Programmieren in ~~*C#*~~ … *TypeScript*

Wie viel bytes benötigen die numerischen Datentypen?

- uint 32 bit = 4 byte
- short (16 bit) = 2 byte
- byte (8 bit) = 1 byte
- long (64 bit) = 8 byte
- float (32 bit) = 4 byte
- decimal (128 bit)= 16 byte

Wieviel Speicherplatz in bytes benötigt die Zeichenkette "Hello, World" ?

> Die Zeichenkette “Hello, World” enthält 12 Character. Jeder der Chars benötigt zwei Byte. Somit benötigt die Zeichenkette 24 Bytes.

Vergleicht den Umfang der darstellbaren Zahlen zwischen int und short, sowie zwischen float und double. Wie groß ist jeweils der größte und der kleinste Wert? Wie groß ist der kleinste positive mit float darstellbare Wert?

> short
> 32.767 - -32.768
>
> int
> 2.147.483.647 - -2.147.483.648
>
> float
> $\pm 3{,}4 * 10^{38}$ - $\pm 1{,}5 * 10^{-45}$
>
> double
> $\pm 1{,}7 * 10^{308}$ - $\pm 5{,}0 * 10^{-324}$

Was heißt Fließkommazahl und was heißt Festkommazahl? Für welchen Anwendungsbereich ist decimal besonders geeignet? Warum?

> Eine Fließkommazahl ist eine Darstellung einer Zahl mit hilfe der Exponentialschreibweise. Hierbei ist die Stelle des Kommas variabel.
> Bei einer Festkommazahl hat das Komma eine feste Position.
> Decimal eignet sich dann wenn wir wissen, wo sich das Komma in den Zahlenwerten befinden wird, Bsp: Geldbeträge.
> Bei TypeScript gibt es nur number.

Warum darf der Programmierer horst seine Lieblingsvariable nicht nach seiner Freundin else benennen? :-)

> Dieser name wird in C# für die else-Condition verwendet, daher ist dieser bereits reserviert und kann nicht genutzt werden. Armer horst : /

Erzeugt in Visual Studio Code ein neues ~~C#-Projekt~~ ... *TypeScript*-Projekt und fügt oben stehende Deklarationen und Initialisierungen der Variablen i, pi, und salute ein.

```
var i: number = 42;
var pi: number = 3.1415;
var salute: string = "Hello World";
```

Im u.a. Code-Schnipsel werden drei Array-Variablen deklariert und initialisiert. Wie heißen die Variablen, was ist der jeweilige Grund-Typ und wieviel Speicherplätze sind jeweils reserviert worden?

```
int[] ia = new int[10];
char[] ca = new char[30];
double[] da = new double[12];
```

> Name: ia, Typ: int[], Speicherplätze: 10
> Name: ca, Typ: char[], Speicherplätze: 30
> Name: da, Typ: double[], Speicherplätze: 12

```
let ia: number[] = new Array(10);
ia[0] = 1;
ia[1] = 0;
ia[2] = 2;
ia[3] = 9;
ia[4] = 3;
ia[5] = 8;
ia[6] = 4;
ia[7] = 7;
ia[8] = 5;
ia[9] = 6;

// oder kürzer: let ia: number[] = [1, 0, 2, 9, 3, 8, 4, 7, 5, 6];

let ergebnis: number = ia[2] * ia[8] + ia[4];
// = 14
```

Erzeugt einen Array vom Grund-Typ double, der drei Speicherplätze enthält in denen in der angegebenen Reihenfolge

die Zahl PI,
die Eulersche Zahl und
die Kepler-Konstante

enthalten sind.

Gebt nach der Initialisierung des o.A. Arrays mit Console.WriteLine(ia.Length); die Anzahl der Einträge aus. Ändert die Anzahl der Einträge und überprüft die Ausgabe.

```
let dArray: number[] = [Math.PI, Math.E, 2.97 * Math.pow(10, -19)];
console.log(dArray.length);
// 3 :)
dArray.push(999);
console.log(dArray.length);
// 4 :)
```

Fügt den o.a. Beispielcode zu Strings einem ~~C#-Projekt~~ *TypeScript-Projekt* zu und überprüft jeweils die Variableninhalte von meinString, c, a_eq_b, a_eq_c und zeichen mit Console.WriteLine oder mit dem Debugger.

```
let meinString: string = "Dies ist ein String";

let a: string = "Dies ist ";
let b: string = "ein String";
let c: string = a + b;
// c = "Dies ist ein String"

let a2: string = "eins";
let b2: string = "zwei";
let c2: string = "eins";
let a_eq_b: boolean  = (a == b);
// = false
let a_eq_c: boolean = (a == c);
// = true

let deinString: string = "Dies ist ein String";
let zeichen: string = meinString[5];
```

Schreibt ein ~~C# Programm~~ *TYPESCRIPT*-Programm, das zwei Zahlen von der Konsole einliest. Diese sollen verglichen werden. Ist die erste größer als die zweite, soll der Text "a ist größer als b" ausgegeben werden, ansonsten der Text "b ist größer als a".

```
let zahlA: number = parseInt(window.prompt("Zahl a", ));
let zahlB: number = parseInt(window.prompt("Zahl b", ));
if (zahlA > zahlB)
    console.log("a ist größer als b");
else
    console.log("b ist größer als a");
```

Ändert das Programm aus dem letzten TODO so ab, dass wenn die erste Zahl größer drei und die zweite Zahl gleich 6 sechs ist, der Text "Du hast gewonnen" ausgegeben wird. Ansonsten soll "Leider verloren" ausgegeben werden.

```
let zahlA: number = parseInt(window.prompt("Zahl a", ));
let zahlB: number = parseInt(window.prompt("Zahl b", ));
if (zahlA > 3 && zahlB == 6)
    console.log("Du hast gewonnen");
else
    console.log("Leider verloren");
```

