



República Bolivariana de Venezuela
Aldea Universitaria Liceo Fray Pedro de Agreda
Trayecto II
Desarrollo de Software

Prof. Elías Cisneros

Lenguaje C++
Parte V: Cadenas de caracteres

Introducción a cadenas de caracteres

A diferencia de la mayoría de otros lenguajes de computadores, C++ no tiene incorporado un tipo de datos de cadena. En su lugar, C++ posibilita las cadenas utilizando arrays de caracteres de una sola dimensión. Una cadena se define como un array de caracteres terminado con un carácter nulo (“\0”). Internamente los valores del arreglo se almacenan en posiciones consecutivas de memoria.

***char* identificador [tamaño];**

En las cadenas o arreglos de caracteres unidimensionales es posible hacer referencia a cada uno de los caracteres individuales que componen la cadena, simplemente indicando la posición o índice. Ejemplo: Supongamos que el arreglo de caracteres llamado cédula contiene el siguiente valor “V17005269”, entonces el arreglo almacenará los valores de la siguiente manera:

<code>char cedula[10];</code>	V	Posición 0
	1	Posición 1
	7	Posición 2
	0	Posición 3
	0	Posición 4
	5	Posición 5
	2	Posición 6
	6	Posición 7
	9	Posición 8
	\0	Posición 9

Obsérvese que en C++ el primer carácter de la cadena está ubicado en la posición 0. El símbolo de



terminación de cadena es el carácter “\0” el cual indica hasta donde se ha escrito dentro de la cadena. Una forma de asignar el valor a una cadena es la siguiente:

```
char cedula[10] = "V17005269";
```

Lectura y escritura de cadenas desde la consola

Para leer una cadena desde el teclado, simplemente basta con poner el nombre del array que recibe la cadena en el lado derecho de una sentencia *cin*.

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5.     char str[80];
6.     cout << "Ingrese una cadena (menos de 80 caracteres)" << endl;
7.     cin>>str;
8.     for(int i=0; i<80;i++)
9.         cout<< str[i]<<" ";
10.    return 0;
11. }
```

La sentencia *cin* no efectuará ninguna comprobación de límites, de manera que es posible que el usuario introduzca más caracteres que los que pueda almacenar la cadena. Por tanto, asegúrese de usar una cadena lo suficientemente grande como para almacenar la entrada que se espera. En el programa anterior, a la cadena que introdujo el usuario se le dio salida un carácter a la vez por pantalla. No obstante, hay una manera más fácil de visualizar una cadena usando el *cout*. El programa anterior se puede reescribir de la siguiente manera.

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5.     char str[80];
6.     cout << "Ingrese una cadena (menos de 80 caracteres)" << endl;
7.     cin>>str;
8.     cout<<str<<endl;
9.     return 0;
10. }
```



Un problema que se presenta al introducir cadenas con *cin*

Si ha probado el programa anterior, tal vez haya notado un problema. Ejecute el programa e intente introducir la cadena “Esta es una prueba”. Como podrá comprobar, cuando el programa vuelve a visualizar su cadena solamente mostrará la palabra “Esta” y no la secuencia completa. La razón para que esto es que el sistema de E/S de C++ deja de leer una cadena cuando encuentra el primer espacio en blanco. Hay varias maneras de resolver el problema planteado, por lo momentos estudiaremos la función *gets* de la biblioteca de funciones de C++. El formato general de esta función es:

```
gets(nombre_array);
```

La función *gets()* lee una cadena desde el teclado hasta que se pulsa la tecla INTRO. Esto significa que leerá una cadena que contiene espacios. Al utilizar esta función no se especifica ningún índice.

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5.     char str[80];
6.     cout << "Ingrese una cadena (menos de 80 caracteres)" << endl;
7.     gets(str);
8.     cout<<str<<endl;
9.     return 0;
10. }
```

Algunas funciones estándar de cadenas

La biblioteca estándar de C++ proporciona muchas funciones relacionadas con las cadenas. Las cuatro más importantes son *strcpy()*, *strcat()*, *strcmp()* y *strlen()*. Estas funciones requieren el archivo de cabecera *string.h*. Revisemos las funcionalidades de cada una de ellas. La función *strcpy()* tiene el siguiente formato general:

```
strcpy(a,de);
```

Copia en a la cadena contenida en de. El contenido de de no experimenta ningún cambio. Por ejemplo, este fragmento copia la cadena “hola” en str y la muestra en pantalla.

