Datenformate

Datenformate

(digitale) Daten bestehen aus einzelnen Bits;
- in Kontroll-Registern sind einzelne Bits bedeutsam;
- für komplexere Datenstrukturen werden Bits üblicherweise in Vierer-gruppen zu einem „Hex-Digit“ kombiniert (üblich ist auch die oktale Darstellung)

Dez	binär	Hex	oktal
00	0000	0x0	0o00
01	0001	0x1	0o01
02	0010	0x2	0o02
03	0011	0x3	0o03
04	0100	0x4	0o04
05	0101	0x5	0o05
06	0110	0x6	0o06
07	0111	0x7	0o07

Dez	binär	Hex	oktal
08	1000	0x8	0o10
09	1001	0x9	0o11
10	1010	0xA	0o12
11	1011	0xB	0o13
12	1100	0xC	0o14
13	1101	0xD	0o15
14	1110	0xE	0o16
15	1111	0xF	0o17

Üblich ist 1 Byte = 8 Bits = 2 HexDigits als Einheit

Datenfomate – negative ganze Zahlen

Für ganze Zahlen mit Vorzeichen wird das 8. Bit als Vorzeichen-Bit betrachtet; Darstellung als sog. Zweierkomplement:

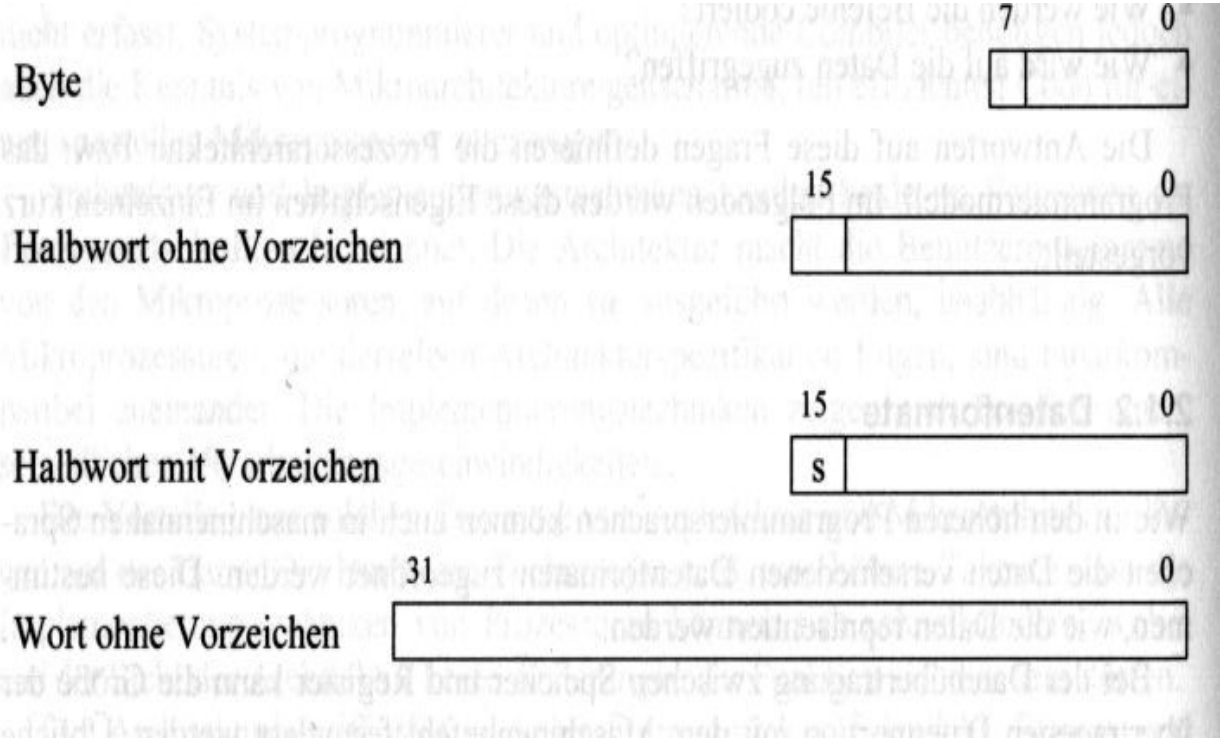
positive Zahl p, dann ist -p = invertierte Bits + 1

