**Cadenas de caracteres**

En C no existe un tipo predefinido para manipular cadenas de caracteres (string). Sin embargo, el estándar de C define algunas funciones de biblioteca para tratamiento de cadenas. Una cadena en C es un array de caracteres de una dimensión (vector de caracteres) que termina con el carácter especial ‘\0’ (cero).

El formato para declarar una cadena es: char nombre[n]; donde: n >= 1 y representa a la longitud-1 real de la cadena. Un ejemplo de declaración de cadena: char cadena [5]; Debido a que en la representación interna de una cadena de caracteres es terminada por el símbolo '\0', para un texto de "n" caracteres, debemos reservar "n+1”. El carácter '\0', aunque pertenece a la cadena, no aparece al utilizar funciones como cout o printf.

En el caso especial de los arrays de caracteres, podemos utilizar varias formas de inicialización:

**char cadena[] = "Hola";**

**char cadena[] = {'H','o','l','a',0};**

**char cadena[] = {'H','o','l','a','\0'};**

sin especificar el tamaño de la cadena,

o especificando el tamaño:

**char cadena[5] = "Hola";**

**char cadena[5] = {'H','o','l','a',0};**

**char cadena[5] = {'H','o','l','a','\0'};**

Durante la inicialización, se reserva automáticamente el número de bytes necesarios para la cadena, esto es, el número de caracteres más uno. Por ejemplo:

**char texto [] = "Hola";**

**texto vale**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **‘h’** | **‘o’** | **‘l’** | **‘a’** | **‘\0’** |

Para acceder a un elemento de una cadena de caracteres puede hacerse de la misma manera que el acceso al elemento de un array. cadena[i]; donde: 0 <=i < n

Por ejemplo:

**char c[8] = "Ejemplo";**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **‘E’** | **‘j’** | **‘e’** | **‘m’** | **‘p’** | **‘l’** | **‘o’** | **‘\0’** |

**c[0] c[1] c[2] c[3] c[4] c[5] c[6] c[7]**

**c[1] vale ‘j’**

**c[4] vale ‘p’**

La biblioteca “**string**” tiene una gran cantidad de funciones prácticas para trabajar con cadenas de caracteres. Para utilizarlas debemos de incluir el fichero que define los prototipos de dichas funciones:

**#include <string.h>**

Algunas de las funciones más importantes son:

• **strlen**(<cadena>): Devuelve la longitud de la cadena sin tomar en cuenta el caracter de final de cadena.

• **strcpy**(<cadena\_destino>, <cadena\_origen>) : Copia el contenido de <cadena\_origen> en <cadena\_destino>.

• **strcat**(<cadena\_destino>, <cadena\_origen>) : Concatena el contenido de <cadena\_origen> al final de <cadena\_destino>.

• **strcmp**(<cadena1>, <cadena2>) : Compara las dos cadenas y devuelve un 0 si las dos cadenas son iguales, un número negativo si <cadena1> es menor que (precede alfabéticamente a) <cadena2> y un número positivo (mayor que cero) si <cadena1> es mayor que <cadena2>.

A diferencia de los arrays de tipos de datos numéricos (arrays de enteros, de números con punto decimal, etc.), en donde cada elemento del array se debe considerar como una variable independiente de los demás, los arrays de caracteres (cadenas) se pueden manipular de dos maneras: de forma conjunta o separada.

Por ejemplo, para mostrar en pantalla un array de caracteres podemos hacerlo dentro de un bucle, desde el primer caracter (indice 0) hasta el último carácter (lo que nos devuelve la función strlen):

for(i=0; i<strlen(cadena); i++)

cout << cadena[i]

o podemos directamente mostrar toda la cadena en un cout

cout << cadena

En el caso de utilizar printf, para mostrarla directamente evitando el caracter a caracter, se debe utilizar el carácter de conversión %s:

printf(“%s”, cadena)