

*„Gute Programmierer erkennt
man an ihrem Datendrang.“
- Klaus Klages*

3 Daten

In diesem Kapitel wollen wir uns mit dem wichtigsten Bereich der Programmierung beschäftigen: den Daten. Egal welches Programm wir betrachten, im Inneren läuft immer der gleiche Vorgang ab: Daten werden eingelesen, verarbeitet, gespeichert und ausgegeben. Damit das reibungslos funktionieren kann, brauchen wir ein grundlegendes Regelwerk, das bestimmt, wie die einzelnen Arten von Daten behandelt, verwendet und gespeichert werden.

3.1 Datentypen

Bevor wir anfangen uns mit praktischen Beispielen zu beschäftigen, müssen wir einiges an Theorie hinter uns bringen, damit sinnvolle Beispiele möglich werden.

3.1.1 Numerische Datentypen

Als erstes wenden wir uns der größten, aber auch einfachsten Gruppe der Datentypen zu: den numerischen Datentypen. Egal ob wissenschaftliche Berechnungen, Finanzmathematik, Computerspiele oder online Shops; Zahlen werden immer gebraucht. Hier ist eine Entscheidung für den richtigen Datentyp für den Anfänger besonders schwer, da es sich um die größte Gruppe der Datentypen handelt und somit oft Unsicherheit bei der Auswahl des passenden Datentyps herrscht.

- 20) Welche Gruppen von numerischen Datentypen kennen Sie? Nennen Sie mindestens die zwei Hauptgruppen.
- 21) Nennen Sie möglichst viele unterschiedliche Datentypen für ganze Zahlen. (Vergessen Sie hierbei nicht die Unterscheidung in rein positive und positive/negative Zahlen.)
- 22) Wieviele sbyte passen in einen int (nicht mathematisch, rein vom verwendeten Speicherplatz)?
- 23) Wieviele byte passen in einen ulong?
- 24) Welcher Bezeichner für einen ganzzahligen Datentyp bildet eine Ausnahme bei der Verwendung von negativen Zahlen?
- 25) Versuchen Sie folgenden Werten einen passenden Datentyp zuzuordnen:
 - a. 17
 - b. 123456
 - c. -987
 - d. -200L
 - e. 0x7AF3UL
 - f. -0X7E
 - g. 0x004488CC
- 26) Welcher Wertebereich ist mit einem byte, sbyte, short bzw. ushort abbildbar?
- 27) Mit welchen Datentypen kann man Gleitkommazahlen darstellen?
- 28) Wie sind die Datentypen float und double aufgebaut?



- 29) Wie ist der Datentyp decimal aufgebaut?
- 30) Welche Auswirkung auf die Genauigkeit der Nachkommastellen hat die Größe einer Zahl, die als double abgespeichert wurde?
- 31) Nennen Sie fünf Anwendungsbeispiele für den Datentyp double.
- 32) Nennen Sie zwei Anwendungsbeispiele für den Datentyp decimal.
- 33) Ist es möglich den Wert eines ulong in einem double abzuspeichern?
- 34) Versuchen Sie folgenden Werten einen passenden Datentyp zuzuordnen:
 - a. 3.141592654
 - b. 2.718281828
 - c. 0.2M
 - d. 987654321.12345
 - e. -123.45
 - f. 0.0
 - g. 34.56f
 - h. -0.5F
 - i. 3.14E+3
 - j. 1.0e120
 - k. -0.5E-1000
- 35) Welche zwei Werte kann der Datentyp bool annehmen?

3.1.2 Referentielle Datentypen


Im nun folgenden Abschnitt wollen wir uns mit der Gruppe der referentiellen Datentypen beschäftigen. Zu diesen komplexeren Datentypen gehören etwa Klassen, Interfaces, Arrays und Delegates. Aufgrund der Kenntnisse, die für die meisten referentiellen Datentypen notwendig sind, werden im folgenden ausschließlich Beispiele für string, class und boxing erarbeitet.



- 36) Erklären Sie den Begriff „string“.
- 37) Versuchen Sie die Begriffe „Klasse“, „Objekt“ und „Instanz“ zu erklären.
- 38) Wozu dient die Klasse object?
- 39) Wann spricht man von boxing?
- 40) Nennen Sie einen praktischen Anwendungsfall von boxing.
- 41) Was ist unboxing?

3.2 Variablen & Konstanten

Die Sprache C# definiert sieben verschiedene Kategorien von Variablen: statische Variablen, Instanzvariablen, Array Elemente, Wertparameter, Referenzparameter, Ausgabeparameter und lokale Variablen. In den nun folgenden Übungen werden nur die einfachsten Formen von Variablen benötigt. In den nachfolgenden Kapiteln werden wir auch die komplexeren Kategorien verwenden.

