


Sobrecarga de Operadores

 Sobrecargar un operador consiste en modificar su comportamiento cuando este se utiliza con una determinada clase.

 Sintaxis de un Operador Sobrecargado:

[acceso] static TipoRetorno operator nombreOperador (Tipo a[, Tipo b])

```
{  
  
    ...  
  
}
```

Nota: El modificador de acceso no podrá ser de un ámbito mayor que el de la clase.

Operadores Sobrecargables

Operadores	Tipo de Operadores
+, -, !, ~, ++, --, true, false	Unarios.
+, -, *, /, %, &, , ^, <<, >>	Binarios.
==, !=, <, >, <=, >=	Comparación. (*)

(*) Nota: Los operadores de Comparación, si son sobrecargados, se deben sobrecargar en pares;

es decir, si se sobrecarga el operador ==, se deberá sobrecargar el operador !=.

Operadores No Sobrecargables

Operadores	Tipo de Operadores
&&, 	Condicionales Lógicos.
[]	Indexador de Array. (*)
()	Casting. (**)
+=, -=, *=, /=, %=, &=, =, ^=, <<=, >>=	Asignación. (***)
=, ., ?:, ->, new, is, sizeof, typeof	Estos operadores no se pueden sobrecargar.

Notas

(*) Indexador de Array:

- Se pueden definir indexadores.


(**) Casting:


- Se pueden definir nuevos operadores de conversión.


(***) Asignación:

- El operador +=, por ejemplo, es evaluado usando el operador +, el cual puede ser sobrecargado.

Operadores de Conversión - Conversiones Definidas

 Las conversiones definidas permiten hacer compatibles tipos que antes no lo eran.

 Los operadores de conversión pueden ser implícitos o explícitos.

 Los operadores de conversión implícitos son más fáciles de usar, aunque los operadores de conversión explícitos son muy usados cuando se quiere que los usuarios estén conscientes que una conversión se llevará a cabo.

Sintaxis - Operadores de Conversión

 Implícitos

```
public static implicit operator tipoRetorno(Tipo a)
{
    ...
}
```

 Explícitos

```
public static explicit operator tipoRetorno (Tipo a)
{
    ...
}
```

Ejemplo # 2: Main (con Explicit)

Metro metros = (Metro) 10;

Centimetro centimetros = (Centimetro) 10;

Metro SumaMetros = metros + centimetros;

Centimetro SumaCentimetros = centimetros + metros;

Console.WriteLine((double) SumaMetros);

Console.WriteLine((double) SumaCentimetros);

No se utilizan constructores.

No se utilizan atributos.

Código más fácil de entender

Ejemplo # 2: Clase Metro

```
public class Metro
{
    public double cantidad;

    // Constructores y Métodos
    public static explicit operator Metro (double cant)
    {
        Metro retValue = new Metro (cant);
        return retValue;
    }
    public static explicit operator double (Metro m)
    {
        return m.cantidad;
    }
}
```

Ejemplo # 2: Clase Centímetro

```
public class Centimetro
{
    public double cantidad;

    // Constructores y Métodos
    public static explicit operator Centimetro (double cant)
    {
        Centimetro retValue = new Centimetro (cant);
        return retValue;
    }
    public static explicit operator double (Centimetro m)
    {
        return m.cantidad;
    }
}
```

Ejemplo # 2: Main (con Implicit)



Si se coloca *implicit* en vez de *explicit*, en la sobrecarga de los operadores de conversión, el Main sería el siguiente:

```
Metro metros = 10;
Centimetro centimetros = 10;

Metro SumaMetros = metros + centimetros;
Centimetro SumaCentimetros = centimetros + metros;

Console.WriteLine(SumaMetros);
Console.WriteLine(SumaCentimetros);
```