Grundlagen der Programmierung

6. Vorlesung
21.11.2017

Zahlendarstellungen

- Binär (Basis 2)

$$b_n \dots b_2 b_1 b_0, b_i \in \{0, 1\}$$

- Hexadezimal (Basis 16)

$$h_n \dots h_2 h_1 h_0, h_i \in \{0, 1, \cdots, 9, \mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}, \mathrm{E}, \mathrm{F}\}$$

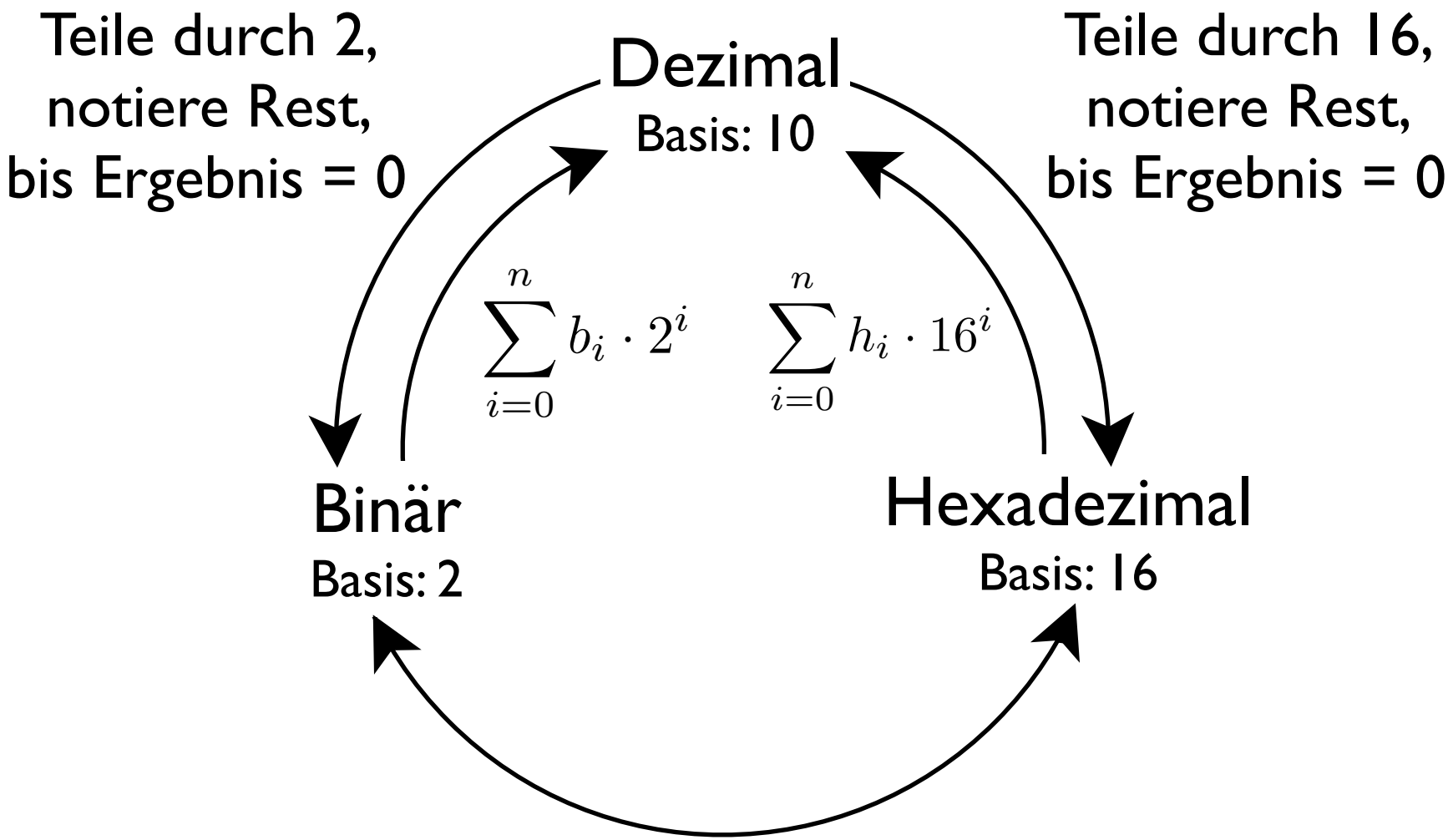
- Dezimal (Basis 10)

$$d_n \dots d_2 d_1 d_0, d_i \in \{0, 1, \cdots, 9\}$$

- Oktal (Basis 8)

$$o_n \dots o_2 o_1 o_0, o_i \in \{0, 1, \cdots, 7\}$$

Zahlenkonvertierungen



4 Ziffern binär ≙ 1 Ziffer hexadezimal

Hexadezimalzahlen

Bin	Dez	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

Bin	Dez	Hex
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

2^3	2^2	2^1	2^0

There are only 10 types of people in the world:
Those who understand binary and those who do not.

