

## Estructuras de datos: Arrays unidimensionales

### Vectores: Ejemplos comentados

#### Notación algorítmica.

Definición de tipo

**tipo**  
**array**[L:U] **de** <tipo\_dato>: <identificador\_tipo\_array>

Declaración de tipo

<identificador\_tipo\_array>: lista\_de\_variables



Los **tipos de datos estructurados** son conjuntos de variables que representan mediante un identificador (nombre) a *múltiples datos* o *elementos*, pudiendo cada uno de estos elementos, ser referenciado independientemente

#### Ejemplos de operaciones con vectores.

**Lectura de vectores:** Técnica de Iteración (*bucles for*)

**Ejemplo 1.** Lectura de 10 valores enteros para asignarlos al vector denominado **n**.

**Algoritmo** leer\_vector\_Numeros

**tipo**

**array**[1..10] **de** entero: arrNumero

**var**

arrNumero: n

entero: i

**inicio**

//recorrido: op. de lectura de array por teclado

**desde**( i ← 1 **hasta** 10 )**hacer**

**escribir**("Item: ", i)

**leer**(n[i])

**fin\_desde**

**fin**

**Ejemplo 2.** Lectura de 7 valores enteros en un vector para almacena los días de la semana: 1° día (Dom) al 7° día (Sab).

**Algoritmo** leer\_vector\_dias

**const** LIM\_SUP = 7

**tipo**

**array**[1..LIM\_SUP] **de** entero: vSemana

**var**

dias: d

entero: i, rango

**inicio**

//elementos del vector

rango ← (LIM\_SUP - 1) + 1

**desde** ( i ← 1 **hasta** rango ) **hacer**

d[i] ← i //1° día (Dom) a 7° día (Sab)

**fin\_desde**

**escribir** ("Fin de semana, día: ", d[6])

**escribir** ("Domingo, día: ", d[1])

**fin**

**Escritura de vectores:** La salida o escritura de vectores es un proceso inverso al de lectura. En la operación de escritura se implementa el método de Iteración.

Técnica de Iteración. (*bucle while*)

**Ejemplo 3.** Escritura de 5 notas de práctica de un alumno.

**Algoritmo** Visualizar\_vector\_Notas

**const** LIM\_INF ← 1, LIM\_SUP ← 5

**tipo**

**array**[LIM\_INF .. LIM\_SUP] **de** real: arrNotas

**var**

arrNotas: n

entero: i

**inicio**

//asignación

n[1] ← 12.5

n[2] ← 14.7

n[3] ← 16.0

n[4] ← 13.5

n[5] ← 18.5

i ← 1 //salida o escritura

**mientras** ( i ≤ 5 ) **hacer**

escribir(n[i])

i ← i + 1

**fin\_mientras**

**fin**



## Recorrido de vectores: ¿Cómo se aplica en procesos de cálculo?

**Ejemplo 4.** Procesamiento del array Ítems y operaciones aplicadas:

- Lectura del vector,
- Calculo de la suma acumulada de valores del vector, y
- Calculo de la media aritmética de una muestra de 10 elementos.

```
Algoritmo media_aritmetica
const LIMITE = 10
tipo
  array[1..LIMITE] de real: arrItems
var
  arrItems: items
  real: suma, media
  entero: i
inicio
  suma ← 0
  //lectura o entrada
  escribir("ingrese 10 datos reales para Array: ")
  desde( i ← 1 hasta LIMITE )hacer
    leer(items[i])
    suma ← suma + ítems[i]
  fin_desde
  //calcular promedio
  media ← suma / LIMITE
  escribir("La media es: ", media )
fin
```

**Ejemplo 5.** Se tienen N temperaturas registradas del mes de Septiembre. Se desea calcular su media y determinar entre todas ellas, cuantos días del mes han sido superiores o iguales a la temperatura media.

### Análisis.

En un primer momento se leen los datos, y se almacenan en un vector: temp[1:N]. Luego, calcular sucesivamente la suma de las temperaturas del mes, para obtener la media.

Por último, mediante un bucle de lectura del vector, comparar cada elemento con la media, y con la ayuda de un contador registre el número de temperaturas mayores o iguales a la media.

## SOLUCIONES

### Primer Método. (Modo source file)

```
Algoritmo tempMedia_01: temperatura media del mes de Septiembre.
const N = 30 //numero de elementos del vector: 30 días

tipo
  array[1..N] de real: arrTemp
var
  arrTemp: vTmp //vector de temperaturas
  real: suma, media //acumulador y temperatura media
  entero: i, k //contador de temperaturas mayor o igual a la media
```

```
inicio
    suma ← 0
    media ← 0
    k ← 0

    //bucle entrada de datos
    desde( i ← 1 hasta N )hacer
        escribir("temperatura dia: ", i)
        leer(vTmp[i])
        suma ← suma + vTmp[i] //acumular temperaturas
    fin_desde
    media ← suma / N //calcular media

    //contar numero de días con temperaturas mayor o igual a la media
    desde( i ← 1 hasta N )hacer
        si( vTmp[i] >= media )entonces
            k ← k + 1
            escribir("dia: ", i, " temperatura: ", vTmp[i])
        fin_si
    fin_desde
    escribir("La media del mes de Septiembre: ", media)
    escribir("Numero de días >= ", media, " es: ", k)
fin
```

### Segundo Método. (Programación modular: Sub-rutinas).

Implementar en el algoritmo tempMedia\_01 funciones y procedimiento con argumentos tipo array.

