# Cadenas

A diferencia de otros lenguajes de programación que emplean un tipo denominado cadena *string* para manipular un conjunto de simbolos, en C, se debe simular mediante un arreglo de caracteres, en donde la terminación de la cadena se debe indicar con nulo. Un nulo se especifica como '\0'. Por lo anterior, cuando se declare un arreglo de caracteres se debe considerar un carácter adicional a la cadena más larga que se vaya a guardar. Por ejemplo, si se quiere declarar un arreglo cadena que guarde una cadena de diez caracteres, se hará como:

char cadena[11];

Se pueden hacer también inicializaciones de arreglos de caracteres en donde automáticamente C asigna el caracter nulo al final de la cadena, de la siguiente forma:

charnombre\_arr[**tam**]=**"cadena"**;

Por ejemplo, el siguiente fragmento inicializa cadena con ``hola'':

char cadena[5]="hola";

El código anterior es equivalente a:

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

Para asignar la entrada estándar a una cadena se puede usar la función scanf con la opción %s (observar que no se requiere usar el operador &), de igual forma para mostrarlo en la salida estándar.

Por ejemplo:

int main()

{

char nombre[15], apellidos[30];

printf("Introduce tu nombre: ");

scanf("%s",nombre);

printf("Introduce tus apellidos: ");

scanf("%s",apellidos);

printf("Usted es %s %s\n",nombre,apellidos);

return 0;

}

El lenguaje C no maneja cadenas de caracteres, como se hace con enteros o flotantes, por lo que algunas de las siguientes instrucciones no son válidas:

int main()

{

char nombre[40], apellidos[40], completo[80];

char nombre[] = "Oscar"; /\*Correcto\*/

char nombre2[] = {'O','s','c','a','r','\0' };/\* Correcto \*/

nombre="José María"; /\* Ilegal \*/

apellidos="Morelos y Pavón"; /\* Ilegal \*/

completo="Gral."+nombre+appellidos; /\* Ilegal \*/

}

GETS

Sirve para leer cadenas de caracteres. Si se dieron cuenta, con el scanf, no puedes poner dos palabras separadas con espacio, solo acepta una palabra. Para resolver ese problema, es mejor utilizar la función gets que se encuentra en la librería <stdio.h>

#include <stdio.h>

#include <conio.h>

int main()

{

char nombre[15], apellidos[30];

printf("Introduce tu nombre: ");

gets(nombre);

printf("Introduce tus apellidos: ");

gets(apellidos);

printf("\nUsted es %s %s\n",nombre,apellidos);

getch();

return 0;

}

SIZEOF

Este comando sirve para saber el tamaño real de la cadena.

|  |
| --- |
| #include<stdio.h>  main()  {  char nombre[] = "Rosy";  printf( "Texto: %s\n", nombre );  printf( "Tamaño de la cadena: %i bytes\n", sizeof nombre );  } |

Resultado al ejecutar:

|  |
| --- |
| Texto: Rosy  Tamaño de la cadena: 5 bytes |

Curiosamente la respuesta nos dice que "Rosy" ocupa 5 bytes. Como cada elemento char ocupa un byte eso quiere decir que la cadena tiene 8 elementos, a pesar de que "Rosy" sólo cuenta con 4 letras. La razón de esta aparente paradoja estriba en que la cadena tiene como carácter final el símbolo **'\0'**, cuyo significado es "fin de cadena". De esta forma, cuando queremos escribir la cadena basta con usar %s y el compilador sabe cuántos elementos debe escribir: hasta que encuentre '\0'.

**La biblioteca string**

Los compiladores de C, C++ dan soporte a la biblioteca de funciones **<string.h>**, misma que accesible por medio de la directiva **#include<string.h>**. No veremos en detalle todas las funciones contenidas en dicha biblioteca, y nos limitaremos a mostrar algunos ejemplos de ciertas funciones importantes.

strlen(): Obtener longitud de cadenas

Sintaxis: size\_tstrlen(constchar \*s);

Comentarios: La función strlen() devuelve la longitud de la cadena s.

*Ejemplo:*

|  |
| --- |
| #include <stdio.h>  #include <string.h>  main()  {  char texto[]="Rosy Avendaño López";  int longitud;  longitud = strlen(texto);  printf( "La cadena \"%s\" tiene %i caracteres.\n", texto, longitud );  } |

strcpy(): Copiar cadenas

Sintaxis: char \*stpcpy(char \*dest, const char \*src);

Comentarios: stpcpy copia la cadena src hacia dest, la función termina hasta haber encontrado en src el caracter de terminación **null**.

