Capítulo 11. Funciones Avanzadas

La sobrecarga de funciones se trata de la posibilidad de tener múltiples funciones con el mismo nombre pero que se diferencian con los parámetros de las misma, por ejemplo, ustedes pueden tener unas funciones así:

1)void miFuncion(); 2)void miFuncion(int var1); 3)void miFuncion(long var2, short var3);

NOTA: el compilador no creará los constructores predeterminados si nosotros declaramos algún otro, por lo que deberá generar el constructor predeterminado nosotros mismo. los constructores también se puede sobrecargarlos, y esta es una poderosa y flexible herramienta para múltiples usos.

Aquí vemos el constructor que creamos con sus inicializadores correspondientes para generar los valores predeterminados, para iniciarlos se declara el valor entre paréntesis al lado de la variable, si se debe inicializar más de una variable se separan con una coma (,).

también pueden inicializarse en el cuerpo de la definición del constructor, pero en realidad de esta forma es un poco más limpia y eficiente, esto siempre queda a criterio del programador.

sabemos que tenemos un constructor y un destructor predeterminado, tambien existe un constructor de copia predeterminado y es llamado cada vez que se crea la copia de un objeto. Todos los constructores de copia toman como referencia objetos de la misma clase, por eso es una buena práctica definirlo como constante para que no pueda ser modificado cuando se pase, un ejemplo:

GATO (const GATO & elGato);

Como vemos el constructor toma como referencia una constante de la misma clase y el objetivo del constructor de copia es hacer una copia del Gato. Esto se llama copia de datos miembro (o superficial) que es la copia de cada variable del objeto que se pasa como parámetro a las variables del nuevo objeto. Esto es correcto para la mayoría de las variables miembro, pero con los apuntadores a objetos en el heap puede generar inconvenientes. Supongamos que GATO tiene una variable su Edad que apunta al heap, el constructor de copia predeterminado copiara la variable del nuevo GATO, Un constructor de copia hace una copia exacta de todos los valores, y en el caso de los apuntadores tambien utilizan la misma dirección de memoria, suponiendo que realizamos esta acción, la variable su Edad del nuevo GATO apuntara a la misma dirección de su Edad del viejo GATO. Y como dijimos en apuntadores, el destructor se encargará de limpiar la memoria, en el caso del GATO original si llama al destructor libera esa memoria, y el nuevo GATO sigue apuntando a esa dirección de memoria y esta acción podría generar un apuntador perdido y por ende haría que el programa se vuelva completamente inestable. La solución a este inconveniente es crear su propio conductor de copia y asignar la memoria según sus necesidades. Esto se conoce como copia profunda.