Crear los siguientes módulos:

- función media() que retorna la media de temperaturas, esta rutina debe tomar como argumento el vector de temperaturas.
- Use la función media() dentro de una rutina de procedimiento definido como mayoresAmedia () para escribir las temperaturas mayores a la media. Este procedimiento tomar como argumento el vector de temperaturas.

```
Algoritmo tempMedia_02: temperatura media del mes de Septiembre.
const N = 30 //numero de elementos del vector: 30 días
tipo
    array[1..N] de real: arrTemp
var
    arrTemp: vTmp //vector de temperaturas
    entero: i
inicio
    // entrada de datos
    desde( i ← 1 hasta N )hacer
        escribir("temperatura dia: ", i)
        leer(vTmp[i])
    fin_desde
    //invocar firma de sub-rutina con parametros
    llamar_a mayoresAmedia(vTmp, N)
fin
```

```
/** rutina de función, con array como argumento */
real funcion media(E/S arrTemp: temp; E entero: rango)
var
    real suma          //acumulador de temperaturas
    entero: i

inicio
    suma ← 0
    // recorrido secuencial de vector y calculo de la temperatura media
    desde( i ←1 hasta rango )hacer
        suma ← suma + temp[i]
    fin_desde
    devolver (suma /rango)
fin_funcion

/** rutina de procedimiento, con array como argumento */
procedimiento mayoresAmedia(E/S arrTemp: temp; E entero: rango)
var
    entero: i, k //contador de temperaturas mayor o igual a la media
    real: promedio

inicio
    k ← 0
    promedio ← media(temp, rango)
    escribir("La media del mes de Septiembre: ", promedio)
    // contar numero de días con temperaturas mayor o igual a la media
    desde( i ← 1 hasta rango )hacer
        si( temp[i] >= promedio )entonces
            k ← k +1
            escribir("dia: ", i, "temperatura: ", temp[i])
        fin_si
    fin_desde
    //imprimir conteo
    escribir("Numero de dias >= ", promedio, " es: ", k)
fin_procedimiento
```

**Fuente de datos masivos.** Los arrays suelen ser estructuras de datos de mucha utilidad para manejar gran cantidad de datos (datos masivo) y con frecuencia estos provienen de fuentes externas (Dispositivos permanentes de almacenamiento externo) como Archivos de datos o administradores de Base de datos DBMS.

**Importante:** La lectura de datos masivos es un proceso de entrada desde una fuente externa (archivos de datos) puede ser simulada por la función *aleatorio()*. También se puede implementar simulaciones para investigación estadística, juegos de azar, etc. Por ejemplo, generar temperaturas del mes en el rango de: 21.00 a 24.00 grados centígrados.

Debe considerar la función *aleatorio()* como una función interna que genera números al azar entre 0 y 1 (reales infinitos). Según esto último, el proceso de generación de aleatorios entre rangos se verifica mediante la siguiente relación:


$$\text{aleatorio()} * (U - L) + L$$

Así por ejemplo: si queremos obtener aleatorios entre 50 y 80, y si la función *aleatorio()* genera números entre 0 y 1, tenemos:

U (máximo) 80 -  
L (mínimo) 50

30 (diferencia)

Entonces:

*aleatorio()* 

1 \* diferencia + mínimo se obtiene el **máximo** (80)  
0 \* diferencia + mínimo se obtiene el **mínimo** (50)

En general:

**`aleatorio() * (U - L) + L`**

**Ejemplo 6.** Números aleatorios. Lanzamientos de dos dados y visualizar valores al azar.

**Algoritmo** Lanzar dados al azar

**const** LIM\_INF = 1, LIM\_SUP = 6

**tipo**

**array**[LIM\_INF..LIM\_SUP] **de** real: vDado

**var**

vDado: dado1. dado2

entero: i

**inicio**

//inicializar dados

i ← LIM\_INF

**mientras**( i ≤ LIM\_SUP )**hacer**

dado1[i]←i

dado2[i]←i

i ← i +1

**fin\_mientras**

//función aleatorio()para generar números al azar en el rango 1-6

m ← aleatorio() \* (6 - 1) + 1

n ← aleatorio() \* (6 - 1) + 1

//visualizar valores de lanzamiento

**escribir**("Lanzar dados: ", dado1[m], dado2[n])

**fin**