## OBJETIVOS

- Trabajar con cadenas de caracteres en lenguaje C.
- Aprender el uso de arrays.
- Aprender a usar parámetros de entrada/salida.

## PARTE 1: EJERCICIOS SOBRE MANEJO DE CADENAS

Como ya deberá saber, en C no existe un tipo básico "String"; sin embargo el uso de arrays de elementos de tipo char permite manejar cadenas. De hecho existe una librería C denominada *string.h* que incluye un conjunto de funciones que permite el manejo de cadenas y que usaremos más adelante.

Estas funciones de manejo de cadenas presuponen que cualquier cadena en C está almacenada en un array de tipo char y que en la posición inmediatamente siguiente al último carácter de la cadena almacenada existe un carácter especial denominado fin de cadena (el carácter '\0'). Por ejemplo, la cadena: "buenos días", estaría almacenada en un array de 15 posiciones definido como: "char cadena[15]", de la siguiente manera:

| b | u | e | n | o | s |  | d | í | a | s | \0 |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

La función *strlen(cadena)*, que permite saber la longitud de la cadena, devolverá 11 (lo que mide "buenos días") y no 15 (lo que se reservó para la variable cadena).

### EJERCICIO 1– BUSCANDO UN CARÁCTER

- Cree un proyecto que se denominará **Cadenas**.
- En un archivo de cabecera denominado **cadenas.h** deberá:
  - Incluir las librerías de manejo de funciones de entrada/salida y de manejo de cadenas.
  - Definir una constante N con valor 129.
  - Definir dos tipos: un enumerado denominado Logico con valores FALSO y CIERTO y otro denominado Cadena para almacenar cadenas de hasta 128 caracteres.
  - Incluir, a medida que lo necesite, las funciones que se le piden en el ejercicio.
- En un archivo denominado **cadenas.c** implemente la función *int buscaCaracter (**const** Cadena, char)* cuya funcionalidad es recorrer la cadena comparando cada elemento con el carácter dado y devolver la posición de la primera aparición de éste (la primera posición será la cero). En caso de no encontrarlo devolverá -1.
- En un archivo **testCadenas.c** codifique:
  a. La función *int main(void)* que invocará a una función *void testBuscaCaracter()*;
  b. La función *testBuscaCaracter()*, que pedirá por teclado una cadena de caracteres y un carácter y visualizará un mensaje con la posición de la cadena donde se encuentra el carácter o, en su caso, el mensaje "Carácter no encontrado".
     Nota: utilice las funciones *gets(cad)* para pedir la cadena y *c=getchar()* para pedir el carácter, ya que la función scanf no reconoce un carácter que se teclee después de cualquier otro valor (asume como dato introducido para el carácter *el blanco, tabulador o el retorno de carro* con el que se separa el carácter del valor introducido anteriormente).

### EJERCICIO 2 – PALABRA O FRASE CON LA CONDICIÓN DE PALÍNDROMO

Se dice que una palabra, frase, secuencia de caracteres o de números es un palíndromo cuando se lee igual de izquierda a derecha, que de derecha a izquierda.

- En el proyecto del ejercicio anterior,
  a) Cree una función *Logico esPalindromo (**const** Cadena)* que devolverá FALSO o CIERTO según resulte ser, o no, un palíndromo la secuencia que reciba como parámetro.
  b) Para probarlo realice las modificaciones oportunas a los archivos *testCadenas.c y cadenas.h*

## PARTE 2: ARRAYS Y PARÁMETROS DE E/S

Cree un proyecto denominado Vectores que contendrá los archivos de código fuente que denominará **vectores.c** y **testVectores.c**, y un archivo de cabecera **vectores.h** en el que entre otras cosas deberá definir:

Una constante TAM_VECTOR con valor 20, y un tipo Vector de dimensión TAM_VECTOR para almacenar valores de tipo double.

También deberá ir añadiendo los prototipos de los ejercicios que se irán realizando a continuación.

Además en los ficheros vectores.c y testVectores.c deberá incluir "vectores.h".

### EJERCICIO 3 – LEYENDO Y ESCRIBIENDO VECTORES

Escriba una función *leeVector* para leer por teclado los elementos de un Vector. Su prototipo será

```
void leeVector(Vector, int);
```

Esta función irá visualizando el texto "Elemento[i]:", donde i es el índice de que se trate, e inmediatamente se introducirá por teclado el correspondiente valor que se irá almacenando en el vector.

El segundo argumento de tipo *int* es un valor que se introducirá en el test y que permitirá saber cuántos elementos se almacenarán en el vector (elementos reales del vector). Comprobar que dicho valor no sea mayor que la dimensión del vector.

***Es importante recordar que el nombre de un array es la dirección donde comienza el array, pero C no sabe dónde termina, por lo que es imprescindible indicar siempre el número de elementos reales del array a tratar***.

¿Qué pasaría si se invoca a leeVector para leer más de TAM_VECTOR elementos?

Escriba una función *escribeVector* para mostrar los elementos del vector por pantalla entre corchetes "[...]" y separados por coma (,). El prototipo será:

```
void escribeVector(const Vector, int);
```

La cláusula *const* indica que el Vector es solo argumento de entrada. Recuerde que tanto para leer como para escribir, el especificador de formato de la cadena de control de las funciones scanf y printf para un tipo double es %lf.