Vorteil: Subtraktion entspricht Addition des 2-Komplements

Dez	**binär**	**Hex**	**oktal**
-8	1000	0x8	0o10
-7	1001	0x9	0o11
-6	1010	0xA	0o12
-5	1011	0xB	0o13
-4	1100	0xC	0o14
-3	1101	0xD	0o15
-2	1110	0xE	0o16
-1	1111	0xF	0o17

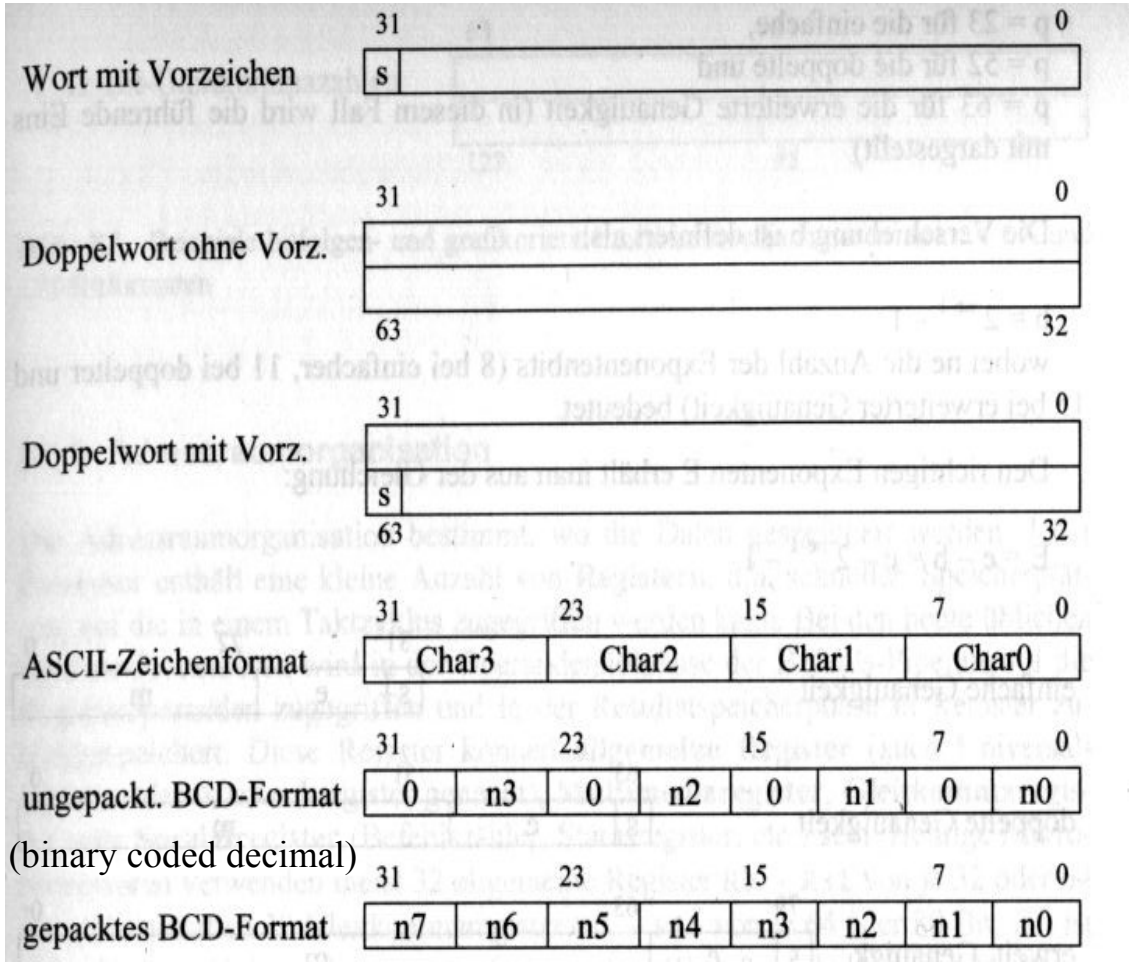
Datenformate – Ganze Zahlen

In der Praxis reicht ein Byte meist nicht aus, komplexere Datentypen entstehen durch Aneinanderreihung von Bytes:



	Wertebereich
Byte	0 bis 255 (-128 bis 127 mit Vorzeichen)
Halbwort ohne Vorzeichen	0 bis 65.535
Halbwort mit Vorzeichen	-32.768 bis 32.767
Wort ohne Vorzeichen	0 bis 4.294.967.295