Ausdrücke

- "Befehle", die ein Ergebnis liefern
  - 3 + 4
  - sin(x)
  - x < 10
  - input()
- Ausdrücke können Teil eines anderen Ausdrucks sein
  - sin(x + y)
  - x = sin(x + y)

Auswertung von Ausdrücken

- Auswertung: von "innen" nach "aussen":

```
(( 3 * 4 ) + ( 5 * 6 ))
((  12   ) + ( 5 * 6 ))
((  12   ) + (  30   ))
(          42         )
```

- “Punktrechnung” vor “Strichrechnung”
  - 5 + 3 * 4 ergibt 17
- Andere Operatoren?
  - https://de.mathworks.com/help/matlab/matlab_prog/operator-precedence.html

Ausgabe

- Biblitheksfunktion `fprintf`
- Syntax: `fprintf`(*zeichenkette*, ...)
- Zeichenkette: in einfache Anführungszeichen (‘)
- Sonderzeichen in der Zeichenkette:
  - `\n`: Newline
  - `\t`: Tabulator
- % leitet *Formatierungsanweisung* ein

fprintf Formatangabe

- *%-0Feldbreite.Genauigkeit Identifier*
- Optionale Angaben:
  - -: linksbündige Ausgabe
  - 0: führende Nullen anzeigen
  - Feldbreite: minimale Anzahl Zeichen in der Ausgabe
  - Genauigkeit (Zahl nach .):
    - float/double: Anzahl Nachkommastellen
    - String: maximale Anzahl Buchstaben

fprintf Identifier

d	integer, dezimal mit Vorzeichen
u	integer, dezimal mit Vorzeichen
x	integer, hexadezimal
c	char (ein Zeichen)
s	Zeichenkette
f	double
e	double in Exponentialdarstellung
g	%f oder %e
%	Prozentzeichen ausgeben

fprintf Beispiele

- `fprintf( ':%s:', 'Hello World');`
  - `:Hello World:`
- `fprintf( ':%15s:', 'Hello World');`
  - `:    Hello World:`
- `fprintf( ':%-15s:', 'Hello World');`
  - `:Hello World    :`
- `fprintf( ':%15.9s:', 'Hello World');`
  - `:      Hello Wor:`

Alternative: disp

- `disp(x)` gibt Inhalt der Variablen *x* aus.
- Mehrere Werte müssen zu einem Feld zusammengefasst werden (mit [...])
- Numerische Werte darin müssen in Strings umgewandelt werden (mit `num2str(...)`)
- ```
  >> n=42;
  >> disp(['n hat den Wert ', num2str(n), '.'])
  n hat den Wert 42.
  >>
  ```
  ```

Eingabe

- `input(`*`prompt`*`)`
- Druckt *`prompt`* auf dem Bildschirm und ließt Eingabe von Tastatur
- Gibt eingegeben Wert zurück (interpretiert)
- `input(`*`prompt`*`,'s')`
- Wie oben, gibt jedoch Zeichenkette zurück

mehr dazu:

```
help fprintf
help input
```

Bedingung

- Ausdruck, dessen Wert sich als "Wahr" oder "falsch" interpretieren lässt
- 0 wird als "falsch" interpretiert, alles andere als "wahr"
- Beispiele:
  - `max_so_far < eingabe`
  - `isnumeric( x )`
  - `eingabe > 0`
  - `eingabe`

entspricht
`eingabe ~= 0`

Vergleichsoperatoren

- < echt kleiner
- > echt größer
- <= kleiner oder gleich
- >= größer oder gleich
- ~= ungleich

- == gleich

= vs == in C

- ```
  int i = 3;
  if ( i = 0 )
     printf( "i ist gleich 0\n");
  else
     printf( "i ist ungleich 0\n");
  printf( "i ist: %d", i );
  ```

hier erfolgt eine Zuweisung!

- Ausgabe:

  ```
  i ist ungleich 0
  i ist: 0
  ```

in Matlab nicht so dramatisch

Ergebnis der Zuweisung:
0 (entspricht false),
also wird der else-Zweig ausgeführt
  ```

Logische Operatoren

- Verknüpfen von Bedingungen
- UND
  Operator &&

a	b	a && b
falsch	falsch	falsch
falsch	wahr	falsch
wahr	falsch	falsch
wahr	wahr	wahr

