4.5. Cadenas de Caracteres

Definición

En C y C++ las cadenas de caracteres no son más que arreglos de caracteres, salvo que a este tipo de arreglos el compilador les da un tratamiento especial. Usted puede manipular las cadenas de caracteres de la misma manera en que manipula cualquier otro tipo de arreglo, sin embargo, es preferible hacer uso de una librería estándar especialmente escrita para manipulación de cadenas de caracteres, me refiero a la librería **<string.h>** y que viene incluida con todo compilador de C o C++.

Recordando la presentación de arreglos hecha en los puntos anteriores podemos ver una cadena de caracteres como un vector de caracteres que ocupa una porción de memoria RAM conteniendo caracteres ASCII. En C y C++ una cadena es estrictamente una secuencia de cero o más caracteres seguidas por un carácter ***NULL*** cuyo código es cero y se puede escribir como todo código ASCII en octal precedido de una barra: ***'\0'***

Supongamos por ejemplo que se almacena en una cadena de longitud máxima 10, la secuencia "EJEMPLO 1":

E	J	E	M	P	L	O	\40	1	\0
0	1	2	3	4	5	6	7	8	9

Observe que las posiciones van desde 0 a 9, también fíjese que el espacio en blanco (cuyo código ASCII en decimal es 32 y en octal es 40) ocupa un lugar y que al final se encuentra el carácter ***NULL.***

Cuando manejamos a las cadenas como simples arreglos de caracteres, es importante preservar el carácter de terminación ***NULL***, ya que con éste es como C define y maneja las longitudes de las cadenas. Todas las funciones de la biblioteca estándar de C lo requieren para una operación satisfactoria. En cambio cuando manipulamos las cadenas con las funciones especiales de dicha biblioteca, nos despreocupamos del carácter ***NULL,*** ya que automáticamente se coloca al final de la cadena resultante.

Por lo tanto se recomienda usar las funciones para manejo de cadenas y no tratar de manipular las cadenas en forma manual desmantelando y ensamblando cadenas.

A continuación veamos el siguiente ejemplo de declaración de cadenas y asignaciones de valores, para aclarar algunos detalles:

```
char  CAD1 [ ] = "FACULTAD";   // manipulación como cadena

typedef  char cadena8[9];

cadena50 CAD2 = {'F' , 'A' , 'C' , 'U' , 'L' , 'T' , 'A' , 'D' , ' \0'} , CAD3;    // como vector

……..

CAD3 = "FACULTAD";    // es incorrecto solo se puede asignar de esta forma en la declaración
```

Como vemos la variable CAD1 se declara como un vector de tipo char sin longitud máxima, pero se le dá un valor inicial "FACULTAD", como cada elemento char ocupa un byte la variable CAD1 ocupará 9 byte en memoria, ya que alberga una cadena de 8 caracteres y en la última posición automáticamente se guarda el carácter ***NULL.*** Observe como se definió CAD2 y CAD3, posterior a una declaración de tipo que hace más eficiente y transparente esta declaración; también se le da a CAD2 un valor inicial igual que a CAD1, pero como vector, asignando uno por uno los valores de cada posición tipo carácter –entre comillas simples- inclusive el carácter **'\0'.** Tenga en cuenta que es incorrecto asignarle a CAD3 una cadena entre comillas dobles, esto es solo posible en la declaración de la variable.

Ambas formas de dar un valor inicial a una cadena son equivalentes, pero es obvio que asignarle un valor entre comillas dobles, como cadena, es mucho más práctico. Sin embargo la segunda forma puede ser interesante cuando se asigna uno o varios caracteres en forma directa indicando la posición, por ejemplo supongamos que queremos cambiar la secuencia "FACULTAD" por "FACILITA", en CAD1, entonces resulta más eficiente hacer una asignación directa, ya que hacer lo mismo con funciones especiales resulta complicado:

```
CAD1[3] = CAD1[5] = 'I' ;

CAD1[6] = 'T' ;

CAD1[7] = 'A' ;
```

Vemos que en este caso se asigna valores tipo char a elementos del arreglo, cosa que es perfectamente válida.

Lectura y Escritura

Para realizar la lectura e impresión de una cadena se puede recurrir a las siguientes funciones de la biblioteca <stdio.h>:

1) Para leer una o varias cadenas desde el teclado, la función "scanf" utiliza la especificación de formato "%s", por ejemplo:

   scanf("%s %s", CAD1,CAD2)

   Nótese que en este caso las variables no van antecedidas del operador de dirección &.

