Wertedatentypen in C

Wertedatentypen belegen bei der Dekleration schon Speicher für den Wert. Sie können daher - im Gegensatz zu Referenztypen - nicht null sein.

Neben den bekannten Datentypen aus Java gibt es in C# noch unsigned Datentypen wie uint, ulong, ... Sie werden allerdings selten verwendet, da die Frameworkmethoden mit "normalen" int, ... arbeiten.

Die Dekleration erfolgt wie gewohnt in Java:

```
int myInt = 1;
bool myBool = true;
long myLong;   // Wird mit 0 initialisiert.
```

Gleitkommatypen

Folgendes Programm liefert eine Endlosschleife, da 0.1 nicht exakt gespeichert werden kann:

```
double sum = 0;
// Vergleiche mit double sind durch die Ungenauigkeit
// kritisch. 1000000 wird nie erreicht.
while (sum != 1000000)
{
    sum += 0.1;
}
```

decimal

Möchte man Dezimalzahlen exakt speichern, kann der Typ *decimal* verwendet werden. Die Performance ist allerdings geringer als bei double:

```
decimal sum2 = 0;
while (sum2 != 1000000)
{
    sum2 += 0.1M;
}
```

Explizite und implizite Typencasts

Kann jeder mögliche Wert zugewiesen werden, ist ein impliziter Typencast möglich. Beispiel:

```
myLong = myInt;
myInt = myLong; // Geht nicht
```

Möchte man trotzdem eine Zuweisung in einen "kürzeren" Typ machen, brauchen wir den expliziten Typencast:

```
myInt = (int) myLong; // Wird notfalls abgeschnitten
```

Nullable Types

Eine Zuweisung von null ist bei Wertetypen wie int, long, ... nicht möglich. Möchte man trotzdem null zuweisen, etwa weil der Wert nicht bekannt ist, kann ein nullable Type definiert werden:

```
int? myInt2;
myInt2 = null;  // Funktioniert
```

Allerdings ist im weiteren Programmverlauf darauf zu achten, dass das Ergebnis auch null liefern kann.

```
myInt = myInt2 + 1;  // Geht nicht, da es null liefern kann.
```

Es gibt allerdings den "null-coalescing Operator" ??. Er liefert einen Standardwert, falls die Variable null ist:

```
myInt2 ?? 0;  // Liefert 0
1 ?? 2;       // Liefert 1
```

Durch diesen Operator können wir auch myInt2 in myInt schreiben:

```
myInt = myInt2 ?? 0 + 1;  // Liefert 1, wenn myInt2 null ist.
```