

Software Development

Variablen und Datentypen

Software-Lösungen. Einfach. Clever.

NICHT
VERGESSEN!

C:\Windows\System32\cmd.exe

```
C:\Users\eis-ma\OneDrive\BVS\Repo\BVS>git pull
```

Inhalt

- Datentypen
- Variablen
- Konstanten
- Kommentare

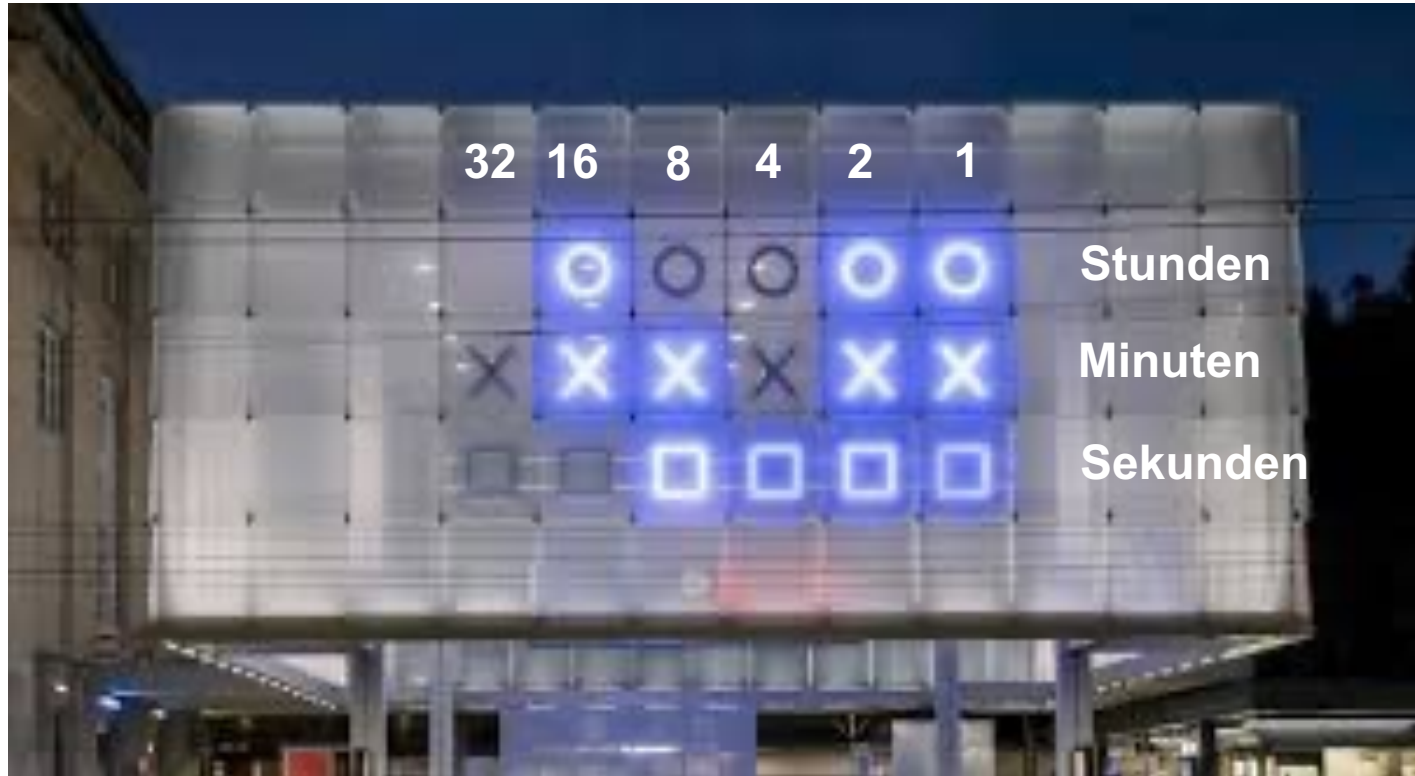
Datentypen

Software-Lösungen. Einfach. Clever.

