2.3.1 Variablen Deklaration & Initialisierung

Variablen, die innerhalb einer Methode wie beispielsweise Main deklariert sind, gelten noch nicht als initialisiert. Sie enthalten keinen gültigen Wert, auch nicht die Zahl 0. Daher kann ihr Inhalt auch nicht ausgewertet werden.

2.3.2 .Net-Namenskonventionen

Ein Bezeichner darf sich nur aus alphanumerischen Zeichen und dem Unterstrich zusammensetzen. Leerzeichen und andere Sonderzeichen wie beispielsweise #, §, $ usw. sind nicht zugelassen.

Ein Bezeichner muss mit einem Buchstaben oder dem Unterstrich anfangen.

Ein einzelner Unterstrich als Variablenname ist nicht zulässig.

Der Bezeichner muss eindeutig sein. Er darf nicht gleichlautend mit einem Schlüsselwort, einer Prozedur, einer Klasse oder einem Objektnamen sein.

2.3.4 Formatausdrücke in den Methoden Write() und WriteLine()

Der Formatausdruck `{}` dient nicht nur der eindeutigen Bestimmung des Elements, er ermöglicht auch eine weitergehende Einflussnahme auf die Ausgabe. Soll der einzusetzende Wert eine bestimmte Breite einnehmen, gilt die syntaktische Variante:

```
{N, M}
```

Dabei gilt Folgendes:

- `N` ist ein nullbasierter Zähler.
- `M` gibt die Breite der Ausgabe an.

Unbesetzte Plätze werden durch eine entsprechende Anzahl von Leerzeichen aufgefüllt. Sehen wir uns dazu ein Codefragment an:

```
int value = 10;
Console.WriteLine("Ich kaufe {0,3} Eier", value);
Console.WriteLine("Ich kaufe {0,10} Eier", value);
```

Listing 2.8 Erweiterte Formatierungsmöglichkeiten

Die Ausgabe des Listings 2.8 lautet hier:

```
Ich kaufe  10 Eier
Ich kaufe        10 Eier
```

Formatangaben

Die Breite darf auch eine negative Zahl sein. Die Ausgabe erfolgt dann linksbündig, daran schließen sich die Leerstellen an.

Sie können den Formatausdruck so spezifizieren, dass numerische Ausgabedaten eine bestimmte Formatierung annehmen. Das führt uns zu der vollständigen Syntax des Formatausdrucks:

```
// Syntax des Formatausdrucks
{N [,M ][: Format]}
```

Format spezifiziert, wie die Daten angezeigt werden. In Tabelle 2.1 werden die möglichen Optionen aufgelistet.

Formatangabe	Beschreibung
C	Zeigt die Zahl im lokalen Währungsformat an.
D	Zeigt die Zahl als dezimalen Integer an.
E	Zeigt die Zahl im wissenschaftlichen Format an (Exponentialschreibweise).
F	Zeigt die Zahl im Festpunktformat an.
G	Eine numerische Zahl wird entweder im Festpunkt- oder im wissenschaftlichen Format angezeigt. Zur Anzeige kommt das »kompakteste« Format.
N	Zeigt eine numerische Zahl einschließlich Kommaseparatoren an.
P	Zeigt die numerische Zahl als Prozentzahl an.
X	Die Anzeige erfolgt in Hexadezimalnotation.

Tabelle 2.1 Formatangaben der Formatausgabe

Escape-Zeichen

Escape-Zeichen	Beschreibung
\'	Fügt ein Hochkomma in die Zeichenfolge ein.
\''	Fügt Anführungsstriche ein.
\\	Fügt einen Backslash in die Zeichenfolge ein.
\a	Löst einen Alarmton aus.
\b	Führt zum Löschen des vorhergehenden Zeichens.
\f	Löst einen Formularvorschub bei Druckern aus.
\n	Löst einen Zeilenvorschub aus (entspricht der Funktionalität der ↵-Taste).
\r	Führt zu einem Wagenrücklauf.
\t	Führt auf dem Bildschirm zu einem Tabulatorsprung.
\u	Fügt ein Unicode-Zeichen in die Zeichenfolge ein.
\v	Fügt einen vertikalen Tabulator in eine Zeichenfolge ein.

