Cadenas de caracteres

Introducción

El lenguaje C, no cuenta como otros lenguajes, con un tipo de dato String para trabajar con cadenas de caracteres. Sin embargo, una cadena está representada por un vector o array de caracteres (char) que termina generalmente con el carácter especial de fin de cadena (\0). Este carácter especial no es visible a través de la pantalla.

Por tratarse justamente de un tipo particular de vector o array es que lo abordaremos más en profundidad en un nuevo apartado.

Una cadena en C es un conjunto de caracteres o valores de tipo char, terminados con el carácter nulo (\0). Internamente, se almacenan en posiciones consecutivas de memoria. Este tipo de estructuras recibe un tratamiento especial.

Declaración

La sintaxis o forma general de declarar un cadena de caracteres es:

```
char <Identificador> [Longitud_de_la_cadena];
```

Cuando declaramos una cadena hay que tener en cuenta que tendremos que reservar una posición más para almacenar el carácter nulo, de modo que si queremos almacenar por ejemplo la cadena "PROGRAMA", tendremos que hacer la siguiente declaración:

```
char cadena[9];
```

Las ocho primeras posiciones se usan para almacenar los caracteres "PROGRAMA" y la posición extra, se reserva para el carácter nulo.

Es importante recalcar que en el lenguaje C, los índices toman valores consecutivos, empezando siempre desde el valor 0, así en nuestro ejemplo el primer valor de la cadena será cadena[0], es decir el valor 'P'.

La asignación directa a una cadena se puede hacer junto con la declaración de dicha cadena, teniendo en cuenta que se debe escribir entre comillas dobles (“”), como se muestra en el ejemplo a continuación:

```
char cadena[9]="PROGRAMA";
```

Veamos otros ejemplos de asignación directa, blanqueo o inicialización de una cadena:

```
char cadena[] = "Hola";
char otra_cadena[] = {'H', 'o', 'l', 'a', '\0'};
char saludo[] = {'H', 'o', 'l', 'a'};
char nueva_cadena[20] = "Una cadena en C";
char cadena_vacia[] = "";
```

Entrada y salida

Entrada y salida de cadenas con formato

Como ya se mencionó en otro apartado el lenguaje C dispone de las funciones printf() y scanf(), la primera se ocupan de mostrar información y la segunda de introducir datos por el teclado, ambas con formato estándar, para cualquier tipo de dato.

Veamos las siguientes líneas de código con cadenas de caracteres:

```
...
char cadena[30];

printf("Escribe una palabra: ");
scanf("%s", cadena);//Se ingresa una cadena de caracteres

printf("Ingreso: %s\n", cadena);//Se muestra la cadena leída
...
```

El especificador de formato en este caso es el ”%s” que indica que lo que se va a leer o mostrar es una cadena de caracteres.

Es muy importante tener en cuenta la siguiente consideración, scanf() toma una palabra como cadena y usa los espacios para separar variables es decir que si la cadena es compuesta, ej.: "hola alumnos", esta función sólo recoge o guarda la palabra "hola" y se ha olvidado de "alumnos".

Es decir que si ejecutamos el ejemplo anterior en pantalla va a ocurrir lo siguiente:

```
Escribe una palabra: hola amigos
Ingreso: hola
```

Una forma de subsanar esta cuestión es a través del uso de la función gets() que se explica a continuación.

Entrada y salida de cadenas sin formato

Además de las funciones de E/S por consola con formato: printf(), scanf(), el lenguaje C también ofrece dos funciones específicas para leer y escribir cadenas que se encuentran dentro de la biblioteca estándar <stdio.h>, gets() y puts()

La función gets(), es una función que se encarga de leer y almacenar una cadena de caracteres introducida mediante el teclado. La variable es guardada hasta que se encuentra con un salto de línea (\n).

La función puts(), se encarga de mostrar una cadena de caracteres específica, es equivalente al printf(), pero se recomienda puts() para el uso de cadenas de caracteres.

