Tutorial de la librería *string* de STL

Introducción

La librería *string* de STL permite manejar cadenas de caracteres de forma eficiente, consistente y sencilla . Las posibilidades que ofrece son numerosas y el alumno podrá hacer uso completo de esta librería cuando conozca detalles de STL. A continuación se ofrece un pequeño tutorial para aprender a manejarlo rápidamente.
Para más información mirar la siguiente página web de STL:

```
http://www.sgi.com/tech/stl/basic_string.html
```

NOTA: El código de este tutorial debe verse de forma continua

#include

Es necesario incluir la librería *string* de STL. Hacer:

```
#include <string>
```

Definición y Construcción de objetos *string*

Existen varias formas de crear objetos de este tipo. Veamos varios ejemplos:

```
string a; // crea un string vacío
string b(a);  // llamada al constructor copia
string c(10,'A'); //crea un string con 10 ocurrencias del carácer 'A'
string d ("Esto es una cadena constante"); // a partir de un const char*
// ahora con una variable char*
char cad[10]="Holaaaa!!";
string e (cad);
// El operador de asignación también está implementado
e = d;
a = cad; //incluso para char*
```

La ventaja es obvia: no necesitamos reservar espacio para albergar las nuevas cadenas. La destrucción de cadenas es también transparente para el usuario.

Operadores sobrecargados

Muchas de las operaciones que pueden realizarse con los string de STL se realizan con operadores sobrecargados. Veamos algunos ejemplos:

Entrada/Salida

Podemos utilizar la E/S estándar de C++.
El siguiente código lee sólo una palabra:

```
string g;
cin >> g;
cout << g << endl;
```

Para evitar problemas con la lectura de cadenas de más de una palabra es conveniente leer líneas usando *getline* de la siguiente forma:

```
string g;
getline (cin, g); //leemos de teclado el string g
```

Concatenación

Para concatener tenemos varias opciones: el *operator+=* concatena a la cadena resultado

```
c+=d;
cout << c; //Resultado:"AAAAAAAAAAEsto es una cadena constante"
```

El *operator+* devuelve el resultado en una nueva cadena. Se pueden concatenar también variables *char**

```
string f = a+b;
cout << f << endl; //Resultado: Holaaaa!!Esto es una cadena constante
```

Operadores relacionales

Todos los operadores de comparación de cadenas están incluídos:

```
if (e==d) cout << "e y d son iguales"<<endl; //lo cual es cierto
// en el siguiente código se ejecuta el else
string h = "abcd";
string i = "abcD";
if (h<i) cout << "abcd es menor que abcD" << endl;
else cout << "abcd no es menor que abcD" << endl;
```

Más operaciones de cadenas

El resto de operaciones con la librería *string* se realizan a través de funciones miembro. Daremos algunas de ellas.

Inserción

```
string j("casa");
j.insert(3,"er"); //inserta una cadena en una posición
cout << j << endl;  //imprime "casera"

string k("madera");
j.insert(4,k,3,6); //inserta el substring k[3..6] en la posición 4 de j
cout << j << endl; // imprime "caseerara"

char cad2[10] = "efghijk";
h.insert(2,cad2); //lo mismo pero para char*
cout << h << endl; //imprime "abefghijkcd"
h.insert(2,cad2,1,2); //también se puede hacer esto con una subcadena de cad2
```

Borrar

Podemos eliminar un rango o a partir de una posición

```
j.erase(3); //borra todo a partir de la posición 3
cout << j << endl; // imprime "cas"
j.erase(3,2); // borra 2 caracteres a partir de la posición 3
cout << j << endl; //imprime "casrara"
```

Reemplazar

replace es parecido a *insert* pero elimina caracteres. Existen más opciones para esta operación en la ayuda de STL.

```
string l ("abcd");
string m ("BC");
l.replace(1,2,m); //reemplaza 2 caracteres de *this en la posición 1 con m
cout << l << endl; // imprime: "aBCd"
```

