**PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ**
**FACULTAD DE CIENCIAS E INGENIERÍA**

LENGUAJE DE PROGRAMACIÓN 1
# PROGRAMACIÓN ORIENTADA A OBJETOS
# SOBRECARGA DE OPERADORES

## Indicaciones Generales:

## Problema

La mayoría de ustedes conoce la existencia en C++ de una biblioteca de funciones denominada ***<string>*** en la que se define una clase denominada también ***string***. Esta clase permite definir objetos para manejar cadenas de caracteres que, a diferencia del C estándar, son "más fáciles de manipular y más seguras". Sin embargo, como la mayoría comprende, esta clase no aparece de la nada, alguien se tomó un tiempo para diseñarla e implementarla en base a elementos más simples del lenguaje.

En este laboratorio usted replicará esta tarea implementado de una manera reducida una clase similar denominada "**Cadena**". La clase **Cadena** definirá los atributos y métodos necesarios que cumplan con los siguientes requerimientos:

**Contenedor:** La cadena se manejará de manera dinámica a través de un puntero. Deberá almacenar también la longitud de la cadena, de modo que no se tenga que contar caracteres cada vez que se le solicite y que se actualice en cada operación. Además, contará con un valor para la capacidad del arreglo que contiene la cadena, la cual no será necesariamente igual que la longitud y que también se debe actualizar.

**Inicialización:** Se podrá inicializar un objeto de esta clase de las formas siguientes:
- **`Cadena cad01;`** el contenedor se inicializa en nulo, la longitud de la cadena será cero al igual que la capacidad del arreglo que contiene el texto.
- **`Cadena cad02 ("Ana Roncal");`** la cadena es almacenada en el objeto de manera exacta, la longitud de la cadena será igual al número de caracteres asignado sin contar el cero, la capacidad será igual al espacio separado.
- **`Cadena cad03 (10);`** a la cadena se le separará un espacio de memoria (capacidad) igual al valor del parámetro. La longitud de la cadena será cero.
- **`Cadena cad04 = cad01;`** copiará de manera idéntica, toda la información de **`cad01`** en **`cad04`**.

**Asignación:** A un objeto de esta clase se le podrá asignar valores de las formas siguientes:
- **`cad01.assign ("Valentina");`** asigna la cadena al objeto. Si el objeto tiene capacidad para contener la cadena se le asigna sin modificar los espacios; si no la puede contener, se separará un espacio de memoria 30% mayor de la longitud de la cadena a asignar.
- **`cad01.assign (cad2);`** hace lo mismo que el método anterior pero la asignación se hace desde el objeto pasado como parámetro.
- **`cad01 = "Valentina";`** igual a lo anterior pero sobrecargando el operador =.
- **`cad01 = cad2;`** igual.
- **`cad01.append ("Valentina");`** agrega la cadena parámetro al final de la cadena del objeto. La asignación de memoria se maneja de igual manera que en la asignación.
- **`cad01.append (cad2);`** hace lo mismo que el método anterior pero la agregación se hace desde el objeto pasado como parámetro.
- **`cad01.swap (cad2);`** intercambia los contenidos de ambas cadenas. La asignación de memoria se maneja de igual manera que en la asignación.

**Concatenación:** Une dos cadenas en una nueva. La operación debe hacerse mediante el operador +; esto es, por ejemplo: **`cad01 + cad02`**. La operación no debe modificar los operandos, sólo debe devolver un objeto de tipo **`Cadena`** con la concatenación de ambas cadenas, de modo que al hacer **`cad03 = cad01 + cad02`**, a **`cad03`** se le asigne la concatenación.

**Comparación:** Se podrá comparar dos objetos de esta clase de las formas siguientes:

- **`cad01.compare ("Naomi");`** compara la cadena ingresada como parámetro con la del objeto `cad01`. Devuelve cero si son iguales, un valor mayor que cero si la cadena del objeto es mayor que la del parámetro y un valor menor que cero si es menor.
- **`cad01.compare (cad02);`** igual que la anterior pero la comparación es a través de otro objeto.
- **`cad01 == cad02`**, **`cad01 > cad02`**, **`cad01 < cad02`**, **`cad01 >= cad02`**, **`cad01 <= cad02`**: se comparan dos objetos **Cadena** mediante la sobrecarga de los operadores. Se devuelve **true** si se cumple la condición y **false** si no.

**Longitud:** Devuelve la longitud de la cadena (no debe contar caracteres aquí), de la siguiente manera: **`cad01.length ()`**.

**Impresión:** Imprime en la salida estándar la cadena de la manera siguiente: **`cad01.print ();`**

**Destructor**: Debe liberar los espacios asignados.

**Pruebas:** Deberá elaborar una aplicación que permita probar de manera sencilla cada uno de los métodos y sobrecargas.

**PRUEBA FINAL** (5 puntos): Se cuenta con un archivo de textos como el que se nuestra a continuación:

```
Pérez    Carpio    José
Roncal Neyra              Ana
…
```

En cada línea se encuentra el apellido paterno, el materno y el primer nombre de una persona. Las palabras estarán separadas por uno o más espacios en blanco. Usted deberá leer el archivo y guardar los nombres en un arreglo de tipo **Cadena** con la forma "Pérez/Carpio/José", ordenarlos empleando Quick Sort o algún otro algoritmo con similar eficiencia (n log n) y finalmente imprimirlos.

Elaborado por: Miguel Guanira E.

San Miguel, 16 de junio del 2020.