Einführung in die Informatik I

Kapitel I.2: Variablen und arithmetische Ausdrücke

Prof. Dr.-Ing. Marcin Grzegorzek

Juniorprofessur für Mustererkennung im Institut für Bildinformatik

Department Elektrotechnik und Informatik

Fakultät IV der Universität Siegen

17.10.2012

Demo: Robbie 15

Robbie 15 on YouTube

Robbie 15

Within the practical course "Robbie 15" an improved concept over the designs from "Robbie 13" is being developed. The robot will once again participate in the Sick Robot Day 2010 autonomous robotics competition on the 2nd of October 2010 with the goal of defending last year's title.

Once again, the main objective is to find and recognize signs with the numbers 0 to 9 on them while avoiding collisions with obstacles and other robots.

SICK robot day 2010 (Waldkirch/Freiburg)

- **2nd Place**

Inhaltsverzeichnis

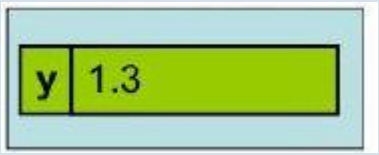
Variablen

- Programme speichern ihre Daten in Variablen.
- Variablen haben einen **alphanumerischen** Namen.
  - *y, axiale_Position, Durchmesser2,…*
- **Erstes Zeichen** muss ein **Buchstabe** sein.
  - *1st_Position* ist kein Variablenname!!!
- Zur besseren Übersichtlichkeit und besserem Verständnis immer **„sprechende“ Variablennamen** wählen.
  - *Axiale_Position anstatt x, y, z, a1, a2,…*
- Bei der Namensgebung wird **Groß-/ Kleinschreibung beachtet**.
  - *Axiale_Position ≠ axiale_Position*

Variablen: Wertzuweisung

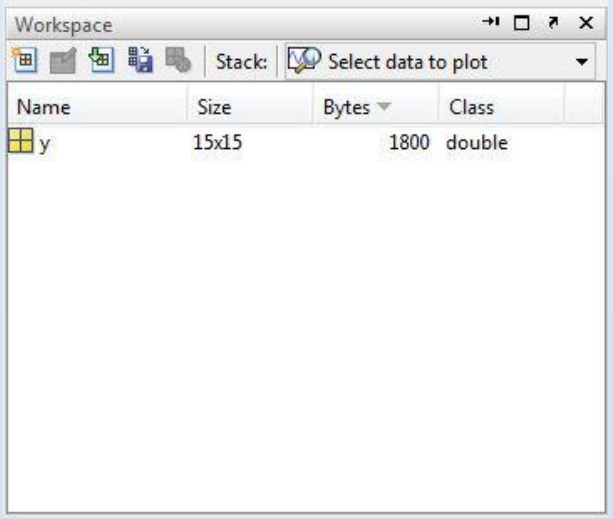
- Beispiel: *y=1.3* (Zur Erinnerung: Der Punkt ist das Dezimaltrennzeichen!)
- Die Variable *y* ist noch unbekannt, weil sie bisher nicht verwendet wurde.
- Im Hauptspeicher muss Speicherplatz für eine Zahl reserviert werden.
- Der Wert *1.3* wird der Variablen *y* zugewiesen.

