Manejo de Cadenas de Caracteres

1. Declaración de una variable.

La declaración de una variable como una cadena de caracteres se realiza de la siguiente forma:

```
char NombreVariable[CantidadCaracteres];
```

Ejemplo 1:

```
char arregloCadena[] = "buenas";
```

Notemos que no hemos especificado índice alguno, el compilador contará la cantidad de letras que tiene la cadena `"buenas"` (son 6) y agregará uno para incluir el carácter nulo (`'\0'`). El carácter nulo indica el término de la cadena, es importante siempre tenerlo en cuenta.

Ejemplo 2:

```
char arregloCadena[7] = "buenas";
```

Notemos que hemos especificado la dimensión para la variable, contemplando los 6 caracteres de la palabra “buenas” más el carácter nulo (`'\0'`).

Ejemplo 3:

```
char arregloCadena[7] = {'b','u','e','n','a','s','\0'};
```

En esta declaración inicializamos cada uno de los elementos del arreglo de manera individual. Notar que inicializamos explícitamente en el elemento 7 del arreglo al carácter nulo '\0'.

Ejemplo 4:

```
char Saludo[5];
Saludo = "HOLA";         /* Error */
```

La asignación directa sólo está permitida cuando se hace junto con la declaración. El anterior ejemplo producirá un error en el compilador, ya que una cadena definida de este modo se considera una constante. La manera correcta sería la siguiente:

```
char Saludo[5];
Saludo[0] = 'H';
Saludo[1] = 'O';
Saludo[2] = 'L';
Saludo[3] = 'A';
Saludo[4] =  0;
```

O bien:

```
char Saludo[5] = "HOLA";
```

Ejemplo 5:

No es posible asignar un texto directamente a una variable del tipo cadena, a menos que se haga al momento de declarar la variable, como se mostró anteriormente. La forma correcta es empleando la función **strcpy**.

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

int main (){
    char Nombre[20];
    strcpy(Nombre,"Helena");
    cout << Nombre;
    return 0;
}
```

2. Arreglos de Cadenas de Caracteres.

Para almacenar varios datos de tipo cadena en un arreglo, podemos hacerlo de varias formas:

1. Definiendo un tipo de dato.

```
typedef char texto[30];
texto vector[5] = {"HELLEN","KAREN","PASCAL","ARLETTE","HELENA"};
```

	vector
0	HELLEN\0
1	KAREN\0
2	PASCAL\0
3	ARLETTE\0
4	HELENA\0

2. Usando un arreglo bidimensional.

```
char vector[5][30];
```

Ejemplo:

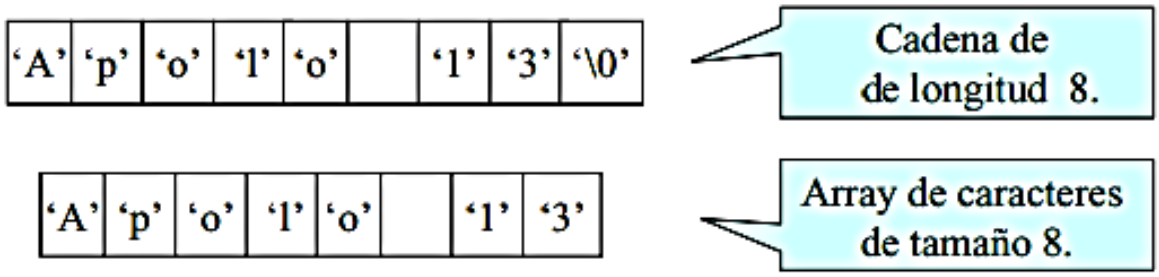
```
#include <string.h>
#include <iostream>

using namespace std;

void main() {
     int i;
     char nom[4][9] = { "Ana", "Juan", "María", "Luis" };
     for( i=0; i<4; i++ )
          if ( strcmp(nom[i],"Juan") != 0 )
               cout << nom[i];
}
```

3. Diferencia entre Cadenas de Texto y Arreglos de Caracteres.

La principal diferencia entre una cadena y un arreglo de caracteres, es que una cadena siempre termina con el carácter nulo '\0' y los arreglos de caracteres no. Como conclusión, podemos decir que - Todas las cadenas de texto son arreglos de caracteres, pero, no todos los arreglos de caracteres almacenan cadenas.



4. Lectura de Cadenas de Caracteres desde teclado

La lectura de una cadena de caracteres ha sido un tema que por mucho tiempo no tuvo una solución satisfactoria. En la actualidad, se cuenta con el método getline() el cual permite una lectura de una cadena de caracteres sin problemas alguno.

La forma general de uso es:

