BASICS

Dozent: Dr. Andreas Jäger

Übersicht

Datentypen (Analogie zur Mathematik)

Anmerkungen bzgl. C (C++)

Variablen (Deklaration, Zuweisung, Berechnung)

Blöcke und Gültigkeitsbereiche

DATA TYPES

Dozent: Dr. Andreas Jäger

Was sind Datentypen?

Analysieren wir den Namen… „DatenTyp“

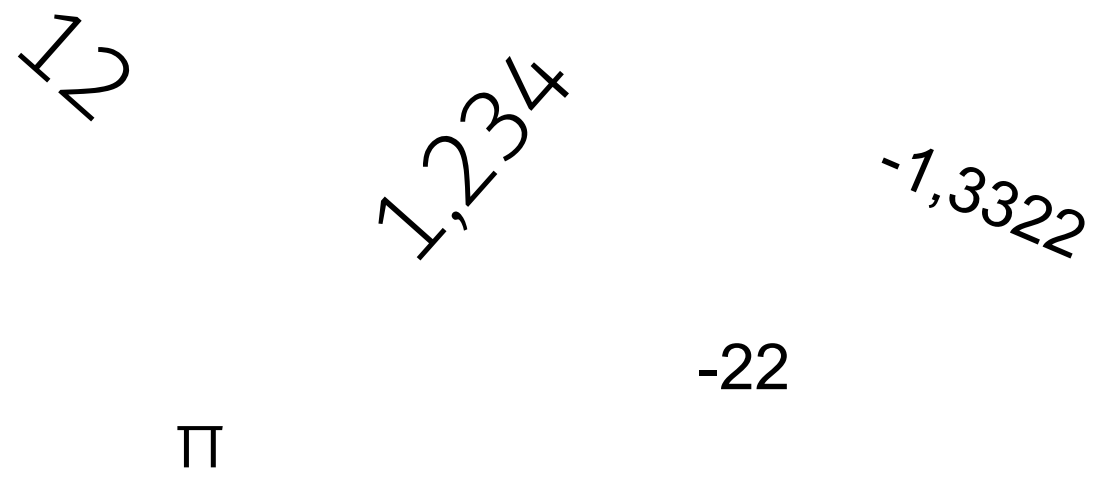
Der „Datentyp“ sagt uns also etwas über die Eigenschaften (Attribute) der Daten (mit denen wir arbeiten) aus.

Formal formuliert halten wir fest:

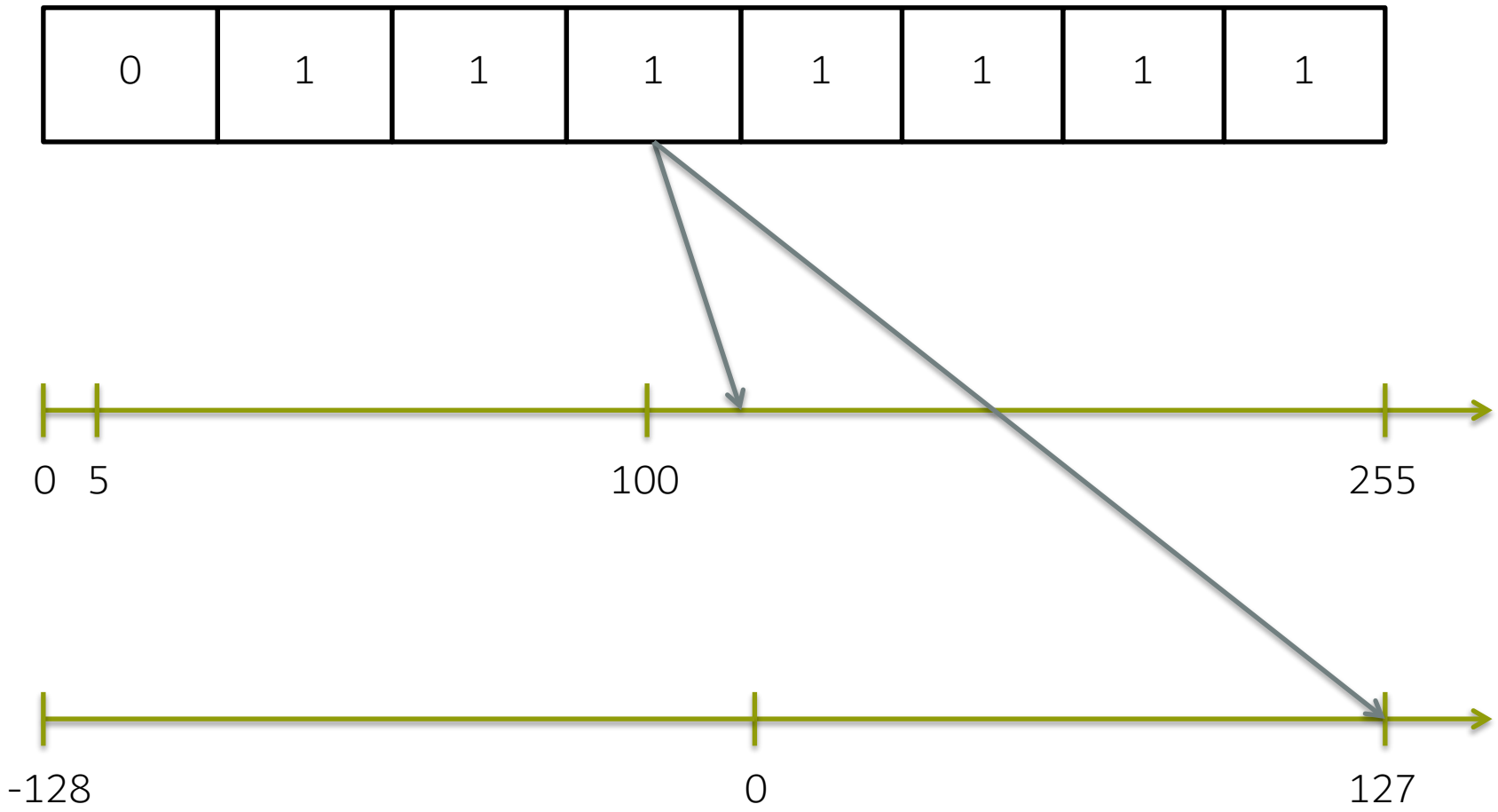
Der Datentyp gibt dem Interpreter oder Compiler bekannt, wie Daten verwendet werden sollen. Der Datentyp schränkt ein, wie Daten verwendet werden können und welche Werte die Daten annehmen können.

Datentypen

Im folgenden werden wir uns zuerst Datentypen anschauen, die zur Speicherung von uns bekannten Größen aus der Mathematik genutzt werden.



Mathe Zahlen



Datentypen - Zahlen

Wir erinnern uns an die unterschiedlichen Zahlenräume aus der Mathematik

N_0	Natürliche Zahlen mit 0 (+)
	0, 1, 2, 3 … ∞
Z	Ganze Zahlen (+ und -)
	-∞, -5, -4 …, 0, 1, 2, …, ∞
R	Reelle Zahlen (Kommazahlen)

Datentypen - Zahlen

Datentyp	Größe	Zahlenbereich	Zahlenraum
unsigned char	1Byte	0 bis 255	N_0
char	1Byte	-128 bis 127	Z, N_0
unsigned short	2Byte	0 bis 65.535	Z^+, N_0
short	2Byte	−32.768 bis +32.767	$Z \quad , N_0$
unsigned int	* (idR 4)	0 bis 4.294.967.295	Z^+, N_0
int	* (idR4)	−2.147.483.648 bis +2.147.483.647	$Z \quad , N_0$
unsigned long	*		Z^+, N_0
long	*		$Z \quad , N_0$
long long	*		...
float	* (idR4)		R
double			R
long double			R

Speichergröße von Basis-Datentypen

Kann ich herausfinden wie groß die Werte sind die ich speichern kann?