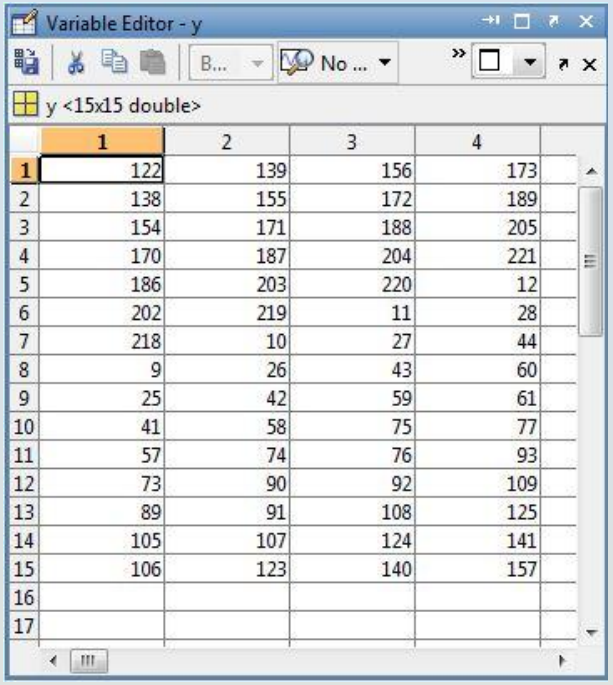
Hauptspeicher



Workspace-Fenster

- Das Workspace-Fenster kann über das Menü *View* geöffnet werden.
- Im Workspace werden die neuen Variablen mit dem Speicherplatzbedarf (Bytes) angezeigt.
- Durch einen Doppelklick auf einen Variablennamen öffnet sich ein neues Fenster mit dem Array Editor.



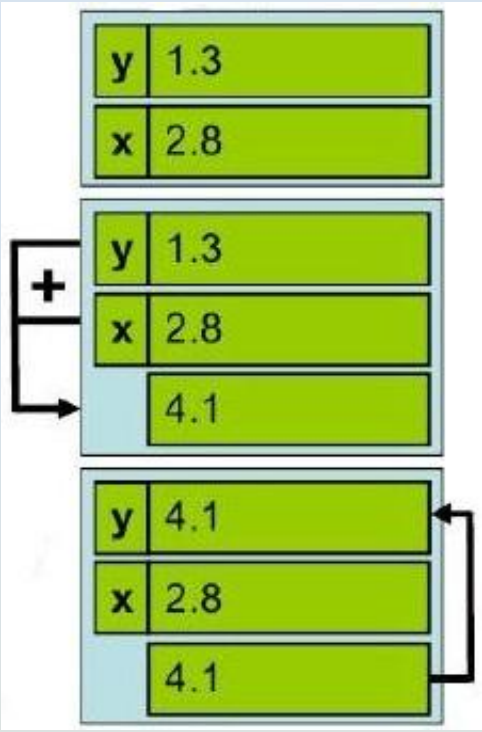


	1	2	3	4
1	122	139	156	173
2	138	155	172	189
3	154	171	188	205
4	170	187	204	221
5	186	203	220	12
6	202	219	11	28
7	218	10	27	44
8	9	26	43	60
9	25	42	59	61
10	41	58	75	77
11	57	74	76	93
12	73	90	92	109
13	89	91	108	125
14	105	107	124	141
15	106	123	140	157
16				
17				

Wiederholte Wertzuweisung

- Die Variable *y* hat aktuell den Wert *1.3*
- Die Variable *x* hat den Wert *2.8*
- Der Wert von *y* soll um den Wert von *x* erhöht werden, also *y=y+x*.
  - Wie geht MATLAB bei der Berechnung von y=y+x vor? (Mathematisch würde diese Gleichung als Ergebnis x=0 liefern), ABER HIER:

1. Zuerst wird y+x berechnet (y und x haben die Werte 1.3 und 2.8) und das Ergebnis (also 4.1) temporär gespeichert.
2. Dieses berechnete Ergebnis wird der Variablen y zugewiesen.
3. Das temporär gespeicherte Ergebnis wird gelöscht.



Vorsicht beim Gleichheitssymbol

- „=“ in der Mathematik: x=x+1 $\Leftrightarrow$ 0=1 (falsche Aussage)
- „=“ in der Programmierung: *x=x+1* (der Wert der Variable x wird um 1 erhöht, also **„=“ bedeutet Wertzuweisung**)
- Wertzuweisung in anderen Programmiersprachen:
  - A=b in FORTRAN, C, Java, VBA,…
  - A:=b in Pascal
  - A $\leftarrow$ B in Pseudo Code (programmiersprachenunabhängige Notation für Programme)

Variablen: Löschen

- Der Befehl *clear<Variable>* löscht eine Variable und gibt deren Speicher frei, z.B. *clear y*
- Nach dem Löschen kann MATLAB nicht mehr auf die Variable zugreifen.
- Bei erneutem Zugriffsversuch wird eine Fehlermeldung ausgegeben: *??? Undefined function or variable ‘y‘.*

Vordefinierte Variablen

- MATLAB hat einige **vordefinierte Variablen** (Erläuterung folgt später).
  - *pi* π
  - *eps, realmin, realmax*,… (werden später behandelt)

- **MATLAB-Problem** (schwer auffindbarer Fehler): Diese in MATLAB **vordefinierten Variablen** können **irrtümlich neu definiert** werden.