Datenformate – Ganze Zahlen, Text



Wertebereich

-2.147.483.648 bis
2.147.483.647

0 bis ~1.8 x 10^{19}

~-0.9 x 10^{19} bis
~0.9 x 10^{19}

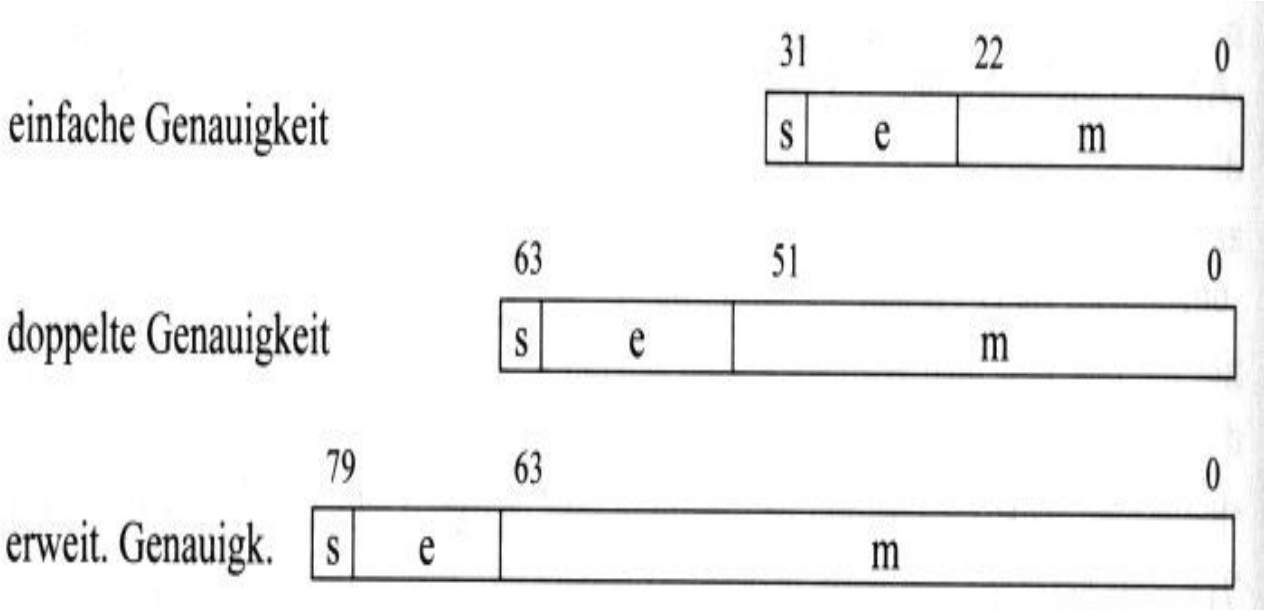
0 bis 9.999

0 bis 99.999.999

Datenformate – reelle Zahlen (Real, Float)

Reelle Zahlen werden beschrieben durch ein Vorzeichen s, eine Mantisse m, einen Exponenten e und eine konstante Verschiebung b:

$$Z = (-1)^s \cdot 1{,}m \cdot 2^{e-b}$$

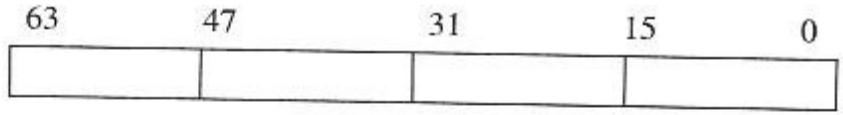


	Wertebereich	Genauigkeit in Dezimalstellen
einfache Genauigkeit	ca. $\pm 10^{\pm 38}$	7 - 8
doppelte Genauigkeit	ca. $\pm 10^{\pm 308}$	15 - 16
erweit. Genauigk.	ca. $\pm 10^{\pm 4932}$	19 - 20

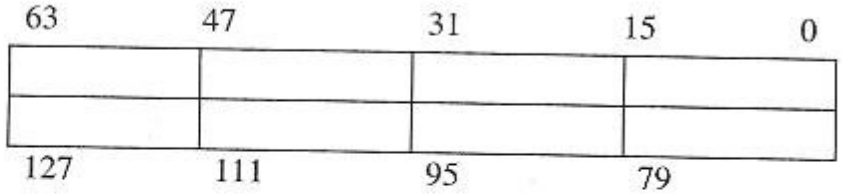
Spezielle Bitmuster für 0, ±∞, ungültige Zahl (NaN)

