



Ejemplo documentación, sobrecarga de operadores, insertores y clase string



Ejemplo de documentación

```

/** Time.h -----
This header file defines the data type Time for processing time.
Basic operations are:
Set:      To set the time
Display:  To display the time
Advance:  To advance the time by a certain amount
LessThan: To determine if one time is less than another
-----*/

#include <iostream>
using namespace std;

struct Time
{
    unsigned hour,
        minute;
    char AMorPM;    // 'A' or 'P'
    unsigned milTime; // military time equivalent
};

```

documentación!

```

/* Set sets the time to a specified values.
*
* Receive:  Time object t
*           hours, the number of hours in standard time
*           minutes, the number of minutes in standard time
*           AMPM ('A' if AM, 'P' if PM
* Pass back: The modified Time t with data members set to
*           the specified values
*****/
void Set(Time & t, unsigned hours, unsigned minutes, char AMPM);

/* Display displays time t in standard and military format using
* output stream out.
*
* Receive:  Time t and ostream out
* Output:   The time T to out
* Pass back: The modified ostream out
*****/
void Display(const Time & t, ostream & out);

/* Advance increments a time by a specified value.
*
* Receive:  Time object t
*           hours, the number of hours to add
*           minutes, the number of minutes to add
* Pass back: The modified Time t with data members incremented
*           by the specified values
*****/
void Advance(Time & t, unsigned hours, unsigned minutes);

```

documentación!



```

/* Determine if one time is less than another time.
*
* Receive:  Times t1 and t2
* Return:   True if t1 < t2, false otherwise.
*****/
bool LessThan(const Time & t1, const Time & t2);

```

```

//===== Time.cpp -- implements the functions in Time.h =====
#include "Time.h"

/** Utility functions -- might be added as basic operations later **/

int ToMilitary(unsigned hours, unsigned minutes, char AMPM);

void ToStandard(unsigned military,
                unsigned & hours, unsigned & minutes, char& AMPM);

//... Definitions of Set, Display, Advance, LessThan, ToMilitary,
// and ToStandard go here --- see the text.

```



```

/** Time.h -----
This header file defines the data type Time for processing time.
Basic operations are:
Set:      To set the time
Display:  To display the time
-----*/

#include <iostream>
using namespace std;

class Time
{
/***** Member functions *****/
public:
/* Set sets the data members of a Time object to specified values.
*
* Receive:  hours, the number of hours in standard time
*           minutes, the number of minutes in standard time
*           AMPM ('A' if AM, 'P' if PM)
* Postcondition:  The Time object containing this function has its
*               myHours, myMinutes, and myAMorPM members set to hours,
*               minutes, and am_pm, respectively, and myMilTime to
*               the equivalent military time
*****/

```



```

/* Display displays time in standard and military format using
* output stream out.
*
* Receive:      ostream out
* Output:      The time represented by the Time object containing
*             this function
* Passes back: The ostream out with time inserted into it
*****/

```

```

void Display(ostream & out) const;

/***** Data Members *****/
private:
    unsigned myHours,
            myMinutes;
    char     myAMorPM; // 'A' or 'P'
    unsigned myMilTime; // military time equivalent
}; // end of class declaration

```



Mostrar un objeto

- En un objeto se puede incluir una función miembro que pase el objeto a un flujo de datos
 - No depende del dispositivo de salida
 - Si se le pasa *cout* como argumento lo muestra por pantalla

```

void Time::Display(ostream & out) const
{
    out << myHours << ':'
        << (myMinutes < 10 ? "0" : "") << myMinutes
        << ' ' << myAMorPM << ".M. ("
        << myMilTime << " mil. time)";
}

```



Sobrecarga de operadores

- En C++ el operador \square se puede implementar con la función `operator\square()`.
- Si la función de sobrecarga del operador es miembro de una clase C, el compilador trata a $a \square b$ como
 - `a.operator\square(b)`
- Si no es miembro de una clase C, el compilador trata a $a \square b$ como
 - `operator\square(a, b)`



Sobrecarga del insertor <<

- El insertor no puede ser una función miembro de la clase
 - `cout << objeto` → `cout.operator<<(objeto)`
- Implicaría una modificación de una clase estándar
 - `operator<<(const ClaseObjeto &)` debería ser un **miembro de la clase** `ostream` (o `cout` ser de tipo `ClaseObjeto`)



Definición de insertores

- Como un método externo a la clase
 - En este caso con `inline` se define dentro del `.h`

```

. . .
} // end of class declaration
. . .
/* operator<< displays time in standard and military
   format.
   Receive:  ostream out and Time object t
   Output:   time represented by Time object t
   Pass back: ostream out with t inserted into it
   Return:   out
*/

inline ostream & operator<<(ostream & out, const Time & t)
{
    t.Display(out);
    return out;
}

```



Definición de insertores

- Como una función amiga
 - Tiene acceso a los miembros privados de datos

En la declaración de la clase

```

/* documentación ... */
friend ostream & operator<<(ostream & out, const Time & t);

```

En la implementación (archivo .cpp)

```

ostream & operator<<(ostream & out, const Time & t)
{
    out << t.myHours << ':'
        << (t.myMinutes < 10 ? "0" : "")
        << t.myMinutes
        << ' ' << t.myAMorPM << ".M. ("
        << t.myMilTime << " mil. time)";
    return out;
}

```



Clase estándar string

- Clase predefinida para almacenar cadenas de caracteres de longitud arbitraria
 - Cabecera `<string>`
 - Espacio de nombres estándar `std`

<code>string cadena;</code>	Define un objeto cadena vacía
<code>string cadena("Balta");</code>	Define una cadena y la inicializa
<code>string nombre(cadena);</code>	Define una cadena y la inicializa
<code>string alias(nombre, 3);</code>	Define una cadena y la inicializa con los 3 primeros caracteres de <code>nombre</code>
<code>string cadAes('\B', 7);</code>	Define una cadena y la inicializa con 7 '\B'S



Operadores sobrecargados para string

operador	significado
>>	Extrae caracteres de un flujo de entrada y los inserta en la cadena hasta encontrar el primer blanco (para leer una línea entera <code>getline</code>)
<<	Inserta la cadena en el flujo de salida
=	Asignación de cadenas
+=	Concatena y asigna dos cadenas
+	Concatena dos cadenas
[]	Permite acceder a un caracter de una cadena utilizando la notación de los arrays (cuidado no hay comprobación de rango válido)
>, >=, <, <=	Operadores relacionales para la comparación de cadenas
<<=, >>=, !=	



Funciones miembro de string

- **Categorías:**
 - **asignación:** `assign`, `copy`, `data`
 - **modificación:** `append`, `clear`, `erase`, `insert`, `replace`, `swap`
 - **Gestión del espacio de almacenamiento:** `capacity`, `empty`, `length`, `resize`, `size`
 - **subcadenas:** `find`, `substr`
 - **comparación:** `compare`