Cadenas de caracteres (Strings) para lenguaje C

El lenguaje C, a diferencia de otros lenguajes, hace un manejo muy particular de las variables para almacenar texto. Un texto no es más que una sucesión de caracteres. Una variable del tipo char permite almacenar un único carácter, por lo tanto, para almacenar un texto se debe definir un vector de caracteres con tantas posiciones como letras pueda tener el texto que se quiere almacenar. Entonces, en un principio se podría decir que una variable del tipo string o cadena de caracteres, no es más que un vector de char. Sin embargo, esta afirmación no es del todo cierta ya que hay una característica que las diferencia. Toda cadena de caracteres define su terminación con un carácter de control '\0'. Este carácter especial indica el fin de la cadena.

Por ejemplo si se quiere ingresar un nombre, no se sabe exactamente el largo del mismo por lo tanto se define un vector con una cantidad suficiente de acuerdo al nombre más largo que se quiera guardar por ejemplo:

```
char nombre[20];
```

Declara un vector de char de 20 posiciones. Si en dicho vector se almacena un nombre corto, por ejemplo "JUAN", dicho nombre solo ocupará 4 de los 20 caracteres, y por lo tanto, luego de la letra N se debe poner el carácter de fin de cadena indicando hasta donde llega dicho nombre.

J	U	A	N	\0														

Entonces la diferencia principal entre un vector de caracteres y un string es la presencia de ese carácter especial de fin de cadena.

Inicialización de Strings

Al declarar la memoria del string es posible asignar un valor, para ello se le asigna una constante del tipo string que se escribe entre comillas. El carácter de fin de cadena se agrega automáticamente luego de la última letra. A continuación, observe algunos ejemplos de inicialización y como queda el vector resultante:

```
char nombre[20] = "ANA MARIA";
```

A	N	A		M	A	R	I	A	\0										

```
char nombre[] = "ANA MARIA";
```

A	N	A		M	A	R	I	A	\0

Si no se especifica el tamaño automáticamente reserva la memoria mínima suficiente para almacenar el texto y el carácter de fin de cadena.

> *Cuidado!! Si se le pone un tamaño al vector y se le asignan más caracteres de los definidos NO guardará el \0, y por lo tanto, ya no puede tratarse como un string ya que la función para mostrar no encontrará el fin de cadena.*

```
char nombre[3] = "MARCELO";
```

M	A	R

Biblioteca para el manejo de texto (string.h)

Debido a que los string son vectores con un carácter especial que indica el final de la cadena hay ciertas cosas como copiar uno a otro, comparar, etc que NO se pueden hacer directamente ya que no son un tipo de dato en sí mismos. Identificando el fin de cadena es posible armar distintas funciones para hacer esas tareas. Por ejemplo, para saber la cantidad de caracteres de un string se puede recorrer posición a posición el vector hasta encontrar el fin de cadena contando cuantas posiciones nos desplazamos. Para copiar un string en otro es similar a la copia de vectores donde se copia posición a posición en este caso hasta encontrar el fin de cadena. Todas estas funciones las podemos desarrollar pero dentro del lenguaje C ya existe una biblioteca que maneja distintas funciones relacionadas con cadenas de caracteres. La biblioteca string.h

Dentro de string.h hay muchas funciones, veremos algunas de ellas:

strlen	Determina la longitud de una cadena
strcpy	Copia una cadena a otra
strcat	Concatena dos cadenas dejando el resultado en la primera
strcmp	Compara dos cadenas
strcmpi	Compara dos cadenas ignorando si son mayúsculas o minúsculas

Función strlen:

Determina la longitud de una cadena, sin contabilizar el carácter nulo de terminación de la misma.

Sintaxis: strlen (cadena)

La función retorna un entero indicando la cantidad de caracteres.

Función strcpy:

Copia desde una cadena de origen hacia una cadena de destino.

Sintaxis: strcpy (cadena destino, cadena origen)

Ejemplo:

```
char original [15], copia [15];
….
strcpy (copia, original);
```

También strcpy se puede usar para asignar una constante a un string cuando no se hace al

Ejemplo:

```
char texto [25];

strcpy(texto, "MENSAJE");
```

Función strcat:

Concatena (añade) una cadena detrás de otras quedando el resultado en la cadena que se encuentra en primer orden.

Sintaxis: strcat (cadena receptora, cadena a añadir)

Ejemplo:

```
char receptor [40] = "Se agrego lo siguiente", dador [] =", me agregue";
strcat (receptor, dador);
```

Es importante asegurarse de que la cadena receptora tenga el espacio suficiente para guardar la cadena a añadir caso contrario sobrescribe memoria no asignada

Función strcmp:

Esta función compara dos cadenas y devuelve el resultado de la comparación.

Sintaxis: strcmp (cadena 1, cadena 2)

El valor que devuelve que será el resultado de la comparación es el siguiente:

Si las cadenas son iguales	devolverá un cero (0)
Si la cadena 1 es mayor que la cadena 2	devolverá un valor positivo
Si la cadena 1 es menor que la cadena 2	devolverá un valor negativo

Ejemplo:

```
char cadena1[30], cadena2[30];
…
if (strcmp (cadena1, cadena2) == 0)
          printf ("\nAmbas cadenas son iguales " );
else
    if (strcmp (cadena1, cadena2) > 0 )
       printf ("\nLa cadena1 es mayor que la cadena2");
    else
       printf ("\nLa cadena2 es mayor que la cadena1");
```

La comparación NO es por largo sino alfabética, comparando el peso en valor ASCII de cada una de las letras, posición a posición, hasta encontrar alguna diferencia. Hay que recordar que el ASCII de las minúsculas es mayor que el de las mayúsculas, por lo tanto, una palabra en minúsculas será mayor que una en mayúscula. A continuación se muestran algunos ejemplos:

Cadena1	**Cadena2**	**Mayor / iguales**
HOLA	HOLA	IGUALES
Hola	hola	Cadena2
hola mundo	hola	Cadena1
ana	anana	Cadena2
teXto	texto	Cadena2

Función strcmpi:

Es exactamente igual a la función strcmp pero ignora si son mayúsculas o minúsculas es decir que el string “ANA” es igual al string “ana” y también es igual al string “anA” o cualquiera de sus variantes.