- 42) Schreiben Sie Variablendefinitionen mit Initialisierung für die Werte aus den Übungen 25 und 34. Geben Sie anschließend den Wert jeder Variablen, gefolgt von einem Zeilenumbruch aus.
- 43) Deklarieren Sie einen string str1 und weisen Sie ihm den Wert „Hello“ zu. Anschließend definieren Sie einen string str2 und weisen ihm den Wert „Hello World!“ zu, verwenden Sie hierfür die vorher erstellte Variable str1.
- 44) Wie unterscheidet sich die Lebensdauer von Variablen, die innerhalb der Main-Methode deklariert wurden von denen, die ausserhalb erstellt wurden? 
- 45) Was unterscheidet eine Konstante von einer Variablen?
- 46) Erstellen Sie eine Konstante „PI“ mit dem Wert 3.1415, die in der gesamten Klasse gilt.
- 47) Erstellen Sie eine Variable vom Typen double mit dem Namen „Winkel“, die den Wert $90 \cdot \text{PI} / 180.0$ erhält. Verwenden Sie dazu die Konstante aus der letzten Übung.
- 48) Welche der folgenden Variablennamen sind nicht gestattet?
 - a. F.Schubert
 - b. ersteZahl
 - c. _Anton
 - d. \$atom
 - e. 12terWert
 - f. neue Eingabe
 - g. _123abc
 - h. cba321_
- 49) Welche der folgenden Variablendeklarationen sind gültig?
 - a. int a;
 - b. int b=a=3;
 - c. double _ganz;
 - d. char erstes zweites drittes;
 - e. bool abc, def, ghi;
 - f. short x=5*3;
 - g. string ausgabe=x.ToString();
- 50) Warum produziert der folgende Quelltext einen Fehler beim Übersetzen?

```
int a;
a = 10;
int b, a;
```

3.3 Typkonvertierung

Vielleicht haben Sie schon einmal die folgende oder eine ähnliche Fehlermeldung beim Programmieren gelesen:

error CS0029: Cannot implicitly convert type 'string' to 'int'

Diese Fehlermeldung sagt aus, dass hier versucht wurde einer Variablen vom Typ int den Wert eines Strings zuzuweisen. Uns Menschen ist der Unterschied zwischen dem Text „123“ und der Zahl 123 nicht immer bewußt bzw. wichtig. In C# ist es allerdings nötig (so wie in den meisten anderen Hochsprachen auch) spezielle Klassen oder Funktionen für die Umwandlung eines Wertes von einem Datentyp in einen anderen zu verwenden.

Im nun folgenden Abschnitt kommen spezielle Übungen mit mehreren Datentypen zur Anwendung. Das Arbeiten mit Daten verschiedenen Typs muß für jeden professionellen Programmierer so selbstverständlich sein wie das kleine Einmaleins.

- 51) Verwenden Sie beliebig viele der Variablen aus Übung 42 und konvertieren Sie diese mit Hilfe der ToString() Methode in einen String. Geben Sie anschließend diesen String aus.
- 52) Erstellen Sie einen String „123“ und Variablen vom Typ byte, short, int, long und double. Konvertieren Sie den String mit Hilfe der Klasse Convert in die jeweiligen Variablen-Typen und geben Sie diese anschließend am Bildschirm aus.
- 53) Erstellen Sie eine Variable vom Typ int. Verwenden Sie anschließend die Methode Console.ReadLine() um eine Zahl aus der Eingabeaufforderung einzulesen und diese in der eben erstellten Variable abzuspeichern. Vergessen Sie hierbei nicht auf die notwendige Konvertierung.
- 54) Schreiben Sie das letzte Beispiel so um, dass anstatt einer ganzen Zahl eine Gleitkommazahl eingegeben werden kann. Konvertieren Sie diese Zahl anschließend in eine ganze Zahl. Geben Sie hierauf beide Zahlen aus.

Bei den den letzten beiden Übungen taucht nun ein interessantes Problem auf: Versuchen Sie z.B. statt einer Zahl ihren Namen einzugeben. Was passiert? Richtig: das Programm bringt eine Fehlermeldung und beendet sich anschließend selbst. Dieses Verhalten ist zwar technisch korrekt, aber nicht gerade angenehm für den Benutzer unseres Programms. Wir werden in einem späteren Kapitel (Exceptions) noch sehen, wie man die Erkennung und Korrektur solcher Fehler bei der Eingabe durchführen kann.

- 55) Wird der folgende Quelltext funktionieren? Begründen Sie Ihre Meinung.

```
int a;
double b;
byte c;
short d;
float e;
string str1 = "123Grad";
string str2 = "123.45";
string str3 = "123 Grad";
a = Convert.ToInt32(str1);
b = Convert.ToDouble(str2);
c = Convert.ToByte(str3);
d = Convert.ToInt16(str2);
e = Convert.ToDouble(str2);
```