# UNIDAD 4.- CADENAS DE CARACTERES

**OBJETIVOS:**

- Que el alumno aprenda a realizar correctamente las funciones en un lenguaje de programación.
- Que el alumno utilice adecuadamente los tipos de funciones en lenguaje C.

**TEMAS:**

**4.1.** Definición.

**4.2.** Lectura e impresión. Representación en lenguaje C.

**4.3.** Funciones predefinidas en el lenguaje C.

## 4.1. DEFINICIÓN DE CADENAS DE CARACTERES

En cualquier lenguaje de programación las cadenas de caracteres tienen una importancia especial, no solo porque es el tipo mediante el cual se almacenan los mensajes a pantalla o entradas del teclado, sino porque un caracter (char) es del tamaño de un byte, y un byte es la menor unidad de información 'natural' para la máquina. Una cadena de caracteres es una colección ordenada de bytes.

Hay lenguajes que tienen un tipo (type) preestablecido para tratar con cadenas de caracteres, es así en las distintas versiones de basic, donde el tipo 'String' es un tipo más (como 'integer'), En cambio en C/C++ podría decirse que el tipo 'String' no existe como tal (en C++ hay implementaciones de la clase 'String' pero no pertenecen propiamente al lenguaje sino a librerías anexas, dependiendo de cada compilador), el recurso usado por la mayoría de las funciones de estos lenguajes es representar una cadena de caracteres como un array de elementos tipo char, un array de caracteres.

Una cadena de caracteres, representada en memoria, es una simple sucesión de bytes, cada caracter se corresponde con un byte, si queremos sacar en pantalla una cadena de caracteres, el problema es: ¿cómo sabe el programa donde finaliza esa cadena?, ¿cuál es su ultimo byte? A esta pregunta diferentes lenguajes plantean diferentes respuestas, según un modelo de 'string', en C se reserva una función especial al caracter cuyo valor ASCII es 0 (cero), ese caracter indicara con su presencia que la cadena finaliza allí, ese char forma parte de la cadena pero por convención no se lo tiene en cuenta al determinar el *largo* de la cadena. De una cadena como "hola" se dice que tiene 4 caracteres, aunque en memoria luego de 'a' se encuentre el '\0' que forma parte de ella.

Una cadena en C es un conjunto de caracteres, o valores de tipo "char", terminados con el carácter nulo, es decir el valor numérico 0. Internamente se almacenan en posiciones consecutivas de memoria. Este tipo de estructuras recibe un tratamiento especial, y es de gran utilidad y de uso continuo.

En C y C++ las cadenas de caracteres no son más que arreglos de caracteres, salvo que a este tipo de arreglos el compilador les da un tratamiento especial. Se puede manipular las cadenas de caracteres de la misma manera en que manipula cualquier otro tipo de arreglo, sin embargo, es preferible hacer uso de una librería estándar especialmente escrita para manipulación de cadenas de caracteres, me refiero a la librería <string.h> y que viene incluida con todo compilador de C o C++.

También se puede ver una cadena de caracteres como un vector de caracteres que ocupa una porción de memoria RAM conteniendo caracteres ASCII. En C y C++ una cadena es

estrictamente una secuencia de cero o más caracteres seguidas por un carácter NULL cuyo código es cero y se puede escribir como todo código ASCII en octal precedido de una barra: '\0'

Suponga por ejemplo que se almacena en una cadena de longitud máxima 10, la secuencia "EJEMPLO 1":

| E | J | E | M | P | L | O | \40 | 1 | \0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Observe que las posiciones van desde 0 a 9, también fíjese que el espacio en blanco (cuyo código ASCII en decimal es 32 y en octal es 40) ocupa un lugar y que al final se encuentra el carácter NULL.

Cuando se manejan las cadenas como simples arreglos de caracteres, es importante preservar el carácter de terminación NULL, ya que con éste es como C define y maneja las longitudes de las cadenas. Todas las funciones de la biblioteca estándar de C lo requieren para una operación satisfactoria. En cambio cuando manipulamos las cadenas con las funciones especiales de dicha biblioteca, nos despreocupamos del carácter NULL, ya que automáticamente se coloca al final de la cadena resultante.

