Introducción a la Programación en C
–Arrays–

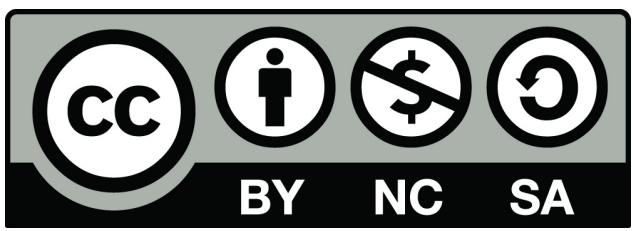
Christopher Expósito-Izquierdo
cexposit@ull.edu.es

Airam Expósito-Márquez
aexposim@ull.edu.es

Israel López-Plata
ilopezpl@ull.edu.es

Belén Melián-Batista
mbmelian@ull.edu.es

José Marcos Moreno-Vega
jmmoreno@ull.edu.es



Contenidos

Introducción:

Array

Es una entidad que representa a un conjunto de datos de un tipo conocido, es accesible a través de un índice y tiene un cierto identificador

	0	1	2	3	4	5	6	7	8	9
miPrimerArray:	7	104	-2	9	61	34	0	-5	-89	92

Introducción:

Cuándo usamos un array:

Cuando existe la necesidad de trabajar con múltiples datos de un **mismo tipo**. Ejemplos:

- Almacenar la temperatura media de cada mes del año
- Guardar los números de teléfono de nuestros amigos
- Registrar las mediciones obtenidas de un cierto experimento
- ...

Definición:

Un array se define tal como sigue:

```
tipoDato identificadorArray[númeroElementos]
```

- `tipoDato`: es el tipo de datos de cada elemento
- `identificadorArray`: es el identificador de la variable utilizada para el array
- `númeroElementos`: es el número de elementos que componen el array. Debe ser una expresión constante entera

Ejemplos:

- `int edades[30];`
- `float temperaturas[12];`
- `char texto[50];`
- ...

Definición:

Es común el uso de macros para acotar el máximo número de elementos a almacenar

```
#include <stdio.h>

#define ALUMNOS 30
#define MESES 12
#define LONGITUD_TEXTO 50

int main(void) {
        int edades[ALUMNOS];
        float temperaturas[MESES];
        char texto[LONGITUD_TEXTO];
        ...
        return 0;
}
```

Acceso a Elementos:

El acceso a los elementos de un array se realiza como sigue:

`identificadorArray[expresión]`

- `identificadorArray`: es el identificador de la variable utilizada para el array
- `[]`: es el operador de acceso a los elementos del array
- `expresión`: es un índice del array. Debe ser entero

Ejemplos:

- `edades[4];`
- `temperaturas[7];`
- `texto[14];`
- ...

Acceso a Elementos:

¿Diferencia entre `edades` y `edades[4]`?

- `edades`: denota a un array
- `edades[4]`: denota a un elemento individual del array

Acceso a Elementos:

¿Diferencia entre `edades` y `edades[4]`?

- `edades`: denota a un array
- `edades[4]`: denota a un elemento individual del array

```
int edad = edades[4];
int edadFuturo = edades[4] + 10;
int dobleEdad = edades[4] * 2;
int edadPasado = edades[4] - 1;
int sumaEdades = edades[4] + edades[13];
int edad = edades[8 * 2];
int edad = edades[i + 9 - 4];
...
```

Acceso a Elementos:

¿Diferencia entre `edades` y `edades[4]`?

- `edades`: denota a un array
- `edades[4]`: denota a un elemento individual del array

- `int edad = edades[4];`
- `int edadFuturo = edades[4] + 10;`
- `int dobleEdad = edades[4] * 2;`
- `int edadPasado = edades[4] - 1;`
- `int sumaEdades = edades[4] + edades[13];`
- `int edad = edades[8 * 2];`
- `int edad = edades[i + 9 - 4];`
- ...