   La otra posibilidad es utilizar la función "gets" que lleva como argumento una sola variable tipo cadena, por ejemplo:

   gets(CAD1)

2) Para escribir una o varias cadenas en la pantalla, la función "printf" utiliza la especificación de formato "%s", por ejemplo:

   printf("%s %s", CAD1, CAD2)

   La otra función que cumple una tarea similar es "puts" que lleva como argumento la variable tipo cadena, por ejemplo:

   puts(CAD1)

Funciones especiales para cadenas:

La biblioteca <string.h> provee de una colección de funciones para manipular cadenas, dentro de las cuales se destacan:

strlen(): Obtener longitud de cadenas

Sintaxis: strlen(S) , devuelve la longitud de la cadena S.

Ejemplo:

```
char  nombre[ ] = "Dante O. Diambra";
printf("La longitud de \"  %s \" es %d",nombre, strlen(nombre));
```

strcpy(): Copiar cadenas

Sintaxis: strcpy(DEST,ORIG) ,copia la cadena ORIG en la variable cadena DEST.

Ejemplo:

```
char  nombre[ ] = "Dante O. Diambra";
char  copia[80];
strcpy(copia, nombre);
printf("La variable  <copia>  tiene el valor :  %s", copia);
```

strcat(): Concatenar cadenas

Sintaxis: strcat(DEST, ORIG) ,agrega la cadena ORIG a continuación de la cadena DEST.

Ejemplo:

```
char  nombre1[ ]  = " Dante O.";
char  copia1[80] = " Diambra";
strcat(copia1, nombre1);
printf("Ahora el apellido está primero: %s" , copia1);
```

strlwr(): Convertir a minúsculas.

Sintaxis: strlwr(S) ,convierte todos los caracteres alfabéticos ('A' .. 'Z') en la cadena S a sus correspondientes caracteres alfabéticos ('a' .. 'z').

Ejemplo:

```
char  nombre[ ]  = "Dante O. Diambra";
strlwr(nombre);
printf("Ahora quedó todo con minúsculas: %s", nombre);
```

strupr(): Convertir a mayúsculas.

Sintaxis: strupr(S) ,convierte todos los caracteres alfabéticos ('a' .. 'z') en la cadena S a sus correspondientes caracteres alfabéticos ('A' .. 'Z').

Ejemplo:

```
char  nombre3[ ]  = " Dante O. Diambra";
strupr(nombre3);
printf("Ahora quedó todo con mayúsculas: %s", nombre3);
```

strstr(): Buscar subcadena

Sintaxis: strstr(S1, S2) ,busca en la cadena **S1** la subcadena **S2**. La búsqueda se lleva a cabo desde el inicio hasta el final de **S1**. Si la operación es exitosa strstr regresa la dirección, dentro del bloque de memoria de la cadena, de la primera ocurrencia de **S2** en **S1**, en caso contrario strstr regresa **NULL**.

Ejemplo:

```
char  CAD[ ]  = "Este es un ejemplo de una búsqueda";
 int pos;

pos= strstr(CAD, "un")-CAD;
if  (pos>=0)  printf(" 'un' está en ' %s ', su primer ocurrencia es en la posición:  %d", CAD,pos);
                    else printf(" 'un'  no está en ' %s ' ", CAD);
```

strcmp(): Compara dos cadenas

Sintaxis: strcmp(S1, S2) , comparación léxica de las cadenas S1 y S2. Devuelve un entero menor, igual o mayor que cero si se encuentra que S1 es, respectivamente, menor que, igual a, o mayor que S2. Para esta comparación se tiene en cuenta el código ASCII de cada carácter.

Ejemplo:

```
char  CAD4[ ]  = "Alba";
char  CAD5[ ]  = "Alberto";
 int compara;

compara= strcmp(CAD4,CAD5);
if  (compara > 0)  printf("%s es mayor que %s",CAD4,CAD5);
     else if (compara == 0) printf("%s es igual que %s",CAD4,CAD5);
             else printf("%s es menor que %s",CAD4,CAD5);
```

En este caso se mostraría como resultado que CAD4 es menor que CAD5.