```
cin.getline(cadena, num, carácter_final);
```

getline lee una serie de caracteres desde el flujo de entrada y los almacena en la variable cadena. Se leen caracteres hasta el final del flujo, hasta el primer carácter que coincide con el carácter final especificado, el cual se desecha o hasta que se han leído num-1 caracteres; en este último caso el estado del flujo pasa a ser de fallo. Esto se puede comprobar con el método **fail**(). El estado de fallo se produce porque se han leído num-1 caracteres que es el máximo que queremos leer y aun han quedado caracteres en el flujo por lo que es posible que se hayan tecleado más caracteres de la cuenta y pueden provocar un error en la siguiente lectura.

getline añade automáticamente el carácter nulo (\0) al final de la cadena extraída.

Utilizando **getline** para leer nombre y apellidos del ejemplo anterior escribiremos:

```
cin.getline(nombre, 50, '\n');
```

El carácter final se puede omitir, si no aparece se supone que es \n :

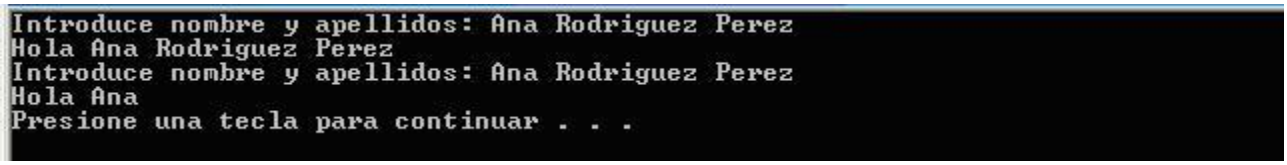
```
cin.getline(nombre, 50);
```

Ejemplos de lectura de caracteres en C++

Ejemplo 1: El siguiente ejemplo muestra la diferencia cuando se leen cadenas de caracteres en C++ usando o no **getline**:

```
#include <iostream>
using namespace std;
int main(void){
    char nombre[40];
    cout <<  "Introduce nombre y apellidos: ";
    cin.getline(nombre,40);            //lectura incluyendo espacios
    cout << "Hola " << nombre << endl;
    cout <<  "Introduce nombre y apellidos: ";
    cin >> nombre;                     //lectura sin incluir espacios
    cout << "Hola " << nombre << endl;
    system("pause");
}
```

La salida de este programa es:



Ejemplo 2: En el siguiente ejemplo se leen cadenas de caracteres hasta que se pulsa F6. Se comprueba y se muestra el estado del flujo **cin** después de cada lectura.

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

int main(void)
{  char cadena[15];
   cout << "Introduzca cadena de caracteres. F6 Para finalizar\n\n";
   do {  cout << "cadena: ";
      cin.getline(cadena,15,'\n');

      //se muestran los bits de estado del flujo cin para
      //comprobar el resultado de la lectura

      cout << "bit eof->" << cin.eof() << " ";
      cout << "  bit fail->" << cin.fail() << endl;

      //si la cadena excede de 15 cin.fail() toma el valor 1
      //si no se ha pulsado F6 en el nombre (fin de lectura)
      //y el nombre excede de la longitud máxima permitida
      //se eliminan el resto de los caracteres
      if(!cin.eof() && cin.fail())
      {
            cin.clear();
            cin.ignore(numeric_limits<int>::max(),'\n');
      }
   }while(!cin.eof());
   system("pause");
}
```

La salida de este programa es la siguiente:

5. La biblioteca string

Los compiladores de C, C++ dan soporte a la biblioteca de funciones **<string.h>**, a la que accede por medio de la directiva **#include <string.h>**. No veremos en detalle todas las funciones contenidas en dicha biblioteca, y nos limitaremos a mostrar algunos ejemplos de ciertas funciones importantes.

5.1. strlen(): Obtener longitud de cadenas.

La función **strlen**() devuelve la longitud de la cadena s.

```
size_t strlen(const char *s);
```

Ejemplo:

```
char nombre[] = "Oscar E. Palacios";
cout << strlen(nombre) << endl;
```

5.2. strcpy(): Copiar cadenas.

stpcpy copia la cadena src hacia dest, la función termina hasta haber encontrado en src el carácter de terminación **null**.

```
char *stpcpy(char *dest, const char *src);
```

Ejemplo:

```
char nombre[] = "Oscar E. Palacios";
char copia[80];
strcpy(copia, nombre);
cout << copia << endl;
```

5.3. strcat(): Concatenar cadenas.

strcat agrega la cadena src a dest, la función termina hasta haber encontrado en src el carácter de terminación **null**.

```
char *strcat(char *dest, const char *src);
```

Ejemplo:

```
char nombre[] = "Oscar E.";
char copia[80] = " Palacios";
strcat(copia, nombre);
cout << copia << endl;
```

5.4. strlwr(): Convertir a minúsculas.