*Ejemplo:*

|  |
| --- |
| #include <stdio.h>  #include <string.h>  main()  {  char texto[] = "Éste es el Grupo de ICI Primero";  char destino[50];  strcpy( destino, texto );  printf( "Valor final: %s\n", destino );  } |

strcat(): Concatenar cadenas

Sintaxis: char \*strcat(char \*dest, const char \*src);

Comentarios: strcat agrega la cadena src a dest, la función termina hasta haber encontrado en src el caracter de terminación **null**.

*Ejemplo:*

|  |
| --- |
| #include <stdio.h>  #include <string.h>  main()  {  Char nombre\_completo[50];  char nombre[]="Rosy";  char apellido[]="Avendanio Lopez";  strcpy( nombre\_completo, nombre );  strcat( nombre\_completo, " " );  strcat( nombre\_completo, apellido );  printf( "El nombre completo es: %s.\n", nombre\_completo );  } |

Como siempre, tenemos que asegurar que la variable en la que añadimos las demás cadenas tenga el tamaño suficiente. Con la primera línea de este programa introducimos el nombre en nombre\_completo. Usamos strcpy para asegurarnos de que queda borrado cualquier dato anterior. Luego usamos un strcat para añadir un espacio y, finalmente, introducimos el apellido.

strlwr(): Convertir a minúsculas.

Sintaxis: char \*strlwr(char \*dest);

Comentarios: strlwr convierte todos los caracteres alfabéticos en mayúsculas ( 'A' ..'Z' ) en dest a sus correspondientes caracteres alfabéticos en minúsculas

( 'a' ..'z' ).

*Ejemplo:*

|  |
| --- |
| #include <stdio.h>  #include <string.h>  main()  {  Char nombre\_completo[]="ROSY Avendanio Lopez";  strlwr(nombre\_completo);  printf("El nombre en minusculas es : %s", nombre\_completo );  } |

strupr(): Convertir a mayúsculas.

Sintaxis: char \*strupr(char \*dest);

Comentarios: strupr convierte todos los caracteres alfabéticos ( 'a' .. 'z' ) en dest a sus correspondientes caracteres alfabéticos ( 'A' ..'Z' ).

|  |
| --- |
| #include <stdio.h>  #include <string.h>  main()  {  Char nombre\_completo[]="ROSY Avendanio Lopez";  strupr(nombre\_completo);  printf( "El nombre en mayusculas es : %s", nombre\_completo );  } |

|  |  |
| --- | --- |
|  | sprintf |

*intsprintf(char \*destino, const char \*format, ...);*

Funciona de manera similar a [printf](http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/control/lengua_C/cadenas.htm#printf) pero, en vez de mostrar el texto en la pantalla, lo guarda en una variable (destino). El valor que devuelve (int) es el número de caracteres guardados en la variable destino.

Con sprintf podemos repetir el ejemplo de strcat de manera más sencilla:

|  |
| --- |
| #include <stdio.h>  #include <string.h>  main()  {  charnombre\_completo[50];  char nombre[]="Rosy";  char apellido[]="AvendanioLopez";  sprintf( nombre\_completo, "%s %s", nombre, apellido );  printf( "El nombre completo es: %s.\n", nombre\_completo );  } |

Se puede aplicar a sprintf todo lo indicado para printf.

|  |  |
| --- | --- |
|  | strcmp |

*intstrcmp(constchar \*cadena1, constchar \*cadena2);*

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 contrario:

* < 0 si cadena1 < cadena2
* ==0 si cadena1 == cadena2
* > 0 si cadena1 > cadena2

|  |
| --- |
| #include<stdio.h>  #include<string.h>  main()  {  char nombre1[]="Rosy";  char nombre2[]="Avendanio";  printf( "Comparación con strcmp: %i\n",  strcmp(nombre1,nombre2));  } |
|  |

El resultado es:

|  |
| --- |
| Comparación con strcmp : 1 |

Ejemplo 2:

#include<stdio.h>

#include<string.h>

#include<conio.h>

main()

{

char a[30], b[30];

printf("Dame la primer cadena: ");

gets(a);

printf("\nDame la segunda cadena: ");

gets(b);

if(strcmp(a, b) == 0)

printf("Las cadenas son iguales");

else

printf("\n las cadenas son diferentes");

getche();

}