Buscar

Existen numerosas opciones para realizar búsquedas, en todas ellas se devuelve la posición donde aparece la primera ocurrencia.

```
l = "abc-abc-abc";
//buscamos una ocurrencia en *this a partir de la posición 0 o de la 2
cout << l.find("bc",0) << endl; //imprime: 1
cout << l.find("bc",2) << endl; //imprime: 5
//Buscamos los 3 primeros caracteres "abcd-ario" como una subcadena
//de *this, comenzando por la posición 2 de *this
cout << l.find("abcd-ario", 2, 3) << endl; //imprime: 4
//buscaremos el caracter 'a' comenzando en la posición 5
cout << l.find('a',5) << endl; //imprime: 8
```

Existen otras versiones de *find* que permiten partir la búsqueda por el final, que buscan la primera cadena diferente, etc.

Otras operaciones válidas para contenedores secuencia

Podemos acceder en cualquier momento a un carácter utilizando el *operator[]*

```
cout << l[2] << endl; // imprime: c
```

Podemos saber si la cadena es o no vacía

```
string n;
if (n.empty() )
 cout << "Vacía"; //se ejecuta esta línea
else cout << "No vacía";
```

Conocer la longitud de la cadena

```
cout << l.length() << endl; //imprime: 11
```

Devolver en cualquier momento un char*

```
const char *cad3;
cad3 =  l.c_str();
cout << cad3[4] << endl; // imprime: a
```

Métodos de la clase string

String es una clase que ofrece facilidades para el manejo de cadenas de caracteres. En C o C++, las cadenas están representadas por el tipo de dato *char* * y existen librerías como <string.h> que contienen funciones que facilitan su uso. En la STL este tipo de funciones están implementadas bajo la clase string (para utilizar strings se incluirá la librería <string>).

Operaciones para el tipo de dato string

Constructores	
string s	Constructor por defecto
string s (“hola”)	Constructor con inicializador
string s (aString)	Constructor de copia
Acceso a elementos	
s[i]	Acceso al elemento i-ésimo del string
s.substr(int pos,int len)	Subcadena que comienza en pos y tiene longitud len
s.c_str()	Devuelve una cadena estilo C igual al string
Inserción y borrado	
s.insert(int pos,string str)	Insetar antes de pos el string str
s.erase (int start, int len)	Eliminar desde s[start] hasta s[start+len]
s.replace(int start, int len,str)	Sustituir desde s[start] hasta s[start+len] por str
Longitud	
s.length()	Longitud del string
s.resize(int,char)	Cambia el tamaño, rellenando con un valor
s.empty()	Cierto si el string es vacío
Asignación	
s = s2	Asignación de strings
s += s2	Concatenación de strings
s + s2	Nuevo string resultado de concatenar s y s2
Comparaciones	
s ==s2 s != s2	Igualdad y desigualdad de strings
s < s2 s <= s2	Comparaciones de strings (orden lexicográfico)
s > s2 s >= s2	Comparaciones de strings (orden lexicográfico)
Iteradores	
string::iterator itr	Declara un nuevo iterador
s.begin ()	Iterador que referencia al primer elemento
s.end ()	Iterador que referencia al siguiente al último

string::reverse_iterator ritr	Declara un nuevo reverse_iterator
s.rbegin ()	Reverse_iterator que referencia al último elemento
s.rend ()	Reverse_iterator que referencia al anterior al primero
Operaciones de búsqueda	
s.find(string str, int pos)	Devuelve la posición en donde comienza la subcadena str desde s[pos].
s.find_first_of(str,pos)	Posición en donde se encuentra el primer carácter que pertenece a str desde s[pos].
s.find_first_not_of(str,pos)	Posición en donde se encuentra el primer carácter que no está en str desde s[pos].
s.find_last_of(str,pos)	Posición en donde se encuentra el último carácter que pertenece a str desde s[pos].
s.find_las_not_of(str,pos)	Posición en donde se encuentra el último carácter que no está en str desde s[pos].
Operaciones E/S	
stream >> str	Entrada de strings
stream << str	Salida de strings
getline(stream,str,char)	Añade a str todos los caracteres de una línea de la entrada estándar hasta encontrar el carácter char. Por defecto char es igual a '\n'.