Analizamos las funciones gets() y puts(), a través del siguiente ejemplo:

```c
#include <stdio.h>
int main()
{
    char nombre[10]; //Declara una cadena

    puts("ingrese un nombre");   //Muestra un texto por pantalla
    gets(nombre);                //Lee una cadena por consola
    puts("la persona se llama"); //Muestra un texto por pantalla
    puts(nombre);                //Muestra la cadena ingresada
    return 0;
}
```

Operaciones con cadenas

El lenguaje C soporta una gran variedad de funciones de manejo de cadenas que me permiten trabajar, con operaciones básicas.

Por ejemplo el siguiente fragmento de código en C da error, puntualmente en las líneas 5, 6, 7, justamente porque las cadenas se tratan de manera diferente.

```
1...
2   char nombre[10];
3   char nomapellido[20];
4   ...
5   nombre = apellido;
6   nomapellido=nombre+apellido;
7   if (nombre==apellido)
8...
```

Como se mencionó el lenguaje C cuenta con una gran variedad de librerías que se encuentran dentro de la biblioteca que permiten realizar diferentes operaciones entre ellas las más comunes como las vistas en el ejemplo.

Dichas funciones se encuentran en la librería "string.h", por lo tanto, para usar estas funciones debemos incluir en la cabecera de nuestro programa la sentencia:

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

Función strcpy

La función strcpy(), permite copiar el contenido de una cadena en otra. Por analogía sería lo mismo que una asignación clásica.

La sintaxis o forma general de esta función es:

```
...
strcpy(cadena1,cadena2);
...
```

Ejemplos:

```
...
strcpy(nombre,"Juan");
strcpy(alias, nombre);
...
```

Función strcat

La función strcat(), permite concatenar dos cadenas. Cabe destacar que C, no hace comprobaciones de límites, así que es nuestra responsabilidad asegurarnos que la primera cadena es lo suficientemente grande como para mantener su contenido original y el de la segunda cadena.

La sintaxis o forma general de esta función es:

```
...
strcat(cadena1,cadena2);
...
```

Ejemplos:

```
...
strcpy("Juan","Perez");
strcpy(nombre,apellido);
...
```

Función strcmp

La función strcmp(), permite comparar dos cadenas.

La sintaxis o forma general de esta función es:

```
...
strcmp(cadena1,cadena2);
...
```

La función strcmp(), compara dos cadenas y devuelve un entero que se interpreta de la siguiente manera:

VALOR	INTERPRETACIÓN
-1	cadena1 es menor que cadena2
0	cadena1 es igual a cadena2
1	cadena1 es mayor que cadena2

Para ser específicos diremos que la comparación devuelve o (cero) si las cadenas de texto son iguales (incluyendo mayúsculas y minúsculas); si la primera cadena es mayor que la segunda, devuelve un número positivo; si es mayor la segunda, devuelve un valor negativo.

Cabe señalar que todas las comparaciones se hacen alfabéticamente (de acuerdo con el orden del diccionario universal). Esta función compara las cadenas, carácter a carácter utilizando el valor ASCII de los mismos. La comparación se detiene cuando se llega al final de la cadena o los caracteres comparados son distintos.

Veamos algunos ejemplos:

```
...
strcmp("a","a");    //0, el ASCII de "a" y "a" son iguales,(97)
strcmp("a","A");    //1, el ASCII de "a" (97)es mayor que "A" (65)
strcmp("abc","abe");//-1 el ASCII de "c"(99) es menor que "e" (101)
...
```

En general para hacer una comparación es necesario usar una estructura condicional (que abordaremos en otro apartado), en este ejemplo se ve su uso:

```
...
If (strcmp(cad1,cad2)==0)
                Printf("Las cadenas son iguales");
     Else
                Printf("Las cadenas son distintas");
...
```

Función strlen

La función strlen(), me permite obtener la longitud de una cadena. Devuelve la longitud de una cadena que finaliza con el carácter nulo, este último no se contabiliza.

La sintaxis o forma general de esta función es:

```
...
strlen(cadena1,cadena2);
...
```

Ejemplo:

```
strlen("hola");//Devuelve como longitud 4
```