# 3 Constructores y parámetros por referencia en C++

Elaborado por: Ukranio Coronilla

Debido a que es muy común inicializar variables dentro del constructor, se dispone de otra forma que ocupa menos líneas de código y que mostraremos a continuación. Suponga que se tiene el siguiente constructor:

```
Fecha::Fecha(int dd, int mm, int aaaa)
{
   mes = mm;
   dia = dd;
   anio = aaaa;
}
```

La siguiente es una forma equivalente:

```
Fecha::Fecha(int dd, int mm, int aaaa) : dia(dd), mes(mm), anio(aaaa)
{/*Vacio intencional*/}
```

Es recomendable hacer en los constructores una validación para verificar que los datos pasados en el argumento son correctos. Por ejemplo:

```
Fecha::Fecha(int dd, int mm, int aaaa) : dia(dd), mes(mm), anio(aaaa)
{
   if((mes < 1) || (mes > 12)){
      cout << "Valor ilegal para el mes!\n";
      exit(1);
   }
   …
}
```

*Ejercicio 1*: Pruebe la nueva forma de declarar un constructor en el código del programa 2-2 y añada el código para verificar que los datos pasados en el argumento sean correctos (sólo valide 1< dia <31, y 0 < año < actual) . Pruebe que funciona la validación.

Un parámetro pasado por referencia es más eficiente que un parámetro por valor, sobre todo cuando se trata de una clase. Por ello se recomienda pasar objetos por referencia aun cuando la función no modifique el valor del objeto.

En C++ el paso de parámetros por referencia se logra añadiendo el símbolo & entre el tipo de dato y la variable; tanto en la declaración de la función como en el encabezado de la definición de la función.  Al compilador no le importa la posición del &, podría estar junto a la variable o junto al tipo de dato o entre ambas.

*Ejercicio 2*: Con ayuda del **comando** `time` (véase con man) y el programa 2-1, elabore un programa para probar que los datos pasados por referencia tienen mejor desempeño que los que

se pasan por valor. Para esto elabore las funciones  con parámetros por valor y por referencia cuyas declaraciones son las siguientes:

```
int masVieja(Fecha fecha1, Fecha fecha2);
int masVieja(Fecha &fecha1, Fecha& fecha2);
```

La función devuelve 1 si fecha1 > fecha2, 0 si son iguales y -1 si fecha1 < fecha2. La función debe llamarse n veces en un ciclo, y los valores miembro deben asignarse con números aleatorios antes de llamar a la función (véase la función rand() ). Es importante hacer una modificación adicional a la clase `Fecha` para observar diferencias importantes en la ejecución, para ello el objeto que se pasa debe tener un tamaño significativo, por ejemplo 1Kb o más.

Ejercicio 3: Realice la misma actividad que en el ejercicio 2 pero utilizando el paso por referencia al estilo del lenguaje C, con la declaración de función siguiente:

```
int masVieja(Fecha *fecha1, Fecha *fecha2);
```