Datenformate – Multimedia-Formate

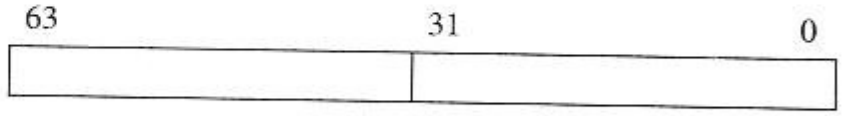
4 Bitfelder mit je 16 Bits



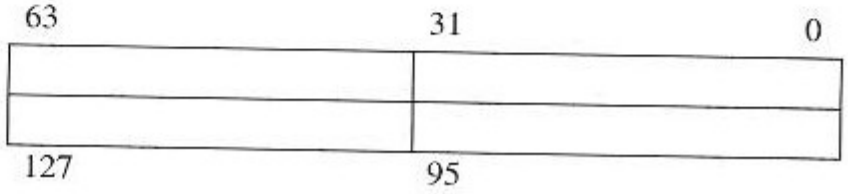
8 Bitfelder mit je 16 Bits



2 32-Bit-Gleitkommazahlen



4 32-Bit-Gleitkommazahlen



Mulimediabefehle führen dieselbe Operation auf allen Teilwörtern aus

Datenformate – Anordnung der Bytes im Speicher ...

... ist leider nicht einheitlich !

Wortadresse:	x0				x4			
Bytestelle im Wort:	7	6	5	4	3	2	1	0

Wortadresse:	x0				x4			
Bytestelle im Wort:	0	1	2	3	4	5	6	7

Abb. 2.4. Big-endian- (oben) und Little-endian- (unten) Formate.

x86-Architektur verwendet little-endian !

Wichtig bei Datentransfers von/zu anderen Plattformen !
Datentyp, zu dem ein Byte gehört, muss bekannt sein.

Datenformate – Text: ASCII = American Standard Code for Information Interchange

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Datenformate – ASCII extended: Ausnutzung des 8. Bits

Erweiterung des Ascii-Standards um graphische und nationale Sonder-Zeichen

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	ÿ	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	█	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
141	ì	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	░	192	└	208	╨	224	α	240	≡		

Source: www.LookupTables.com

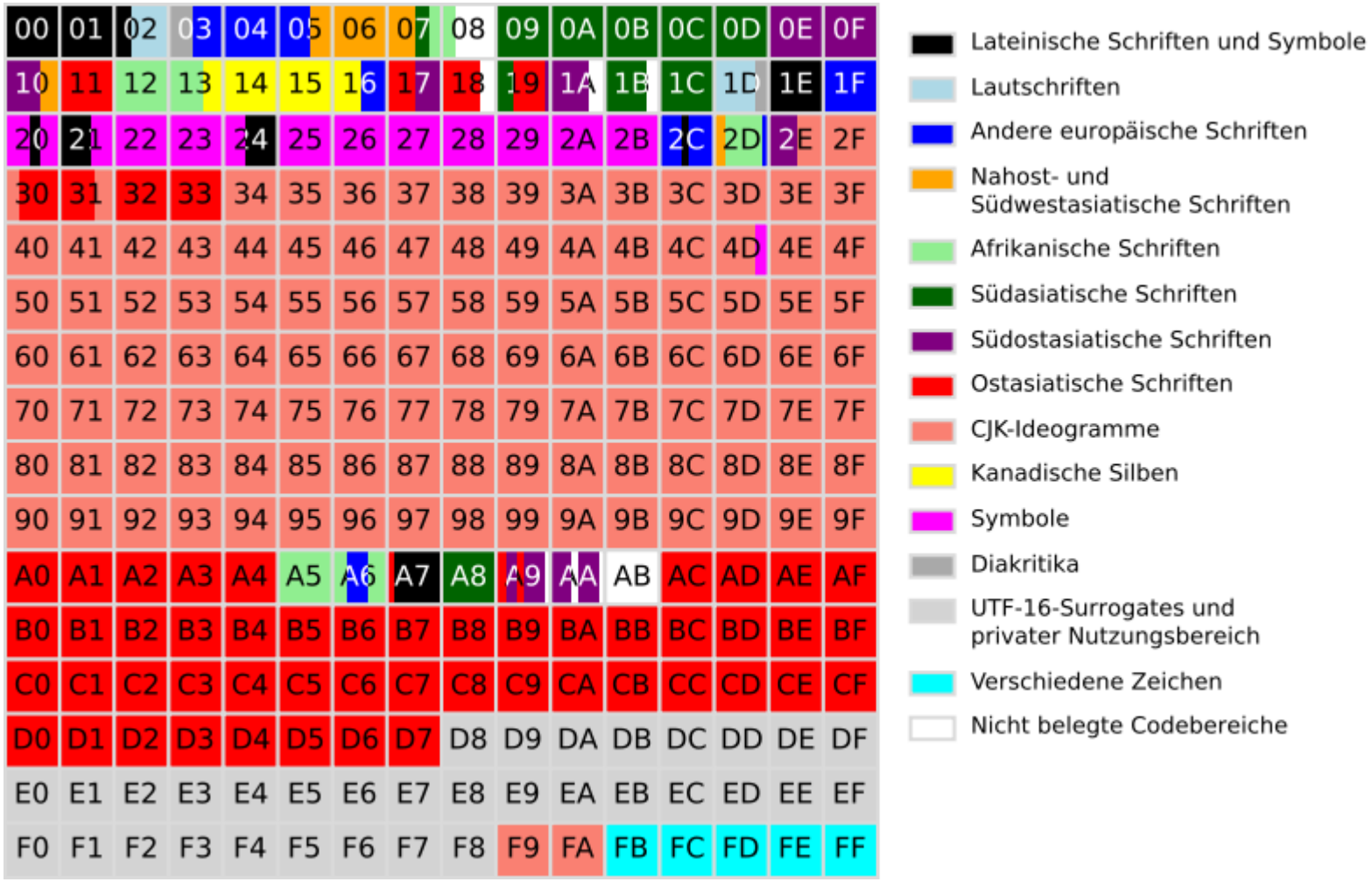
Leider nicht ganz einheitlich gehandhabt.

