Formatierung von Zahlen in C#

Häufig haben wir das Problem, dass eine lange Fließkommazahl als Ergebnis entsteht, z. B. bei einer Division. Wenn man diese in einem Label ausgeben will, stören die vielen Nachkommastellen.

Der Quelltext sieht normalerweise folgendermaßen aus:

```
double x=2.0/3;
label1.Text=x.ToString();
```

Um die Ausgabe zu verändern, schreiben wir einen so genannten Formatstring zwischen die Klammern bei ToString. Dabei haben folgende Zeichen eine besondere Bedeutung:

Zeichen	Bedeutung
0	Die Null wird als Stellen- oder Nullplatzhalter verwendet. Verfügt die Zahl an der Position über eine Stelle, an der im Formatstring die 0 erscheint, erscheint diese Stelle im Ergebnis. Falls nicht, erscheint an dieser Stelle eine Null.
#	Das Rautenzeichen wird als Stellen- oder Leerplatzhalter verwendet. Die Raute funktioniert genauso wie die 0. Es erscheint aber ein Leerzeichen, wenn die Stelle nicht belegt ist.
.	Erster Punkt im Formatstring: Trennungszeichen zwischen ganzer Zahl und Nachkommastellen
,	Trennungszeichen für Tausenderblöcke

Für die Bedeutung weiterer Zeichen findet man in der Hilfe oder im Internet weitere Informationen (für die Suche angeben: „c# format strings“), z. B. auf Englisch bei Microsoft unter
http://msdn2.microsoft.com/en-us/library/0c899ak8.aspx

Im Unterricht beschränken wir uns nur auf die einfachste Variante.

Beispiel (als Konsolen-Anwendung):

```
public static void Main(string[] args)
{
  double x=2.0/3;
  double y=5;
  String s;

  // 3 Nachkommastellen anzeigen, wenn vorhanden
  s=x.ToString("0.###");
  Console.WriteLine(s);
  // 3 Nachkommastellen anzeigen, wenn vorhanden
  s=y.ToString("0.###");
  Console.WriteLine(s);
  // 2 Nachkommastellen, wenn vor Komma 0, diese weglassen
  s=x.ToString("#.00");
  Console.WriteLine(s);
  // keine Nachkommastellen, 2 Stellen vor dem Komma
  s=x.ToString("00");
  Console.WriteLine(s);

  // Mit 10000 multiplizieren
  x=x*10000;

  // 3 Nachkommastellen anzeigen, wenn vorhanden
  s=x.ToString("0.###");
  Console.WriteLine(s);
```



```
  // Tausender-Trennzeichen anzeigen wenn vorhanden
  s=x.ToString("#,##0.00");
  Console.WriteLine(s);
  // Tausender-Trennzeichen anzeigen wenn vorhanden
  s=y.ToString("#,##0.00");
  Console.WriteLine(s);
  // keine Nachkommastellen
  s=x.ToString("0");
  Console.WriteLine(s);

  // Auf Tastendruck warten, damit die Anzeige auf
  // dem Bildschirm bleibt
  Console.ReadLine();
}
```

Ausgabe des Beispiels:

```
0,667
5
,67
01
6666,667
6.666,67
5,00
6667
```

Bisher behandelte Elemente in C#

Element	Erläuterung

`int x;`
`double y;`
`String s;`
	Variablendeklaration einer int-Variable namens x (ganze Zahlen), einer double-Variable y (Fließkommazahlen) und einer Zeichenkette s.

`textBox1.Text` `label1.Text`	Die Eigenschaft Text des jeweiligen Objektes enthält den in der textBox1 bzw. im label1 angezeigten Text.
`s=y.ToString();`	Umwandlung einer Zahl in eine Zeichenkette. Das Ergebnis wird in s gespeichert.
`label1.Text=y.ToString()`	Umwandlung einer Zahl in eine Zeichenkette. Das Ergebnis wird im label1 ausgegeben.
`x=int.Parse(s);`	Es wird versucht, die Zeichenkette s in eine ganze Zahl umzuwandeln.
`x=int.Parse(textBox1.Text)`	Es wird versucht, den Text in der TextBox in eine ganze Zahl umzuwandeln.
`y=double.Parse(textBox1.Text)`	Es wird versucht, die Zeichenkette s in eine Fließkommazahl umzuwandeln.
`MessageBox.Show("Test");`	Der Text Test wird in einem Meldungsfenster angezeigt.
`double ergebnis=4/5;`	Liefert als Ergebnis 0 in der Variablen ergebnis, da eine ganzzahlige Division durchgeführt wird
`double ergebnis=4.0/5;`	Liefert als Ergebnis 0,8 in der Variablen ergebnis.
`// Kommentar …`	Den Rest der Zeile hinter // wird als Kommentar aufgefasst und von C# ignoriert.
for-,while-,do-while-Schleife, if	Siehe entsprechende Skript-Blätter

