1. Grundlegende Konzepte der Informatik

Inhalt

- Algorithmen
- Darstellung von Algorithmen mit Programmablaufplänen
- Beispiele für Algorithmen
- Aussagenlogik
- **Zahlensysteme**
- Kodierung

Zahlensysteme

Menschen nutzen zur Angabe von Werten und zum Rechnen vorzugsweise das **Dezimalsystem**

Beispiel

435 Fische aus dem Teich gefischt, d.h. $4 \cdot 10^2 + 3 \cdot 10^1 + 5 \cdot 10^0$

Digitale Rechner speichern Daten im **Dualsystem**

$435_{dez} = 1100110011_{binär}$

1100110011 bedeutet $1 \cdot 2^9 + 1 \cdot 2^8 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^1 + 1 \cdot 2^0$

Andere Zahlensysteme sind z.B. das **Oktalsystem** und das **Hexadezimalsystem**

Zahlensysteme

Dezimalsystem, Binärsystem, Oktalsystem und Hexadezimalsystem sind s.g. Positionssysteme

Basis: B (10,2,8,16)

Ziffern: 0 bis B-1

Ziffern an Stellen von rechts nach links mit Faktor B^0, B^1, B^2 usw. bewertet

Es gibt Zahlensysteme, die keine Positionssysteme sind:

Römische Zahlenschrift, z.B. MMXIV für 2014

Zahlensysteme: Binärsystem

Binärdarstellung:

- ist die Interne Repräsentation der Daten, die vom Typ Integer und Double sind
  (inkl. vorzeichenlose und längere Formate)
- Werte werden bei Eingabe in Binärzahlen umgewandelt und intern so weiterverarbeitet
- einige Rechenoperationen und Vergleichsoperationen können auch eleganter und schneller programmiert werden, wenn man die binäre Repräsentation ausnutzt.

Binärdarstellung

Umrechnung Dezimal → Binär

Eingabe: Dezimalzahl d
Ausgabe: Binärzahl auf Feld b
mit niederwertigster Stelle rechts

Prinzip: Teile Dezimalzahl wiederholt ganzzahlig durch 2 und notiere den Rest der Division von rechts nach links. Wiederhole solange wie die ganzzahlige Division noch ein Ergebnis ungleich Null liefert. Die notierten 0- und 1-Werte ergeben die Binärzahl.

Binärdarstellung

Umrechnung Dezimal → Binär

Beispiel: d=35

35/2=17, Rest = 1, notiere _ _ _ _ _ 1
17/2=8, Rest = 1, notiere _ _ _ _ 1 1
8/2=4, Rest = 0, notiere _ _ _ 0 1 1
4/2=2, Rest = 0, notiere _ _ 0 0 1 1
2/2=1, Rest = 0, notiere _ 0 0 0 1 1
1/2=0, Rest = 1, notiere 1 0 0 0 1 1

Binärdarstellung

Umrechnung Binär → Dezimal

- Per Wertigkeit der einzelnen Bits $b_n\ b_{n-1} \ldots b_1\ b_0$:
  $b_n * 2^n + b_{n-1} * 2^{n-1} + \ldots. b_1 * 2^1 + b_0 * 2^0$

  Beispiel: $100011 \rightarrow 1*2^5 + 1* 2^1 + 1*2^0 = 35$

- Durch Multiplizieren mit 2 und aufaddieren:
  Binärzahl von links nach rechts durchlaufen, Summe (anfänglich 0) und Bitwert addieren.
  Summe jedes mal mit 2 multiplizieren,
  außer bei letzter Bitstelle

  Beispiel: 100011

  ((((((**1***2+**0**)*2+**0**)*2+**0**)*2+**1**)*2)+**1** = 35

Binar und Oktaldarstellung

Oktalzahlen werden anhand eines Postionssystems über 8 Ziffern gebildet:

Ziffern 0 bis 7

Zahl := $o_n\ o_{n-1} \ldots o_1\ o_0$

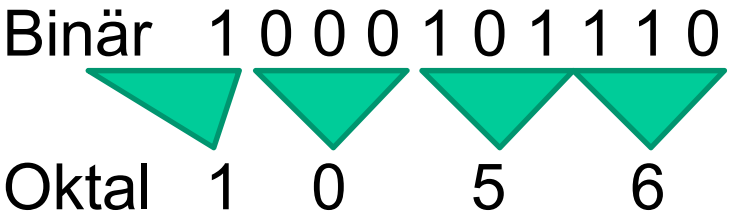
Der Wert der Zahl ergibt sich aus
$o_n * 8^n + o_{n-1} * 8^{n-1} \ldots o_1 * 8^1 + o_0 * 8^0$