Comentarios: strlwr convierte todos los caracteres alfabéticos ('A' .. 'Z') en dest a sus correspondientes caracteres alfabéticos ('a' .. 'z').

```
char *strlwr(char *dest);
```

Ejemplo:

```
char nombre[] = "Oscar E. Palacios";
strlwr(nombre);
cout << nombre << endl;
```

5.5. strupr(): Convertir a mayúsculas.

strupr convierte todos los caracteres alfabéticos ('a' .. 'z') en dest a sus correspondientes caracteres alfabéticos ('A' .. 'Z').

```
char *strupr(char *dest);
```

5.6. strchr(): Buscar carácter (hacia adelante)

strchr busca en **s** el caracter **c**. La busqueda se lleva a cabo desde el inicio hasta el final de s.Si la operación es exitosa strchr regresa un puntero hacia la primera ocurrencia de **c** en **s**, en caso contrario strchr regresa **null**.

```
char *strchr(char *s, int c);
```

Ejemplo:

```
char nombre[] = "Oscar E. Palacios";
char *p;

p = strchr(nombre, 'E');
if (p) {
```

```
        cout << "nombre contiene a E" << endl;
            cout << "indice = " << (p - nombre) << endl;
        }
    else cout << "E no está en nombre" << endl;
```

5.7. strrchr(): Buscar carácter (hacia atrás)

strchr busca en **s** el caracter **c**. La busqueda se lleva a cabo desde el final hasta el inicio de s. Si la operación es exitosa strchr regresa un puntero hacia la última ocurrencia de **c** en **s**, en caso contrario strchr regresa **null**

```
char *strrchr(char *s, int c);
```

Ejemplo:

```
char nombre[] = "Oscar E. Palacios";
char *p;

p = strrchr(nombre, 'E');
if (p) {
         cout << "nombre contiene a E" << endl;
         cout << "indice = " << (p - nombre) << endl;
       }
else cout << "E no está en nombre" << endl;
```

5.8. strstr(): Buscar subcadena.

strstr busca en **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 un puntero hacia la primera ocurrencia de **s2** en **s1**, en caso contrario strstr regresa **null**.

```
char *strstr(const char *s1, char *s2);
```

Ejemplo:

```
char s[] = "Un barco de tristeza";
char *p;

p = strstr(s, "barco");
if (p) {
         cout << "barco está en s" << endl;
         cout << "indice = " << (p - s) << endl;
       }
else cout << "barco no está en s" << endl;
```

6. La biblioteca stdlib

- **atoi:** convierte string a `int` type.
- **atol:** convierte string a `long` type.
- **atof:** convierte string a `float` type.

7. La biblioteca ctype

7.1. Función toupper()

Convierte un carácter a mayúscula.

Sintaxis: `int toupper(int ch);`

Descripción: toupper es una función que convierte el entero ch (dentro del rango EOF a 255) a su valor en mayúscula (A a Z; si era una minúscula de, a a z). Todos los demás valores permanecerán sin cambios.

Valor de retorno: toupper devuelve el valor convertido si ch era una minúscula, en caso contrario devuelve ch.

7.2. Función tolower()

Convierte un carácter a minúscula.

Sintaxis: `int tolower(int ch);`

Descripción: tolower es una función que convierte el entero ch (dentro del rango EOF a 255) a su valor en minúscula (A a Z; si era una mayúscula de, a a z). Todos los demás valores permanecerán sin cambios.

Valor de retorno: tolower devuelve el valor convertido si ch era una mayúscula, en caso contrario devuelve ch.

7.3. Macros is<conjunto>()

Las siguientes macros son del mismo tipo, sirven para verificar si un carácter concreto pertenece a un conjunto definido. Estos conjuntos son: alfanumérico, alfabético, ascii, control, dígito, gráfico, minúsculas, imprimible, puntuación, espacio, mayúsculas y dígitos hexadecimales. Todas las macros responden a la misma sintaxis:

Sintaxis: `int is<conjunto>(int c);`

Función	Valores
isalnum	(A - Z o a - z) o (0 - 9)
isalpha	(A - Z o a - z)
isascii	0 - 127 (0x00-0x7F)
iscntrl	(0x7F o 0x00-0x1F)
isdigit	(0 - 9)
isgraph	Imprimibles menos ' '
islower	(a - z)
isprint	Imprimibles incluido ' '
ispunct	Signos de puntuación
isspace	Espacio, tab, retorno de línea, cambio de línea, tab vertical, salto de página (0x09 a 0x0D, 0x20).
isupper	(A-Z)
isxdigit	(0 to 9, A to F, a to f)

Valores de retorno: Cada una de las macros devolverá un valor distinto de cero si el argumento c pertenece al conjunto.