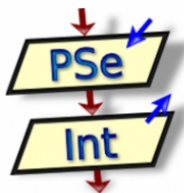


CURSO DE PROGRAMACIÓN FULL-STACK

MATERIAL DE TRABAJO

Arreglos con PseInt





Objetivos de la Guía

En esta guía aprenderemos a:

- Definir arreglos de acuerdo con el 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 permitían 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.

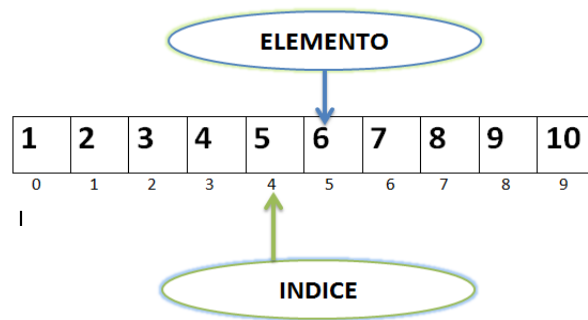
Un array o arreglo (matriz o vector) es un conjunto finito y ordenado de elementos homogéneos. La propiedad “**ordenado**” significa que el elemento primero, segundo, tercero, ..., enésimo de un arreglo puede ser identificado. Los elementos de un arreglo son **homogéneos**, es decir, del mismo tipo de 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: $v = [1, 2, 3, 4 \dots n]$

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 número entero o una variable entera, el tamaño no puede ser un número 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



