PRÁCTICA 4ª: Ampliación de la clase implementada en la práctica 3 (**en sus dos versiones**).

OBJETIVOS: Manejo de los conceptos de constructor y destructor. Parámetros por defecto. Constructor copia.

TEMPORIZACIÓN:
Publicación del enunciado: Semana del 30 de septiembre.
Entrega: Semana del 21 de octubre junto con las prácticas 3 y 5.
Límite de entrega (con penalización): Semana del 28 de octubre.

BIBLIOGRAFÍA
Programación orientada a objetos con C++
Autor: Fco. Javier Ceballos
Editorial: RA-MA.

Se añadirán a la clase `CHora` implementada en la práctica 3ª los siguientes elementos:

- Un constructor que permita iniciar un objeto con ninguno, todos o algunos de los datos (horas, minutos, segundos y formato). Esto implica que se han de definir argumentos por defecto.
- Un constructor copia (no utilizar el operador = para la implementación de este constructor).
- Un destructor.
- La sobrecarga del operador de asignación.

Hacer que los constructores, el destructor y el operador de asignación visualicen un mensaje indicando que han sido invocados.

Hacer que el método `EsHoraCorrecta` de `CHora` sea público.

El programa principal se escribirá en un fichero `práctica4.cpp` y mostrará el siguiente menú desde su función `main`:

1. Crear un objeto local. Para ello defina un objeto **auto** (variable automática) en los bloques donde sea necesario. *(Esta opción mostrará el siguiente submenú)*

    a. Con una hora predeterminada.
    b. Introduciendo la hora.
    c. Introduciendo la hora y los minutos.
    d. Introduciendo la hora, los minutos y los segundos.
    e. Introduciendo la hora, los minutos, los segundos y el formato.
    f. Volver al menú principal.

2. Crear un objeto dinámicamente. Para ello, definir un puntero a nivel de **main** y crearlo invocando a **new** donde sea necesario. *El objeto será destruido en la opción 5*.

3. Constructor copia.
   *(Creará un objeto **auto** a partir del objeto dinámico del punto 2)*

4. Operador de asignación.
   *(Copiará en un objeto **auto** el dinámico del punto 2)*

5. Terminar *(liberará los objetos dinámicos)*

Cada opción invocará a la función externa *VisualizarHora* de la práctica 3.

En la práctica 2 escribió los ficheros `utils.h` y `utils.cpp` que incluían las funciones `LeerInt`, `LeerDouble`, `LeerCadena`, `CrearMenu`, `Minúsculas`, etc. Utilizando el concepto de función sobrecargada, escriba las sobrecargas de `bool LeerDato(`*`tipo`*`& dato)` que sustituyan a las funciones `Leer...` anteriores. Por ejemplo:
`LeerInt()` será sustituida por `bool LeerDato(int& dato)`
`LeerDouble()` será sustituida por `bool LeerDato(double& dato)`
`LeerCadena()` será sustituida por `bool LeerDato(string& dato)` y por
`bool LeerDato(char *dato)`
...

Todas estas funciones, junto con las funciones `CrearMenu`, `Minúsculas` y cualquier otra que se estime necesaria, se implementarán como métodos **static** de una clase `CUtils` perteneciente al espacio de nombres `utils`:

```
namespace utils
{
  class CUtils
  {
    public:
      ...
  };
};
```

En la versión de la práctica con `char *` debe ponerse un cuidado especial en el manejo de la memoria dinámica del miembro `m_pszFormato`. Es habitual poner a `0 (NULL)` los datos miembro de tipo puntero que no apuntan a ningún espacio de memoria reservada. Sin embargo, en esta práctica es más sencillo reservar memoria en los constructores y liberarla en el destructor (sin llegar a usar `NULL` para ese puntero en ningún momento), ya que las cadenas de formato de fecha tienen siempre la misma longitud.

Todas las funciones que reciban punteros como parámetros deberán contemplar la posibilidad de recibir el valor `NULL`.

En la versión de la práctica que utiliza el tipo `string` en lugar de `char *`, razone qué métodos de la clase `CHora` pueden eliminarse por resultar innecesarios.