Verwendet o.a. Code in einem lauffähigen C#-Programm und probiert es aus.

```
let i: number = parseInt(prompt("Zahl eingeben",));
switch (i)
    {
       case 1:
          console.log("Du hast EINS eingegeben");
          break;
       case 2:
          console.log("ZWEI war Deine Wahl");
          break;
       case 3:
          console.log("Du tipptest eine DREI");
          break;
       default:
          console.log("Die Zahl " + i + " kenne ich nicht");
          break;
    }
```

Erweitert den Code um einen weiteren Switch für eine Zahl Eurer Wahl.

```
let i: number = parseInt(prompt("Zahl eingeben",));
switch (i)
    {
       case 1:
          console.log("Du hast EINS eingegeben");
          break;
       case 2:
          console.log("ZWEI war Deine Wahl");
          break;
       case 3:
          console.log("Du tipptest eine DREI");
          break;
        case 4:
           console.log("4... wow");
           break;
       default:
          console.log("Die Zahl " + i + " kenne ich nicht");
          break;
    }
```

Ändert den Typ der Variablen i von int nach string und ändert die case-Labels, so dass diese aus Strings bestehen.

```
let s: string = prompt("Zahl eingeben",);
switch (s)
    {
       case "1":
          console.log("Du hast EINS eingegeben");
          break;
       case "2":
          console.log("ZWEI war Deine Wahl");
          break;
       case "3":
          console.log("Du tipptest eine DREI");
          break;
        case "4":
           console.log("4... wow");
           break;
       default:
          console.log("Die Zahl " + i + " kenne ich nicht");
          break;
    }
```

Was passiert, wenn man an einer Stelle das break vergisst? Denkt euch Fälle aus, bei denen das sinnvoll sein kann.

Der folgende Fall wird ausgeführt.

Versucht, die oben mit der switch / case Anweisung implementierte Funktionalität mit if/ else Anweisungen zu implementieren.

```
let i: number = parseInt(prompt("Zahl eingeben",));
if (i == 1) {
    console.log("Du hast EINS eingegeben");
} else if (i == 2) {
    console.log("ZWEI war Deine Wahl");
} else if (i == 3) {
    console.log("Du tipptest eine DREI");
} else {
    console.log("Die Zahl " + i + " kenne ich nicht");
}
```

Erzeugt ein ~~*C#*~~ … *TypeScript* Programm, das in einer while-Schleife die Zahlen von 1 bis 10 auf der Konsole ausgibt.

```
let i: number = 1;
while(i < 11)
{
    console.log(i);
    i++;
}
```

Wie lauten hier die Teile <INITIALISIERUNG>, <BEDINGUNG> und <INKREMENT>?

INITIALISIERUNG: let i: number =1;
BEDINGUNG: (i < 11)
INKREMENT: i++

Lasst u.a. Code laufen. Was tut er?

```
let someStrings: string[] =
[
"Hier",
"sehen",
"wir",
"einen",
"Array",
"von",
"Strings"
];

for (let i: number = 0; i < 5; i++) {
console.log(someStrings[i]);
}
```

Er gibt nacheinander die Strings von “Hier” bis “Array” aus.

Wandelt die for-Schleife in eine while-Schleife um.

```
let someStrings: string[] =
[
"Hier",
"sehen",
"wir",
"einen",
"Array",
"von",
"Strings"
];


let i: number = 0;
    while(i < 5)
    {
    console.log(someStrings[i]);
    i++
    }
```

Lasst statt der erste 5 *alle* in someStrings gespeicherten Werte ausgeben, indem die Bedingung statt eines konstanten Wertes die jeweilige Anzahl der Inhalte von someStrings ausgibt (geht mit someStrings.Length).

```
let someStrings: string[] =
[
"Hier",
"sehen",
"wir",
"einen",
"Array",
"von",
"Strings"
];


let i: number = 0;
    while(i < 5)
    {
    console.log(someStrings[i]);
    i++
    }
```

Lasst die unteren Beispiele für do while und break mit dem Array aus dem for-Beispiel laufen und überzeugt Euch, dass die gleiche Ausgabe erzeugt wird.

```
let i: number = 0;
    do
    {
        console.log(someStrings[i]);
        i++;
    } while (i < 5);
```

```
let i: number = 0;
    while (true)
    {
        console.log(someStrings[i]);
        if (i >= 4)
            break;
        i++;
    }
```

Sind die beiden Beispiele wirklich äquivalent zur for-Schleife?

- Verändert in allen zwei Fällen die Abbruchbedingung so, dass *alle* Array-Einträge ausgegeben werden und zwar in Abhängigkeit von der Anzahl der Array-Einträge (someStrings.Length).

```
let i: number = 0;
    do
    {
        console.log(someStrings[i]);
        i++;
    }
    while (i < someStrings.length);
```

```
let i: number = 0;
    while (true)
    {
        console.log(someStrings[i]);
        if (i >= someStrings.length - 1)
            break;
        i++;
    }
```

- Was passiert in den drei Fällen, wenn someStrings *keine* Einträge enthält?

  Undefined error bzw. index out of bounce

Diskutiert Vor- und Nachteile der Schleifenkonstruktion mit for aus dem Beispiel oben gegenüber dem Beispiel mit foreach hinsichtlich

- Klarheit

  foreach erscheint weniger klar als *for*

- Schreibaufwand

  foreach ist geringer

- Flexibilität (was ist mit Zugriffen in umgekehrter Reihenfolge, Zugriff nur auf die ersten paar Einträge, ...)

  Da bei *foreach* keine Indexvariable vorhanden ist fallen dynamische Zugriffe auf bestimmte Stellen weg,