Cadenas de caracteres

Si bien introdujimos el tema cuando tratamos tipos de datos primitivos, volvemos ahora sobre el mismo para tratarlo en profundidad ya que C no cuenta con un tipo específico para almacenar cadenas.

Una cadena de textos es un conjunto de caracteres, tales como "abcde". El lenguaje C soporta cadenas, utilizando array de caracteres (char).

Este tipo de estructuras funcionan igual que los demás *arrays* con la diferencia que ahora se manejan letras en vez de números.

Las cadenas en C son vectores de caracteres (elementos de tipo char) con una particularidad, el texto de la cadena termina siempre en un carácter nulo (\0).

Podemos manipular una única cadena que se comportaría o tendría las mismas características que un array unidimensional (vector).

Declaración

Para recordar diremos que la sintaxis o forma general de declarar un array de caracteres unidimensional es la siguiente:

```
char <Identificador_de_variable> [tamaño];
```

El identificador de variable es el nombre con el que vamos a identificar a la cadena y el tamaño indica la longitud máxima de la cadena.

Ejemplo:

```
char nombre [10]; //reserva espacio para 10 caracteres
```

Es muy común en C utilizar este tipo de datos pero con la necesidad de manipular más de una cadena, la estructura que representa esto es un array de caracteres bidimensional o dicho de otra forma un matriz de cadenas.

La sintaxis o forma general de declarar un array de cadenas bidimensional es la siguiente:

```
char <Identificador_de_variable> [tamaño 1][tamaño 2];
```

Igual que en el array unidimensional, el identificador de variable es el nombre con el que vamos a identificar o nombrar la estructura en este caso el tamaño 1 indica la cantidad de cadenas a procesar y el tamaño 2 la longitud total de cada cadena teniendo en cuenta una posición más para el valor nulo.

Ejemplo:

```
//reserva espacio para 3 nombres de  como máximo 9 caracteres.
char nombres[3][10];
```

Si por ejemplo, hacemos la siguiente asignación,

```
char nombres[3][10]={"LUZ","SOLEDAD","CARLA"};
```

la estructura gráficamente se ve de la siguiente forma:

	0	1	2	3	4	5	6	7	8	9
0	L	U	Z	\0						
1	S	O	L	E	D	A	D	\0		
2	C	A	R	L	A	\0				

Entrada y salida

Con formato

Generalmente, las cadenas se manipulan con printf y scanf con su correspondiente marca de formato "%s", sin que se presenten mayores dificultades.

Lo que resulta problemático es ingresar cadena utilizando la función scanf, cuando dichas cadenas contienen espacios sanf falla, es que sólo puede leer palabras, no frases separadas por espacios, esto nos obliga a utilizar otras funciones que mencionaremos luego en este mismo apartado, funciones sin formato.

Empecemos por considerar las dos funciones estándar de entrada y salida con su correspondiente especificador de formato "%s".

El siguiente ejemplo muestra básicamente una lectura y escritura utilizando las funciones printf y scanf con array de caracteres (unidimensional):

```
#include<stdio.h>
int main()
{
  char nombre[10];//Declaramos un cadena de tamaño 10
  int c;
  scanf("%s",nombre);
  printf( "\nSu nombre es: %s\n", nombre);
  return 0;
}
```

Lo mismo ocurre cuando queremos escribir o leer más de un arrays de cadenas (matrices), en este caso como vamos a leer más de una cadena podemos recurrir al uso de ciclos.

El ejemplo a continuación muestra esta situación:

```
#include<stdio.h>
int main()
{
  //Declaramos una matriz de 3 elementos de longitud máxima 9
  char nombres[3][10];
  int i;
  for(i=0;i<3;i++){
        printf( "Introduzca el nombre de la posicion %d ",i);
        scanf("%s",nombres[i]);
  }
   printf( "\nLos nombres ingresados son: \n");
  for(i=0;i<3;i++){
        printf("%s \n",nombres[i]);
  }
  return 0;
}
```

En ambos ejemplos se utilizó scanf que no presentaría problemas siempre y cuando el usuario tomara la precaución de ingresar un nombre simple.

Cabe recordar en esta instancia que cuando declaramos variable de tipo cadenas hay que tener la precaución de considerar que la longitud abarque el carácter nulo, ya que si no se encuentra la función no sabe dónde termina la cadena.

Sin formato

Ya mencionamos anteriormente que la biblioteca estándar de C contiene un conjunto de funciones grande y diversificado. Dentro de la librería stdio.h encontramos las funciones gets y puts que son el equivalente al scanf y printf, solo que especificas de cadenas de caracteres, y no tienen en cuenta el especificador de formato.

Si establecemos una analogía con printf y scanf, diremos que puts básicamente se comporta como printf, el problema sustancial se ve en el uso de la función scanf que solo lee palabras y no permite leer una secuencia de caracteres compuestos, es decir no permite espacios en blanco.

Aquí es donde se distingue del uso de la función gets que lee todos los caracteres que hay hasta encontrar un salto de línea.

El ejemplo a continuación muestra el uso de las funciónes puts y gets:

```c
#include <stdio.h>
#include<stdlib.h>
int main(void)
{
    char nombre[15];

    system("cls");

    puts "Introduce tu nombre y apellido completo:";
    gets(nombre);
    puts("Tu nombre y apellido es:");
    puts(nombre);

    return 0;
}
```

En el caso de usar una matriz de caracteres solo es necesario utilizar un ciclo para controlar la cantidad de elementos a leer, la longitud se controla con el carácter nulo.

El siguiente ejemplo muestra cómo leer y escribir más de una cadena de caracteres con gets y puts:

```c
#include <stdio.h>
int main(void)
{
    char nombres[2][15];
    int i;
    for (i = 0; i < 2; i++)
    {
        gets(nombres[i]);
    }
    for (i = 0; i < 2; i++)
    {
        puts(nombres[i]);
    }
    return 0;
}
```

Inicialización

Existen distintas formas de inicializar cadenas que dependerá del uso que le vamos a dar, veamos algunos ejemplos a continuación

```c
...
//Se carga cada carácter con el carácter nulo de forma manual
char a[7] = { 'c', 'a', 'd', 'e', 'n', 'a', '\0' };
//El compilador inserta automáticamente el carácter nulo al final de la cadena
char a [7]= "cadena";
//En estos dos últimos hay que tener la precaución de que las cadenas no excedan la 2da longitud y que además contemple el valor nulo.
char nombres[3][10]={"Carlos","Hugo","Sergio"};
char animales[][10]={"Perro","León","Lobo"};
...
```

Cabe recordar que este tipo de inicialización, utilizando {} se puede hacer solo en la zona de declaración de variables.

Otra forma de inicializar este tipo de estructuras es, a través de un ciclo y utilizando funciones de cadenas vistas anteriormente para asignar valores en el cuerpo del programa, tal y como lo muestra el siguiente ejemplo, que en este caso realiza un blanqueo de la variable.

```
...
char nombres[2][15];
int i;
for (i = 0; i < 2; i++){
        strcpy(nombres[i], "");
    }
...
```

Otro ejemplo: inicializa la cadena nombre con una valor de como máximo 9 caracteres

```
...
char nombre[10];
strcpy(nombre, "cadena");
...
```