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

Ingeniería Informática

PROGRAMACIÓN ORIENTADA A OBJETOS

UNIDAD 5

La clase string

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);`

```
...
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 << "Not found" << 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)

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];
...
```

Clases y structs

Podemos observar que la *clases* y las *structs* son similares en sintaxis y más aún: las *structs* admiten funciones miembros privadas y públicas!.
La forma más general de una struct es:

```
struct s (
          datos miembros públicos;
          funciones miembros públicos;
     private:
          datos miembros privados;
          funciones miembros privados;
        );
```

Una diferencia entre clases y structs es que si no se especifica etiqueta de permisos, los miembros son públicos, a diferencia de las clases.
Las funciones miembros de un struct se implementan en forma idéntica a las funciones miembro de una clase.

El motivo de esta aparente confusión o ambigüedad se debe a que el diseño de C++ siempre ha intentado resguardar la compatibilidad con lenguaje C. Muchos programadores prefieren usar solo class para definir objetos y dejan las structs para definir entidades que carecen de funciones miembro.

Actividades

Ejercicios

Ejercicio 5.1.
Proponga una clase con métodos para incorporar un arreglo lineal de N elementos conteniendo palabras de hasta 12 caracteres, para obtener el primero de la lista de acuerdo a un orden alfabético, y extraer las palabras que comienzan con la sílaba 'mar'.

Ejercicio 5.2
Escriba un programa OO basado en una clase que posea un método llamado **contar**(..). Dicho método debe recibir como parámetro una frase cualquiera y determinar: a) La cantidad de vocales de la frase; b) La cantidad de consonantes; c) La cantidad de letras. Complete al clase con los atributos y métodos necesarios y proponga un programa cliente que la utilice.

Ejercicio 5.3.
Escriba un programa OO con un clase que permita ingresar como parámetro de uno de sus métodos una lista de apellidos y nombres de 10 personas (apellido y nombre se asignan a una sola variable). Utilice para cada persona una variable de tipo string. La clase debe obtener un listado con los 10 apellidos y luego un listado con

los 10 nombres. Considere apellidos formados por una única palabra (la presencia del primer espacio en blanco indica el fin del apellido). Pruebe la clase en un programa.

Ejercicio 5.4.
Proponga una clase que tenga como atributo una lista de apellidos y nombres de 10 profesores de la FICH-UNL (apellido y nombre se asignan a una sola variable). La clase obtener las direcciones de correo electrónicos (e-mails) de ellos. El dominio asignado a la Facultad para el e-mail es: fich.unl.edu.ar, y el nombre de usuario se forma con la inicial del nombre y el apellido. Proponga un programa cliente que haga uso de la clase.
Ejemplo: Si el dato es Marelli Jorge, debe obtenerse: jmarelli@fich.unl.edu.ar

Ejercicio 5.5.
Diseñe una clase que permita analizar los datos de n grupo de personas residentes en la ciudad de Santa Fe. Se ingresan por cada persona los datos siguientes: Apellido, Nombres, Calle y Número. Proponga métodos para detectar y mostrar los nombres de las personas que viven en la misma calle, indicando primero la calle correspondiente (no incluir aquellas personas que sean únicas en su calle.

Ejercicio 5.6.
Escriba un programa C++ que lea una lista de N palabras y acumularlas en un único objeto de tipo string llamado **lista_s**. Mostrar el objeto **lista_s** de modo que: a) las palabras aparezcan una junto a la otra separadas por un blanco