Daten in Programmen

- Daten werden in Bits und Bytes in Rechnern dargestellt.
- Ein Byte umfasst dabei 8 Bits und kann im Zahlenbereich der natürlichen Zahlen Werte von 0 bis 255 beinhalten.

Zahlen

- Zahlen bestehen aus Ziffern, wobei die Ziffern die Koeffizienten der Potenzen zu einer festgelegten Zahlenbasis sind.
- Die häufigste Zahlenbasis ist dabei die 10, wir sprechen von den Dezimalzahlen
- Betrachten wir die Zahl 108:

1	$*10^2$	→	100
0	$*10^1$	→	0
8	$*10^0$	→	8
		Summe:	108

Binärzahlen

- Dezimalzahlen basieren auf der Zahlenbasis 10 und verfügen über die Ziffern 0 bis 9.
- Binärzahlen basieren auf der Zahlenbasis 2 und verfüfgen nur über die Ziffern 0 und 1.
- Betrachten wir die Zahl 01101100

0	$*2^7$	→	0
1	$*2^6$	→	64
1	$*2^5$	→	32
0	$*2^4$	→	0
1	$*2^3$	→	8
1	$*2^2$	→	4
0	$*2^1$	→	0
0	$*2^0$	→	0
		Summe:	108

Umrechnung Dezimal → Binär

- Zur Umrechnung wird die Dezimalzahl durch die neue Zahlenbasis (2) ganzzahlig dividiert und der Rest notiert.
- Sodann wird der Quotient der vorangegangenen Division, solange er ungleich 0 ist, durch die neue Zahlenbasis (2) ganzzahlig dividiert und jweils der Rest notiert.
- Die notierten Reste ergeben die Ziffernfolge der Zahl im Zielzahlensystem, allerdings in der umgekehrten Reihenfolge.

Zahl			Quotient	Rest
108	:2	→	54	0
54	:2	→	27	0
27	:2	→	13	1
13	:2	→	6	1
6	:2	→	3	0
3	:2	→	1	1
1	:2	→	0	1

Ergebnis:
01101100 (von unten nach oben)

Octalzahlen

- Octalzahlen sind die Zahlen zur Zahlenbasis 8, sie umfassen die Ziffern 0 bis 7.
- In C-Programmen werden Octalzahlen mit einer führenen 0 gekennzeichnet
  - 108 - Dezimalzahl
  - 0154 - Octalzahl mit dem Wert 108.

1	$*8^2$	→	64
5	$*8^1$	→	40
4	$*8^0$	→	4
Sum			108

Octalzahlen

- Jede Ziffer einer Octalzahl wird in 3 bits codiert

001	101	100
1	5	4

- Die Zahlen 01101100 als Binärzahl, die 0154 als Octalzahl und die 108 als Dezimalzahl repräsentieren den selben Zahlenwert, er wird nur unterschiedlich dargestellt.

Hexadezimalzahlen

- Hexadezimalzahlen basieren auf der Zahlenbasis 16.
- Hexadezimalzahlen bestehen aus den Ziffern 0 bis 9 und a bis f für die Ziffern mit den Werten 10 bis 15.
- In C-Programmen werden Hexadezimalzahlen durch ein vorangestelltes 0x gekennzeichnet (z.B.: 0x6c)

A	10
B	11
C	12
D	13
E	14
F	15

Hexadezimalzahlen

- Jede Ziffer einer Hexadezimalzahl wird mit 4 Bit, einem HalbByte, einer Tetrade von Bits oder einem Nibbel (Nyble) codiert
- Zusammen ergeben die beiden Nibbel wieder die Binärzahl 01101100, die der dezimalen 108 entspricht.