Ejemplos de programas con cadenas:

A continuación se muestran los ejemplos anteriores como un solo programa de prueba, para el compilador Dev-C++ :

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

char  CAD1[] = "FACULTAD";   // manipulación como cadena
typedef  char cadena8[9];
cadena8 CAD2 = {'F' , 'A' , 'C' , 'U' , 'L' , 'T' , 'A' , 'D' , '\0'} , CAD3;    // como vector

main ()
{
// CAD3 = "FACULTAD";      es incorrecto solo se puede asignar de esta forma en la declaración
//-------------------------------------------------------------------------------------------
printf("%s %s \n\n",CAD1,CAD2);
CAD1[3] = CAD1[5] = 'l' ;
CAD1[6] = 'T' ;
CAD1[7] = 'A' ;
printf("%s %s \n\n",CAD1,CAD2);
//-------------------------------------------------------------------------------------------
scanf("%s %s", CAD1,CAD2);
printf("%s %s \n\n",CAD1,CAD2);
//-------------------------------------------------------------------------------------------
gets(CAD1);
puts(CAD1);
//-------------------------------------------------------------------------------------------
char  nombre[] = "Dante O. Diambra";

printf("La longitud de  \" %s \" es %d \n\n",nombre, strlen(nombre));
//-------------------------------------------------------------------------------------------
char  copia[80];
strcpy(copia, nombre);
printf("La variable <copia> tiene el valor: %s \n\n", copia);
//-------------------------------------------------------------------------------------------
char  nombre1[]  = " Dante O.";
char  copia1[80] = "Diambra";

strcat(copia1, nombre1);
printf("Ahora el apellido esta primero: %s \n\n" , copia1);
//-------------------------------------------------------------------------------------------
char  nombre2[]  = "Dante O. Diambra";

strlwr(nombre2);
printf("Ahora quedo todo con minusculas: %s \n\n", nombre2);
//-------------------------------------------------------------------------------------------
char  nombre3[]  = "Dante O. Diambra";

strupr(nombre3);
printf("Ahora quedo todo con mayusculas: %s \n\n", nombre3);
//-------------------------------------------------------------------------------------------
char  CAD[] = "Este es un ejemplo de una busqueda";
int pos;

pos = strstr(CAD, "un")-CAD;
if  ( pos >= 0)  printf("'un' esta en '%s', su primer ocurrencia es en la posicion:  %d \n\n",CAD, pos);
                        else printf("'un' no esta en '%s' \n\n",CAD);
pos = strstr(CAD, "in")-CAD;
if  ( pos >= 0)  printf("'in' esta en '%s', su primer ocurrencia es en la posicion:  %d \n\n",CAD, pos);
                        else printf("'in' no esta en '%s' \n\n",CAD);
//-------------------------------------------------------------------------------------------
char  CAD4[ ]  = "Alba";
char  CAD5[ ]  = "Alberto";
int compara;

compara= strcmp(CAD4,CAD5);
if  (compara > 0)  printf("%s es mayor que %s \n\n",CAD4,CAD5);
     else if (compara == 0) printf("%s es igual que %s \n\n",CAD4,CAD5);
               else printf("%s es menor que %s \n\n",CAD4,CAD5);
//-------------------------------------------------------------------------------------------
getchar();
}
```

El siguiente es un programa para ordenamiento de una lista de nombres, que utiliza el método de la burbuja mejorado, para el compilador Dev-C++ :

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

#define MAX 100

typedef char cadena[20];
typedef cadena tipovec[MAX];

void leervec(int &n, tipovec vec)
{
    int i;
    printf("Introducir cantidad de nombres de la lista: ");
    scanf("%d", &n);
    for (i=0;i<n;i++)
        {printf("\nVEC(%d)=",i);
         scanf("%s", &vec[i]);}
}

void mostrarvec(int n, tipovec vec)
{
    int i;
    printf("\n\n\nLa lista ordenada es:");
    for (i=0;i<n;i++)
        printf("\nVEC(%d)=%s",i,vec[i]);
}

main (void)
{
  tipovec VEC;
  int L,N,I;
  cadena AUX;
  bool B;
  clrscr();
  leervec(N,VEC);
  L=N-1;
  do {
     B=false;
     for (I=0;I<L;I++)
       if (strcmp(VEC[I],VEC[I+1])>0)
          {strcpy(AUX,VEC[I]);
           strcpy(VEC[I],VEC[I+1]);
           strcpy(VEC[I+1],AUX);
           B=true;}
     L=L-1;}
  while (B);
  mostrarvec(N,VEC);
  getch();
}
```