Abstracciones (mente Humana)

Herencia: mecanismo de abstracción (programación/clasificación)

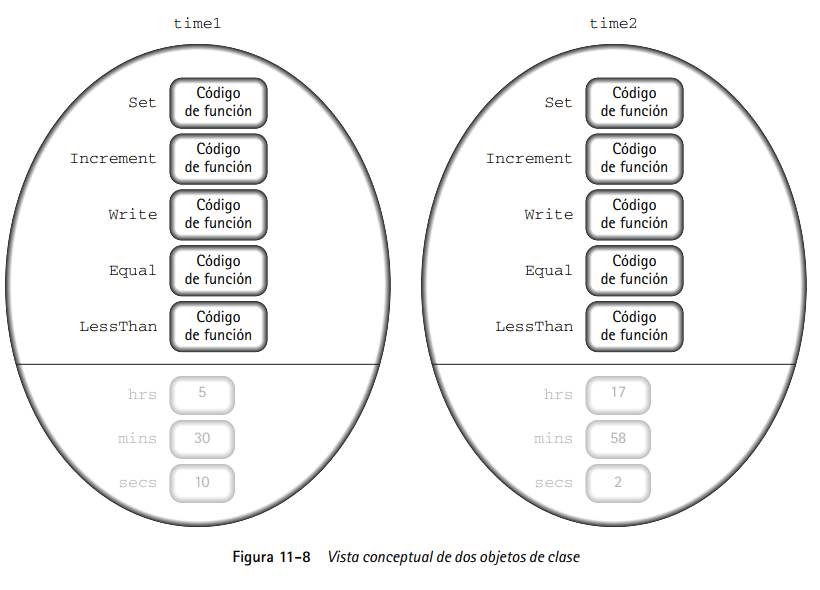
Es importante hacer hincapié en que una clase es un tipo, no un objeto de datos. Como cualquier tipo,  
una clase es un modelo a partir del cual se crean (o ejemplifican) muchos objetos de ese tipo.

class Time  
{  
public:  
 void Set( int, int, int );  
 void Increment();

void Write() const;  
 bool Equal( Time ) const;  
 bool LessThan( Time ) const;  
private:  
 int hrs;  
 int mins;  
 int secs;  
};

Las declaraciones  
Time time1;  
Time time2;  
crean dos objetos de la clase Time: time1 y time2. Cada función tiene sus propias copias de hrs,  
mins y secs, los miembros de datos privados de la clase.

(En realidad, el compilador de C++ no consume memoria al colocar copias de una función miembro, por ejemplo, Increment, en time1 y time2. El compilador genera sólo una copia física de  
Increment, y cualquier objeto de clase ejecuta esta única copia de la función. No obstante, el diagrama de la fi gura 11-8 es una buena imagen mental de dos objetos de clase distintos.)



Operaciones integradas en objetos de clase  
Por muchas razones, las clases definidas por el programador son como tipos integrados. Usted puede  
declarar tantos objetos de una clase como quiera. Puede pasar objetos de clase como argumentos a  
funciones y devolverlos como valores de función. Como cualquier variable, un objeto de clase puede  
ser automático (creado cada vez que el control alcanza su declaración y destruido cuando termina el  
programa).

Encapsulación Ocultar la ejecución de un módulo en un  
bloque separado con una interfaz formalmente especificada.

Debido a que la ocultación de información es fundamental para la abstracción de datos, la mayoría de las clases exhibe un patrón característico: la parte privada contiene datos y la parte pública las funciones que manejan los datos.

Como las funciones miembro de clase pueden tener acceso a datos privados, el creador de la clase  
puede ofrecer un producto confiable, sabiendo que el acceso externo a los datos privados es imposible. Si es aceptable permitir que el cliente inspeccione (pero no modifique) miembros de datos privados, una clase podría proporcionar funciones de observación. La clase Time tiene tres funciones de  
esa clase: Write, Equal y LessThan. Debido a que esas funciones de observador no están dedicadas  
a modificar los datos previos, se declaran con la palabra const seguida de la lista de parámetros:

void Write() const;  
bool Equal( Time ) const;  
bool LessThan( Time ) const;

C++ hace referencia a estas funciones como funciones miembro const. Dentro del cuerpo de una  
función miembro const, ocurre un error en tiempo de compilación si alguna sentencia intenta modificar un miembro de datos privado. Aunque aún no lo requiere el lenguaje, es buena práctica declarar como const las funciones miembro que no modifican datos privados.

Un miembro de una clase que ha sido definido utilizando la palabra clave public puede ser accedido desde afuera de la clase, siempre y cuando esté dentro del ámbito del objeto de la clase.

Datos Dinámicos: p 655 de ‘Programación y Resolución de Problemas con C++’

Diseño orientado a objetos  
Secciones 14.6 y 14.7 de ‘Programación y Resolución de Problemas con C++’