6	$*16^1$	→	96
12	$*16^0$	→	12
		Summe:	108

6	C
0110	1100

Zahlensysteme in c

Zahlensystem	Zahlenbasis	Ziffern	Zahl in c
binär	2	0,1	
octal	8	0 - 7	0154
dezimal	10	0 - 9	108
hexadezimal	16	0-9, a-f	0x6c oder 0x6C

Zahlen, größer 255

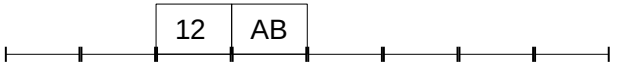
- In einem Byte können Zahlen zwischen 0 und 255 dargestellt werden.
- Sollen größere Zahlen im Rechner dargestellt werden, so muss man mehrere Bytes aneinander reihen.
- Der Zahl 4779 entspricht in hexadezimaler Darstellung die Zahl 0x12ab, die sich in zwei Bytes darstellen lässt. Binär: 0001 0010 1010 1011

Darstellung von Zahlen im Speicher

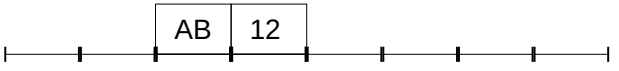
- Den Speicher eines Rechners kann man sich wie ein Band von aneinandergereihten Speicherplätzen vorstellen, die durchnummeriert sind.
- Die laufende Nummer bezeichnet man als die Adresse des Speicherplatzes.
- Bei den meisten heutigen Rechnersystemen ist jeder Speicherplatz ein Byte groß.

Darstellung von Zahlen im Speicher

- Belegt eine Zahl (4779 oder 0x12ab) nun zwei oder mehr Bytes im Speicher, so belegt sie mehrere Speicherplätze.
- Hierfür gibt es zwei gebräuchliche Arten:
  - **little endian** (intel 80xxx, PC und kompatible)
  - **big endian** (motorola Prozessoren, java)



big endian



little endian

Little endian

- Die Zahl beginnt an der kleinsten Adresse mit dem geringwertigsten Byte (sie beginnt mit dem kleinen Ende, deshalb little endian).
- Die weiteren Bytes enthalten die höherwertigen Teile der Zahl auf den jeweils hören Adressen.
- Innerhalb eines Bytes werden zwei Hexadezimalziffern, wie bereits dargestellt, gespeichert.
- Heute sehr gebräuchlich, da sich Operationen gut cascadieren lassen, da neiderwertigerer Teil der Zahl auf niederer Adresse und höherwertigerer Teil der Zahl auf höherer Adresse abgelegt ist.

Big endian

- Die Zahl endet an der größten Adresse mit dem geringwertigsten Byte (sie beginnt mit dem großen Ende, deshalb big endian).
- Die Zahl lässt sich bei aufsteilgenden Adressen von links nach rechts „normal“ lesen.
- Kommt selten vor.

Darstellung negativer Zahlen

- Bislang wurde nur die Darstellung von Zahlen aus dem Raum der Natürlichen Zahlen (nicht negative Zahlen) betrachtet.
- Negative Zahlen werden im sogenannten **Zweierkomplement** dargestellt.

Bildung des Zweierkomplements

- Die zu negierende Zahl wird zunächst bitweise negiert, dabei wird aus jedem Bit 0 ein Bit 1 und umgekehrt.
- Im zweiten Schritt wird eine 1 addiert

```
Beispiel: -108
108
01101100 (binär)
1. Schritt Negation (0→1, 1→0)

01101100 → 10010011

2. Schritt Increment (+1)

 10010011
+00000001
Üb      1 1
---------
 10010100

10010100 entspricht nun der -108
```

Binäre Addition:
0+1→ 1
0+0→ 0
1+1→ 0, Übertag 1

Die Probe

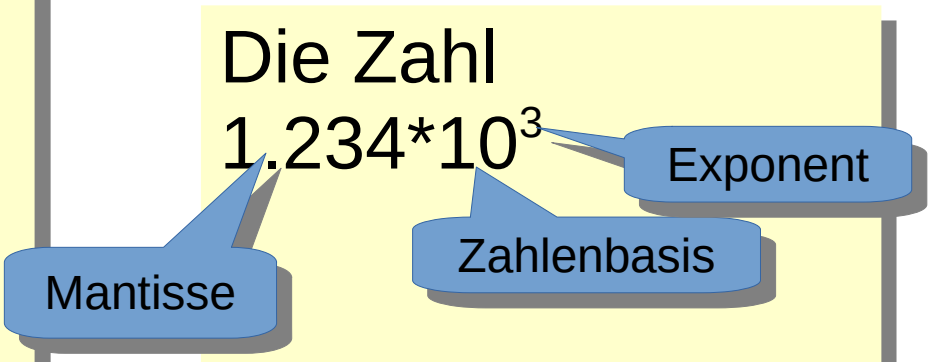
- Zur Probe brechnen wir nun die Summe aus 108 + -108, die 0 ergeben sollte.
- Abgesehen vom Überlauf (1 ander 9. Stelle) ist das der Fall. Der Überlauf wird verworfen, somit ist das Ergebnis 0.

