

C# Operatoren

Primär	(x) f(x) a[x] new typeof sizeof checked unchecked
Unär	+ - ~ ! ++x --x x++ x-- (T)x
Multip./Divis.	* / %
Addition/Subtr.	+ -
shift	<< >>
kleiner/größer	< > <= >= is
gleich/ungleich	== !=
Logisch AND	&
Logisch XOR	^
Logisch OR	
Bedingtes AND	&&
Bedingtes OR	
Bedingungsop.	c ? x:y
Zuweisung	= += -x *= /= %= <<= >>= &= x= =

Die Operanden für das bedingte AND (&&) und OR (||) werten den 2. Operanden nur dann aus, wenn er einen Einfluß auf das Endergebnis hat. (wird auch als Short Circuit bezeichnet).

C# Arithmetische Ausdrücke

Operandentypen

- numerisch oder char
- bei ++ und -- numerisch oder enum (funktioniert auch bei float und double!)

Ergebnistyp

Kleinsten numerischer Typ, der beide Operandentypen einschließt.

C# Vergleichsausdrücke

Operandentypen

- bei <, >, <=, >=: numerisch, char, enum
- bei ==, !=: numerisch, char, enum, bool, Referenzen
- bei x is T: x: Ausdruck mit beliebigem Typ, T: Referenztyp
z.B.: obj is Rectangle
objOfValueType is IComparable
3 is object
arr is int[]

Ergebnistyp

bool

Mit Hilfe des is-Operators kann man testen, ob ein Objekt zu einem bestimmten Typ kompatibel ist

C# Boolesche Ausdrücke (&&, ||, !)

Operandentypen

bool

Ergebnistyp

bool

Kurzschlußauswertung (bedingte Auswertung)

if (a && b) ... wenn a==false → b wird nicht mehr ausgewertet → false-Zweig

if (a || b)... wenn a==true → b wird nicht mehr ausgewertet → if-Zweig

Nützlich z.B. bei

if (p != null && p.val > 0) ...

if (x == 0 || y / x > 2) ...

C# Überlaufprüfungen

Normalerweise wird Überlauf nicht erkannt

```
int x = 1000000;  
x = x * x; // -727379968, kein Fehler
```

Überlaufprüfung

```
x = checked(x * x); // liefert System.OverflowException
```

```
checked {  
    ...  
    x = x * x; // liefert System.OverflowException  
    ...  
}
```

alle im checked-Block enthaltenen Anweisungen werden nun überprüft.
es gibt auch eine Compiler-Option, um Überlaufprüfung generell einzuschalten:
csc /checked Test.cs

C# Diverses

typeof

- liefert *Type*Objekt zu einem Typ

(*Type*Objekt eines Objekts *x* kann mit *x.GetType()* abgefragt werden).

```
Type t = typeof(int) ;  
Console.WriteLine(t.Name); // liefert Int32
```

sizeof

- gibt die Größe eines Typs zurück
- kann nur auf Werttypen angewendet werden
- kann nur im unsafe-Kontext benutzt werden (unportable oder gefährliche Konstrukte)
Muß übersetzt werden mit `csc /unsafe xxx.cs`

```
unsafe {  
    Console.WriteLine(sizeof(int) );  
    Console.WriteLine(sizeof(MyEnumType) );  
    Console.WriteLine(sizeof(MyStructType) );  
}
```