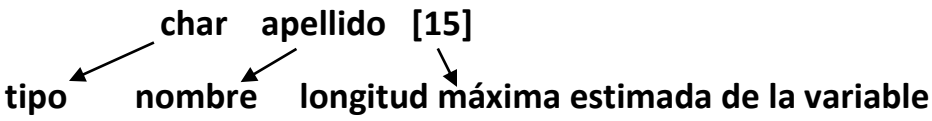
Cadena de caracteres

Introducción

Una cadena es una serie de caracteres tratados como una sola unidad, que puede incluir letras, dígitos y varios caracteres especiales como +, -, *, /, $, % y otros. Una leyenda o constante de caracteres se escribe entre comillas dobles "BUEN DIA".

En C, una cadena de caracteres es un arreglo de caracteres donde arreglo es una estructura de datos consistentes en elementos relacionados del mismo tipo y bajo un mismo nombre. Los arreglos ocupan lugar en memoria que depende de la cantidad de elementos que posee. Para indicar el nombre puede tener como máximo quince caracteres.

Se define así:



Para ingresar una cadena de caracteres usamos la función scanf con el siguiente formato:

scanf (“%s”, apellido);

%s es el formato correspondiente para string o cadena de caracteres.

La variable apellido, no lleva el **&** adelante pues C guarda automáticamente la dirección del primer elemento del arreglo y, como guarda en forma consecutiva los caracteres, solo sumando la dirección de memoria inicial al número de caracteres sabe cuál es la última dirección.

Si al ingresar una cadena de caracteres se deja un espacio intermedio, la variable solo guarda los caracteres hasta el espacio. Para que la variable guarde todos los caracteres, incluyendo el espacio, se utiliza el siguiente formato:

```
scanf ( "   % [ ^\n ] ", nombre
```

El **%** indica que es un formato, **[ ]** entre los corchetes se pueden enumerar todos los caracteres que se permiten ingresar. Esta opción no es aconsejable, ya que son muchos caracteres los que tendrían que escribirse; es mejor indicar cuál es el caracter que no se quiere considerar, es decir, el caracter que va a indicar el fin de entrada de datos. Para ello, se coloca el símbolo **^** y el carácter de fin (**\n** indica que la finalización de entrada de datos está dada con un ENTER).

[^\n] considera todos caracteres menos el ENTER.

Importante

Entre las primeras comillas **"** y el signo de **%** se debe dejar un espacio para que cada vez que cargue un nombre limpie el buffer de entrada de datos. Para mostrar una cadena de caracteres se usa, printf con el formato **%s** en el lugar que debe aparecer el contenido de la variable.

printf ("El apellido es %s", apellido);

Funciones para cadenas de caracteres

Para usar funciones relacionadas con cadenas de caracteres hay que incluir la biblioteca string.h, en la que estas están definidas.

Copiar

Para asignar una variable de tipo char a otra se utiliza la función strcpy.

La sintaxis de dicha función es:

```
strcpy (S1, S2);
```

strcpy significa string copy, es decir, copia una cadena de caracteres en otra.

La instrucción permite guardar en contenido de s2 en s1.

- **s1:** cadena de caracteres destino
- **s2:** cadena de caracteres origen

s2 y s1 deben ser del mismo tamaño**.**

Ejemplo

Leer los nombres y promedios de diez alumnos. Mostrar el nombre del mejor alumno.

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

int main()
{
  int i;
  char nom [30], mejor [30];
  float prom, auxprom;
  auxprom = 0.0;
  for ( i = 1; i <= 10; i ++)
  {
            printf ("Ingrese el nombre del alumno:");
            scanf ( "  %[^\n]", nom);
            printf ("Su promedio es:");
            scanf ("%f", & prom);
            if (prom > auxprom)
            {
                        auxprom = prom;
                        strcpy (mejor, nom);
            }
  }
 printf ("El mejor alumno es %s con un promedio de %.2f", mejor, auxprom);
return 0;
}
```

Comparar

La función strcmp (string compare) sirve para comparar el contenido de dos cadenas de caracteres.

Su sintaxis es:

