El lenguaje de programación C
- Vectores y matrices -

Isidro González Caballero
(gonzalezisidro@uniovi.es)

7

Introducción a la Física Computacional

Curso 2010 - 2011

Universidad de Oviedo

Introducción

- En C a los vectores también se les llama *arrays* o *arreglos*
- Las matrices serán vectores de vectores → Ver más adelante
- Los arrays son
  - Conjuntos de variables del mismo tipo…
  - … que tienen el mismo nombre…
  - … y se diferencian en el índice
- Es un modo de manejar una gran cantidad de datos del mismo tipo bajo un mismo nombre o identificador
- Para realizar operaciones matemáticas sobre un array (como en Matlab) debemos operar sobre cada elemento del array

Declaración de un array

```
tipo nombre[tamaño];
```

- *tipo*: Tipo que tienen el conjunto de variables
  → `int`, `float`, `double`, ...
- *nombre*: Identificador con el nombre del array
- *tamaño*: Cantidad de espacios de memoria que queremos reservar para este array
  - **Importante**: Debe ser un entero constante!!!
    Conocido en tiempo de compilación

Inicialización de un array

```
tipo nombre[tamaño] = {a, b, c,…};
```

- El número de valores entre llaves tiene que ser menor o igual al *tamaño*
  - Si es menor el resto de los valores se quedan sin inicializar
- Existe una forma alternativa:

```
tipo nombre[] = {a, b, c,…};
```

  - Si no se especifica el tamaño se reservarán tantos espacios como elementos haya entre llaves

Acceso a los elementos de un array

- Para un array de tamaño ***N*** y nombre ***V*** accederemos al elemento ***i*** como ***V[i]***
- Ese valor puede ser leído (imprimido, asignado a otra variable,…) o modificado (dándole un valor) como cualquier otra variable

```
V[i] = valor;
```

- **Importante**: Los índices en los arrays de C van desde 0 hasta (N-1)

Ejemplo

```c
/* Usando un array de enteros */
int main() {
  /* Declarando el valor del array */
  int losnumeros[10];
  int i = 0;
  /* Modificando el valor del array */
  for (i = 0; i < 10; i++)
    losnumeros[i] = i;
  /* Imprimiendo el valor del array */
  for (i = 0; i < 10; i++)
    printf("El elemento %d vale %d\n", i, losnumeros[i]);
  return 0;
}
```

```
El elemento 0 vale 0
El elemento 1 vale 1
El elemento 2 vale 2
…
```

Ejemplo… con truco

```c
#define MYSIZE 10
int main() {
  /* Declarando el valor del array */
  int losnumeros[MYSIZE];
  int i = 0;
  /* Modificando el valor del array */
  for (i = 0; i < MYSIZE; i++)
    losnumeros[i] = i;
  /* Imprimiendo el valor del array */
  for (i = 0; i < MYSIZE; i++)
    printf("El elemento %d vale %d\n", i, losnumeros[i]);
  return 0;
}
```

```
El elemento 0 vale 0
El elemento 1 vale 1
El elemento 2 vale 2
…
```

Arrays como argumentos de funciones

- Para pasar un array a una función no hace falta especificar su tamaño:

```
retorno nombrefuncion(tipo array[],…);
```

- Los arrays en C están íntimamente relacionados con los punteros → Lo veremos más adelante
  - La sintaxis general para usar arrays como argumentos de funciones será

```
retorno nombrefuncion(tipo *array,…);
```

  - **Importante**: En este formato, si se modifican valores del array dentro de la función, estas se propagan fuera → Paso de argumentos por referencia

Ejemplo función

```c
#define MYSIZE 10
void imprime_array_enteros(int V[], int size);

int main() {
  /* Declarando el valor del array */
  int losnumeros[MYSIZE];
  int i = 0;
  /* Modificando el valor del array */
  for (i = 0; i < MYSIZE; i++)
    losnumeros[i] = i;
  imprime_array_enteros(losnumeros, MYSIZE);
  return 0;
}

/* Imprime el valor del array V de tamaño size*/
void imprime_array_enteros(int V[], int size) {
int i = 0;
for (i = 0; i < size; i++)
    printf("El elemento %d vale %d\n", i, V[i]);
}
```

Arrays multidimensionales

- Declaración

```
tipo nombre[N1][N2][…];
```

- Utilización:

```
Nombre[i1][i2]
```

  - 0 < i1 < N1
  - 0 < i2 < N2
  - …

- Funciones

```
retorno nombrefuncion(tipo array[][N2],…);
```