Folgende Header können genutzt warden

<limits.h> (climits in C++) und <float.h> (cfloat header in C++)

Hier befinden sich Macros, mit denen man die spezifischen Größen herausfinden kann.

Speichergröße von Basis-Datentypen

Beispiele:

Signed

SCHAR_MIN, SHRT_MIN, INT_MIN, LONG_MIN, LLONG_MIN(C99)

SCHAR_MAX, SHRT_MAX, INT_MAX, LONG_MAX, LLONG_MAX(C99)

Unsigned

UCHAR_MAX, USHRT_MAX, UINT_MAX, ULONG_MAX, ULLONG_MAX(C99)

CHAR_BIT ,CHAR_MIN, CHAR_MAX

Anmerkung - char

char ist ein Datentyp, um Zeichen darzustellen.

→ Zeichen sind jedoch ebenfalls Zahlen!

ASCII definiert 256 unterschiedliche Zeichen, welche durch 1 Byte repräsentiert werden.

Hinweis:
Kleine Zahlen sollten dennoch eher mit dem Datentyp **short** gespeichert werden, aufgrund der Ausgabefunktion.

Integer und Kommazahlen

Integer (Ganze Zahlen)

char, short, int, long

Fließkommazahlen (Reelle Zahlen)

float, double, long double

Übung

Ordnen Sie folgenden Zahlenkombinationen mögliche Datentypen zu

5, 9, 22, 99	
0,7.3, 12, 77.59	
-127, -12, 0, 22	
31, 92, -259, 302012	
‘a‘, ‘c‘, ‘i‘	
151 / 3.0	
-931, -7, 0	

Übung

Ordnen Sie folgenden Zahlenkombinationen mögliche Datentypen zu

5, 9, 22, 99	unsigned short
0,7.3, 12, 77.59	float (oder double)
-127, -12, 0, 22	short
31, 92, -259, 302012	int
‘a‘, ‘c‘, ‘i‘	char
151 / 3.0	float
-931, -7, 0	short

Gleitkommazahlen

Wie werden jetzt Kommazahlen gespeichert?

Es gibt hierzu mehrere IEEE Standards.

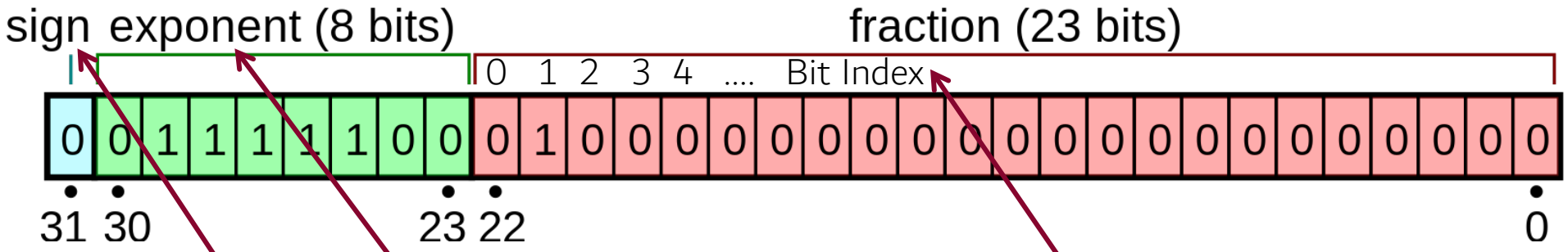
Single-precision floating-point format (32bit)

(watch: fast inverse square root quake 3)

Double-precision floating-point format (64bit)

Quadruple-precision floating-point format (128bit)

