Syntax

- die "Sätze" (Statements bzw. Anweisungen) der Sprache C# bestehen aus diversen Komponenten:
  - Literale: das sind Werte, die man entweder buchstäblich angibt oder die nicht aus anderen Werten berechnet werden, also atomare Bestandteile der Sprache.
    - Beispiele: `1`, `1.25`, `"abc"`, `'a'`, `true`, `false`

Eingebaute Datentypen

- `int` - vorzeichenbehaftete ganze Zahl mit 32 Bit
- `double` - vorzeichenbehaftete Zahl mit Nachkommastellen (64 Bit)
  - kann ungefähr 15 signifikante Dezimal-Ziffern darstellen
- `string` - Repräsentiert Zeichenketten
- `bool` - präsentiert die Wahrheitswerte `true` und `false`
- `char` - präsentiert ein einzelnes Zeichen

Operator

- Operatoren bilden aus ihren Operanden einen Wert eines bestimmten Datentyps

```
1 + 1           // int + int => int
1.0 + 1         // double + int => double
"abc" + "uvw"   // string + string => string
3 / 4           // int / int => int
3.0 / 4         // double / int => double
3D / 4          // double / int => double
```

Typkonvertierung

- Compiler konvertiert Werte unterschiedlichen Datentyps nur dann automatisch (implizit), falls die Umwandlung zu *keinem* Datenverlust führen kann
- Ist ein Datenverlust bei einer Umwandlung möglich, müssen wir *explizit* konvertieren

- die explizite Typkonvertierung ist nicht für beliebige Datentypen möglich. Zum Beispiel kann man einen `string` nicht per expliziter Umwandlung in einen `int` konvertieren.

```
int integer = 3.0; // Fehler: Compiler konvertiert double nach int nicht automatisch, da Datenverlust möglich
int integer = (int)3.0; // OK: explizite Konvertierung von int nach double. Datenverlust wird in Kauf genommen.
string name = (string)1.25; // Fehler: keine explizite Typkonvertierung möglich
```

Parsing

- der Vorgang, bei dem aus einer Zeichenkette Daten extrahiert und in einen anderen Datentypen umgewandelt werden, heißt *Parsing*
- viele eingebaute Datentypen unterstützen eine `Parse` Methode, mit der Zeichenketten geparst werden können
- standardmäßig werden beim Parsing die aktuell verwendeten regionalen Einstellungen (*Culture*) verwendet
    - parst man zum Beispiel eine Zeichenkette in eine Gleitkommazahl, dann muss für den Dezimaltrenner das Komma verwendet werden, sofern man die deutschen regionalen Einstellungen verwendet. In der US-Amerikanischen Kultur ist der Punkt zu verwenden.

```
string input = "123";
int number = int.Parse(input);  // Parsen einer Zeichenkette in eine ganze Zahl
double d = double.Parse(input); // Parsen einer Zeichenkette in eine Gleitkommazahl
DateTime dt = DateTime.ParseExact("13.04.2023", "dd.MM.yyyy", CultureInfo.CurrentCulture); // string -> DateTime
```

Umwandlung eines Wertes in einen String

- jeder Datentyp besitzt die Methode `ToString` mit der sich ein Objekt in eine Zeichenkette konvertieren lässt
- standardmäßig berücksichtigt `ToString` die regionalen Einstellungen (im deutschsprachigen Raum wird Komma statt Punkt als Dezimaltrenner

verwendet)

```
1.ToString();          // erzeugt "1"
(2.45).ToString();     // erzeugt "2.45"
(2.45).ToString("F1"); // erzeugt "2.5"
DateTime.Now.ToString("yyyy-MM-dd");
```