Tabelle 2.2 Die Escape-Zeichen

Escape-Zeichen ignorieren

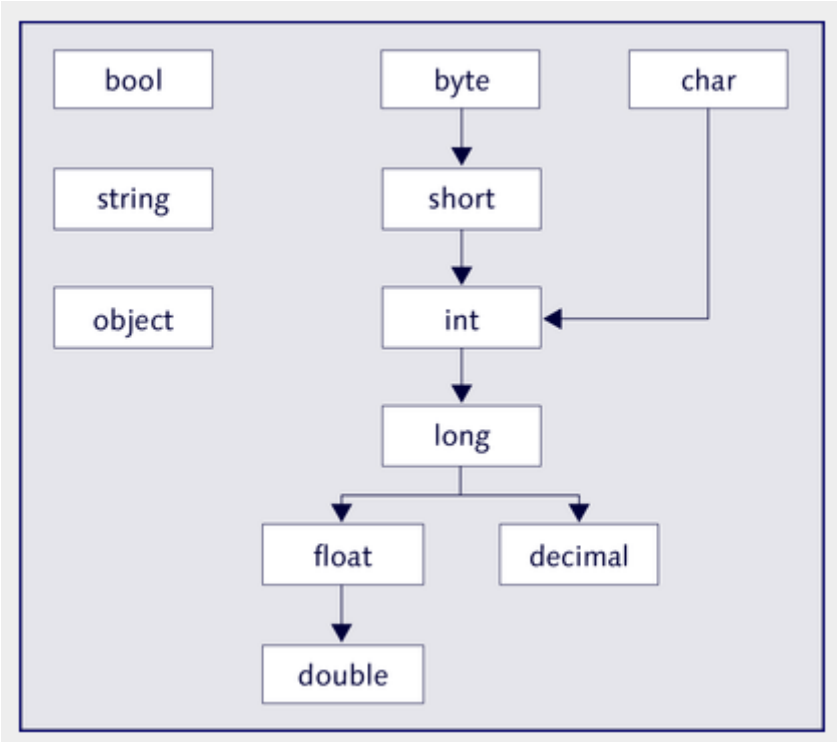
Um die Interpretation als Escape-Sequenz für eine gegebene Zeichenfolge vollständig abzuschalten, wird vor der Zeichenfolge das Zeichen »@« gesetzt.

```
Console.Write(@"C#\nmacht\nSpaß.");
```

2.3.5 Boolean

In vielen Programmiersprachen wird `false` numerisch mit 0 beschrieben und `true` durch alle Werte, die von 0 abweichen. .NET ist hier viel strenger, denn `true` ist nicht 1 und auch nicht 67, sondern ganz schlicht `true`.

2.3.6 Implizite Konvertierung



Explizite Konvertierung

In zwei ganz wesentlichen Punkten unterscheidet sich die Konvertierung mit den Methoden der `Convert`-Klasse von der mit dem Konvertierungsoperator:

- Es können Konvertierungen durchgeführt werden, die mit dem Typkonvertierungsoperator »`()`« unzulässig sind. Allerdings sind die Methoden der Klasse `Convert` nur auf elementare Datentypen beschränkt.
- Grundsätzlich werden alle Konvertierungen mit den Methoden der `Convert`-Klasse auf einen eventuellen Überlauf hin untersucht.