  >> pi ergibt 3.1415

  >> pi=7 neuer Wert von π

  >> pi ergibt jetzt 7.0000

Arithmetische Operatoren (1)

- MATLAB kennt die **üblichen Grundrechenarten** (arithmetische Operationen) zwischen zwei Zahlen
- Das Ergebnis der Operation ist wieder eine Zahl. Die Operation wird mit einem Operatorsymbol zwischen den beiden Zahlen angegeben.
  - `+` Addition
  - `-` Subtraktion
  - `*` Multiplikation
  - `/` Division (Achtung, nicht \)
  - `^` Potenzieren (Achtung: Eingabe mit Leerzeichen)

Arithmetische Operatoren (2)

- Beispiele:
  - *pi+5, pi*pi, pi^2, 3/pi*
- Multiplikation: Das **Symbol *** kann bei der Produktbildung **nicht** wie in der Mathematik **weggelassen** werden!

  In MATLAB ergibt:
  - *xy* einen Variablennamen mit zwei Buchstaben
  - *x*y* das Produkt von x und y
  - *xy*y* das Produkt von xy und y
  - *x y* eine Fehlermeldung (*Missing operator…*)

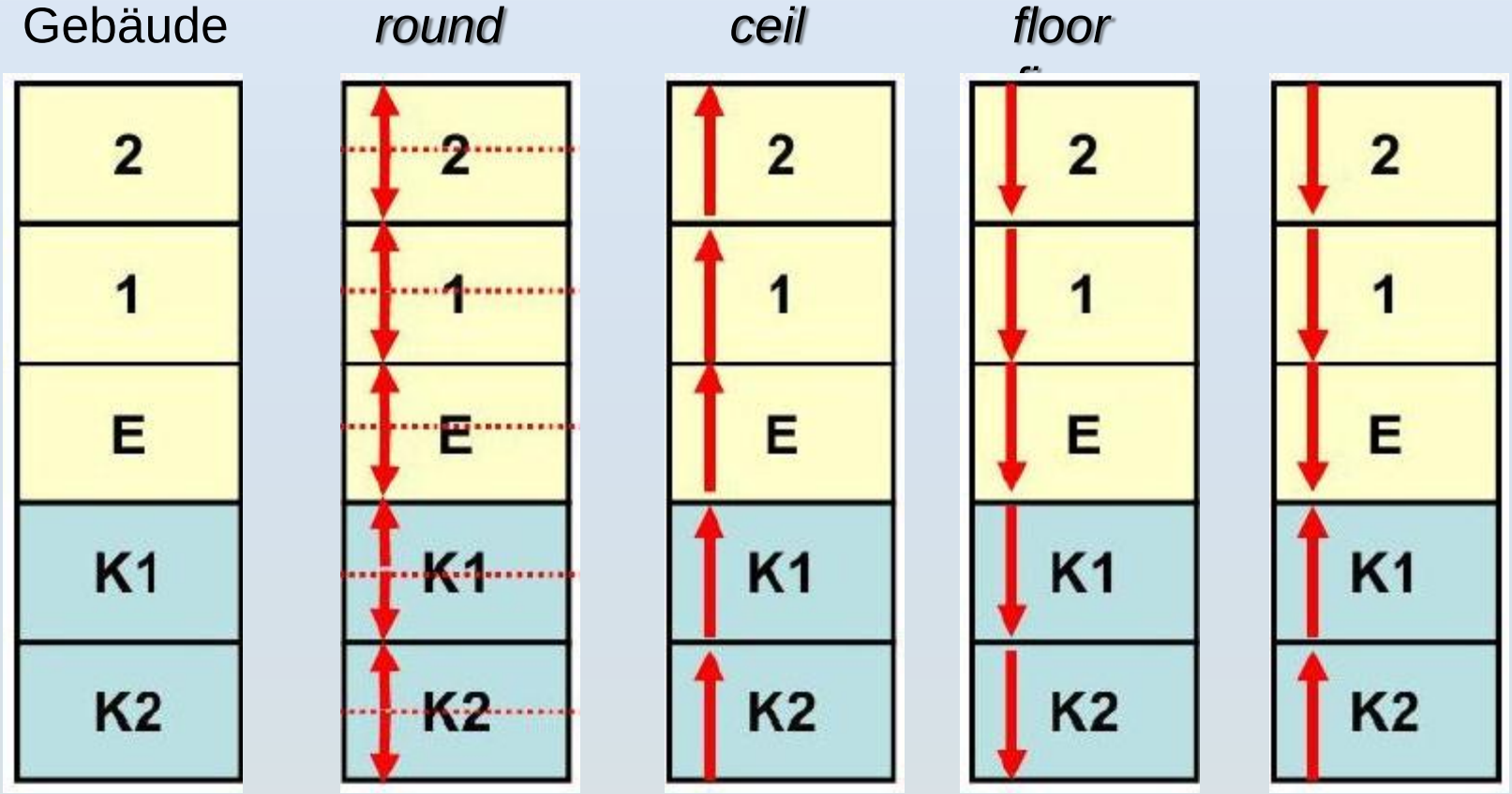
Mathematische Funktionen

- Trigonometrische Funktionen, x im Bogenmaß (rad)
  - *sin(x)*
  - *cos(x)*
  - *tan(x)*
- Andere Funktionen
  - *abs(x)* absoluter Betrag
  - *sqrt(x)* Quadratwurzel
  - *exp(x)* natürliche Exponentialfunktion
  - *log(x)* natürlicher Logarithmus
  - *log10(x)* Zehnerlogarithmus

Rundungsfunktionen (1)

- Runden zur nächsten Zahl (mathematisches Runden)
  - *round(x)*, z.B. *round(0.5)=1, round(-0.5)=-1*
- Runden zur nächst größeren ganzen Zahl
  - *ceil(x)*, z.B. *ceil(0.5)=1, ceil(-0.5)=0*
- Runden zur nächst kleineren ganzen Zahl
  - *floor(x)*, z.B. *floor(0.5)=0, floor(-0.5)=-1*
- Nachkommastellen abschneiden
  - *fix(x)*, z.B. *fix(0.5)=0, fix(-0.5)=0*

Rundungsfunktionen (2)