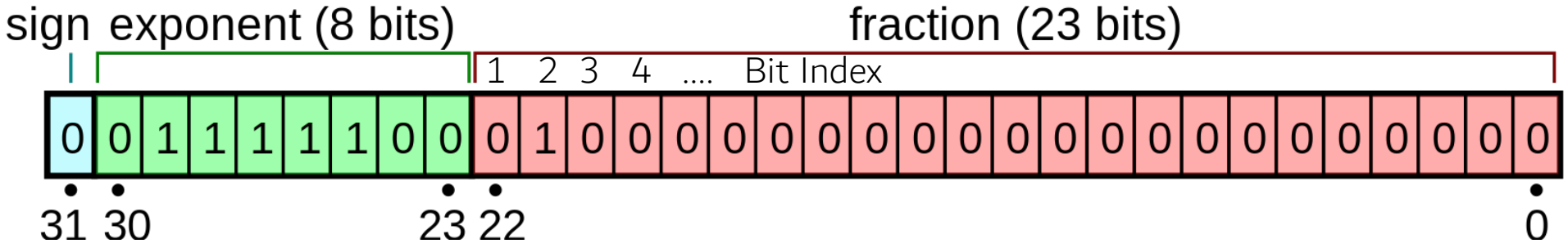
Single-precision floating-point format



So wie genau funktioniert die Kodierung…

$$Wert = (-1)^{sign} \cdot 2^{exponent\,-127} \cdot (1+\sum_{i=1}^{23} b_{23-i} \cdot 2^{-i})$$

Single-precision floating-point format



$$Wert = (-1)^{sign} \cdot 2^{exponent\,-127} \cdot (1+\sum_{i=1}^{23} b_{23-i} \cdot 2^{-i})$$

Berechnen Wir das Beispiel:

$$sign = 0 \rightarrow (-1)^0 = 1$$

$$exponent = 01111100 = 2^6 + 2^5 + 2^4 + 2^3 = 255 - 2^7 - 2^1 - 2^0 = 124$$

$$1 + \sum_{i=1}^{23} b_{23-1} \cdot 2^{-i} = 1 + b_{23-2} \cdot 2^{-2} = 1 + \tfrac{1}{4} = 1,25$$

Single-precision floating-point format

$$Wert = (-1)^{sign} \cdot 2^{exponent\,-127} \cdot (1+\sum_{i=1}^{23} b_{23-i} \cdot 2^{-i})$$

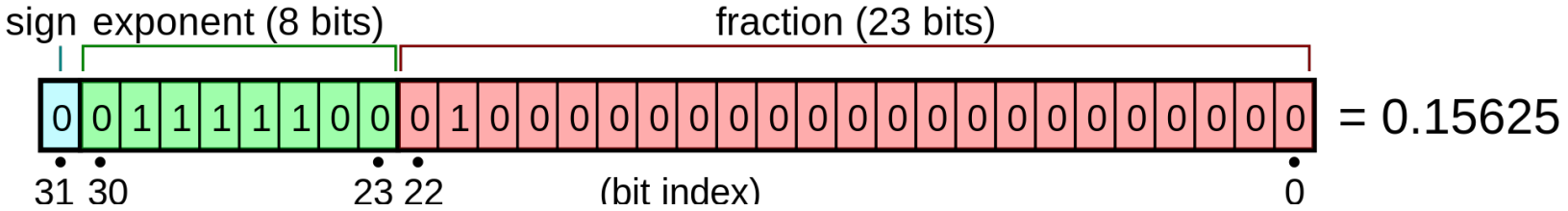
Führen wir nun alles zusammen, ergibt sich:

$$sign = 0 \rightarrow (-1)^0 = 1$$

$$exponent = 01111100 = 2^6 + 2^5 + 2^4 + 2^3 = 255 - 2^7 - 2^1 - 2^0 = 124$$

$$1 + \sum_{i=1}^{23} b_23 - 1 \cdot 2^{-i} = 1 + b_{23-2} \cdot 2^{-2} = 1 + \frac{1}{4} = 1,25$$

$$1 \cdot 2^{124-127} \cdot 1,25 = 2^{-3} \cdot 1,25 = 0,125 \cdot 1,25 = 1,5625$$