```
strcmp ( S1, S2 );
```

Compara la cadena S1 con S2 y devuelve un valor entero:

0 si ambas cadenas son iguales

> 0 si la cadena S1 es mayor a S2

< 0 si la cadena S1 es menor a S2

¿Qué significa que una cadena sea mayor que otra?

Cada caracter (letra, dígito o símbolos especiales) se corresponde con un código numérico universal como ASCII o EBCDIC, que siguen el orden alfabético. La función strcmp compara numéricamente caracter a caracter y, si todos los códigos son iguales, devuelve un 0; si el caracter de S1 es anterior al de S2, devuelve un número menor que cero (< 0) y, de lo contrario, devuelve un número mayor a cero (> 0).

Ejemplo 1

- en S1 está guardada la cadena N A D A
- en S2 está guardada la cadena N U L O

La instrucción strcmp (S1, S2) realiza la comparación de la siguiente forma:

- N y N son iguales
- "A " está antes que " U", es decir, el código ASCII de la “A” es menor al de la “U”. Al restarlos el resultado es negativo, no compara más y devuelve < 0.

Se puede interpretar que si strcmp devuelve un valor negativo, la primera cadena está alfabéticamente antes que la segunda.

Ejemplo 2

- en S1 está guardada la cadena E S P E C I E
- en S2 está guardada la cadena E S P E C I A L

Hasta la " I ", las cadenas son iguales. Luego se compara la “E” de S1 con la "A" de S2. En este caso, strcmp devuelve un numero positivo (> 0), ya que el código ASCII de la "E" es mayor al de la "A" y al restarlos el resultado es positivo.

Ejercicio de aplicación

Ingresar los nombres de los 10 alumnos de un curso y decir cuántos se llaman JUAN

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

int main()
{
   char nombre[15];
   int cont=0, i;
   for(i=0; i<10;i++)
   {
      printf("Ingrese el nombre del alumno\n");
      scanf("%s",nombre);
      if(strcmp(nombre,"Juan")==0 || strcmp(nombre,"JUAN")==0)
      {
         cont++;
      }
   }
   printf("\nEn el curso hay %d alumnos llamados Juan", cont);
   return 0;
}
```

Observación

En el strcmp del ejercicio anterior, comparamos la variable nombre con una cadena determinada, por eso la cadena está escrita entre comillas.

Tratamiento de la cadena caracter a caracter

Nuestro propósito es, dada una cadena de caracteres, procesar caracter a caracter y mostrar algunas conclusiones.

Ejemplo

- Si ingresamos: ***La vida es maravillosa***

Queremos conocer cuántas palabras terminan con la letra a, cuántas palabras tiene la frase, cuántos blancos hay, sin utilizar arreglos ni matrices para guardar la frase. La idea es procesar cada caracter y, una vez leído, se pierde (no se puede volver para atrás).

¿Cómo lo hacemos?

Hay dos funciones en C que permiten leer caracter a caracter y mostrarlo en pantalla.

La función **getchar()** permite leer un caracter (solo uno). Esta función internamente traduce el carácter leído al número que le corresponde en la tabla ASCII, es decir, procesa ese número. Esto se hace así ya que con un entero podemos representar tanto el conjunto de caracteres que cabe en el tipo carácter (normalmente el conjunto ASCII de caracteres) como el carácter EOF de fin de fichero. Estos caracteres se suelen representar como un entero que va del 0 al 127 o 256. El carácter EOF entonces es representado con un -1

La función **putchar()** muestra el carácter en pantalla. Internamente traduce el valor numérico a la letra que le corresponde en la tabla.

El esquema básico en C es:

```
int main()
{
char c;
c=getchar();
while(c!='.')
{
//procesamiento de caracter
c=getchar();
}
//mostrar resultados.
}
```

Veamos entonces un ejemplo: Ingresar una frase terminada en punto y contar cuántos blancos tiene la frase.

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int c, cont=0;
   printf("Ingrese un frase terminada en punto\n");
   c=getchar();
   while (c!= '.')
   {
       if (c== ' ')
          cont++;
       c=getchar();
   }
   printf("\nla cantidad de blancos es %d",cont);
}
```

Ingresamos la frase y c guarda el primer carácter de la frase.

Se empieza a recorrer la frase hasta encontrar el punto

Si el caracter es un blanco, se incremente el contador

La variable c guarda el próximo carácter de la frase.

Veamos ahora pequeños códigos que les van a ayudar a construir los programas del trabajo práctico.

```c
while (c!= ' ')
 {
    c=getchar();
 }
```

Este ciclo recorre una palabra.

```c
while (c== ' ')
 {
    c=getchar();
 }
```

Este ciclo recorre los blancos de una frase.

Este ciclo recorre una palabra y la variable aux guarda la última letra de esa palabra.

```
while (c!= ' ')
 {
    aux=c;
    c=getchar();
 }
```

```
if (c=='s')
{
    c=getchar();
    if (c=='a')
      cont++;
}
```

En este caso, al anidar los if se busca en la frase el grupo ‘sa’ y los cuenta. Puede estar en cualquier lugar de la frase