Dezimal	binär
108 + -108 ------ 0	01101100 +10010100 --------- 100000000

- Mit Binärzahlen wird immer in einer festgelegten Verarbeitungsbreite gerechnet, üblich sind 1,2,4 oder 8 Byte.
- Es obliegt dem Programm, ob eine Bitkombination als immer postive (natürliche Zahl) oder, wenn sie mit einer 1 an der höchstwertigen Stelle beginnt, als positive oder negative Zahl interpretiert wird.
- Die Bitfolge `10010100` kann in einem Byte als natürliche Zahl 148 oder ganze Zahl -108 interpertiert werden.
- Zahlen, die an der höchstwertigen Stelle mit einer 0 beginnen, sind immer postive Zahlen.

Gleitpunktzahlen

- Darstellung gebrochener, rationaler Zahlen
- Zahlen werden durch eine Zahl, die als ganze Zahl oder Dezimalbruch dargestellt werden (Mantitte genannt) und einem Exponenten zu einer vereinbarten Zahlenbasis, deren Potenz mit der Mantisse multipliziert wird gebildet



Gleitpunktzahlen

- Zahlen in dieser Form sind bekannt als Zahlen im wissenschaftlichen Format. Sie finden bei gebräuchlichen Taschenrechnern anwendung in der Form 1.234E4
- In Rechnersystemen werden Gleitpunktzahlen gemeinhin nach der Norm IEEE 754 (ANSI/IEEE Std 754-1985; IEC-60559:1989…) gebildet

Gleitpunktzahlen

- Die Mantisse wird binär codiert, dabei werden die negativen Potenzen zur 2 verwendet.
- Addiert man alle Werte ab der Zeil 2^0 so sollte sich der Wert 2 näherungsweise ergeben, in unserem Fall ergibt sich 1,984375. Tatsächlich ist die Zahl 2 als Gleitpunktzahl nur näherungsweise darstellbar.

2^3	8
2^2	4
2^1	2
2^0	1
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625
2^{-5}	0.03125
2^{-6}	0.015625

Gleitpunktzahlen

- Der Exponent wird binär codiert und als positive Zahl mit einer Verschiebung gebildet.
- Mikt einer Veerschiebung des Dezimalpunktes in der Matisse nach links wird der Exponent incrementiert, bei einer Verschiebung des Dezimaölpunktes nach rechts wird der Exponent decrementiert. Der Wert der gesamten Zahl böleibt dabei unverändert.

Gleitpunktzahlen

- In Gleitpunktzahlen werden Mantisse und Exponent so gebildet, dass die Mantisse (binär dargestellt) mit **1.**xxx beginnt. Man spricht von einer normierten Gleitpunktzahl.
- Da die erste Stelle nun immer 1 ist, wird sie nicht mit gespeichert. Somit gewinnt man eine höhere Genauigkeit bei gleichem Speicherbedarf.

Gleitpunktzahlen

Werden Gleipunktzahlen additiv vernüpft, müssen die Exponenten der beiden Operanden einander gleichen. Dazu muss die Mantisse verschoben werden. Da die Mantisse nur eine endliche Länge hat, kann es passieren, dass signifikante Stellen verloren gehen oder die Mantisse un d damit die betrachtete Zahl 0 wird. Es ergeben sich Rechenfehler, wenn große und kleine Zahlen additiv veknüpft werden. Aus diesem Grunde werden Gleitpunktzahlen in der Finanzmathematik eher vermieden.

Gleitpunktzahlen

v z	Exponent	Mantisse

Übliche Längen:

Datentyp	float	double
Vorzeichen der Zahl	1 bit	1 bit
Exponent	8 bit	11 bit
Mantisse	23 bit	52 bit

Weitere leicht verständliche Erläuterungen unter:
https://www.elektronik-kompendium.de/sites/dig/1807231.htm