




Software Development

Quellenangaben:

- Jürgen Kotz, München, Visual C# 2019, Carl Hanser Verlag München
- Michael Bonacina, C# Programmieren für Einsteiger (2. Aufl.), BMU Verlag
- <https://docs.microsoft.com/de-de/dotnet/csharp/programming-guide>

NICHT
VERGESSEN!

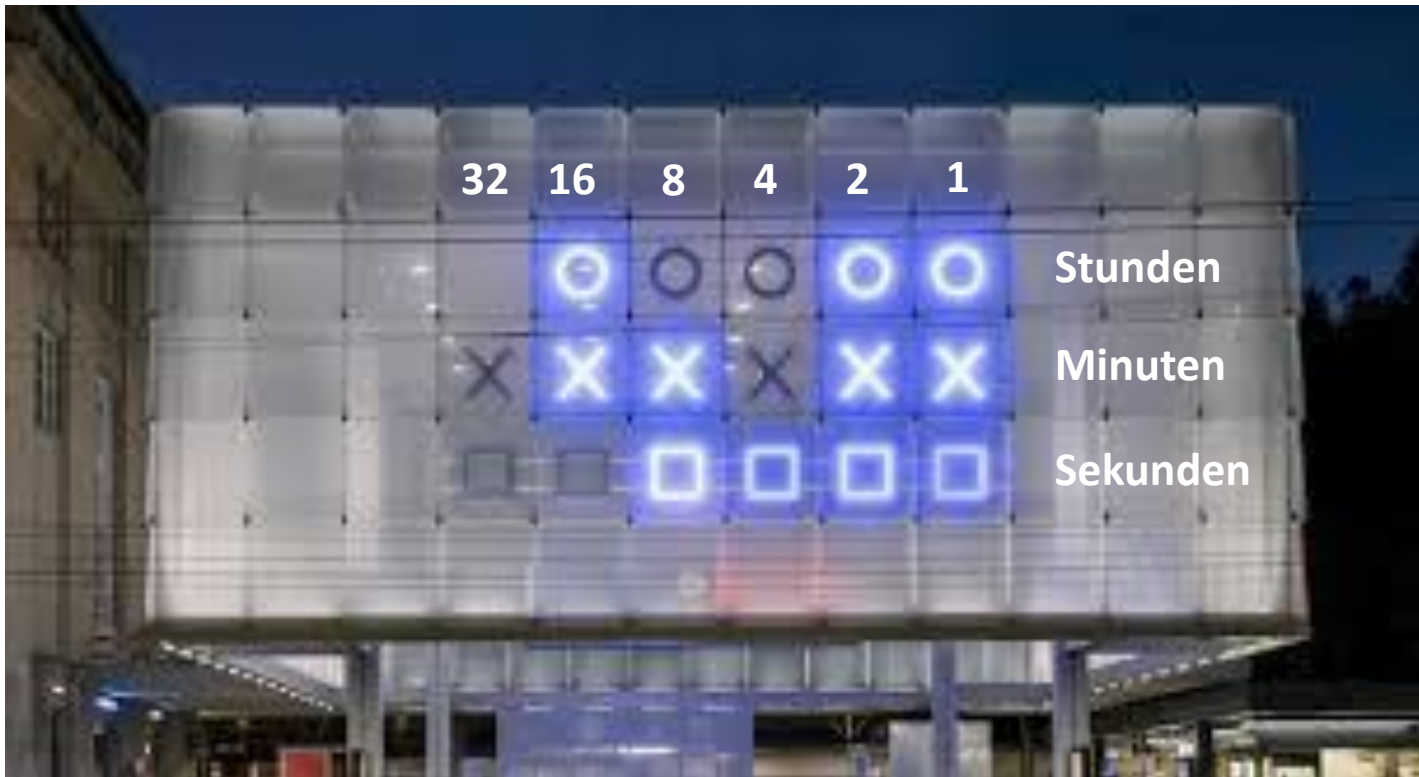
 C:\Windows\System32\cmd.exe

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

Datentypen



Datentypen



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

- 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

Deklaration

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;
```

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 !
```

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";
```

Nullable Types

C# erfordert eine explizite Initialisierung von Variablen.

C #

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

Initialisieren von Wertetypen mit null

C #

