Expressions

Operators and their Priority

Primary	(x) x.y f(x) a[x] x++ x-- new typeof sizeof checked unchecked
Unary	+ - ~ ! ++x --x (T)x
Multiplicative	* / %
Additive	+ -
Shift	<< >>
Relational	< > <= >= is as
Equality	== !=
Logical AND	&
Logical XOR	^
Logical OR	|
Conditional AND	&&
Conditional OR	||
Conditional	c?x:y
Assignment	= += -= *= /= %= <<= >>= &= ^= |=

Operators on the same level are evaluated from left to right (e.g. in x + y - z).

The unary operators +, -, ~, ! as well as type casts are evaluated from right to left:

```
- (int) x  ⇔ - ((int) x)
```

Arithmetic Expressions

Operand types must be

- numeric or *char*
- operands of ++ and -- must be numeric or enumeration constants
  (++ and -- work also on *float* and *double*!)

Result type

Smallest numeric type that includes both operand types, but at least *int*.

Note

```
- uint    => long
- ulong   => illegal

uint    •  (sbyte | short | int)          => long
ulong   •  (sbyte | short | int | long)   => illegal
decimal •  (float | double)               => illegal
```

Comparisons

Operand types

- <, >, <=, >=:	numeric, *char*, *enum*
- ==, !=:	numeric, *char*, *enum*, *bool*, reference types
- x is T:	*x*: expression of arbitrary type, *T*: reference type

e.g.:
```
obj is Rectangle
objOfValueType is IComparable
3 is object
arr is int[]
```

Result type

bool

Boolean Expressions (&&, ||, !)

Operand types

bool

Result type

bool

Short-circuit evaluation (conditional evaluation)

```
a && b    =>   if (a) b else false
a || b    =>   if (a) true else b
```

Useful in

```
if (p != null && p.val > 0) ...
if (x == 0 || y / x > 2) ...
```

Bit Expressions (&, |, ^, ~)

Operand types

- & (and):
  | (or): } integer type, *char*, *enum*, *bool*
  ^ (xor):
- ~ (not): integer type, *char*, *enum*
- if the operand types are not identical the operand with the smaller type is converted to the larger type prior to the operation

Result type

- largest of the two operand types
- for numeric types and *char* the result type is at least *int*

Shift Expressions

Operand types for x << y and x >> y

- *x*: integer type oder *char*
- *y*: *int*

Result type

type of *x*, but at least *int*

Note

>> does a *logical shift* for unsigned types and an *arithmetic shift* for signed types

Overflow Checks

Overflow is not checked by default

```
int x = 1000000;
x = x * x;   // -727379968, no error
```

Overflow checks can be turned on

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

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

Overflow checks can also be turned on with a compiler switch

```
csc /checked Test.cs
```

typeof and sizeof

typeof

- Returns the *Type* descriptor for a given type
  (the *Type* descriptor of an object *o* can be retrieved with *o.GetType()*).

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

sizeof

- Returns the size of a type in bytes.
- Can only be applied to value types.
- Can only be used in an unsafe block (the size of structs may be system dependent).
  Must be compiled with   csc /unsafe xxx.cs

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