Por lo tanto se recomienda usar las funciones para manejo de cadenas y no tratar de manipular las cadenas en forma manual. El siguiente ejemplo realiza declaración de cadenas y asignaciones de valores, para aclarar algunos detalles:

```
char  CAD1 [ ] = "FACULTAD";   // manipulación como cadena

typedef  char cadena8[9];

cadena50 CAD2 = {'F' , 'A' , 'C' , 'U' , 'L' , 'T' , 'A' , 'D' , ' \0'} , CAD3;    // como vector

……..

CAD3 = "FACULTAD";    // es incorrecto solo se puede asignar de esta forma en la declaración
```

La variable CAD1 se declara como un vector de tipo char sin longitud máxima, pero se le dá un valor inicial "FACULTAD", como cada elemento char ocupa un byte la variable CAD1 ocupará 9 byte en memoria, ya que alberga una cadena de 8 caracteres y en la última posición automáticamente se guarda el carácter NULL.

Observe como se definió CAD2 y CAD3, posterior a una declaración de tipo que hace más eficiente y transparente esta declaración; también se le da a CAD2 un valor inicial igual que

a CAD1, pero como vector, asignando uno por uno los valores de cada posición tipo carácter –entre comillas simples- inclusive el carácter '\0'. Tenga en cuenta que es incorrecto asignarle a CAD3 una cadena entre comillas dobles, esto es solo posible en la declaración de la variable.

Ambas formas de dar un valor inicial a una cadena son equivalentes, pero es obvio que asignarle un valor entre comillas dobles, como cadena, es mucho más práctico. Sin embargo la segunda forma puede ser interesante cuando se asigna uno o varios caracteres en forma directa indicando la posición, por ejemplo cambiar la secuencia "FACULTAD" por "FACILITA", en CAD1, entonces resulta más eficiente hacer una asignación directa, ya que hacer lo mismo con funciones especiales resulta complicado:

```
CAD1[3] = CAD1[5] = 'I' ;

CAD1[6] = 'T' ;

CAD1[7] = 'A' ;
```

En este caso se asigna valores tipo char a elementos del arreglo, cosa que es perfectamente válida.

**Array y cadena de caracteres: algunas diferencias**

En muchos textos sobre C y C++ se encuentra la siguiente afirmación: "En C/C++ una cadena de caracteres (string) es un array de caracteres" es claro en qué sentido esta afirmación es válida, una cadena de caracteres es un array y no una variable de tipo individual, como un entero o un float, sin embargo, hay ligeras diferencias entre los dos conceptos, el de array y el de cadena de caracteres.

Mientras que un **array de caracteres:** es un conjunto ordenado de 'n' bytes de cualquier valor, una **cadena de caracteres** es solo un subconjunto de ese array, desde el primer char hasta el primer '\0' encontrado en el array.

La manera de definir una cadena es la siguiente:

**char <identificador> [<longitud máxima>];**

Cuando se declara una cadena hay que tener en cuenta que tendremos que reservar una posición para almacenar el carácter nulo, de modo que si queremos almacenar la cadena "FRT", tendremos que declarar la cadena como:

**char cadena[4];**

Tres caracteres para "FRT" y uno extra para el carácter '\0'. Observe que no es preciso poner el carácter '\0' (se añade por defecto)

También nos será posible hacer referencia a cada uno de los caracteres individuales que componen la cadena, simplemente indicando la posición. Por ejemplo el tercer carácter de nuestra cadena de ejemplo será la 'T', podemos hacer referencia a él como cadena[2]. Los índices tomarán valores empezando en el cero, así el primer carácter de nuestra cadena sería cadena[0], que es la 'F'.

Una cadena puede almacenar informaciones como nombres de personas, mensajes de error, números de teléfono, etc.

La asignación directa sólo está permitida cuando se hace junto con la declaración. Por ejemplo:

**char cadena[4] = “FRT”;**

## 4.2. LECTURA E IMPRESIÓN

Para realizar la lectura e impresión de una cadena se puede recurrir a las siguientes funciones :

1) Para leer una o varias cadenas desde el teclado, la función “scanf” utiliza la especificación de formato “%s”, por ejemplo:
   scanf(“%s %s”, CAD1,CAD2)
   Nótese que en este caso las variables no van antecedidas del operador de dirección &.

