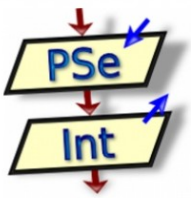
ARREGLOS CON PSEINT



EGG

Objetivos de la Guía

En esta guía aprenderemos a:

- Definir arreglos de acuerdo al tipo de dato que contendrán.
- Dimensionar arreglos.
- Rellenar arreglos.
- Mostrar correctamente arreglos por pantalla.
- Utilizar funciones y subprogramas, y estructuras de control para trabajar con arreglos y matrices.

¿QUÉ SON LOS 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. Si vieramos las variables como cajas, estas nos permitian guardar un solo dato. Los arreglos son como cajas, pero dentro hay compartimentos, que permiten guardar varios datos, siempre y cuando sean del mismo tipo.

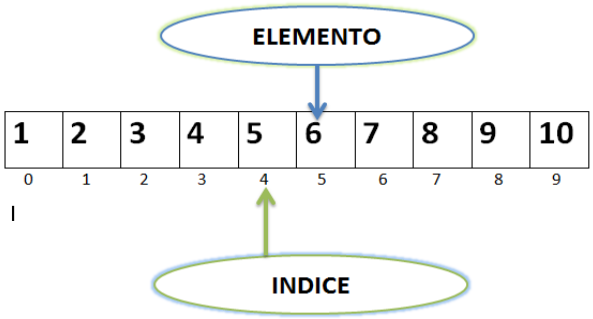
Un *array* o arreglo (matriz o vector) es un conjunto *finito* y *ordenado* de elementos *homogéneos*. La propiedad "**ordenado**" significa que el elemento primero, segundo, tercero, ..., enésimo de un arreglo puede ser identificado. Los elementos de un arreglo son **homogéneos**, es decir, del mismo 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

¿QUÉ SON LOS 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.



Matemáticamente se ve algo así: v = [1, 2, 3 ,4 ...n]

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.

```
Algoritmo vectores
    Definir vector Como Entero
    Dimension vector[5]
FinAlgoritmo
```

MANOS A LA OBRA!

EJERCICIO DEFINIR VECTOR

Define un vector que alojará números enteros y otro de cadena. Dimensiona ambos de la longitud que tu desees. Ahora en lapiz y papel, escribe la dimensión de tus vectores y sus subíndices.

ASIGNACIÓN DE VALORES

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.

```
Algoritmo vectores

    Definir vector Como Entero

    Dimension vector[5]

    vector[0] = 1
    vector[1] = 2
    vector[2] = 3
    vector[3] = 4
    vector[4] = 5

FinAlgoritmo
```

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.

```
Algoritmo vectores
    Definir vector,i Como Entero
    Dimension vector[5]
    Para i<-0 Hasta 4 Con Paso 1 Hacer
        vector[i] = i
    Fin Para
FinAlgoritmo
```

Nuestra variable i pasara por todos los subíndices de nuestro arreglo, ya que ira desde 0 hasta 5. 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

MANOS A LA OBRA!

EJERCICIO LLENAR VECTOR

Ahora es tu turno. Llena uno de los vectores que definiste y dimensionaste anteriormente, de forma manual y el otro con un Bucle Para.

DETECCIÓN DE ERRORES

Copia y pega el código que está a continuación, tu tarea es arreglar el código.

```
Algoritmo vectores
Definir vector Como Entero
Dimension vector()
Para i<-0 Hasta 5 Con Paso 1 Hacer
      vector(0)=0
Fin Para
FinAlgoritmo
```

MOSTRAR O TRAER ELEMENTOS DE UN ARREGLO

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.

```
Definir vector,var Como Entero
Dimension vector[5]
Escribir vector[2]
var = vector[3]
```

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 Sin Saltar "[" vector[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.

Utilizamos Escribir Sin Saltar para que el programa no muestre saltos de línea al imprimir el vector.

MANOS A LA OBRA!

EJERCICIO MOSTRAR VECTOR

Ahora te toca a ti mostrar tus vectores. Además, define una nueva variable y aloja allí algún valor del vector.

USO EN SUBPROGRAMAS

Los arreglos, cualquiera sea su dimensión, 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
```

¿Cómo se ve en PseInt?

```
SubProceso ejemploSubProceso ( vector Por Referencia )
    //Acciones
FinSubProceso
Algoritmo vectores
    Definir vector Como Entero
    Dimension vector[5]
FinAlgoritmo
```