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 por 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
```

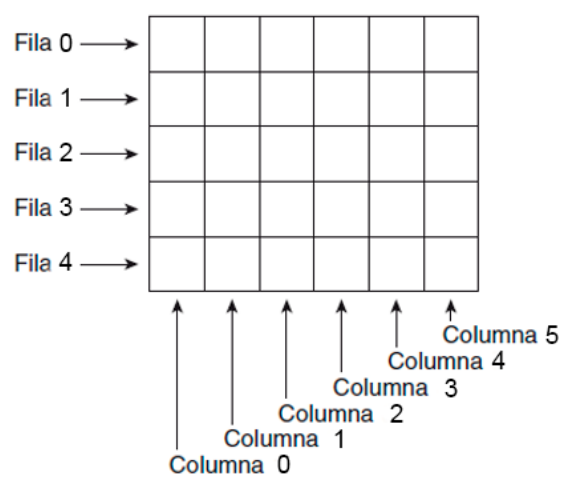
¿Cómo se ve en PseInt?

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

ARREGLOS BIDIMENSIONALES: MATRICES

¿QUÉ SON LAS 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 arreglos 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

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

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.

EJERCICIOS PRÁCTICOS

Para cada uno de los siguientes ejercicios realizar el análisis del problema e indicar cuáles son los datos de entrada y cuáles son los datos de salida. Escribir luego el algoritmo en PSeInt haciendo uso de vectores y/o matrices según corresponda en cada caso. **¡¡¡Recuerda que sabes usar sub-programas!!!**

Arreglos: Vectores

1. Realizar un programa que rellene un vector con 5 valores ingresados por el usuario y los muestre por pantalla.
2. Realizar un programa que lea 10 números reales por teclado, los almacene en un arreglo y muestre por pantalla la suma, resta y multiplicación de todos los números ingresados al arreglo.
3. Realizar un programa que rellene un vector de tamaño N, con valores ingresados por el usuario. A continuación, se debe buscar un elemento dentro del arreglo (el número a buscar también debe ser ingresado por el usuario). El programa debe indicar la posición donde se encuentra el valor. En caso de que el número se encuentre repetido dentro del arreglo se deben imprimir todas las posiciones donde se encuentra ese valor.

Finalmente, en caso de que el número a buscar no está adentro del arreglo se debe mostrar un mensaje.

4. Realizar un programa con el siguiente menú y le pregunte al usuario que quiere hacer hasta que ingrese la opción Salir:
 - A. Llenar Vector A. Este vector es de tamaño N y se debe llenar de manera aleatoria usando la función Aleatorio(valorMin, valorMax) de PseInt.
 - B. Llenar Vector B. Este vector también es de tamaño N y se llena de manera aleatoria.
 - C. Llenar Vector C con la suma de los vectores A y B. La suma se debe realizar elemento a elemento. Ejemplo: $C = A + B$
 - D. Llenar Vector C con la resta de los vectores B y A. La resta se debe realizar elemento a elemento. Ejemplo: $C = B - A$
 - E. Mostrar. Esta opción debe permitir al usuario decidir qué vector quiere mostrar: Vector A, B, o C.
 - F. Salir.

NOTA: El rango de los números aleatorios para los Vectores será de [-100 a 100]. La longitud para todos los vectores debe ser la misma, por lo tanto, esa información sólo se solicitará una vez.

5. Realizar un programa que rellene un vector de tamaño N, con valores ingresados por el usuario. A continuación, se deberá crear una función que reciba el vector y devuelva el valor más grande del vector.

6. Disponemos de un vector unidimensional de 20 elementos de tipo carácter. Se pide desarrollar un programa que:

- Pida una frase al usuario y luego ingrese la frase dentro del arreglo letra por letra. Ayuda: utilizar la función Subcadena de PSeInt.
- Una vez completado lo anterior, pedirle al usuario un carácter cualquiera y una posición dentro del arreglo, y el programa debe intentar ingresar el carácter en la posición indicada, si es que hay lugar (es decir la posición está vacía o es un espacio en blanco). De ser posible debe mostrar el vector con la frase y el carácter ingresado, de lo contrario debe darle un mensaje al usuario de que esa posición estaba ocupada.

Por ejemplo, suponiendo la siguiente frase y los subíndices del vector:

H	o	l	a		m	u	n	d	o		c	r	u	e	l	!			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Si se desea ingresar el carácter “%” en la posición 10, entonces el resultado sería:

H	o	l	a		m	u	n	d	o	%	c	r	u	e	l	!			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

7. Crear un subproceso que rellene dos arreglos de tamaño n, con números aleatorios. Después, hacer una función que reciba los dos arreglos y diga si todos sus valores son iguales o no. La función debe devolver el resultado de esta validación, para mostrar el mensaje en el algoritmo. **Nota:** recordar el uso de las variables de tipo lógico.

Arreglos: Matrices

- Realizar un programa que rellene una matriz de 3x3 con 9 valores ingresados por el usuario y los muestre por pantalla.
- Escribir un programa que realice la búsqueda lineal de un número entero ingresado por el usuario en una matriz de 5x5, llena de números aleatorios y devuelva por pantalla las coordenadas donde se encuentra el valor, es decir en que fila y columna se encuentra. En caso de no encontrar el valor dentro de la matriz se debe mostrar un mensaje.
- Dada una matriz de orden n * m (donde n y m son valores ingresados por el usuario) realizar un subprograma que llene la matriz de numeros aleatorios. Despues, crearemos otro subprograma que calcule y muestre la suma de los elementos de la matriz. Mostrar la matriz y los resultados por pantalla.
- Rellenar en un subproceso una matriz cuadrada con números aleatorios salvo en la diagonal principal, la cual debe rellenarse con ceros. Una vez llena la matriz debe generar otro subproceso para imprimir la matriz.

12. Rellenar una matriz, de 3 x 3, con una palabra de 9 de longitud, pedida por el usuario, encontrando la manera de que la frase se muestre de manera continua en la matriz.

Por ejemplo, si tenemos la palabra habilidad, nuestra matriz se debería ver así:

H	A	B
I	L	I
D	A	D

Nota: recordar el uso de la función Subcadena().

13. Una matriz mágica es una matriz cuadrada (tiene igual número de filas que de columnas) que tiene como propiedad especial que la suma de las filas, las columnas y las diagonales es igual. Por ejemplo:

2	7	6
9	5	1
4	3	8

En la matriz de ejemplo las sumas son siempre 15. Considere el problema de construir un algoritmo que compruebe si una matriz de datos enteros es mágica o no, y en caso de que sea mágica escribir la suma. Además, el programa deberá comprobar que los números introducidos son correctos, es decir, están entre el 1 y el 9. El usuario ingresa el tamaño de la matriz que no debe superar orden igual a 10.

EJERCICIOS EXTRA

Estos van a ser ejercicios para reforzar los conocimientos previamente vistos. Estos pueden realizarse cuando hayas terminado la guía y tengas una buena base sobre lo que venimos trabajando. Además, si ya terminaste la guía y te queda tiempo libre, puedes continuar con estos ejercicios extra. Por último, recuerda que la prioridad es ayudar a los compañeros de tu equipo y que cuando tengas que ayudar, lo más valioso es que puedas explicar el ejercicio con la intención de que tu compañero lo comprenda, y no sólo mostrarlo. ¡Muchas gracias!

Arreglos: Vectores

1. Realizar un programa que rellene dos vectores al mismo tiempo, con 5 valores aleatorios y los muestre por pantalla.
2. Realizar un programa que rellene un vector de tamaño N, con valores ingresados por el usuario y muestre por pantalla el promedio de la suma de todos los valores ingresados.
3. Crear dos vectores que tengan el mismo tamaño (el tamaño se pedirá por teclado) y almacenar en uno de ellos nombres de personas como cadenas. En el segundo vector se debe almacenar la longitud de cada uno de los nombres (para ello puedes usar la función Longitud() de Pselnt). Mostrar por pantalla cada uno de los nombres junto con su longitud.
4. Crear un vector que contenga 100 notas de 100 supuestos estudiantes, con valores entre 0 y 20 generadas aleatoriamente mediante el uso de la función azar() o aleatorio() de Pselnt. Luego, de acuerdo a las notas contenidas, el programa debe indicar cuántos estudiantes son:
 - a) Deficientes 0-5
 - b) Regulares 6-10
 - c) Buenos 11-15
 - d) Excelentes 16-20
5. Tomando en cuenta el ejercicio 6, mejore el mecanismo de inserción del carácter, facilitando un potencial reordenamiento del vector. Digamos que se pide ingresar el carácter en la posición X y la misma está ocupada, entonces de existir un espacio en cualquier posición X-n o X+n, desplazar los caracteres hacia la izq o hacia la derecha para poder ingresar el carácter en cuestión en el lugar deseado. El procedimiento de reordenamiento debe ubicar el espacio más cercano.

Por ejemplo, suponiendo la siguiente frase y los subíndices del vector:

H	o	l	a		m	u	n	d	o		c	r	u	e	l	!			
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Si se desea ingresar el carácter “%” en la posición 8, entonces el resultado con desplazamiento sería:

h	o	l	a		m	u	n	%	d	o		c	r	u	e	l	!		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Notar que el desplazamiento se hizo hacia la izquierda porque el espacio de la posición 10 estaba más cerca de la posición 8 que el espacio de la posición 4. **Nota:** en caso del que el desplazamiento sea a la izquierda y se requiera que se remueva la letra, se hará. Por ejemplo, para poner un “%” en la posición 6, haríamos un desplazamiento a la izquierda y borraríamos la letra h

6. Crear una función que devuelva la diferencia que hay entre el valor más chico de un arreglo y su valor más grande.
7. Programe una función que calcule el producto de un arreglo de números enteros. Para esto imagine, por ejemplo, que para un vector V de tamaño 4, el producto de todos los valores es igual a $(V[1]*V[2]*V[3]*V[4])$