Comandos para manejo de cadenas de carácteres basadas en arreglos

Ya sabemos que en el lenguaje C no existía el tipo *string*, así como el tipo *bool*. Para manejar cadenas de carácteres se utilizaban *arreglos de tipo char*. Esto sigue siendo así, aunque ahora, en C++, se cuenta con el tipo *string*, que lo veremos más adelante.

Ya sabemos que los arreglos se declaran de tamaño fijo. Si se declara suficientemente grande, como para poder resolver problemas de la naturaleza que el programa abordará, esto permitirá manejar cadenas de cualquier longitud que se necesite en el programa. Como se mencionó arriba. el uso de este tipo de arreglos sigue siendo válido aunque, actualmente, C++ ya incorpora el tipo *string*, que nos da muchas facilidades para el manejo de cadenas.

Podemos manejar un arreglo de carácteres de la misma forma que un arreglo de enteros, pero tengamos en cuenta que en ella guardaremos información como el nombre de una persona, la dirección donde ella vive o el nombre de un determinado producto que está a la venta. Notemos que en diferentes momentos de la ejecución del programa, la variable de cadena podría tener datos de diferente longitud, así que debe poderse determinar hasta qué casilla se encuentra el último carácter válido de la cadena. Si estuviéramos introduciendo nombres de personas desde teclado deberíamos de indicar, para cada nuevo nombre, cuántas letras contiene, lo que volvería muy tedios el programa y muy poco ágil su manejo para el usuario.

El lenguaje C/C++ determina de manera automática el final de la cadena dentro del arreglo, de la siguiente manera: si de digita, por ejemplo, un nombre desde teclado, luego de colocar los carácteres en el arreglo, se colocará un carácter indicador de fin de cadena en la casilla justo detrás del último carácter válido. Este carácter de simboliza en C/C++ así: **\0**. Esto es, la casilla que le sigue al último carácter válido de la cadena debe contener este carácter para que el programa pueda detectar el final de la misma. Por ejemplo:

Arreglo:

T	e		a	m	o	\0	...

Note que, al igual que el carácter nueva línea, o del de tabulación, este carácter va precedido de la plica invertida, ****, lo que le indica al lenguaje que se trata de un carácter especial.

Las funciones predefinidas para el manejo de cadenas están preparadas para manipular este símbolo especial cuando se realizan operaciones con cadenas. También nosotros, podemos recorrer y manejar manualmente las cadenas, hasta encontrar el carácter de fin de cadena. También podríamos manejar un arreglo de carácteres utilizando un *n* que nos indique su tamaño, pero todas estas herramientas se utilizan de acuerdo a cómo nosotros necesitemos resolver el problema que estamos abordando.

Desde el lenguaje C, se cuenta con muchas funciones predefinidas, originalmente agrupadas en la librería *string.h*, que nos permiten manipular cadenas. Estas librerías siguen siendo válidas en C++, si declaramos la misma cabecera o la cabecera *cstring*.

C:

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

C++:

#include <cstrin> ó *#include <string.h>*

A continuación se muestran algunas funciones que se vienen usando desde lenguaje C. Estas trabajaban directamente con arreglos de cadena, pues recordemos que no existía aún el tipo *string*. Note que la mayoría retorna punteros a carácter, precisamente porque los nombres de arreglos son, en sí mismos, punteros:

Función	Uso
int strlen(char *s)	Retorna la longitud de la cadena s, que consiste en la cantidad de caracteres antes del carácter de terminación de cadena.
char *strcpy(char *s1, char *s2)	Copia la cadena s2 en la cadena s1. Retorna un puntero a s1.
char *strncpy(char *s1, char *s2, int n)	Copia n carácteres de s2 en s1. Retorna un puntero a s1.
char *strcat(char *s1, char *s2)	Concatena las cadenas s1 y s2 y el resultado es colocado en s1. Retorna un puntero a s1.
char *strncat(char *s1, char *s2, int n)	Concatena n carácteres de s2 al final de s1. Retorna un puntero a s1.
int strcmp(char *s1, char *s2)	Retorna 0 si s1 y s2 son iguales, -1 si s1 es menor que s2 y 1 si s1 es mayor que s2. La comparación es de acuerdo a como se colocarían en el diccionario. Se dice que una cadena es menor que otra, si se debiera colocar antes que esta en orden alfabético.
int atoi(char *str) int x = atoi("97");	Convierte una cadena de dígitos a número entero. Es un acrónimo de ascii to integer.
char *_itoa(int, char *str, base) arr = _itoa(45, 10);	Convierte un entero a su representación en cadena de carácteres. Es un acrónimo de integer to ascii.
double atof(char *str) float x = atof("97.28");	Convierte una cadena de dígitos, que representa un real, a número real en doble precisión. Es un acrónimo de ascii to float
char *sprintf(char *, "%f", n)	Convierte el número real que recibe como tercer argumenta a una cadena almacenada en el primer argumento. Esta es una forma de impresión con formato, pero en lugar de desplegar en pantalla, se guarda en una cadena.

En las funciones que modifican el contenido de una cadena, hay que tener cuidado de que no se rebase el tamaño máximo del arreglo que la contiene. También hay que tomar en cuenta que siempre hay que dejar al menos la última casilla del arreglo disponible para poder colocar el carácter de fin de cadena. Manejar una cadena sin carácter de fin de cadena puede provocar fallos o comportamientos impredecibles en un programa. Por ello recordemos siempre declarar los arreglos de cadenas con longitud suficiente.

Al manipular cadenas de esta forma, ha de tenerse cuidado de que el carácter de fin de cadena siempre se coloque después de la última casilla válida. En muchas ocasiones es completa responsabilidad de nosotros hacerlo. Pero las funciones de librería están preparadas para manejar esta situación, aunque debemos ser previsores.

