Unidad Nº3: Encapsulamiento y Polimorfismo

*Preguntas orientadoras*

1. El modificador const situado a la derecha de los paréntesis, en la declaración de un método es para que no se modifique el objeto con el cual estamos llamando a la función. Por ejemplo: Yo tengo un objeto de la clase complejo y quiero saber cuál es ese mismo complejo cambiado de signo, es decir, tengo c1 = 4+2i y quiero saber –c1, entonces yo voy a llamar a el método que sobrecargue el operador “– “que tendrá una declaración de la forma: complejo operator- () const; donde el const me dice que yo quiero que la función me devuelva el complejo cambiado de signo, pero que el objeto en si no se modifique, es decir que yo cuando salga de la línea que pide el “-c1 = -4-2i”, c1 siga valiendo 4+2i.
2. Cuando dos clases deben ser declaradas mutuamente friend se debe utilizar una declaración anticipada (forward declaration) de una de ellas en el archivo de declaración:  
   class <clase2>; // declaración anticipada para el compilador  
   class <clase1>

{

/\* funciones y variables miembro de clase1\*/

friend clase2;

};  
class <clase2>

{

/\* funciones y variables miembro de clase2 \*/

friend clase1;

};

1. Diferencias entre un constructor de copia y el operador de asignación (=):

Un “constructor de copias” de copia (copy constructor): inicializa un nuevo objeto copiándolo desde otro objeto de la misma clase. Tiene un único argumento que es una referencia constante al objeto fuente. El compilador proporciona un constructor de copia de oficio que realiza una copia bit a bit de las variables miembro (no deseado con punteros = no copia los contenidos de la memoria sino las direcciones). Al trabajar con punteros el constructor de copia debe realizarlo el programador para gestionar espacios de memoria distintos y luego copiar sus contenidos.

El operador de asignación “=” copia los datos de un objeto de forma idéntica a otro objeto. El operador de asignación copia un objeto a otro miembro. Si no sobrecarga el operador de asignación, realiza una copia bit a bit. Cuando se realiza una asignación bit a bit, el objeto comparte la misma ubicación de memoria y los cambios de un objeto, se reflejan en el otro objeto.

En conclusión: El constructor de copia es mejor para copiar un objeto a otro cuando el objeto contiene punteros.



Un constructor de copia se invoca en las siguientes condiciones:

1. Cuando se inicializa un nuevo objeto con uno existente.
2. Cuando a una función se le pasan objetos como argumentos por valor.
3. Cuando una función tiene un objeto como valor de retorno.
4. Al sobrecargar funciones miembro de una clase, deben diferir en la cantidad o el tipo de parámetros que reciben. Ya que el nombre debe ser el mismo, la única forma de que el compilador sepa cual usar en tiempo de ejecución, es viendo la cantidad o el tipo de parámetros en la llamada a dicha función.
5. Los valores predeterminados en una función se utiliza para además de inicializar, poder llamar a la función con un numero distinto de parámetros o hasta sin parámetros y que tomen esos valores por default, esto no lo podría hacer por más que yo sobrecargue una función. Por ejemplo, yo tengo un numero complejo y por default pongo que valgan “0” tanto la parte imaginaria como la compleja, y yo quiero en la llamada a el constructor inicializar solamente la parte real y que la imaginaria quede en “0” puedo llamar al constructor como: complejo (7.5) y el complejo será: 7.5 + 0i. O hasta simplemente llamarla sin parámetros y va a tomar los valores por default.
6. Los únicos operadores que no se pueden sobrecargar son el punto (.), el if ternario (?:), el sizeof, el scope resolution (::) y puntero a miembro de un objeto (.\*).
7. const\_cast: es un operador de uso muy específico /\*\*\*\*\* Ejemplo que sirve para eliminar el modificador const de una referencia o un puntero. Actúa durante la compilación. Sintaxis: const\_cast<tipo> (<objeto>);  
   Es importante advertir que el operador no cambia el tipo del operando; no hace que una variable constante pueda volverse no-constante y ver alterado  
   su valor. Es útil para pasar un objeto constante a una función que espera una referencia. Aunque su uso no es común también se aplica al modificador volatile.