Die Operatoren »checked« und »unchecked«

```
// Beispiel: ..\Kapitel 2\CheckedSample

static void Main(string[] args) {

  // Zahleneingabe anfordern

  Console.Write("Geben Sie eine Zahl im Bereich von ");
  Console.Write("0...{0} ein: ", Int16.MaxValue);

  // Eingabe einem short-Typ zuweisen

  short value1 = Convert.ToInt16(Console.ReadLine());

  // Überlaufprüfung einschalten

  byte value2 = checked((byte)value1);
  Console.WriteLine(value2);
  Console.ReadLine();
}
```

Listing 2.15 Arithmetischen Überlauf mit »checked« prüfen

Nach dem Starten der Anwendung wird der Benutzer dazu aufgefordert, eine Zahl im Bereich von 0 bis zum Maximalwert eines `short` einzugeben. Entgegengenommen wird die Eingabe durch die Methode `Console.ReadLine`, die ihrerseits die Eingabe als Zeichenfolge, also vom Typ `string` zurückliefert. Um die gewünschte Zahl einer `short`-Variablen zuweisen zu können, muss explizit konvertiert werden. Beachten Sie bitte, dass wir dazu die Methode `ToInt16` der Klasse `Convert` einsetzen müssen, da eine Konvertierung eines `string` in einen `short` mit dem Typkonvertierungsoperator nicht zulässig ist:

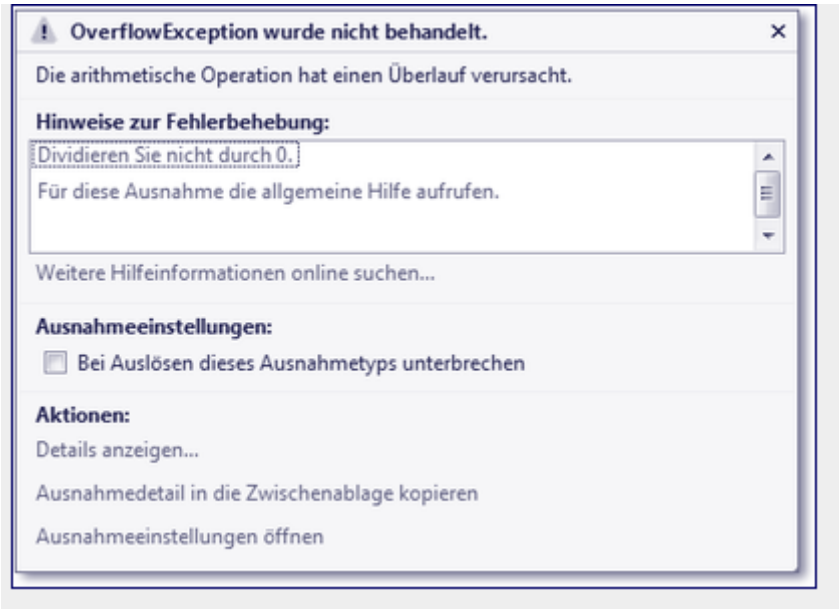
```
short value1 = Convert.ToInt16(Console.ReadLine());
```

Gibt der Anwender eine Zahl ein, die den Wertebereich des `short`-Typs überschreitet, wird ein Laufzeitfehler ausgelöst und die Laufzeit der Anwendung beendet. Falls der Wertebereich nicht überschritten wird, wird die dann folgende Anweisung ausgeführt:

```
byte value2 = checked((byte)value1);
```

In dieser Anweisung steckt allerdings eine Gemeinheit, denn nun soll der Inhalt der `short`-Variablen einer `byte`-Variablen zugewiesen werden. Je nachdem, welche Zahl der Anwender eingegeben hat, wird die Zuweisung fehlerfrei erfolgen oder – bedingt durch die Überprüfung mit `checked` – zu einem Fehler führen. Löschen Sie `checked` aus dem Programmcode, wird die Zuweisung einer Zahl, die den Wertebereich eines `byte`-Typs überschreitet, keinen Fehler verursachen.

`checked` ist ein Operator und wird verwendet, um einen eventuell auftretenden arithmetischen Überlauf zu steuern. Tritt zur Laufzeit ein Überlauf ein, weil der Anwender eine Zahl eingegeben hat, die den Wertebereich des Typs überschreitet, in den konvertiert werden soll, wird ein Laufzeitfehler ausgelöst, der unter .NET auch als Ausnahme bzw. Exception bezeichnet wird. Geben wir beispielsweise an der Konsole die Zahl 436 ein, werden wir die folgende Mitteilung erhalten:



„unchecked“ unterdrückt eine Exception.