- ODER
  Operator ||

a	b	a || b
falsch	falsch	falsch
falsch	wahr	wahr
wahr	falsch	wahr
wahr	wahr	wahr

Logische Operatoren

- NICHT Operator ~

a	~a
falsch	wahr
wahr	falsch

- `((eingabe > 0) && (eingabe < 10))`
- `((c == 'q') || (c == 'Q'))`

Logische Operatoren

- Verknüpfen **Wahrheitswerte**
- D.h. beide Operanden werden als Wahrheitswerte interpretiert
- z.B.
  - `((eingabe < 0) || (eingabe > 2))`
  - ```
    (     (eingabe == 1)
       || (eingabe == 2)
       || (eingabe == 3))
    ```
- NICHT: `eingabe == 1 || 2 || 3`
- (entspricht `((eingabe == 1) || wahr || wahr)`
    ```

Logische Operatoren in C

```
int i = 0;
int u = 10;

if ( (u/i > 0)
     && (i != 0) )
 printf("ok\n");
else
 printf("not ok\n");
```

```
> ./andtest2
Floating point exception
>
```

```
int i = 0;
int u = 10;

if ( (i != 0)
   && (u/i > 0) )
 printf("ok\n");
else
 printf("not ok\n");
```

```
> ./andtest
not ok
>
```

Logische Operatoren

- Auswertungsreihenfolge beachten!
- Bei log. Operatoren: von links nach rechts
- Short-Circuit Evaluation:
  - Auswertung nur so lange, bis Gesamtergebnis feststeht
  - $C = A \;\&\&\; B$
    - A ist wahr $\Rightarrow$ B muss ausgewertet werden
    - A ist falsch $\Rightarrow$ C ist falsch
  - $C = A \;||\; B$
    - A ist falsch $\Rightarrow$ B muss ausgewertet werden
    - A ist wahr $\Rightarrow$ C ist wahr

Summe berechnen

- Bekannt:
  Lies ganze Zahlen von der Tastatur ein und summiere diese, bis 0 eingegeben wurde.
  Gib die Summe aus.

Aufgabe: Maximium bestimmen

- Lies ganze Zahlen von der Tastatur ein bis 0 eingegeben wurde. Gib das Maximum der eingeg. Zahlen aus.
- Änderung (zu Summe): Die Ergebnisvariable wird nicht mehr in jedem Durchlauf aktualisiert, sondern in Abhängigkeit von der Eingabe.

Aufgabe: Maximium bestimmen

```
max_so_far = 0;
eingabe = 1;
while eingabe ~= 0
    eingabe = input('Zahl: ');
    if max_so_far < eingabe
        max_so_far = eingabe;
    end
end
fprintf('Maximum: %d\n', max_so_far);
```

Kontrollstruktur: if ... else

- Allgemein:

```
if bedingung
  block
else
  block
end
```

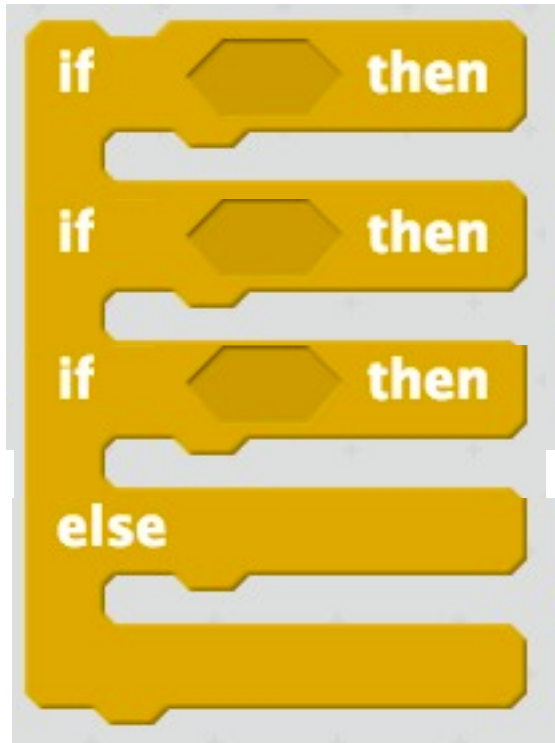
- else-Zweig kann entfallen

```
if bedingung
  block
end
```

Kontrollstruktur: if ... elseif ... ... else

- geschachtelt:

```
if  bedingung1
  block
elseif  bedingung2
  block
elseif  bedingung3
  block
...
else
  block
end
```



Aufgabe: Mini-Taschenrechner

- Lies eine float-Zahl, einen Operator (+, -, * oder / als char) und eine weitere float-Zahl von Tastatur ein.
- Gib das Ergebnis des Operator angewendet auf die beiden Operanden aus.
- Gib eine Fehlermeldung aus, wenn ein ungültiger Operator eingegeben wurde.