```
int z = null;  // falsch
```

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 Console.pdf

Windows Forms

Kurze Einführung Forms anhand Beispiel

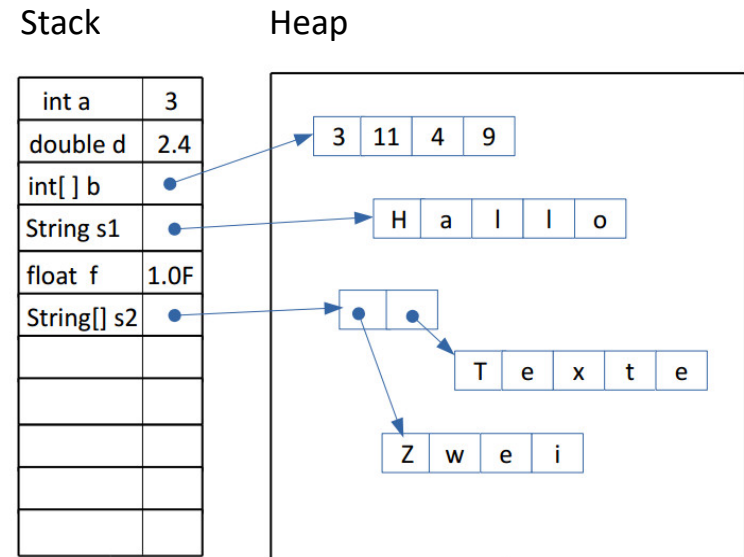


Übungen

Übung Datentypen Forms.pdf

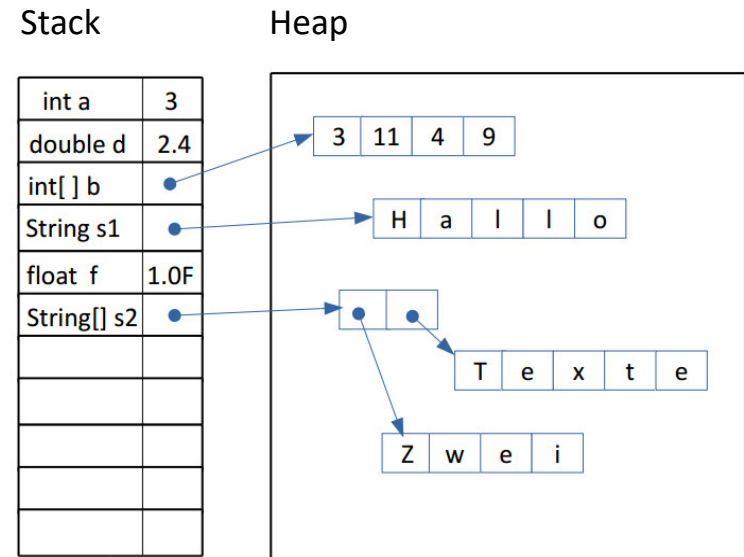
Speicherverwaltung Stack

- Alle Daten werden im Speicher gehalten. Dieser ist aufgeteilt in den Stack (Stapel) und den Heap (Halde).
- Auf dem Stack werden alle Daten verwaltet, deren Typen eine feste Grösse aufweisen (Basistypen wie int, float, double, ...).



Speicherverwaltung Heap

- Der Heap beinhaltet alle restlichen Daten (deren Typen keine feste Grösse aufweisen, wie z.B. Zeichenketten (Strings) oder Objekte (Instanzen von Klassen)). Die Daten auf dem Heap werden aus dem Stack referenziert.



Werttypen <- -> Referenztypen

- **Werttypen**

Werttypen (engl. value type) sind Typen, die im Typsystem eine feste Länge (Anzahl Bytes) aufweisen.

- Der Speicherort von Werttypen ist der Stack.

- **Referenztypen**

Referenztypen (Verweistyp, Objekttyp, engl. reference type) sind Typen, die im Typsystem keine feste Grösse aufweisen.

- Der Speicherort von Referenztypen ist der Heap.
- Für das Anlegen von Objekten muss die Laufzeitumgebung eine Speicheranforderung an die Heap-Verwaltung absetzen.

Zuweisung von Wert- oder Referenztypen

C #

```
static void Main(string[] args)
{
    Console.WriteLine("Wert-Typ --> Integer");
    int a = 5;
    int b = a;
    Console.WriteLine($"a={a}, b={b}");
    a = 7;
    Console.WriteLine($"a={a}, b={b}");
    Console.WriteLine("Referenz-Typ --> Array");
    double[] s = { 12, 2.2 };
    double[] t = s;
    Console.WriteLine($"s={s[0]}, t={t[0]}");
    s[0] = 17;
    Console.WriteLine($"s={s[0]}, t={t[0]}");
    Console.ReadLine();
}
```

// Zuweisung von Wert
// Werte von a und b ausgeben
// Wert von a ändern
// Wert von a und b sind verschieden

// Zuweisung von Referenz

// Element des Arrays ändert den Wert 17
// s und t sind gleich

```
Wert-Typ --> Integer
a=5, b=5
a=7, b=5
Referenz-Typ --> Array
s=12, t=12
s=17, t=17
```

Speicherbild

- Die int Variable b ist eine echte Kopie. Änderungen auf dem Original berühren die Kopie b nicht.
- Die Kopie t zeigt auf das gleiche Feld wie s (ist keine "echte" Kopie).
- Beim Ändern von Werten , welche auf dem Heap liegen, werden auch die Inhalte der Kopien ebenfalls verändert.
- Im Code ist der Unterschied nicht einfach zu erkennen.