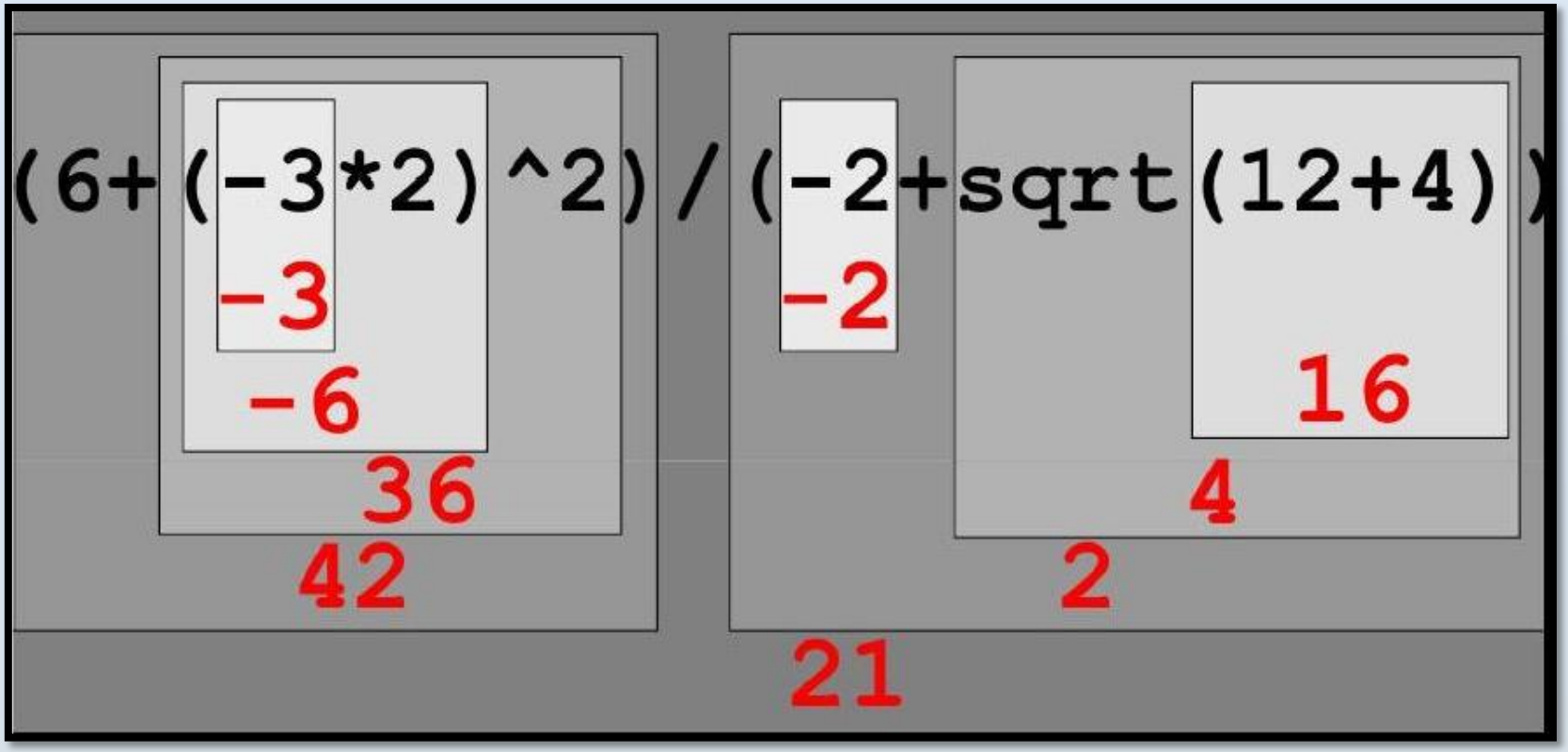
Operator-Prioritäten

- Arithmetische Ausdrücke: Mathematische Formeln mit Zahlen Variablen, Operatoren, Funktionen und Klammern.
  - *5+4*sin(x)/3*pi^32*
  - *sin(x-y/(z+3))/cos(3*y)*
  - *3*32^1/2 ≠ 3*32^(1/2)*
- **Prioritätenregelung** bestimmt die Auswertungsreihenfolge für Operatoren

1.	Potenzieren	^	
2.	Vorzeichenoperationen (unäres Minus/ Plus)	+	-
3.	Punktoperationen	*	/
4.	Strichoperationen (binäres Minus/ Plus)	+	-

- Prioritäten können **durch Klammerung aufgehoben** werden.

Beispiel: Operator-Prioritäten und Klammerung



- Mathematische Notation $\frac{6 + (-3 \cdot 2)^2}{-2 + \sqrt{12 + 4}}$

Beispiel: Operator-Prioritäten und Klammerung

- In den meisten Programmiersprachen wird der Bereich der reellen Zahlen durch zwei weitere Werte erweitert:
  - *inf* (infinity) steht für den Wert „unendlich“
  - *NaN* (not a number) steht für den Wert „undefiniert“
- Der Wert inf tritt z.B. auf,
  - wenn durch Null dividiert wird
    1/0 → *inf*, *-1/0* → *-inf*
  - wenn der Zahlbereich unter-/überschritten wird
    *1E300*1E300* → *inf*, *-1E300*1E300* → *-inf*
    (Beachte: 1E300=1*10^300)
  - wenn mit inf weitergerechnet wird
    inf+inf → *inf*, *inf*inf* → *inf*
- Der Wert NaN tritt auf, wenn ein Rechenergebnis nicht mehr sinnvoll festgelegt werden kann

  0/0 → NaN, *inf*0 → NaN*, *inf-inf → NaN*