Y para poder asignar varios elementos a nuestra matriz, usaríamos dos bucles Para anidados., ya que un Para recorrerá las filas *(variable i)* y otro las columnas *(variable j).*

```
Para i<-0 Hasta 2 Con Paso 1 Hacer
   Para j<-0 2 Con Paso 1 Hacer
       nombre_matriz(i)(j) = 10
   Fin Para
Fin Para
```

## MOSTRAR O TRAER ELEMENTOS DE UN ARREGLO

### VECTORES

A la hora de querer mostrar o traer algún elemento de nuestro arreglo, lo único que tenemos que hacer es escribir el nombre de nuestro arreglo y entre llaves o paréntesis pasarle un subíndice de ese arreglo para que traiga el elemento que se encuentra en ese subíndice,

```
Escribir nombre_arreglo(0)

Variable = nombre_arreglo(0)
```

Si quisiéramos mostrar todos los elementos de nuestro arreglo, deberíamos usar una estructura Para, que recorrerá todos los subíndices de nuestro arreglo y así poder mostrarlos todos.

```
Para i<-0 Hasta 4 Con Paso 1 Hacer
      Escribir nombre_arreglo(i)
Fin Para
```

Nuestra variable i pasara por todos los subíndices de nuestro arreglo, ya que ira desde 0 hasta 4. Esto es porque como los arreglos arrancan de 0, debemos calcular que, si el tamaño que le definimos al arreglo es de 5, necesitamos que nuestro Para vaya de 0 a 4

### MATRICES

Cuando queremos mostrar o traer un elemento de un arreglo bidimensional o matriz, vamos a necesitar pasarle dos subíndices, uno para las filas y otro para las columnas.

```
Escribir nombre_matriz(0)(0)

Variable = nombre_matriz(0)(0)
```

Ahora, si quisiéramos mostrar todos los elementos de nuestro arreglo bidimensional o matriz, vamos a tener que utilizar dos estructuras Para para traer todos los elementos de nuestra matriz, ya que un Para recorrerá las filas y otro las columnas.

```
Para i<-0 Hasta 2 Con Paso 1 Hacer
   Para j<-0 2 Con Paso 1 Hacer
       Escribir Sin Saltar nombre_matriz[i][j]
   Fin Para
   Escribir " "
Fin Para
```

Nota: este ejemplo funciona con una matriz cuadrada donde el tamaño de las filas sea el mismo que de las columnas.

Ejemplo para abrir en Pseint: Vector

Ejemplo para abrir en Pseint: Matriz

## USO EN SUBPROGRAMAS

Los arreglos se pueden pasar como parámetros a un subprograma (función o procedimiento) del mismo modo que las variables escalares. Sin embargo, hay que tener en cuenta que los arreglos, a diferencia de los tipos de datos simples, pasan siempre como parámetro "Por Referencia", ya que usualmente en nuestros subprogramas usamos los arreglos para rellenar, mostrar nuestros arreglos, etc.

```
Funcion variable_de_retorno <- Nombre (vector por referencia)
      Definir variable_de_retorno como Tipo de Dato
      <acciones>
Fin Funcion

SubProceso Nombre (matriz por referencia)
      <acciones>
FinSubProceso
```

# PREGUNTAS DE APRENDIZAJE

1. **Dada la siguiente sentencia correcta: uno[dos] = verdadero, se puede afirmar...**
   a) dos es una variable de tipo lógico
   b) uno es una variable de tipo lógico
   c) uno es una variable de tipo de vector de lógico
   d) dos es una variable de tipo vector de lógico

2. **Dada la siguiente sentencia correcta: uno[dos] = "auto", se puede afirmar...**
   a) uno es una variable de tipo de vector de carácter
   b) uno es una variable de tipo carácter
   c) dos es una variable de tipo entero
   d) dos es una variable de tipo vector de carácter

3. **Señale cuál de las siguientes afirmaciones es verdadera:**
   a) Los vectores sólo almacenan elementos del mismo tipo
   b) Las cadenas se implementan como un arreglo de caracteres
   c) Los vectores pueden almacenar datos de distinto tipo
   d) Los vectores no pueden pasarse como parámetro a un subprograma

4. **Para sumar dos matrices de orden 3x3:**
   a) Se requieren dos bucles
   b) Se requieren tres bucles
   c) Sólo se requiere un bucle
   d) No requiere ningún bucle

5. **Indique cuál de las siguientes sentencias es verdadera:**
   a) Un arreglo es una estructura de datos heterogénea
   b) Un arreglo es una estructura de datos homogénea
   c) Los arreglos no pueden tener más de tres dimensiones.
   d) Ninguna de las anteriores

6. **Para recorrer una matriz de orden NxMxP:**
   a) Se requieren dos bucles
   b) Sólo se requiere un bucle
   c) Se requieren tres bucles
   d) No requiere ningún bucle

7. **Un subíndice puede ser representado por:**
   a) Una variable de tipo entero
   b) Una constante numérica de tipo entero
   c) Una expresión algebraica cuyo resultado sea equivalente a un valor entero
   d) Todas las anteriores

8. **Dado el siguiente fragmento de código:**

```
...
   Dimension dias(7)
   Definir dias Como Real
   Definir tempAux Como Real
   Definir sabado como Entero

   dias(1) = 17            //asignación con índice constante
   sabado = 6
   leer dias(sabado)       //asignación mediante una función
   tempAux = dias(sabado)  //acceso con índice variable
...
```

**Indique con una cruz cuál/es de las siguientes sentencias es incorrecta:**

a) `días(0) = "lunes"`
b) `tempAux = días`
c) `tempAux = días(0)`
d) `tempAux = días(sabado - 1)`
e) `tempAux = días(sabado + 5)`
f) `tempAux = días(días(sábado))`