Si ingresamos desde teclado la cadena *Periquito* en un arreglo, el carácter '\0' será colocado automáticamente en la casilla con subíndice 9. Pero si luego comenzamos a hacer crecer o disminuir la cadena sin utilizar funciones predefinidas, sino con meras asignaciones directas, deberemos colocar el carácter de fin de cadena donde corresponda.

Por ejemplo:

Si ya tenemos la cadena *Periquito* en el arreglo de carácteres *cad*, y queremos cambiarla por *Perico*, podríamos hacer:

```
cad[4] = 'c';
cad[5] = 'o';
cad[6] = '\0';
```

Notar que el carácter de fin de cadena estaba originalmente en la casilla con subíndice 9, pero ha de insertarse en forma directa más a la izquierda.

Si ya tenemos la cadena *Camote* en el arreglo de carácteres *cad*, y queremos cambiarla por *Camotes*, podríamos hacer:

```
cad[6] = 's';
cad[7] = '\0';
```

Notar que el carácter de fin de cadena estaba originalmente en la casilla con subíndice 6, pero ha de insertarse en forma directa una casilla más a la derecha.

En ambos casos no importa que queden más carácteres a la derecha del fin de la cadena, pues el lenguaje C/C++ reconoce el fin de la cadena hasta este delimitador. Lo demás es basura, pero no es tomado en cuenta por las funciones predefinidas y nosotros debemos tener el cuidado de no tomarlo en cuenta si manipulamos manualmente las cadenas.

Ejemplo:

```
#include <iostream>
#include <cstring>
using namespace std;

int main(void)
{
   int i;
   char s1[10], s2[10];


   cout << "Digite una palabra: ";
   cin >> s2;

   /**********************/
```

```
    cout << "La longitud de s2 es: " << strlen(s2) << endl;

    /************************/

    strcpy(s1, s2);
    cout << "s1 es: " << s1 << endl;

    /************************/

    strncpy(s1, "abcdef", 3);
    cout << "s1 es: " << s1 << endl;

    /************************/

    cout << "Digite una palabra para s1: ";
    cin >> s1;

    cout << "Digite una palabra para s2: ";
    cin >> s2;

    strcat(s1, s2);
    cout << "s1 es: " << s1 << endl;

    /************************/

    int n = 2;

    strncat(s1, s2, n);
    cout << "s1 es: " << s1 << endl;

    /************************/

    cout << "Digite dos palabras para comparar:" << endl;
    cin >> s1;
    cin >> s2;

    int valor = strcmp(s1, s2);
    if(valor == 0)
       cout << "Las cadenas son iguales" << endl;
    else
       if(valor == -1)
          cout << "s1 es menor que s2" << endl;
       else
          cout << "s1 es mayor que s2" << endl;

    return 0;
}

/************************/

#include <iostream>
//#include <string>
using namespace std;

int main(void)
{
    char cadena[10] = "1234", cad[10], *pCad;
    int i = 5, entero;
    double real;
```

```
    cout << "***** Uso de atoi: (cadena a entero)" << endl;
    entero = atoi(cadena);
    cout << "El entero es: " << entero << endl;

    cout << "***** Uso de _itoa: (entero a caena)" << endl;
    _itoa(i, cad, 10);
    cout << "La cadena es: " << cad << endl;

    cout << "***** Uso de atof: (cadena a real)" << endl;
    real = atof("523.27");
    cout << "El real es: " << real << endl;

    cout << "***** Uso de sprintf: (real a cadena" << endl;
    sprintf(cad, "%f", 523.27);
    cout << "La cadena es: " << cad << endl;

    cout << endl << endl;
    return 0;
}
```

El lenguaje C++ proporciona muchas funciones para el manejo de cadenas y la forma de invocarlas es con el formato de acceso a miembro, como se hace en la programación orientada a objetos. Veremos esto posteriormente.

Lectura de cadenas desde teclado

La lectura de una cadena podemos realizarla, al menos de estas tres formas:

Este comando solo lee palabras, sin espacios intermedios:

```
cin >> cad;
```

La variable *cad* puede haber sido declarada como un arreglo de caracteres o como una variable de tipo *string*. En ambos casos este comando funciona igual.

Si al digitar su contenido introduce un espacio en blanco, notará que la lectura del contenido de la variable solo considera los caracteres antes del espacio en blanco.

¿Qué pasa si lo que necesitamos es digitar una frase que contiene espacios intermedios? Entonces debemos utilizar otro comando, como el siguiente:

```
cin.getline(cad, 10);
```

La variable *cad* debe haber sido declarada como arreglo de carácteres. A ella se le asignarán un máximo de 9 caracteres de la entrada y en la décima casilla se colocará un carácter de fin de cadena. Los demás caracteres no serán considerados como contenido de esta variable. Si se digitan menos carácteres, obviamente todos serán almacenados en la variable y el carácter de fin de cadena se colocará a continuación del último carácter ingresado.

Una tercera forma de ingresar una frase es esta:

```
getline(cin, cad);
```

La variable *cad* debe haber sido declarada de tipo *string*. A ella se le asignarán todos los caracteres que se ingresen por el teclado, note el primer argumento, hasta que se presione <Enter>. El carácter de fin de cadena se colocará, de forma automática, a continuación del último carácter válido. En otras palabras, se descarta el <Enter> y se sustituye por un carácter de fin de cadena.

Conviene decir que las variables de tipo *string* se declaran así:

```
string Variable;
```

Note que no se coloca ningún indicador de longitud máxima. A continuación hablaremos de este nuevo tipo de dato.