Datentypen



Datentypen



19:27:15

Datentypen

Ganzzahl-Typen

Typ	Minimum	Maximum	Grösse	Vorzeichen
byte	0	255	8 Bit	Nein
sbyte	-128	127	8 Bit	Ja
short	-32'768	32'767	16 Bit	Ja
int	-2'147'483'648	2'147'483'647	32 Bit	Ja
long	-9'223'372'036'854'775'808	9'223'372'036'854'775'807	64 Bit	Ja

Datentypen

Gleitkomma-Typen

Typ	Bereich	Nachkommastellen (Dezimal-Stellen)	Grösse	Verwendung
float	$-3.4 \times 10^{38} - 3.4 \times 10^{38}$	7	32 Bit	float x = 1.234F;
double	$-1.7 \times 10^{308} - 1.7 \times 10^{308}$	15-16	64 Bit	double y = 1.234; double z = 3D;
decimal	$-7.9 \times 10^{28} - 7.9 \times 10^{28}$	bis 29	128 Bit	decimal d = 9.1m

Datentypen

Weitere vordefinierte Typen

Typ	Grösse	Werttyp	Beschreibung
object	-	nein	Referenz auf ein Objekt
string	-	nein	Referenz auf einen String (Text-Zeichen)
bool	8 Bit	ja	Logischer Wert true oder false
char	16 Bit	ja	Unicodezeichen 0 ... 65.535

Variablen

Software-Lösungen. Einfach. Clever.

Variablen

- Vorübergehende Speicherung von Daten zur Laufzeit im Speicher
- Grösse des Speicherplatzes für diese Variable hängt vom Typ der Variable ab
- Variable besitzt eindeutigen Namen (Bezeichner)
- Über den Bezeichner(Namen) der Variable kann das Programm auf den darin gespeicherten Wert zugreifen
- Als Zeichen sind Gross- und Kleinbuchstaben, Unterstrich «_» und Ziffern 0..9 zulässig
- Jeder Bezeichner muss mit einem Buchstaben oder Unterstrich beginnen
- Case-Sensitiv (Gross-Kleinschreibung wird unterschieden)
- Schlüsselwörter vordefinierte reservierte Bezeichner
- Schlüsselwörter dürfen **nicht** für selbst definierte Bezeichner verwendet werden

Variablen

Deklaration von Variablen

Deklaration erfolgt über folgende Syntax

Datentyp Variablenname;

C #

```
int anzahl, summe, groesse;  
double breite;  
string nachName;
```

Initialisierung einer Variable bei Deklaration

C #

```
int anzahl = 99;
```

oder zu einem späteren Zeitpunkt

C #

```
int anzahl;  
anzahl = 99;
```

Variablen

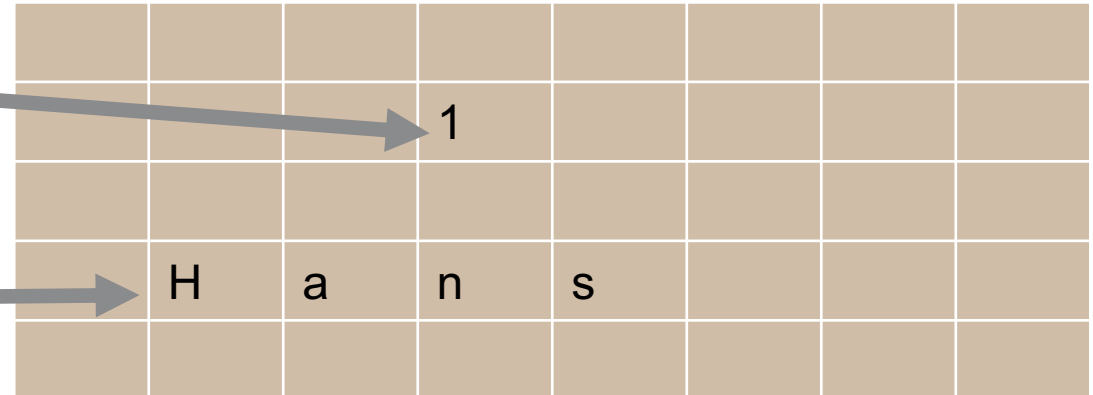
Deklaration von Variablen

Variablen Name

Speicher

```
int meineZahl = 1;
```

```
string meinName = "Hans";
```



Zeichen und Zeichenketten

char

C #

```
char c = 'A';           // Zeichenliteral  
char c = '\x0041';      // hexadezimal  
char c = '\u0041';      // Unicode  
char c = (char) 65;     // Typecasting liefert 'A'
```