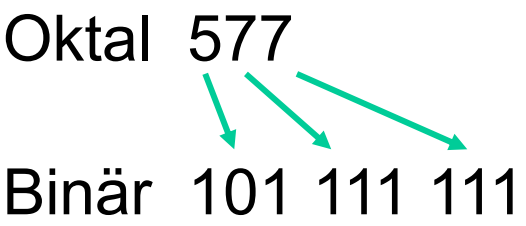
Eine Oktalzahl kann aus der Binärzahl abgelesen werden und umgekehrt. Drei Binärstellen ergeben durch ihren Wert jeweils eine Stelle der entsprechenden Oktalzahl.

Die Gruppierung der Binärstellen muss von rechts nach links erfolgen (beginnend bei niederwertigster Stelle)

Binär- und Oktaldarstellung

Beispiele:





Binär- und Hexadezimaldarstellung

Hexadezimalzahlen werden anhand eines Positionssystems über 16 Ziffern gebildet:

Ziffern 0 bis F: 0 1 2 3 4 5 6 7 8 9 A B C D E F

Zahl := $h_n\ h_{n-1}\ \ldots\ h_1\ h_0$

Der Wert der Zahl ergibt sich aus
$h_n*16^n + h_{n-1}*16^{n-1} \ldots h_1*16^1 + h_0 * 16^0$

Eine Hexadezimalzahl kann aus der Binärzahl abgelesen werden und umgekehrt. Vier Binärstellen ergeben durch ihren Wert jeweils eine Stelle der entsprechenden Hexadezimalzahl.

Die Gruppierung der Binärstellen muss von rechts nach links erfolgen (beginnend ab niederwertigster Stelle)

Binar und Hexadezimaldarstellung

Beispiele:

Binär 1 0 0 0 1 0 1 1 1 0

Hex 2 2 E

Hex 5FA

Binär 0101 1111 1010

Umwandlungen zwischen Positionssystemen (1)

Generalisierung:
Binär-, Oktal-, Dezimal- und Hexadezimalsysteme sind Positionssysteme mit unterschiedlicher Basis b:

Wert = $z_n * b^n + z_{n-1} * b^{n-1} + \ldots z_1 * b^1 + z_0 * b^0$

z_i sind jeweils Ziffern mit Wertevorrat im Intervall [0,b-1]

Zwischen zwei Systemen s1 und s2 ergibt sich eine Zusammenführbarkeit von Stellen wenn eine Basis eine ganzzahlige Potenz der anderen ist, d.h. es gilt

$$b_{s2} = (b_{s1})^n$$

n Ziffern des Systems s1 ergeben eine Ziffer im System s2

Umwandlungen zwischen Positionssystemen (2)

Gebräuchliche Umwandlungen:

s1	s2	n	
2	8	3	Binär ↔ Oktal
2	16	4	Binär ↔ Hexadezimal

Weitere mögliche Umwandlungen:

s1	s2	n	
2	4	2	Binär ↔ 4er System
4	16	2	4er ↔ Hexadezimal

Das Binärsystem wird benutzt, da man leicht zwei Werte in einer Speicherzelle speichern bzw. auf einer Leitung übertragen kann. Mittlerweile gibt's auch Speicherchips und Übertragungsverfahren, die mir mehrwertiger Logik arbeiten, z.B. MLC-Flash-Memories.

Datentypen

Obwohl alle Zahlen prinzipiell im Binärsystem dargestellt werden, ist bei der Programmierung eine Unterscheidung verschiedener Datentypen nötig:

Nichtnumerische Werte:

- Boolean (Wahrheitswerte)
- Zeichen und Zeichenketten (Strings)

Ausgewählte Bitmuster repräsentieren die Symbole für Buchstaben und Ziffern (siehe Kodierung)

Numerische Werte:

- Ganzzahlig
- Gebrochene Zahlen
- Varianten für verschiedene Wertebereiche und Genauigkeiten

Ganzzahlige Datentypen: unsigned integer

32-Bit: *unsigned integer*

Der gespeicherte Zahlenwert errechnet sich aus den zugehörigen Zweierpotenzen:

