

# Taller de Programación

## 04 - Estructuras de Datos

Mgs. Lic. Marcos Prunello

2018

- Hasta ahora usamos objetos que contenían una única pieza de información o dato.
- Pero la verdadera utilidad de la computación radica en poder trabajar con conjuntos de datos.
- Un **arreglo** (o *array*) es una colección ordenada de valores del mismo tipo.
- Los arreglos son muy útiles para organizar y almacenar información en la memoria de la computadora.

- Un **arreglo** tiene dos características fundamentales:
  - *Ordenamiento*
  - *Homogeneidad*
- Antes de utilizar un arreglo se lo debe **dimensionar**: indicarle a la computadora que reserve una zona de la memoria para su uso.
- Los arreglos pueden ser *unidimensionales*, *bidimensionales* o *multidimensionales*.

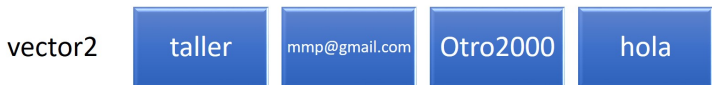
# Arreglos unidimensionales

- Conocido como **vector**.
- Tiene  $n$  elementos ordenados todos del mismo tipo.
- Ejemplo 1: vector de tipo numérico llamado *vector1* con 5 elementos:



Figure 1: Ejemplo vector numérico

- Ejemplo 2: vector de tipo caracter llamado *vector2* con 4 elementos:



# Arreglos unidimensionales: uso de índices

- Se accede al valor guardado en cada posición del vector con índices.
- Por ejemplo, la acción `ESCRIBIR vector1[3]` nos mostrará el valor 2.71.

nombre:	vector 1				
índice:	1	2	3	4	5
valor:	-4.5	12	2.71	-6	25

Figure 3: Ejemplo vector numérico con posiciones indexadas

# Arreglos unidimensionales: declaración

- Como todas las variables a usar, los vectores deben ser declarados en el algoritmo.
- Además, debens ser **dimensionados**, para reservar la memoria que requiere según su tamaño.
- En el ejemplo del *vector1*:

```
DIMENSIONAR numérico vector1(5)  
vector1[1] <- -4.5  
vector1[2] <- 12  
vector1[3] <- 2.71  
vector1[4] <- -6  
vector1[5] <- 25
```

# Arreglos unidimensionales: uso de estructuras iterativas

- Podemos asignar valores a las posiciones del vector empleando estructuras de control iterativas:

```
DIMENSIONAR numérico vector3(30)
PARA i DESDE 1 HASTA 30 HACER
    vector3[i] <- i
FIN PARA
```

- *i* se usa como índice para el espacio en el vector que será modificado y también para calcular el valor por asignar.

# Arreglos unidimensionales: uso de estructuras iterativas

- En el próximo ejemplo se deja que el usuario determine la dimensión del vector y que provea cada uno de los valores para el mismo:

```
VARIABLE numérico tam
LEER tam
DIMENSIONAR numérico vector4(tam)
PARA i DESDE 1 HASTA tam HACER
    LEER vector4[i]
FIN PARA
```



# Arreglos bidimensionales

- Conocida como **matriz**.
- Requieren dos índices o parámetros para acceder a sus elementos (**fila** y **columna**).
- Utilizar dos estructuras *PARA... FIN PARA* anidadas para recorrer todos los elementos de la matriz:

```
DIMENSIONAR numérico matriz1(3, 4)
PARA i DESDE 1 HASTA 3 HACER
  PARA j DESDE 1 HASTA 4 HACER
    matriz1[i, j] <- i * j
  FIN PARA
FIN PARA
```

# Arreglos bidimensionales

	Columna 1 (j = 1)	Columna 2 (j = 2)	Columna 3 (j = 3)	Columna 4 (j = 4)
Fila 1 (i = 1)	1	2	3	4
Fila 2 (i = 2)	2	4	6	8
Fila 3 (i = 3)	3	6	9	12

The diagram illustrates a 3x4 matrix with values and arrows indicating row and column indices. The matrix is organized into three rows and four columns. The columns are labeled 'Columna 1 (j = 1)', 'Columna 2 (j = 2)', 'Columna 3 (j = 3)', and 'Columna 4 (j = 4)'. The rows are labeled 'Fila 1 (i = 1)', 'Fila 2 (i = 2)', and 'Fila 3 (i = 3)'. The values in the matrix are: Row 1: 1, 2, 3, 4; Row 2: 2, 4, 6, 8; Row 3: 3, 6, 9, 12. Yellow arrows point from the row labels to the corresponding rows and from the column labels to the corresponding columns. Additionally, yellow arrows point from the first cell of each row to the last cell of the same row, and from the last cell of each row to the first cell of the next row, illustrating a row-major traversal pattern.