2) La otra posibilidad es utilizar la función “gets” que lleva como argumento una sola variable tipo cadena, por ejemplo:
   gets(CAD1)

3) Para escribir una o varias cadenas en la pantalla, la función “printf” utiliza la especificación de formato “%s”, por ejemplo:
   printf(“%s %s”, CAD1, CAD2)

4) La otra función que cumple una tarea similar es “puts” que lleva como argumento la variable tipo cadena, por ejemplo:
puts(CAD1)

## 4.3. FUNCIONES PREDEFINIDAS EN EL LENGUAJE C

Ahora strlen() indica que el largo de cadena es 4, y las funciones de impresión en pantalla funcionaran normalmente, todo gracias a la presencia del char '\0'.

Existen unas cuantas funciones en la biblioteca estándar de C para el manejo de cadenas:

- ***strlen***
- ***strcpy***
- ***strcat***
- ***strcmp***
- ***strlwr***
- ***strupr***
- ***strstr***

Para usar estas funciones hay que añadir la directiva:

```
#include <strings.h>
```

**strlen**

Esta función nos devuelve el número de caracteres que tiene la cadena (sin contar el '\0').

```
#include <stdio.h>
#include <strings.h>

main()
{
    char texto[]="Sol";
    int longitud;

    longitud = strlen(texto);
    printf( "La cadena \"%s\" tiene %d caracteres.\n", texto, longitud );
}
```

**strcpy**

Copia el contenido de *cadena2* en *cadena1*. *cadena2* puede ser una variable o una cadena directa (por ejemplo "hola"). Se debe tener cuidado de que la cadena destino (cadena1) tenga espacio suficiente para albergar a la cadena origen (cadena2). Si la cadena origen es más larga que la destino, se eliminan las letras adicionales.

```
char textocurso[] = "Este es un curso de C.";
char destino[50];

strcpy(destino, textocurso );
printf( "Valor final: %s\n", destino );
```

**strcat**

Copia la *cadena2* al final de la *cadena1*.

```
char nombre_completo[50];
char nombre[]="Juan";
char apellido[]="Perez";

strcpy(nombre_completo, nombre );
strcat(nombre_completo, " " );
strcat(nombre_completo, apellido );
printf("El nombre completo es: %s.\n", nombre_completo );
```

**strcmp**

Compara *cadena1* y *cadena2*. Si son iguales devuelve 0. Un número negativo si *cadena1* va antes que *cadena2* y un número positivo si es al revés:

- cadena1 == cadena2 -> 0
- cadena1 > cadena2 -> número negativo
- cadena1 < cadena2 -> número positivo

```
#include <stdio.h>
#include <string.h>
```

```
main ()
{
   char ape1[30], ape2[30];
   int valor;

   printf("\n\nIngrese apellido 1:");
   gets(ape1);

   printf("\n\nApellido 2:");
   gets(ape2);

   //compara la primera cadena con la segunda
         valor=strcmp(ape1,ape2);
   if (valor==0)
     printf("Iguales");
   if (valor<0)
     printf("Menor %s",ape1);
   if (valor>0)
     printf("Mayor %s",ape1);

}
```

**strlwr**

Convertir a minúsculas.

Sintaxis: strlwr(S), convierte todos los caracteres alfabéticos ( 'A' .. 'Z' ) en la cadena S a sus correspondientes caracteres alfabéticos ( 'a' .. 'z' ).

*Ejemplo:*

```
char nombre[ ] = "Dante O. Diambra";
strlwr(nombre);
printf("Ahora quedó todo con minúsculas: %s", nombre);
```

**strupr**

Convertir a mayúsculas.

Sintaxis: strupr(S), convierte todos los caracteres alfabéticos ( 'a' .. 'z' ) en la cadena S a sus correspondientes caracteres alfabéticos ( 'A' .. 'Z' ).

*Ejemplo:*

```
char nombre3[ ] = " Dante O. Diambra";
strupr(nombre3);
printf("Ahora quedó todo con mayúsculas: %s", nombre3);
```

**strstr()**

Buscar subcadena

Sintaxis: strstr(S1, S2) ,busca en la cadena S1 la subcadena S2. La búsqueda se lleva a cabo desde el inicio hasta el final de S1. Si la operación es exitosa strstr regresa la dirección, dentro del bloque de memoria de la cadena, de la primera ocurrencia de S2 en S1, en caso contrario strstr regresa NULL.

*Ejemplo:*

```
char CAD[ ] = "Este es un ejemplo de una búsqueda";
int pos;
pos= strstr(CAD, "un")-CAD;
if (pos>=0) printf(" 'un' está en ' %s ', su primer ocurrencia es en la posición: %d", CAD,pos);
else printf(" 'un' no está en ' %s ' ", CAD);
```