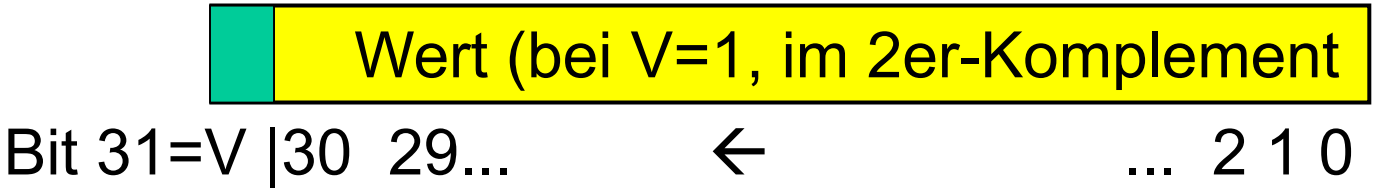
$$b_{31}2^{31}+ b_{30}2^{30} +.....+ b_{2}2^{2}+ b_{1}2^{1}+ b_{0}2^{0}$$

Die b_i stellen die betreffenden Bitwerte an der i-ten Stelle dar.

Maximaler Wert: 4294967295 = $2^{32}-1$
Minimaler Wert: 0

Ganzzahlige Datentypen: integer

32-Bit: *signed integer*



Das werthöchste Bit (hier 31) ist das Vorzeichen-Bit V.

Die Größe errechnet sich *bei positiven Zahlen* aus den zugehörigen Zweierpotenzen:

$$b_{30}2^{30}+ b_{29}2^{29} +.....+ b_{2}2^{2}+ b_{1}2^{1}+ b_{0}2^{0}$$

Bei *negativen Zahlen* sind die b_i im sogenannten *Zweier-Komplement* dargestellt.

Ganzzahlige Datentypen: integer

Durch Aufteilung des Darstellungsbereichs in negative und positive Zahlen ändern sich der maximale und der minimale darstellbare Wert wie folgt:

Maximaler Wert: 2147483647= 2^{31} -1
Minimaler Wert: -2147483648 = -2^{31}

Negative Zahlen im Zweierkomplement

Das Zweierkomplement einer Zahl erhält man, indem man die Binärdarstellung der Zahl bitweise komplementiert (Einer-Komplement) und danach eine „1“ addiert.
Beispiel: Zahl = -1

Binärdarstellung:	0000 0000 0000 0001
Einerkomplement:	1111 1111 1111 1110
1 - Addition:	**1111 1111 1111 1111**

Diese Darstellung ist die –1!

Man sieht, das durch die Komplementierung immer das V-Bit auf „1“ gesetzt ist !

Negative Zahlen im Zweierkomplement

Eine Zahl im Zweierkomplement kann per Addition mit anderen Zahlen verknüpft werden.

Beispiel 1: 24 + (-7)

```
24 = 00011000
 7 = 00000111
-7 = 11111001

 00011000
 11111001
 --------
100010001   = 17 (ohne 1 links)
```

Die Subtraktion kann auf eine Addition einer negierten Zahl abgebildet werden.

Negative Zahlen im Zweierkomplement

Eine Zahl im Zweierkomplement kann per Addition mit anderen Zahlen verknüpft werden.

Beispiel 2: 24 + (-28)

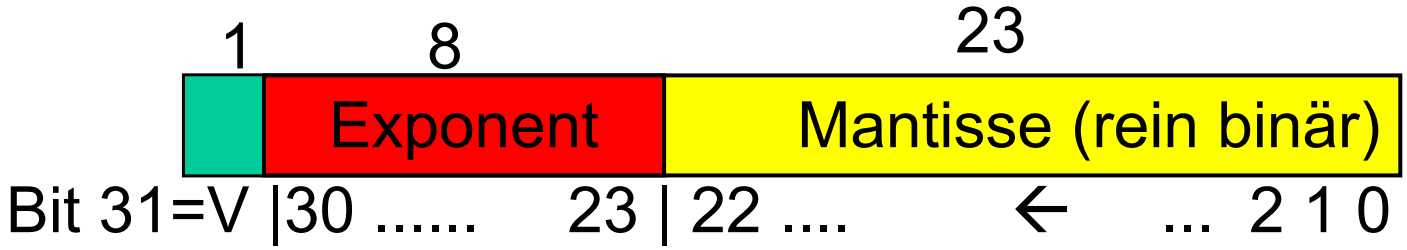
```
 24 = 00011000
 28 = 00011100
-28 = 11100100

 00011000
 11100100
 --------
011111100   = -4  (00000100 negiert und +1)
```

Gleitkomma-Datentypen: single

Gleitkomma-Datentypen in verschiedenen Genauigkeits-Kategorien (single, double)

32-Bit: *single*



Das Vorzeichenbit V(Bit 31) bezieht sich nur auf den Wertanteil der Zahl. Der Wert der Mantisse m errechnet sich aus den zugehörigen negativen Zweierpotenzen:

$$m = b_{22}2^{-1} + b_{21}2^{-2} + \ldots\ldots + b_2 2^{-21} + b_1 2^{-22} + b_0 2^{-23}$$

Die b_i stellen die betreffenden Bitwerte an der i-ten Stelle dar.