Detaillierte Erklärung zu Zeichencodierung unter

<https://www.w3.org/International/articles/definitions-characters/index.de>

Zeichen und Zeichenketten

string

C #

```
string s = "Hallo";
```

Escape-Sequenzen

Escape-Sequenz	Bedeutung
\'	Einfaches Anführungszeichen
\"	Doppeltes Anführungszeichen
\\	Backslash
\a	Signalton
\b	Backspace
\f	Seitenvorschub
\n	Neue Zeile
\r	Wagenrücklauf
\t	Horizontaler Tabulator

C #

```
string pfad = "C:\Benutzer\Marcel";  
// wird bereits von der Entwicklungsumgebung  
zurückgewiesen  
  
// korrekt  
string pfad = "C:\\Benutzer\\Marcel";  
  
// oder auch  
string pfad = @"C:\Benutzer\Marcel";
```


Object-Datentyp

C #

```
int i = 5;  
object o = i;
```

Umgekehrter Weg (Fortsetzung obiges Beispiel)

C #

```
int j = o;           // Fehler !!!!!  
int j = (int) o;     // mit Typecasting o.k !
```

Typkonvertierung (Cast)

Implizit

- Umwandlung Datentyp von kleinem Wertebereich zu einem Grössen
 - Z.B. int -> long, short -> int, etc.

C #

```
int i = 5;  
long l;  
l = i; // implizite Typumwandlung -> funktioniert
```

Typkonvertierung (Cast)

Explizit

- Umwandlung Datentyp von grossem Wertebereich zu einem Kleineren
 - Z.B. long -> int, int -> short etc.

C #

```
short s;  
int i = 10;  
s = i; // -> Fehler !!!!
```

C #

```
short s;  
int i = 10;  
s = (short)i; // explizite Typumwandlung -> funktioniert
```

!!! Expliziter Cast nur wenn man sicher ist, dass kein Überlauf zur Laufzeit passieren kann.

Konstanten deklarieren

Deklaration erfolgt über folgende Syntax

Const *Datentyp* Konstantenname = Wert;

C #

```
const int C = 119;  
const float PI = 3.1415F;  
const double X1 = 3 x 0.4, X2 = 5.3 + 0.68;  
const string S = "Hallo";
```

