CURSO C++ 139_170

Vídeo 139: Sobrecarga de operadores Introducción

En C++ los operadores se implementan como si fueran funciones, así que sobrecargando las funciones predeterminadas de los operadores, podemos crear versiones personalizadas.

Esto se entiende mejor con un ejemplo. El compilador viene con una función integrada del operador "+" para operandos enteros:

```
std::cout << x + y;
```

si x e y fueran int. la suma se vería en realidad como

```
int operator+ (int x, int y){}
```

¿Cómo evalúa el compilador una expresión que contenga un operador? Estas son las reglas que sigue:

1. Si todos los operandos son tipos de datos fundamentales, llama a la función integrada. Si no existe, lanza un error de compilación.

2. Si alguno de los operandos son tipos definidos por el usuario, el compilador busca si el tipo tiene una función de operador sobrecargada coincidente a la que pueda llamar.

3. Si después del paso 2 no encuentra coincidencias, intentará convertir el tipo definido por el usuario a un tipo fundamental, si no puede lanzará un error.

Otro tema a tener claro son las limitaciones de la sobrecarga de operadores.

1. Casi todos los operadores pueden sobrecargarse, a excepción de los siguientes: `Condicional (?:) , sizeof, scope (::), selector de miembro (.), selector de puntero miembro (.*), typeid y los operadores de conversión.`

2. Solo se pueden sobrecargar los operadores ya existentes, no crear nuevos operadores.

3. Al menos uno de los operandos en un operador sobrecargado debe ser un tipo definido por el usuario.

4. No es posible cambiar el número de operandos que admite un operador.

5. Todos los operadores mantienen su prioridad y asociatividad predeterminadas.

Como mejores prácticas intenta mantener la funcionalidad de los operadores sobrecargados similar a la de los originales. Si no es así, es mejor crear funciones con nombre.

Vídeo 140: Sobrecarga de operadores con funciones amigas

En C++ hay tres modos diferentes de sobrecargar los operadores con:

- Funciones normales
- Funciones amigas
- Funciones miembro

Sobrecargar por ejemplo la suma es tan sencillo como crear una función usando `operator+`

En el siguiente ejemplo se usa una clase llamada `Euros` y la variable miembro `int m_euros{};` El siguiente prototipo de función iría dentro de la clase:

```
friend Euros operator+ (const Euros& e1, const Euros& e2);
```

Y se define la función fuera de la clase. Se usa `friend` al ser una función amiga. Solo comentar que una función amiga puede definirse también por completo dentro de una clase, pero si lleva la palabra clave "`friend`", esa función no se considera miembro de la clase, aunque esté definida dentro de ella.

Una vez definida, podremos sumar dos operandos de tipo `Euros` sin problemas ya que, en este caso, `m_euros` es de tipo int y el compilador sabe como sumar tipos fundamentales..

Para la resta, multiplicación y división se sobrecarga del mismo modo que la suma, cambiando el símbolo por el que le toque (- * /).

Cuando C++ evalúa una expresión "x + y", si x e y son del mismo tipo, es lo mismo "x + y" que "y + x". Pero si los operandos son de tipos diferentes, "x + y" llama a una función diferente de "y + x".

Por lo tanto, siguiendo con el ejemplo anterior, si en vez de sumar dos tipo Euros queremos sumar un tipo int y un tipo Euros, debemos crear dos funciones, una para cada caso.

```
friend Euros operator+ (int valor, const Euros& e1);
friend Euros operator+ (const Euros& e1, int valor);
```

Vídeo 141: Sobrecarga de operadores con funciones normales