Arreglos

Fundamentos de Programación

¿Qué es un Vector?

- Un vector fila es una matriz de una fila (1 x n)
- Un vector columna es una matriz de una columna (n x 1)
- En Scilab no hay tratamiento especial vectorial
- Todo se maneja como una matriz

Vectores

- **En un vector**:
  - Cada escalar dentro del vector tiene su **posición** (primero, segundo, tercero…)
  - Y todos los elementos del vector son reales.

- La mejor forma de visualizar un vector es:
  - Como un grupo de cajas, una detrás de otra


...

    - Donde cada caja representa un dato del vector o un elemento.
  - Podemos concluir que un vector tiene:
    - **Tamaño**: cuantas cajas va a tener, el número de datos.
    - **Dimensiones**: vector (1 x n) o (n x 1).
    - **Nombre**: el único nombre bajo el cual vamos a dirigirnos al mismo.

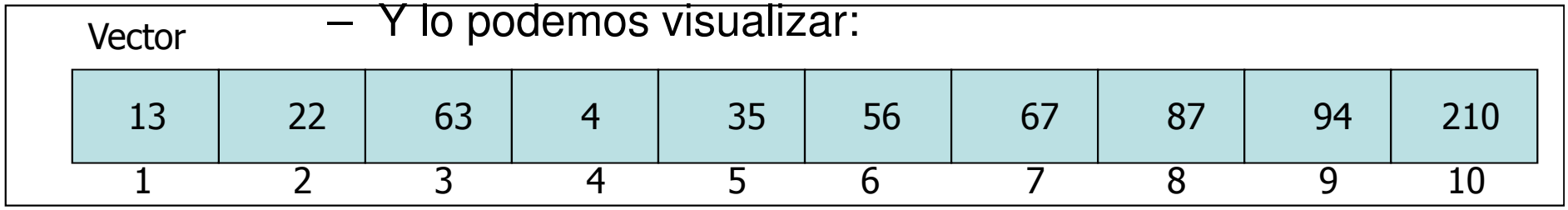
Declaración de Vectores (1 x n)

- Para declarar un vector, se debe indicar
  - Nombre y
  - Elementos
- Un vector (1 x n) de 10 elementos enteros, se decla
  - a=[13 22 63 4 35 56 67 87 94 210]

    a =

    13. 22. 63. 4. 35. 56. 67. 87. 94. 210.
  - Y lo podemos visualizar:

se introduce cada elemento del vector, separados por espacios, entre un par de corchetes



- Cada **elemento** del vector va a estar identificado por un v numérico, llamado **índice**.
- En Scilab el primer elemento de un arreglo tiene el índice
- Para obtener las **dimensiones** de un arreglo, se utiliza la
  size(nombre)
- Para obtener el **tamaño** de un arreglo, se utiliza la funcion
  length(nombre)

```
size(a)
 ans  =

    1.    10.
```

```
length(a)
 ans  =

    10.
```

Declaración de Vectores (n x 1)

- Un vector (n x 1) de 10 elementos enteros, se declara:
  - a=[1;2;3;4;5;6;7;8;9;10]

```
a =

    1.
    2.
    3.
    4.
    5.
    6.
    7.
    8.
    9.

   10.
```

se introduce cada elemento del vector, separados por ; entre un par de corchetes

- Para obtener las **dimensiones** de un vector, se utiliza la funcion size(nombre)
- Para obtener el **tamaño** de un arreglo, se utiliza la funcion length(nombre)

```
size(a)
 ans  =

    10.    1.
```

```
length(a)
 ans  =

    10.
```

Declaración de Vectores

- Supongamos que se desee crear un vector con elementos entre 0 y 20 uniformemente espaciados en incrementos de 2:

```
t=0:2:20
 t  =
    0.    2.    4.    6.    8.    10.    12.    14.    16.    18.    20.
```

Manejo de Vectores

- La manipulación de los vectores es tan sencilla como su creación:

```
a=[13 22 63 4 35 56 67 87 94 210]
a  =
   13.   22.   63.   4.   35.   56.   67.   87.   94.   210.
```

- Si se desea asignar el valor de 2 al primer elemento del arreglo:

```
      a (1,1) =2
 a  =
    2.   22.   63.   4.   35.   56.   67.   87.   94.   210.
```

- Si se desea asignar el valor de 56 al quinto elemento del arreglo:

```
 a (1,5) =56
  a  =
    2.   22.   63.   4.   56.   56.   67.   87.   94.   210.
```

Manejo de Vectores

- Supongamos que se desea sumar 2 a cada elemento en el vector a. El comando será así:

```
b=a+2
 b  =
    3.    4.    5.    6.    7.    8.    9.    10.    11.    12.
```

- Ahora, suponga que se desea sumar 2 vectores entre si. Si ambos vectores poseen la misma longitud, esto es sencillo. Simplemente se adicionan:

```
c=a+b
c  =
    4.    6.    8.    10.    12.    14.    16.    18.    20.    22.
```

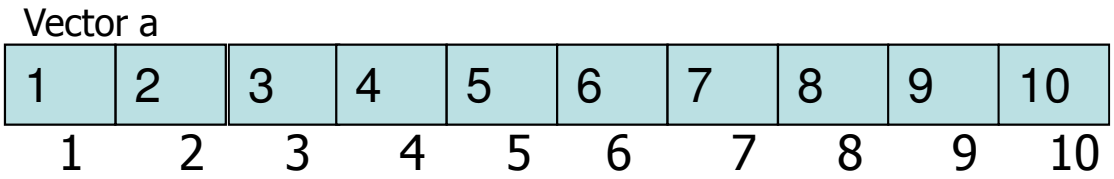
- La sustracción de vectores se realiza análogamente.