```
char str[80];
strcpy(str,"hola");
cout<< str;
```



La función `strcat()` añade el contenido de una cadena al final de otra. Este proceso se llama concatenación, su formato general es:

```
strcat(a,de);
```

Une el contenido de `de` con el contenido de `a`. NO hace comprobación de límites, de manera que se debe asegurar que `a` es lo suficientemente grande como para almacenar su contenido actual más el que está recibiendo. El siguiente fragmento de código imprime ¡Hola, que tal!

```
char str[80];

strcpy(str,"¡Hola,");
strcat(str," que tal!");
cout<<str;
```

La función `strcmp()` compara dos cadenas. Por lo general su formato es:

```
strcmp(s1,s2);
```

La función devuelve cero si las cadenas son iguales. Devuelve menor que cero si `s1` es menor que `s2` y mayor que cero si `s1` es mayor que `s2`. Las cadenas se comparan de forma lexicográfica, es decir, según el orden de un diccionario. Por lo tanto, una cadena es menor a otra cuando aparece en un diccionario antes que esta. Además, la comparación distingue entre mayúsculas y minúsculas, en que los caracteres en minúsculas son mayores que los que en mayúscula.

```
cout << strcmp("uno","uno");
```

La función `strlen()` devuelve la longitud, en caracteres de una cadena. Su formato general es:

```
strlen(str);
```

La función `strlen()` no cuenta el carácter de terminación nulo. Esto significa que si `strlen()` es llamada usando la cadena "test", devolverá 4. A continuación un programa de muestra de las funciones vistas anteriormente:

1. `#include <iostream>`
2. `#include <string.h>`
3. `using namespace std;`
4. `int main()`
5. `{`
6. `char str1[80], str2[80];`
7. `int i;`
8. `cout<<"Introduzca la primera cadena: ";`



```
9.  gets(str1);
10. cout<<"Introduzca la segunda cadena: ";
11.  gets(str2);
12.  //Ver la longitud de las cadenas
13.  cout<< str1 <<" tiene " << strlen(str1) << " caracteres de longitud"<<endl;
14.  cout<< str2 <<" tiene " << strlen(str2) << " caracteres de longitud"<<endl;
15.  //Comparación de cadenas
16.  i= strcmp(str1,str2);
17.  if(i==0){
18.      cout<<"Las cadenas son iguales"<<endl;
19.  }
20.  else{
21.      if (i>0){
22.          cout<<str1<<" es menor que "<<str2<<endl;
23.      }
24.      else{
25.          if(i<0){
26.              cout<<str1<<" es mayor que "<<str2<<endl;
27.          }
28.      }
29.  }
30.  /*Concatenar str2 con el final de str1 si caso hay
31.  espacio suficiente*/
32.  if (strlen(str1) + strlen(str2) < 80){
33.      strcat(str1,str2);
34.      cout<<str1<<endl;
35.  }
36.  //Copiar str2 en str1
37.  strcpy(str1,str2);
38.  cout<< str1<< " "<< str2<< endl;
39.  return 0;
40. }
```

Uso de funciones TOUPPER() y TOLOWER()

La biblioteca estándar de C++ contiene dos funciones que son bastante útiles al trabajar con caracteres y cadenas: `toupper()` y `tolower()`. Estas funciones tienen el siguiente prototipo:

```
int toupper(int ch);
int tolower(int ch);
```

La función `toupper()` devuelve el equivalente en mayúscula del carácter pasado en `ch`. `tolower()` devuelve el equivalente en minúscula de `ch`. Si `ch` no es una letra del alfabeto, entonces `ch` es devuelto sin cambios. Ejemplo, el siguiente fragmento devuelve una 'x' minúscula a `ch`:



```
char ch;  
ch=tolower("X");
```

Tanto `toupper()` como `tolower()` requieren el archivo de cabecera `ctype.h`. La razón por la que `toupper()` y `tolower()` son tan útiles es que permiten a un programa ignorar la distinción de mayúsculas o minúsculas de un carácter o de varios caracteres. Esto es especialmente útil cuando se trata de entradas del usuario. Por ejemplo, en el siguiente programa aritmético que se indica a continuación es necesario introducir órdenes en minúsculas de manera que se puedan hacer coincidir usando `strcmp()`, es por ello la importancia de poder comparar letras en minúsculas o mayúsculas indistintamente.

