PRÁCTICA 3ª: Creación y definición de una clase para el manejo de horas.

OBJETIVOS: Concepto de clase, atributos (datos miembro) y métodos (funciones miembro). Funciones externas. Métodos **`const`**.

TEMPORIZACIÓN:
Publicación del enunciado: Semana del 30 de septiembre.
Entrega: Semana del 21 de octubre junto con las prácticas 4 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 debe implementar la clase especificada por:

```
class CHora
{
  private:
    int m_nHoras;        // AM/PM: 1 a 12,  24 HORAS: 0 a 23
    int m_nMinutos;      // AM/PM/24 HORAS: 0 a 59
    int m_nSegundos;     // AM/PM/24 HORAS: 0 a 59
    char *m_pszFormato; //almacena los valores "AM", "PM" o "24 HORAS"

    bool AsignarFormato(char *pszFormato);
      // Verifica que pszFormato no es nulo.
      // Copia en m_pszFormato el formato dado por pszFormato,
      // asignando la cantidad de memoria necesaria, y lo convierte
      // a mayúsculas. Si fuera necesario, previamente se liberará
      // la memoria que hubiera asignada. Devuelve true si la
      // operación se realiza con éxito y false en caso contrario.

    bool Formato24() const;
      // Devuelve true si el formato es "24 HORAS".
      // En otro caso, devuelve false.

  protected:
    bool EsHoraCorrecta() const;
      // Verifica si una hora es correcta:
      // 1) verificar que el formato es 24 HORAS, AM o PM
      // 2) y verificar que las horas, minutos y segundos están dentro
      // de los límites según el formato sea o no 24 HORAS.
      // Devuelve true si es correcta y false en otro caso.

  public:
    void Iniciar();
      // Será invocada cada vez que se defina un objeto.
      // Pone a 0 todos los atributos de CHora.
    bool AsignarHora(int  nHoras,
                     int  nMinutos,
                     int  nSegundos,
                     char *pszFormato);
      // Recibe 4 argumentos, correspondientes a una hora,
```

```
      // y los almacena en las variables
      // m_nHoras, m_nMinutos, m_nSegundos y m_pszFormato
      // correspondientes al objeto que recibe el mensaje.
      // Invoca a AsignarFormato y a EsHoraCorrecta.
      // Devuelve false si no pudo ser asignado el formato
      // o el valor retornado por EsHoraCorrecta.

    void ObtenerHora(int& nHoras,
                     int& nMinutos,
                     int& nSegundos,
                     char *pszFormato) const;
      // Obtener una hora. Permite obtener los datos hora,
      // minutos, segundos y formato correspondientes
      // al objeto que recibe el mensaje.

    void Destruir();
      // Libera la memoria reservada dinámicamente para un objeto y
      // pone el puntero m_pszFormato a cero.

};
```

Guarde la declaración de la clase en un fichero `CHora.h`, y la definición en `CHora.cpp`.

El formato de la hora, AM, PM o 24 HORAS, se almacenará siempre en mayúsculas, pero se permitirá introducirlo en mayúsculas o minúsculas. De esta forma, las comparaciones se simplifican:

```
bool CHora::EsHoraCorrecta() const
{
  Si el formato es "AM" o "PM" o "24 HORAS"
     Si el formato no es de 24 horas
        Devolver true si la hora es correcta o false si no lo es
     Si el formato es de 24 horas
        Devolver true si la hora es correcta o false si no lo es
  Devolver false
}
```

Las tareas realizadas por los métodos `Iniciar` y `Destruir` son tareas propias de los constructores y destructores de las clases, por lo tanto, añada el constructor y el destructor de `CHora` para que invoquen, respectivamente, a `Iniciar` y `Destruir`. Esto le permitirá eliminar las llamadas a estos métodos; coméntelas (//).

El programa principal se escribirá en un fichero `práctica3.cpp` y deberá mostrar el siguiente menú:

```
1. Introducir hora
2. Visualizar hora
3. Terminar
```

Nota: si, para realizar las pruebas, tiene que declarar en `main` una matriz de N caracteres para almacenar el formato de la fecha, es preferible declararla así: `char formato[N]`, antes que así: `char* formato = new (nothrow) char[N]`, ya que esta última forma nos obliga a gestionar memoria.

También, cuando se invoque a la función `strcpy` debe garantizarse que el origen y el destino referencian bloques de memoria válidos, porque si alguno de ellos fuera `NULL` se generaría un error durante la ejecución.

```
if (destino != 0 && origen != 0) strcpy(destino, origen);
```

Para mostrar la hora se utilizará una función externa con el siguiente prototipo:

```
void VisualizarHora(const Chora& hora);
```

Utilice los ficheros `utils.h` y `utils.cpp` de la práctica anterior. Puede añadir a estos ficheros nuevas funciones o mejorar las existentes. Por ejemplo, estos ficheros deben definir, al menos, las siguientes funciones:

```
int CrearMenu(char *opciones_menu[], int num_opciones);
int LeerInt();
float LeerFloat();
void LeerCadena(char *c, int n);
char* ConverMayus(char* str);
std::string& ConverMayus(std::string& str);
```

La función `LeerCadena`, utilizando el método `basic_istream::getline`, permita leer una cadena de una determinada longitud o hasta encontrar un carácter delimitador (por ejemplo **`cin.getline(str, n, '\n')`**); en el caso de que la longitud de la cadena introducida sea mayor o igual que **`n`** se activará `ios::failbit`. La función `ConverMayus` está sobrecargada para cadenas tipo `char*` y para cadenas tipo `string`:

```
char* ConverMayus(char* str)
{
  if ( ... ) return 0;
  ...
  return str;
}

string& ConverMayus(string& str)
{
  for (int i ...)
    ... = toupper(str.at(i));
  return str;
}
```

Realizar otra versión del programa utilizando el tipo `string` en lugar de `char *`. No implemente los métodos de `CHora` anteriormente descritos que no sean necesarios al utilizar el tipo `string`, como, por ejemplo, `AsignarFormato`. En esta versión utilizará las funciónes `getline(cin, var_string)` para leer un `string` y `string& ConverMayus(string& str)` para convertir un `string` a mayúsculas.

```
class CHora
{
  private:
    int m_nHoras;
    int m_nMinutos;
    int m_nSegundos;
```

```
    std::string m_sFormato;

...
```