Acceso a Elementos:

A cada elemento de un array se accede a través de un índice:

- El índice inferior, el primero, es siempre 0
- El índice superior, el último, corresponde con `númeroElementos-1`

El intento de acceso a elementos fuera del rango permitido da lugar a (i) comportamientos inesperados en el programa en ejecución o (ii) a un error en tiempo de ejecución (*segmentation fault*)

Inicialización:

Un array se puede inicializar como una lista de valores:

- `int array1[] = {1, 2, 3, 4};`
- `char array2[] = {'a', 'b', 'c', 'd'};`
- `int array3[4] = {1, 2, 3, 4};`
- `int array4[4] = {0, 0, 0, 0};`
- `int array5[4] = {1, 2};`
- `int array6[4] = {1, 2, 0, 0};`

Inicialización:

```
/*
   Inicializa un array de 1000 alturas y lo muestra por pantalla
 */
#include <stdio.h>

#define ALTURAS 1000

int main(void) {
        int alturas[ALTURAS];
        int i;
        for (i = 0; i < ALTURAS; i++) {
                alturas[i] = i;
        }
        for (i = 0; i < ALTURAS; i++) {
                printf("%d\n", alturas[i]);
        }
        return 0;
}
```

Arrays Multidimensionales:

Un array multidimensional se define como sigue:

```
tipoDato identificador[elementos1][elementos2]...[elementosN]
```

- `tipoDato`: es el tipo de datos de cada elemento
- `identificador`: es el identificador de la variable utilizada para el array
- `elementos1`: indica el número de elementos de la primera dimensión del array
- `elementos2`: indica el número de elementos de la segunda dimensión del array
- `...`
- `elementosN`: indica el número de elementos de la n-ésima dimensión del array

Arrays Multidimensionales:

Ejemplos:

- `int matrizCuadrada[5][5];`
- `int matriz[10][20][15];`
- `int dimensiones[100][3];`
- `...`

Inicialización:

```
int matrix[3][4] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12}
};
```

Strings:

String

Es una secuencia finita de caracteres consecutivos

- Un *string* o array de caracteres se puede inicializar en el momento de su declaración con una cadena literal. Dicha cadena es una secuencia de caracteres encerrada entre dobles comillas

- Las cadenas literales acaban con el carácter nulo `'\0'`. Por tanto, la cadena `"Primero"` tiene longitud 8

```
char curso[8] = "Primero";
```

y entonces en `curso[0]` está `'P'`, en `curso[1]` está `'r'`,... en `curso[6]` está `'o'` y en `curso[7]` está `'\0'`

Strings: Caracteres Especiales

Los strings o arrays de caracteres pueden contener unos caracteres especiales (secuencias de escape) que tienen un efecto determinado:

- \n
- \t
- \v (tabulación vertical)
- \b (retrocede un espacio)
- \r (va al comienzo de la línea)
- \a (sonido de alerta del sistema)
- \'
- \"
- \\
- \? (para símbolos especiales)

Errores Habituales:

- **Olvidar que el primer índice de un array es 0**. El primer índice de un array es 0. En muchas ocasiones se comete el error de creer que es `1`
- **Acceder a un array fuera de rango**. Cuando se accede a los elementos de un array no se hace ninguna comprobación. Suele ser un error frecuente acceder a posiciones que no pertencen al array. Hay que asegurarse de que siempre se utiliza como índice un valor comprendido entre `0` y `numElementos - 1`
- **Utilizar como índice un valor que no sea entero**. Los índices que se utilizan para acceder deben ser de tipo entero
- **Acceder a un array multidimensional de forma inadecuada**. Cuando se quiere acceder a la componente `i, j` de un array `m`, no se debe emplear la expresión `m[i, j]`, sino `m[i][j]`

Introducción a la Programación en C
–Arrays–

Christopher Expósito-Izquierdo
cexposit@ull.edu.es

Airam Expósito-Márquez
aexposim@ull.edu.es

Israel López-Plata
ilopezpl@ull.edu.es

Belén Melián-Batista
mbmelian@ull.edu.es

José Marcos Moreno-Vega
jmmoreno@ull.edu.es