```
1. #include <iostream>  
2. #include <string.h>  
3. #include <ctype.h>  
4. using namespace std;  
5. int main()  
6. {  
7.     char orden[80];  
8.     int juego=1;  
9.     int gana_caracas=0;  
10.    int gana_magallanes=0;  
11.    int opcion=0;  
12.    int tamano_cadena=0;  
13.    //unsigned i=0;  
14.    cout<<"En la temporada de Besibol Caracas y Magallanes juegan 14 juegos"<<endl;  
15.    while(juego<=14) {  
16.        cout<<"Según su pronóstico que equipo ganará en juego: "<<juego<<endl;  
17.        cout<<"Escriba Caracas o Magallanes (o Salir): "<<endl;  
18.        cin>>orden;  
19.        //convertir en minusculas  
20.        tamano_cadena= strlen(orden);  
21.        for( int i=0;i< tamano_cadena;i++){  
22.            orden[i]=tolower(orden[i]);  
23.        }  
24.        //Revisar si el usuario desea salir  
25.        if( strcmp(orden,"salir")==0){  
26.            cout<<"Usted ha abortado el programa"<<endl;  
27.            break;  
28.        }  
29.        if(strcmp(orden,"caracas")==0){  
30.            opcion=1;  
31.        }
```



```
32.     else{
33.         if(strcmp(orden,"magallanes")==0){
34.             opcion=2;
35.         }
36.         else
37.         {
38.             opcion=3;
39.         }
40.     }
41.     switch(opcion){
42.         case 1:
43.             gana_caracas=gana_caracas+1;
44.             juego++;
45.             break;
46.         case 2:
47.             gana_magallanes=gana_magallanes+1;
48.             juego++;
49.             break;
50.         default:
51.             cout<<"Opción Inválida, vuelva a introducir su opción "<<endl;
52.             break;
53.     }
54. }
55. cout<<"Caracas ganó "<<gana_caracas <<" juegos"<<endl;
56. cout<<"Magallanes ganó "<<gana_magallanes <<" juegos"<<endl;
57. cout << "Hasta Pronto" << endl;
58. return 0;
59. }
```

Otras funciones de cadenas en la librería estándar de C++

La biblioteca estándar de C++ contiene varias funciones útiles que le permiten categorizar los caracteres. Todas estas funciones empiezan por *is*. Todas aceptan un argumento de carácter y devuelven un resultado bien sea true (1) o false (0). Las funciones de prueba de caracteres se muestran a continuación. Investigue la descripción de cada una de las funciones indicadas a continuación:

Nombre función	Descripción
isalnum(ch)	
isalpha(ch)	
iscntrl(ch)	
isdigit(ch)	



isgraph(ch)	
islower(ch)	
isprint(ch)	
ispunct(ch)	
isspace(ch)	
isupper(ch)	
isxdigit(ch)	

Ejercicios con cadenas

1. Construya un programa en C++ que solicite al usuario sus nombres y apellidos en variables separadas. Concatene ambas cadenas y muestre el resultado por pantalla.
2. Realice un programa donde se ingrese una cadena de 300 caracteres llamada dirección. Determine cuantas veces aparece la palabra “de” en dicha cadena.
3. Dada una cadena llamada “entrada”, construya un programa que invierta el orden de los caracteres y los almacene en otra cadena llamada “salida”. Ejemplo entrada=mango , salida=ognam.
4. Construya un programa que proporcione dos opciones al usuario: Compra de boleto “ida” con un precio de BsF 0,5 o “ida-vuelta” con un precio de BsF 0,9. Según la opción utilizada por el usuario calcule el monto a cobrar al usuario por concepto de venta de boleto. Utilice la función *strcmp()*.
5. Mediante la utilización de un ciclo while construya un algoritmo para calcular el tamaño de un cadena de máximo 100 caracteres.
6. Construya un algoritmo para calcular la cantidad de espacios en blanco que existen en una cadena de 300 caracteres.
7. Construya un programa que elimine los espacios en blanco que existen en una cadena de 300 caracteres, es decir realice compactación de cadenas.
8. Construya un programa que cuente la cantidad de vocales que existen en una cadena de 300 caracteres.
9. Construya un programa que reemplace los espacios en blanco de una cadena por asteriscos “*”. La cadena es de 300 caracteres.
10. Realizar un programa para comprobar si una palabra es palíndroma, es decir que se lee igual por la izquierda que por la derecha, ejemplo RADAR, AREPERA, RECONOCER, ALA.