

Clases y Sobrecarga de operadores

Programación Avanzada
Año 2009
Cuat. II



Clases

- Las clases son conceptos extendido de estructura de datos en DOO y POO, donde además de poseer también puede contener funcionalidades.
- Un **Objeto** es una instancia de una clase.

Formato de declaración de una Clase

```
class class_name {  
    especificador_de_acceso_1:  
    member1;  
    especificador_de_acceso_2:  
    member2;  
    ...  
} object_names;
```

Especificadores de Acceso

- **private:** los miembros de la clase son accesibles sólo dentro de otros miembros de la misma clase o de sus amigas
- **protected:** los miembros de la clase son accesibles dentro de otros miembros de la misma clase, de sus amigas y además de las clases derivadas.
- **public:** los miembros son accesibles desde cualquier ámbito donde el objeto sea visible

Ejemplo:

```
class CRectangle {  
    int x, y;  
    public:  
        void set_values (int,int);  
        int area (void);  
} rect;
```

Constructores

- Son funciones miembro utilizadas para inicializar variables y asignar memoria.
- El formato es el de un método miembro que tiene el mismo nombre de la clase. Ver ejemplo.

Destruyores

- Funciones miembro llamada automáticamente cuando se destruye un objeto.

Puntero *this*

- Apuntador que permite a un objeto tener acceso a su propia dirección de memoria.
- No forma parte del objeto. Tener en cuenta para la función ***sizeof***.
- Pasa como primer argumento implícito en las funciones miembro

Consideraciones sobre el puntero *this*

- No se puede modificar.
- Debe usarse (*this).fmiembro(...);
- No funciona con funciones estáticas.

¿Qué es la sobrecarga de operadores?

Según **Bruce Eckel**: “La sobrecarga de operadores es *azúcar sintáctica*. Es sólo otra forma de hacer una llamada a una función”

Fundamentos de la sobrecarga de operadores

- Notación más clara para tipos de datos definidos por el programador.
- Adecuada para clases matemáticas
- Permite expresiones claras para nuevos tipos de datos creados por el usuario

Sintaxis

- La sintaxis de la sobrecarga depende de dos factores:
 - Si el operador es unario o binario.
 - Si el operador esta definido como función Miembro o No Miembro.

Sintaxis

- Tipos de operadores en C++ que pueden sobrecargarse:
 - Unarios: **++**, **--**, **!**, etc
 - Binarios: **+**, *****, **/**, etc
- Tipo **operator@**(parámetros)
 - @ puede ser cualquier operador: **+**, **++**, *****, etc.

Formas de Sobrecargar operadores

- A través de métodos miembro
- A través de métodos NO miembro con funciones amigas.
- A través de métodos NO miembro con funciones NO amigas (globales)

Ejemplos de sobrecarga

- **operador+** como miembro.
- **operador+** como no miembro.
- **operador<<** como no miembro (friend).
- **operator++** como miembro
- **operator()** como miembro
- **Operator+=** como miembro

Restricciones en sobrecarga de operadores

- Los operadores = y & en general no se sobrecargan.
- No se puede cambiar la precedencia de un operador mediante sobrecarga
- La asociatividad de un operador no se cambia con la sobrecarga

Restricciones en sobrecarga de operadores

- No se puede cambiar la precedencia de un operador mediante sobrecarga
- La asociatividad de un operador no se cambia con la sobrecarga
- Hay operadores que no se pueden sobrecargar (ej: `.*`, `..`, `::`, etc)

Restricciones en sobrecarga de operadores

- Los operadores de versiones unarias y binarias deben sobrecargarse por separado
- No se pueden crear operadores a través de la sobrecarga.

Casos de sobrecarga

Expresión	Operador (@)	Función miembro	Función global
@a	+ - * & ! ~ ++ -	A::operator@()	operator@(A)
a@	++ -	A::operator@(int)	operator@(A, int)
a@b	+ - * / % ^ & < > == != <= >= << >> && ,	A::operator@(B)	operator@(A, B)
a@b	= += -= *= /= %= ^= &= = <<= >>= []	A::operator@(B)	-
a(b, c...)	()	A::operator()(B, C...)	-
a->b	->	A::operator->()	-