

Programación 4

LABORATORIO 1

Consideraciones Generales:

- La entrega podrá realizarse hasta el **lunes 28 de marzo a las 15:00 hrs.**
- El código fuente y los archivos Makefile de cada ejercicio deberán ser entregados mediante un formulario Web en la página del curso.
- Las entregas que no cumplan estos requerimientos no serán consideradas. El hecho de no realizar una entrega implica la insuficiencia del laboratorio completo.

Ejercicio 1

- Implementar en C++ el datatype String, que represente cadenas de caracteres de largo variable. Debe ser posible obtener un String tanto a partir de una cadena de caracteres (char *) como a partir de otro String.
- Agregar la sobrecarga de los siguientes operadores (tanto para String como para cadenas de caracteres):
 - Asignación.
 - Concatenación (operador +).
 - Comparación.
 - Acceso a un caracter del String (operador []).
- Agregar la sobrecarga de los operadores de inserción y extracción de flujo.
- Agregar las siguientes operaciones:
 - *length*, que retorna la cantidad de caracteres de la cadena.
 - *substring*, que retorna el substring que se encuentra entre dos posiciones dadas de la cadena (incluyendo los caracteres en dichas posiciones).
- Agregar manejo de excepciones de forma que:
 - Al acceder a una posición del String inválida se lance la excepción "std::out_of_range".
 - Al recibir un parámetro inválido en una operación se lance la excepción "std::invalid_argument".
- Realizar un procedimiento `main()` que permita, mediante un menú, probar todas las funcionalidades implementadas en las partes anteriores.

Ejercicio 2

Se desea implementar un mecanismo que permita poder ordenar un arreglo de objetos de la misma clase. Con el objetivo de que el código sea reusable se definió una clase *Comparable* con una operación ordenar que toma un arreglo de objetos de dicha clase y los ordena de menor a mayor.

La operación *comparar()* sirve para establecer un orden parcial entre los elementos retornando un entero tal que:

$a.comparar(b) > 0$ si $a > b$

$a.comparar(b) = 0$ si $a = b$

$a.comparar(b) < 0$ si $a < b$

Comparable
+comparar(c : Comparable) : int
+ordenar(a : Comparable [])

Algoritmo sugerido:

```

para cada i en [0..largo(a)-2]:
  indiceMin := i
  para cada j en [i+1..largo(a)-1]:
    si a[j].comparar(a[indiceMin]) < 0:
      indiceMin := j

temp := a[i]
a[i] := a[indiceMin]
a[indiceMin] := temp

```

Se pide:

- 1) Implementar en C++ la clase Comparable.
- 2) Dada una definición de una clase *Entero* como sigue:

Entero
-valor : int
+getValor() : int

Implementarla en C++ y hacer los cambios correspondientes para que Entero sea derivada de Comparable.

- 3) Hacer los cambios correspondientes en el String implementado en el ejercicio 1 para que derive de Comparable.
- 4) Realizar un procedimiento `main()` que permita, mediante un menú, probar todas las funcionalidades implementadas en las partes anteriores.