Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Ingeniería Informática

PROGRAMACIÓN ORIENTADA A OBJETOS

UNIDAD 5

La clase string

2011

UNIDAD 5

La clase string

Objetos string

La clase string de C++

La clase **string** de ANSI/ISO C++ implementa el tipo de datos basados en secuencias de caracteres y a través de sus funciones miembros permite procesar con facilidad cadenas de caracteres. Evitaremos en muchos casos la poca versatilidad de la biblioteca string.h con la cual operamos cstrings (strings al estilo C).

Esta clase pertenece a la biblioteca standard de ANSI C++ (no pertenece a la STL) y se la incluye en el programa con la cláusula: `#include <string>`
Al ser una clase, el tipo **string** define objetos o instancias y las funciones que estudiaremos para procesar cadenas, al ser miembros de esta clase se invocarán con la notación de punto: `objeto_string.función(...)`

Funciones miembro

La clase string define muchos métodos. Se describen a continuación algunos de los más usuales

---

Inicializando un string con su constructor
Un objeto string puede inicializarse de varias formas. Si no se especifica un valor inicial se asignará al objeto el string vacío (de longitud cero).

```
.....
string str1;
str1="Universidad";
string str2("Programación");
string str3(str2); //constructor de copia;
string str4="TE:0342-4556711";
.....
```

Inicializando un string con un substring y con un carácter
Otra forma de inicializar un objeto string es usando un constructor que permite copiar un substring del romer parámetro.

```
.....
string str1 = "Universidad”;
string str2 (str1,3,8); //extraer 8 caracteres desde el 3
cout<<strr2<<endl; // muestra “versidad”
.....
string str3 (1,'Z'); // asigna ‘Z’ a str3;
string str4 (‘Q’); // error: expresión no admitida
string str5=’R’;  // error: expresión no admitida
.....
```

Longitud de un objeto string

La longitud de un string puede obtenerse de las funciones `lenght() y size()`. Ambas devuelven un entero sin signo de tipo size_type con la cantidad de caracteres del objeto.

`size_type length() const;`
`size_type size() const;`

```
....
string str = "programa";
string::size_type largo;
largo = str.length(); // asigna 8 a largo
largo = str.size();   // asigna 8 a largo
....
```

Conversión a strings al estilo C

En ocasiones, por cuestiones de compatibilidad con programas que emplean cadenas al estilo C, debemos convertir un objeto string a cstrings. Uno de estos casos es el nombre físico de un archivo. C++ solo admite una cadena al estilo C para expresar el nombre (incluyendo la ruta de directorios) de un archivo físico. En C++ es posible convertir un objeto string a una cadena cstring a través de l la función miembro c_str( ) de la clase sting.

`const char* c_str() const;`

```
....
string cadena;
cout << "Entre el nombre del archivo: ";
cin >> cadena;
ofstream salida(cadena.c_str());
....
```

Insertar un string en otro

Para insertar un string en determinada posición de un objeto string existe la función miembro **`insert`**:

`string &insert(size_type pos, const string& str);`

```
....
string str1 = "Universidad del Litoral";
string str2 = "Nacional"+" ";
str1.insert (12,str2);
cout << str1 << endl; // "Universidad Nacional del Litoral"
....
```

Borrar parte de un string

Para borrar parte de un string se emplea la función miembro **erase**:

```
string &erase(size_type pos, size_type n);
```

```
....
string str1 = "Universidad";
str1.erase (3,5);
cout << str1 << endl; //muestra "Unidad"
....
```

Reemplazar

Para reemplazar una parte del string con otro substring C++ dispone de la función:

```
string &replace(size_type pos, size_type n,const string &str);
```

```
....
string str1 = "Programacion";
string str12 = "dor";
str1.replace (8,4,str2);
cout << str1 << endl; // "Programador"
....
```