Single-precision floating-point format

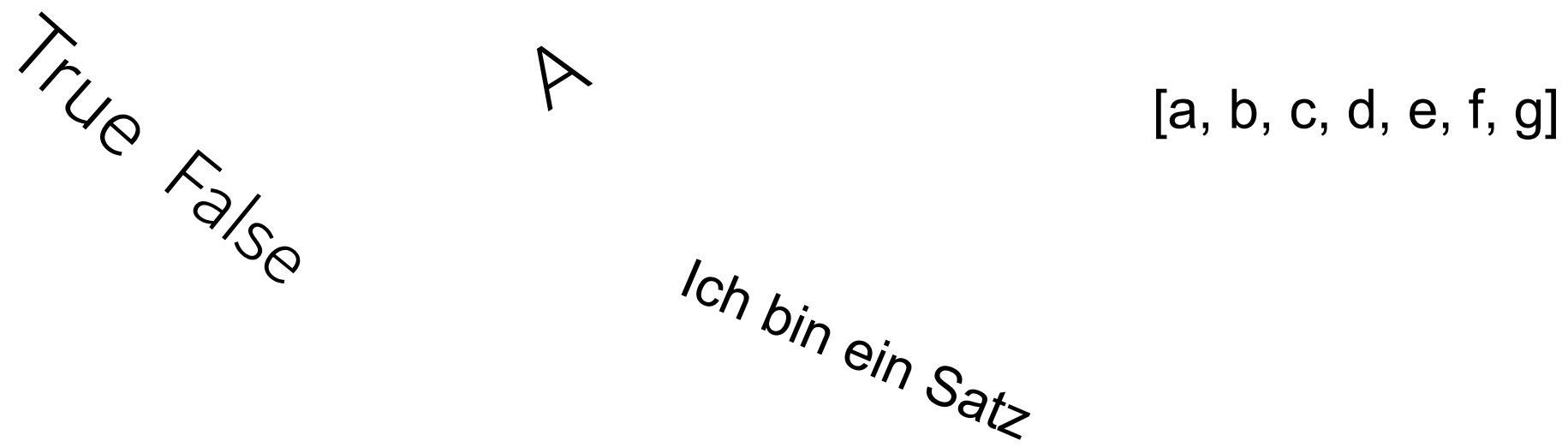
Frage: wie könnte man auf die in der Formel -127 verzichten

$$Wert = (-1)^{sign} \cdot 2^{exponent-127} \cdot (1+\sum_{i=1}^{23} b_{23-i} \cdot 2^{-i})$$

Speicherung des Exponenten Byte als int (nicht unsigned int)

Datentypen – mehr als nur Zahlen…

Im folgenden betrachten wir weitere Standard Datentypen, die C mitbringt.



Boolean

- Dient zur Speicherung von Wahrheitswerten (also wahr oder falsch)
- Intern Repräsentiert durch 1 bit (0[false]; 1[true])

Zu beachten: Jede Zuweisung ≠ 0 wird als 1 gespeichert.

Nachfolgender Code würde ausgeführt.

```
bool b = 256;
if (b) { /* do something really cool*/ }
```

Char

- Datentyp zum speichern von (genau einem) Zeichen.
- Wie wir bereits gelernt haben, wird zur Speicherung ein Byte verwendet.
- Das Zeichen, das representiert wird hängt von der Kodierung ab.
- C (C++) verwendet standardmäßig ASCII (ISO 8859-1)

Anmerkung:

Es existieren UNICODE Datentypen z.B. wchar_t (verwendet mehrere Byte)

Anmerkungen zu C (C++)

C ist case-sensitiv, d.h. die Sprache unterscheidet Groß- / Kleinschreibung

- > *Switch* ist also nicht gleich *switch*
- > Und *INT* ist nicht gleich *int*
- > Sowie *MeineFunktion* ist nicht gleich *Meinefunktion*

C ist typstreng

- ➢ Jeder Wert ist von einem bestimmten Typ
- ➢ d.h. kann nicht ohne Konvertierung (typecast) als Wert eines anderen Typs angesehen werden.