**EJERCICIO 4 – MEDIA ARITMÉTICA**

Añada al proyecto Vectores una función `mediaAritmetica` que reciba como parámetro el vector del ejercicio anterior, y devuelva la media aritmética de sus elementos. Compruebe su ejecución.

**EJERCICIO 5 – PRODUCTO ESCALAR**

Añada al proyecto Vectores una función `productoEscalar` que reciba dos vectores de igual tamaño y devuelva el producto escalar de ambos:

$$(a_1, a_2, a_3, ..., a_n) \cdot (b_1, b_2, b_3, ..., b_n) = a_1 b_1 + a_2 b_2 + ...a_n b_n = \sum a_i \cdot b_i$$

Compruebe su ejecución.

## PARTE 3: EJERCICIOS PARA HACER EN CASA

**EJERCICIO 6 – CALCULANDO EL DÍGITO DE CONTROL DE UN ISBN**

El **International Standard Book Number** (en español, 'número estándar internacional de libro'), abreviado **ISBN**, es un identificador único para cualquier libro cuyo uso se prevea de carácter comercial (ya en el curso se ha utilizado este identificador).

Escriba dentro del proyecto Cadenas, siguiendo el esquema de los ejercicios anteriores, las funciones necesarias para que se lean de teclado el código de grupo, el del editor y el del libro (9 dígitos en total) y visualice una cadena con la concatenación de todo el ISBN (10 dígitos en total).

El prototipo de la función que calcula el dígito de control es:

*char digitoControlISBN(const Cadena);*

Esta función debería comprobar si la cadena de entrada tiene (en este ejercicio) 9 caracteres. En caso contrario la función devolverá el carácter '\0'.

El formato de un código ISBN de 10 dígitos (los hay de trece) es el siguiente:

- Código de grupo (1 dígito)
- Código del editor (4 dígitos)
- Código del libro (4 dígitos)
- Dígito de control (1 dígito)

**Por ejemplo**, el código ISBN para un determinado libro es 0675209935. El carácter o dígito de control (5 en el ejemplo) se obtiene en dos pasos:

1. Se multiplica cada dígito por el índice correspondiente a su posición, y se suman los números resultantes. En el ejemplo:

```
  0   6   7   5   2   0   9   9   3
x 1   2   3   4   5   6   7   8   9
-------------------------------------------------
 0 + 12 + 21 + 20 + 10 +  0 + 63 + 72 + 27 = 225
```

2. Se divide la suma por 11, y se toma el resto como dígito de control, teniendo en cuenta que si el resto es 10, se utiliza el carácter X como "dígito" de control. En el ejemplo, el resto de la división entera de 225 por 11 es 5, que es el dígito de control.

**Importante**: observe que la función *char digitoControlISBN(const Cadena)* devuelve un tipo char, por lo que necesitará, de un lado, pasar a int los valores char que forman la cadena de entrada y, de otro, una vez calculado el dígito de control, pasarlo a char para que se pueda devolver en el return. Apóyese en que:

- `(int)(carácter-'0')` convierte un dato de tipo carácter, que representa un número entero, en ese número como tipo de dato entero.
- `(char)(entero+'0')` convierte un entero entre 0 y 9 en su representación como carácter.

## EJERCICIO 7 – VECTOR CON LAS POSICIONES DONDE APARECE UN CARÁCTER EN UNA CADENA

Añada una nueva funcionalidad al proyecto Cadenas mediante una nueva función.

Defina *Pint* como un nuevo tipo "puntero a entero" int*, y otro tipo Vector para almacenar hasta 128 números enteros.

Implemente la función *void posicionesDeCaracter(**const** Cadena, char , Vector, Pint),* que recibiendo como parámetros de entrada una Cadena y un carácter, devuelva como parámetros de salida un vector con todas las posiciones donde se encuentra el carácter dentro de la cadena y el tamaño real de dicho vector.

**Importante**:

1. Las posiciones deberán estar ajustadas a las primeras posiciones del array (como si de una lista Java se tratase, sin huecos en medio); es por ello que cuando el método haya "cargado" el vector, también debe actualizar el tamaño "real" del vector.
2. Recuerde que para manejar el valor de un puntero, por ejemplo, asignar el valor 5 a un puntero p, se haría: *p=5; o para sumar una unidad positiva se haría: (*p)++;
3. Un ejemplo de invocación a posicionesDeCaracter será: *posicionesDeCaracter(cad,c,v,&n)*

Compruebe el resultado mediante el oportuno test.

## EJERCICIO 8– ELEMENTOS MAYORES

Escriba en el proyecto Vectores una función `mayoresVector` que reciba como parámetros de entrada un Vector v de "doubles", su respectivo número real de elementos y un valor double "x", y devuelva como parámetros de salida un Vector w con todos los elementos de v mayores o iguales a x y el número real de elementos que se han incluido en w; para ello, recuerde que deberá definirlo como puntero a entero y usar los operadores adecuados para su manejo.

Compruebe el resultado utilizando la función `escribeVector`.