7 La clase string

Elaborado por: Ukranio Coronilla

A pesar de que las cadenas pueden usarse en C++ como lo hace el lenguaje C, es más conveniente utilizar la clase estándar ***string*** para manejar cadenas (llamaremos **C-string** al tipo de cadenas que se utilizan en C). Las cadenas en C requieren poner atención sobre los espacios de memoria para su almacenamiento, mientras que con la clase ***string*** no es necesario.

Los objetos de la clase ***string*** pueden ser concatenados con + o igualados con =. Por ejemplo para los objetos s1, s2 y s3:

```
s3 = s1 +s2;
```

Automáticamente se reserva memoria para que el objeto s3 mantenga la cadena s1 concatenada con s2. Aunque en C una cadena se escribe entre comillas, es válido por un cast interno convertirla a objeto ***string***:

```
s3 = "Esta es una cadena";
```

Otra forma equivalente de inicializar como se haría con un objeto es:

```
string s3("Esta es una cadena");
```

El siguiente es un objeto ***string*** vacío:

```
string frase;
```

el cual puede llenarse usando objetos ***string*** y cadenas C-string como sigue:

```
frase = s3 + "Cómo estás?";
```

Para hacer uso de la clase string es necesario incluir la libreria :

```
#include <string>
```

Los objetos ***string*** pueden funcionar con `cin >>` y `cout <<` como con cualquier tipo básico de datos. Sin embargo al usarse con `cin >>`, el objeto ***string*** solo almacenará el primer token. Para almacenar una línea completa se utiliza la función `getline` como sigue:

```
string line;
getline(cin, line);
```

En este caso la función `getline` lee caracteres del objeto `cin` (objeto fuente de caracteres provenientes del teclado o de un archivo) y los inserta en el objeto string `line` hasta que se encuentre el carácter salto de línea '\n', el cual no se incluye en el objeto line.

Ejercicio 1: En base a la explicación anterior pruebe que el siguiente código no funciona como se espera y explique por qué:

```
int n;
string line;
cin >> n;
getline(cin, line);
```

La clase ***string*** posee más de 100 funciones miembro para la manipulación de cadenas. Y también es posible acceder a elementos individuales, por ejemplo para el objeto ***string*** `nombre` el i-esimo caracter está contenido `en nombre[i]`. El número de caracteres almacenados en la cadena nombre se obtendría mediante:

```
nombre.length();
```

Ejercicio 2: Como sucede en C, es posible que se acceda a una posición no valida de la cadena, pruebe el siguiente código:

```
string cadena("ESCOM");
cout << cadena[6] << endl;
```

lo cual puede dar origen a una gran cantidad de errores en la programación. Para evitarlo se utiliza la función `at()`, pruebe su funcionamiento con el siguiente código :

```
string cadena("ESCOM");
cout << cadena.at(6) << endl;
```

A continuación se muestra una lista parcial de funciones miembro para la clase ***string***, suponiendo a str un objeto tipo ***string*** (para mayor información véase http://www.cplusplus.com/reference/string/) :

`str.substr(posicion, length)`	Devuelve una subcadena iniciando en posición y de longitud length.
`str.empty( )`	Devuelve true si es un ***string*** vacío, en caso contrario devuelve false.
`str.insert(pos, str2)`	Inserta str2 en str, comenzando en la posición pos.
`str.remove(pos,length)`	Elimina la subcadena de tamaño length e iniciando en la posición pos.
`str.find(str1)`	Devuelve el índice de la primera ocurrencia de str1 en str.
`str.find(str1, pos)`	Devuelve el índice de la primera ocurrencia de str1 en str, comenzando la búsqueda desde la posición pos.

A veces es necesario utilizar tanto la clase ***string*** como al tipo C-string. Para convertir un objeto ***string*** a una cadena tipo C-string es necesario utilizar la función `strcpy()` junto con la función miembro `c_str()`. El siguiente ejemplo muestra cómo se hace:

```
char aCString[];
string stringVariable;
       :
       :
strcpy(aCString, stringVariable.c_str( ));
```

Ejercicio3: Compararemos el desempeño de dos programas C++, en uno haremos uso exclusivo de las C-string y en otro solo de la clase ***string***. El programa consiste en generar n “palabras” de tres letras cada una y de contenido aleatorio (todas en mayúsculas) e irlas concatenando en una cadena gigante, manteniendo un espacio en blanco de separación entre cada palabra.

Posteriormente debe hacerse la búsqueda de la subcadena “IPN” en la cadena gigante y contabilizar el número de apariciones. ¿De acuerdo a la teoría de la probabilidad cuantas palabras deberían generarse para que se dé una ocurrencia? Mida los tiempos con el comando time y saque sus conclusiones.

Consideraciones:

Se recomienda utilizar las funciones ***append*** y ***find*** de la clase string, sin embargo se pueden utilizar otras.

El tiempo de ejecución del programa debe ser lo suficientemente grande para lograr una buena comparación.

Para evitar inequidad al usar las C-string se deberá hacer reserva dinámica de memoria cada que se incremente el tamaño de la cadena mediante realloc. Un pequeño ejemplo es el siguiente:

```
char *cadenota = NULL;
int posición;
cadenota = (char*)realloc(cadenota, posicion);
memcpy(cadenota, cadena, 4);
```

Optimice en ambos casos el código lo más posible.

IMPORTANTE:

Siempre utilice la clase string en sus clases para evitar cualquier problema al instanciar clases compuestas o heredadas, aunque puede manipular los caracteres con la forma de C.

Traer instalado el compilador java en Linux (javac) para la siguiente clase.