VARIABLEN

Dozent: Dr. Andreas Jäger

Variable

Was sind Variablen

Variable

Wird auch als Skalar bezeichnet.

Es ist eine Speicheradresse die an einen Bezeichner(Symbolic name; Identifier) gekoppelt ist.

Über den Variablennamen wird auf den Wert der Speicheradresse zugegriffen.

Der Name geht auf den „variablen Inhalt“ zurück.

Variablen in C/C++

Variablen sind (in C/C++) immer einem Datentyp zugeordnet.

C ist Typenstreng d.h. Variablen und deren Typ müssen bekannt sein.

Variablen müssen bekannt sein heißt, Sie müssen deklariert sein.

Deklarations Syntax:

<typ> <variable>;

Deklaration + Definition Syntax:

<typ> <variable> = <Wert>;

Variablen in C/C++ Namenbedingungen

Bedingungen für Namen

Beginnt mit **alphabetischen Zeichen oder _**

Keine Sonderzeichen (Ausnahme Unterstrich _)

Keine Operatoren (+,-,*,/,%,&,=,|,!)

Kein reserviertes Schlüsselwort

Schlüsselworte

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

GÜLTIGKEITSBLÖCKE

Dozent: Dr. Andreas Jäger

Anweisungsblöcke

In C werden Anweisungsblöcke verwendet, um:

- Anweisungen von Iterationen und Selektionen zusammen zu fassen.
- Funktionsrümpfe zu definieren.
- Gültigkeitsbereiche (von Variablen,...) zu definieren.

Anweisungsblöcke

Beginn eines Blocks {

Ende eines Blocks }

globaler Block (ohne {})

Bsp:

```
void main(void)
{
//Block des Hauptprogramms
}
```

Gültigkeitsbereiche

```
{
  long gueltig = 2;
  {
    long innen = gueltig + 2;
    gueltig = gueltig + 1;
  }
  cout<<gueltig   // 3
  cout<<innen  // Compilerfehler innen unbekannt!
}
```

OPERATOREN

Dozent: Dr. Andreas Jäger

Zuweisung

<variable> = <wert>;

L-Value = R-Value

L-Value muß bei einer Zuweisung eine Variable sein.

R-Value kann sowohl direkter Wert, Variable, Rückgabewert einer Funktion oder Ergebnis einer Berechnung sein.

Beispiel

```
{
  int assignMeSth;
  int assigneMeTo = 1;
  assignMeSth = assigneMeTo;
  assignMe = ++assignMe;
  assignMe = assigneMe + 1;
  assignMe = incrementMe(assignMe);
}
```

Basis-Operationen

Addition

```
short a = 25 + 17;          // a = 42
```

Subtraktion

```
short b = 25 – 17;          // b = 8
```

Multiplikation

```
short c = 25 * 17;          // c = 425
```

Division

```
double d = 25.0 / 17.0;     // d = 1.470588235
```

Anmerkung - typecast

int / int = int

float / float = float

Um eine Rechenart zu erzwingen, muss evtl. also ein „typecast“ durchgeführt werden:

```
short i1 = 10;
short i2 = 3;
float f = (float) i1 / i2;
```

Anmerkung - typecast

typecast sind nicht sicher!

Datenverlust

(in seltenen Fällen sogar runtime errors)

VIELEN DANK
FÜR
IHRE AUFMERKSAMKEIT!