# About Numbers

One of the key aspects of working with numbers in C# is the distinction between integers (numbers with no digits after the decimal separator) and floating-point numbers (numbers with zero or more digits after the decimal separator).

The two most commonly used numeric types in C# are int (a 32-bit integer) and double (a 64-bit floating-point number).

int i = 123;

double d = 54.29;

Both integers and floating-point numbers can use the \_ character as a *digit separator*, which can help when defining large numbers:

int largeInt = 1\_000\_000;

// => 1000000

double largeDouble = 9\_876\_543.21;

// => 9876543.21

Arithmetic is done using the standard **[arithmetic operators](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/arithmetic-operators" \t "https://exercism.org/tracks/csharp/concepts/_blank)** (+, -, \*, etc.). Numbers can be compared using the standard **[comparison operators](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/comparison-operators" \t "https://exercism.org/tracks/csharp/concepts/_blank)** (<, >=, etc.) and the **[equality-](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/equality-operators" \t "https://exercism.org/tracks/csharp/concepts/_blank)** operator (==) and **[inequality](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/equality-operators" \t "https://exercism.org/tracks/csharp/concepts/_blank)** operator (!=).

5 \* 6;

// => 30

1.2 > 0.8

// => true

2 != 4

// => true

When converting between numeric types, there are two types of numeric conversions:

1. Implicit conversions: no data will be lost and no additional syntax is required.
2. Explicit conversions: data could be lost and additional syntax in the form of a cast is required.

As an int has less precision than a double, converting from an int to a double is safe and is thus an implicit conversion. However, converting from a double to an int could mean losing data, so that requires an explicit conversion.

int i = 9;

double d = 2.66;

// Safe conversion, thus implicit conversion

double fromInt = i;

// Potentially unsafe conversion, thus explicit conversion

int fromDouble = (int)d;