

### 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 04 de octubre.

**Entrega:** Semana del 8 de noviembre junto con las prácticas 4 y 5.

**Límite de entrega (con penalización):** Semana del 29 de noviembre

#### BIBLIOGRAFÍA

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
    string m_pszFormato;   //almacena los valores "AM", "PM" o "24 HORAS"

    bool AsignarFormato(string 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,
                    string 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,
                 string pszFormato) const;
// Obtener una hora. Permite obtener los datos hora,
// minutos, segundos y formato correspondientes
// al objeto que recibe el mensaje.

void Destruir();
// Muestra un mensaje indicando que el objeto se
// está destruyendo

};
```

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

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();  
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` 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;  
}
```

**Utilizará las funciones `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;  
  
    ...  
}
```