Manejo de Vectores

- ¿Como accedemos a uno de los elementos del vector?
  - Usamos el nombre del vector y el índice que identifica al elemento: **nombre_vector(indice)**
  - Si se desea obtener el valor del segundo elemento del vector:

```
a(1,2)
 ans  =
   2.
```

```
a(2)
 ans  =
   2.
```



  - Es muy importante recordar que:
    - El índice de un elemento, no es el valor (contenido) de dicho elemento.
    - El índice puede ser cualquier expresión que retorne un valor entero.

```
a(2+3)
 ans  =

   5.
```

  - Para asignar valores a los elementos del vector se puede usar un lazo for:

```
for i = 1:1:10
        a(1,i)= 0
end
```

Leer e Imprimir un Vector

- Si tenemos 10 elementos en un vector, y queremos pedir que los ingresen por teclado, debemos repetir el ingreso 10 veces:

```
f=[ ]

for i=1:10
      a= input('Ingrese elemento: ')
      f=[f a]
end
```

- Para imprimir todos los elementos de un vector:

```
disp(f)
```

f = [f a]

Esta sintaxis nos permite concatenar dos vectores.
(Un escalar es un vector de un solo elemento)

Ejercicio

- Programa que permite el ingreso de las notas de un curso de 20 alumnos. Una vez ingresados, debe mostrarse el promedio de las mismas.

```
f=[ ]
total=0
num= input("Ingrese el numero de notas: ")
for i=1:num
    mprintf("Ingrese nota %i:",i)
    a= input("")
    f=[f a]
end
for i=1:num
     total=total+f(i)
end
disp(total/num)
```

Matrices

- La generación de matrices en SCILAB funciona de la misma manera que con los vectores, con la peculiaridad que cada fila de elementos se encuentra separada por un punto y coma:

```
A=[1 2 3 ;4 5 6 ;7 8  9]
 A  =
    1.    2.    3.
    4.    5.    6.
    7.    8.    9.
```

Si se desea asignar el valor de 44 al elemento (2,3) de la matriz A:

```
A(2,3)=44
 A  =
    1.    2.    3.
    4.    5.    44.
    7.    8.    9.
```

A

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)
(3,1)	(3,2)	(3,3)

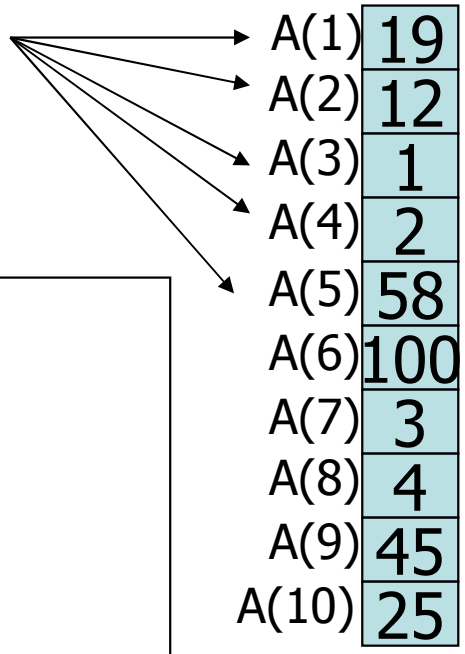
A(1,1)	**A(1,2)**	**A(1,3)**
A(2,1)	**A(2,2)**	**A(2,3)**
A(3,1)	**A(3,2)**	**A(3,3)**

Búsqueda en un Vector

- Se refiere al proceso para encontrar un elemento particular en un vector.
- Una de las estrategias mas comunes y simples para buscar un dato en un vector es:
  - Revisar uno por uno los elementos del mismo, este método se conoce como **búsqueda lineal**.
- Escribir un script que determine si un valor dado se encuentra en un vector de elementos enteros, y si es así, indique su posición.

Solución

Dato a buscar: 58




```
encontrado=0
A=[19;12;1;2;58;100;3;4;45;25]
valor=input("Valor a buscar: ")
for i=1:10
        if (valor==A(i)) then
                disp(i)
                encontrado=1
        end
end
if (encontrado==0) then
        mprintf("El valor no se encuentra en el vector")
end
```

Notaciones Matriciales

```
a=[1 4 5 6; 2 5 4 5; 3 4 5 6; 3 5 5 6]
 a  =
    1.    4.    5.    6.
    2.    5.    4.    5.
    3.    4.    5.    6.
    3.    5.    5.    6.
```

- a(2,4) Retorna el elemento ubicado en la posicion (2,4)
- a(3,:) retorna la tercera fila de la matriz “a”
- a(:,3) retorna la tercera columna de la matriz “a”
- a(1:2,2:4) retorna la sub-matriz de tamaño 2x3 formado por los elementos que están en las filas 1,2 y en las columnas 2,3,4

```
a(1:2,2:4)
 ans  =
    4.    5.    6.
    5.    4.    5.
```