Variablen

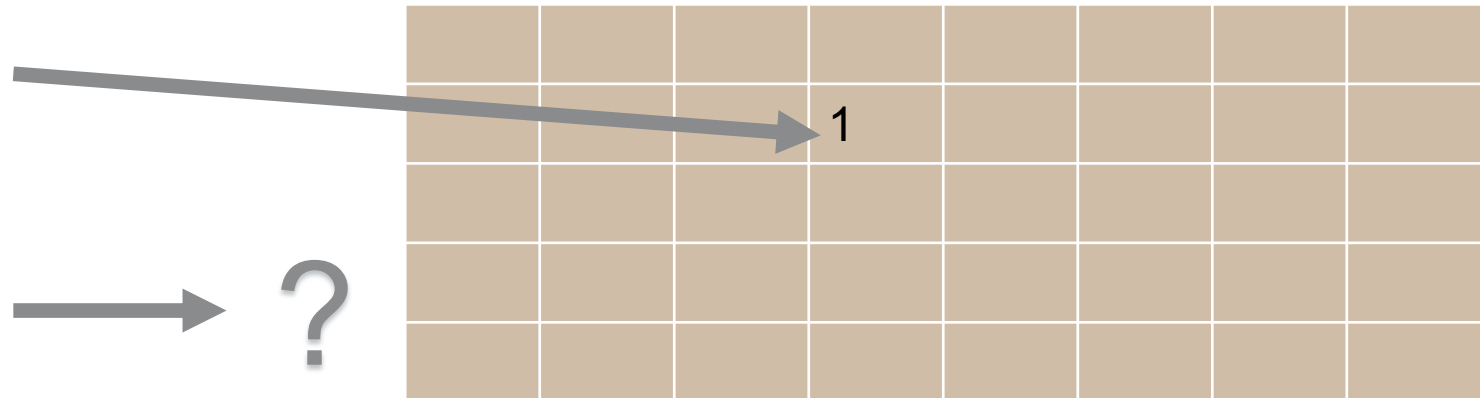
Nullable Types

Variablen Name

Speicher

```
int meineZahl = 1;
```

```
int meineZahl;
```



Wert ist *null*

Nullable Types

C# erfordert eine explizite Initialisierung von Variablen.

C #

```
int z;  
z++;      // Fehler!!, weil z nicht initialisiert ist !!!  
...  
int z = 0;  
z++;      // richtig
```

Initialisieren von Wertetypen mit null

C #

```
int z = null;  // Fehler !!
```

Mit einem nachgestellten Fragezeichen (?)

C #

```
int? z = null; // richtig  
z = 1;  
...  
System.Nullable<Int32> z = null;
```

Kommentare

- Einzeilige Kommentare

```
private double chf = 1.0; // Variablendeklaration
```

- Mehrzeilige Kommentare

```
/* Dieser Kommentar  
besteht aus mehreren  
Zeilen */
```


Übungen

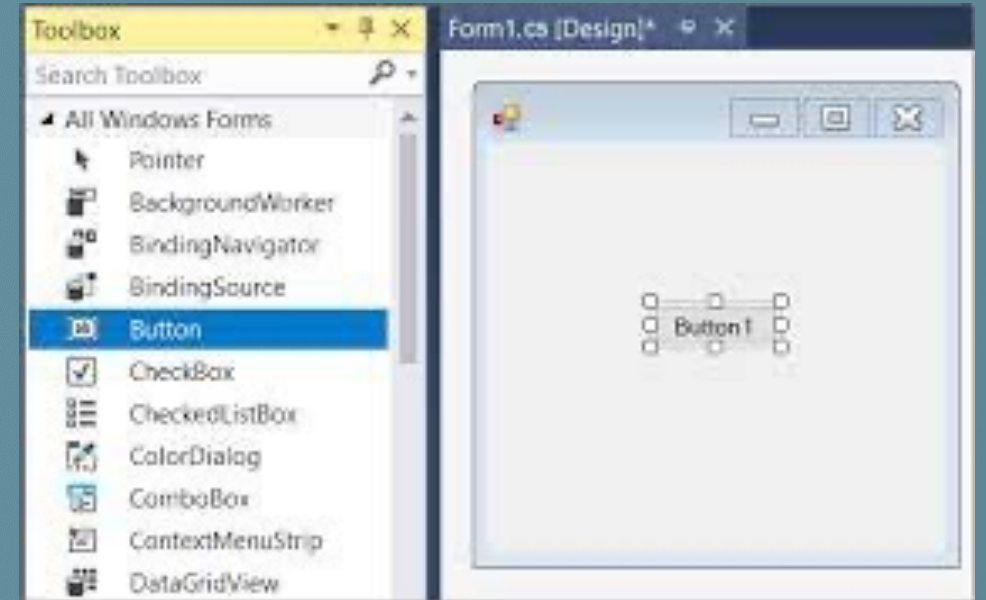
Datentypen



Software-Lösungen. Einfach. Clever.

Windows Forms

Kurze Einführung Forms anhand Beispiel
(nur mit Windows Betriebssystem)



Software-Lösungen. Einfach. Clever.

Übungen

Übung Datentypen Forms



Software-Lösungen. Einfach. Clever.