Gleitkomma-Datentypen: single

Zum Wert der Mantisse m wird immer eine 1 zusätzlich angenommen. Den Wert 1 + m nennt man Signifikand. Die 1 ist also nicht mit abgespeichert, sondern nur implizit vorhanden.

Der Exponent ist vorzeichenbehaftet und zur Basis 2. Dies wird erreicht, indem man den Exponenten um eine feste Größe (BIAS) erhöht darstellt. Der BIAS berechnet sich aus der Anzahl der Bits n des Exponenten:

$$BIAS = 2^{n-1} - 1$$

Mit n=8 ergibt sich 127.

Demnach muss der Binärwert des Exponenten e (die Bitstelle 2^3 hat dabei den Stellenwert 2^0) um den BIAS vermindert werden, um den realen Wert des Exponenten zur Basis 2 zu erhalten.

Gleitkomma-Datentypen: single

Der Wert Z einer 32-Bit Single-Zahl ist damit

$$Z = V \; Signifikand * 2^{e - BIAS}$$

Das Vorzeichen V ist + , falls der Bitwert von b_{31}=0 ist, sonst –

Beispiel:

V	Exponent	Mantisse
1	1000 0001	0100 0000 0000 0000 0000 000
-	e = 129	$m = 0*2^{-1} + 1* 2^{-2} + 0*2^{-1}\ 0* 2^{-3} +...+ 0*2^{-23}$
-	e-BIAS= 2	m = 1*0.25 = 0.25

$Z = - (1+0.25) * 2^{2} = - 1.25 * 4 = -5$

Gleitkomma-Datentypen: single

Damit ergeben sich für 32-Bit Single-Zahlen folgende Darstellungsbereiche:

Maximaler Wert: 3.40282e+38
absolut minimaler Wert: 1.17549e-38
(in dieser Darstellung ist e zur Basis 10 !)

Durch die begrenzte Anzahl von Bits zur Darstellung der Mantisse, ist die Anzahl genau darstellbarer Ziffern in einer Dezimaldarstellung ebenfalls begrenzt. Dabei werden die niederwertigen Ziffern betroffen.

Anzahl der genau dargestellten Dezimalziffern: 6

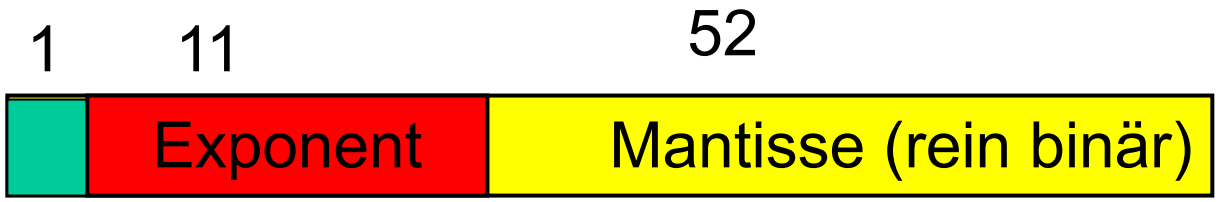
Das heißt:

100100.8 und 100100.9 nicht mehr sicher unterscheidbar, da niederwertigste Ziffer sich an Position 7 befindet.

Bei 20 011 292 als Geldbetrag wären die Zehner- und Einerstelle nicht genau abgebildet.

Gleitkomma-Datentypen: double

64-Bit: *double*



Das Vorzeichenbit V(Bit 63) bezieht sich nur auf den Wertanteil der Zahl. Der Wert der Mantisse m errechnet sich aus den zugehörigen negativen Zweierpotenzen:

$$m = B_{51}2^{-1} + b_{50}2^{-2} + \ldots\ldots + b_2 2^{-50} + b_1 2^{-51} + b_0 2^{-52}$$

Die b_i stellen die betreffenden Bitwerte an der i-ten Stelle dar.

Gleitkomma-Datentypen: double

Damit ergeben sich für 64-Bit double-Zahlen folgende Darstellungsbereiche:

Maximaler Wert: 1.797693e+308
absolut minimaler Wert: 2.225074e-308
(in dieser Darstellung ist e zur Basis 10)

Anzahl der genau dargestellten Dezimalziffern: 15