Figure 4: Ejemplo: matriz1

# Arreglos bidimensionales

- En el ejemplo anterior los valores fueron asignados recorriendo la matriz por filas.
- Otra forma es recorrer la matriz por columna.
- La estructura *PARA... FIN PARA* que representa a los índices de las columnas debe ser la externa y la que representa a los índices de las filas, la interna:

```
DIMENSIONAR numérico matriz1(3, 4)
PARA j DESDE 1 HASTA 4 HACER
  PARA i DESDE 1 HASTA 3 HACER
    matriz1[i, j] <- i * j
  FIN PARA
FIN PARA
```

# Arreglos bidimensionales

	Columna 1 (j = 1)	Columna 2 (j = 2)	Columna 3 (j = 3)	Columna 4 (j = 4)
Fila 1 (i = 1)	1	2	3	4
Fila 2 (i = 2)	2	4	6	8
Fila 3 (i = 3)	3	6	9	12

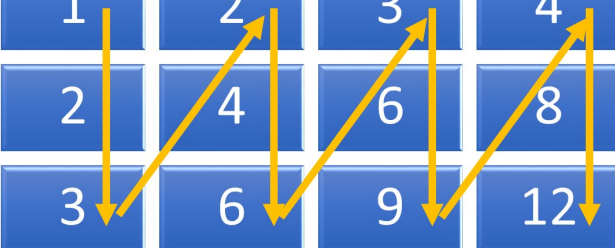


Figure 5: Ejemplo: matriz1 recorrida por columnas

# Arreglos multidimensionales

- Tiene más de dos dimensiones.
- No tan usados como los vectores y matrices.
- La representación matemática o visual ya no es tan sencilla.
- Ejemplo: se desea contar el número de autos que ingresaron a una playa de estacionamiento por hora a lo largo de varios años.
- Se puede emplear un arreglo `numeroAutos`, donde la 1° dimensión indique el año, la 2° el mes, la 3° el día y la 4° la hora.
- El elemento `numeroAutos[2, 4, 23, 14]` contendrá el número de autos que ingresaron a la hora 14, del día 23, en el mes 4 del segundo año.

# Ejemplo: invertir (dar vuelta) los elementos de un vector

Por ejemplo, dado el vector  $v$ :



Figure 6: Vector  $v$  original

Queremos modificarlo para obtener:



Figure 7: Vector  $v$  reordenado

# Ejemplo: invertir (dar vuelta) los elementos de un vector

- Intercambiar de a dos los valores en ciertas posiciones del vector, por ejemplo, el primero y el último.
- Emplear una variable auxiliar para guardar temporalmente el valor de alguna de las celdas:



Figure 8: Vector v reordenado

# Ejemplo: invertir (dar vuelta) los elementos de un vector

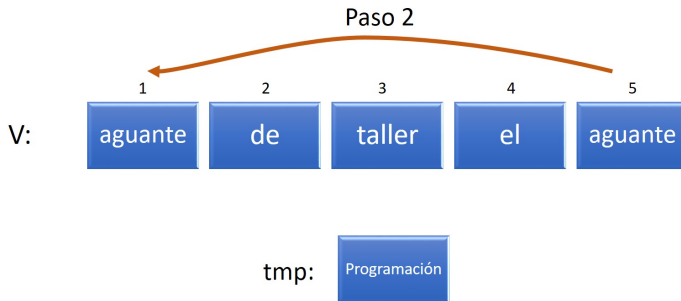


Figure 9: Vector v reordenado



## Ejemplo: invertir (dar vuelta) los elementos de un vector

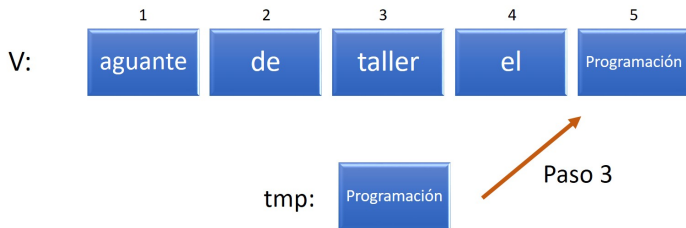


Figure 10: Vector v reordenado

- Sólo falta intercambiar las posiciones 2 y 4.
- Como el número de elementos en el vector es impar, el valor en la posición central queda allí.

# Ejemplo: invertir (dar vuelta) los elementos de un vector

El algoritmo general es:

```
ALGORITMO: "Invertir (dar vuelta)
           los elementos de un vector"
```

```
COMENZAR
```

```
  \\ Declarar variables
```

```
  VARIABLE numérico n
```

```
  LEER n
```

```
  DIMENSIONAR caracter v(n)
```

```
  \\ Asignar valores al vector
```

```
  PARA i DESDE 1 HASTA n HACER
```

```
    LEER v[i]
```

```
  FIN PARA
```

```
  \\ Reordenar
```

```
  PARA i DESDE 1 HASTA ENTERO(n / 2) HACER
```

```
    tmp <- v[i]                \\ Paso 1
```

```
    v[i] <- v[n - i + 1]       \\ Paso 2
```

```
    v[n - i + 1] <- tmp        \\ Paso 3
```

```
  FIN PARA
```

```
  \\ Mostrar el vector reordenado
```

```
  PARA i DESDE 1 HASTA n HACER
```

```
    ESCRIBIR v[i]
```

```
  FIN PARA
```

```
FIN
```

## Ejemplo: invertir (dar vuelta) los elementos de un vector

- El código correspondiente en SAS/IML es:

```
/* Invertir (dar vuelta) un vector */  
proc iml;  
    v = {"programacion" "de" "taller" "el" "aguante"};  
    n = ncol(v);  
    do i = 1 to int(n / 2);  
        tmp = v[i];  
        v[i] = v[n - i + 1];  
        v[n - i + 1] = tmp;  
    end;  
    print v;  
quit;
```