Buscar una subcadena

La función miembro **find** busca la primer ocurrencia de un substring dentro de un string, a partir de una posición dada. Si lo encuentra devuelve la posición del primer carácter de dicha ocurrencia; si no, devuelve un valor especial llamado **string::npos**.

```
size_type find (const string& str, size_type pos);
```

La función **rfind** busca la última ocurrencia de un substring (la primera comenzando desde la derecha)

```
size_type rfind (const string& str, size_type pos);
```

Ambas funciones distinguen entre mayúsculas y minúsculas por lo que el código del siguiente ejemplo arrojaría como resultado el mensaje “No se encontró”.

```
...
string str1 = "abcdefghi";
string str2 = "def";
string::size_type pos = str1.find (str2,0);
cout << pos << endl; // devuelve 3
pos = str1.find ("AB",0);
if (pos == string::npos) cout << "No se encontró" << endl;
...
```

Extraer una subcadena

La función **substr** obtiene un substring de longitud n extrayéndolo de un objeto string.

```
string substr(size_type pos, size_type n);
```

```
...
string str1 = "abcdefghi"
string str2 = str1.substr(6,2);
cout << str2 << endl; // "gh"
...
```

Funciones no miembro

Además de las funciones miembro mencionadas, la clase string permite trabajar con funciones que no son miembro de la clase pero poseen una gran utilidad
La principal de ellas es **getline**( ), y la emplearemos con frecuencia al operar archivos de texto.

`istream& getline(istream& is, string& str, char delim ='\n');`

Esta función lee caracteres desde un flujo de entrada (stream) y los asigna a un string. La asignación termina cuando sucede alguna de las siguientes situaciones:
a) Aparece el fin del archivo en el flujo de entrada
b) Cuando se ha transferido el máximo número de caracteres que puede almacenar el string.
c) Cuando se lee el caracter que hace de delimitador para la asignación. Este caracter no es agregado al string y es removido del flujo de entrada. Por defecto es `'\n'`

El uso más común de **`getline`**( ) es leer un archivo de texto línea a línea. En el ejemplo siguiente se lee una serie de líneas desde la consola y se almacenen en una sola variable; luego se muestra la lista de líneas a través de esa variable.

```
...
int main(int argc, char* argv[]) {
        string linea, todas;
        while (getline(cin,linea,'\n'))
             todas=todas+linea+'\n';
        cout<<todas;
        return 0;;
}
```

Operadores

La clase string permite operar con los siguientes operadores de C++:
=, +, +=, ==, !=, <, >, <=, >=, <<, >>, [ ] (subindicación)

El operador de asignación (**=**) copia el contenido de una cadena en otra. Los operadores **+** y **+=** permiten concatenar cadenas, agregando el contenido de una al final de la otra.
Los operadores de comparación (**==**,**!=**,**<**,**>**,**<=**,**>=**) comparan dos cadenas caracter a caracter según el código ASCII.

```
string s1("c++"),s2("programador"),s3("programa");
bool b1 = s1<s2; // b1 toma true
bool b2 = s2>s3; // b2 toma true
bool b3 = s2.substr(0,8)==s3; // b1 toma true
```

Los operadores **<<** y **>>** son sobrecargas para que las clases istream y ostream (por ejemplo, cin y cout) acepten strings. Operan de la misma forma en que lo hacen sobre cstrings (arreglos de caracteres), leyendo por palabras, o mostrando todo el contenido.
El operador **[]** permite obtener y modificar un caracter. Por ejemplo, para pasar una cadena a minúsculas se puede obtener cada carácter con este operador, convertirlo a minúsculas con la función **char tolower(char)** y asignar el resultado nuevamente con el mismo operador.

```
string s1("PROGRAMA");
for (unsigned int i=0;i<s1.size();i++)
       s1[i] = tolower(s1[i]);
cout<<s1; // muestra "programa"
```

Arreglos de strings

Es posible organizar un arreglo de objetos strings, en forma similar a un arreglo de datos simples.

```
...
string  lista[10];
for (int i=0; i<10; i++)
    cin>>lista[i];
...
```