Datenformate – ISO 8859-1

8-bit ISO 8859-1 Latin 1 characters

160		176	°	192	À	208	Ð	224	à	240	ð
161	¡	177	±	193	Á	209	Ñ	225	á	241	ñ
162	¢	178	²	194	Â	210	Ò	226	â	242	ò
163	£	179	³	195	Ã	211	Ó	227	ã	243	ó
164	¤	180	´	196	Ä	212	Ô	228	ä	244	ô
165	¥	181	µ	197	Å	213	Õ	229	å	245	õ
166	¦	182	¶	198	Æ	214	Ö	230	æ	246	ö
167	§	183	·	199	Ç	215	×	231	ç	247	÷
168	¨	184	¸	200	È	216	Ø	232	è	248	ø
169	©	185	¹	201	É	217	Ù	233	é	249	ù
170	ª	186	º	202	Ê	218	Ú	234	ê	250	ú
171	«	187	»	203	Ë	219	Û	235	ë	251	û
172	¬	188	¼	204	Ì	220	Ü	236	ì	252	ü
173	-	189	½	205	Í	221	Ý	237	í	253	ý
174	®	190	¾	206	Î	222	Þ	238	î	254	þ
175	¯	191	¿	207	Ï	223	ß	239	ï	255	ÿ

Datenformate – UniCode (16 bit)



Datenformate – Text im Speicher

```
> hexdump -C Faust.txt
00000000  48 61 62 65 20 6e 75 6e  2c 20 61 63 68 21 20 50  |Habe nun, ach! P|
00000010  68 69 6c 6f 73 6f 70 68  69 65 2c 0a 4a 75 72 69  |hilosophie,.Juri|
00000020  73 74 65 72 65 69 20 75  6e 64 20 4d 65 64 69 7a  |sterei und Mediz|
00000030  69 6e 2c 0a 55 6e 64 20  6c 65 69 64 65 72 20 61  |in,.Und leider a|
00000040  75 63 68 20 54 68 65 6f  6c 6f 67 69 65 0a 44 75  |uch Theologie.Du|
00000050  72 63 68 61 75 73 20 73  74 75 64 69 65 72 74 2c  |rchaus studiert,|
00000060  20 6d 69 74 20 68 65 69  df 65 6d 20 42 65 6d fc  | mit heißem Bemü|
00000070  68 6e 2e 0a 44 61 20 73  74 65 68 20 69 63 68 20  |hn..Da steh ich |
00000080  6e 75 6e 2c 20 69 63 68  20 61 72 6d 65 72 20 54  |nun, ich armer T|
00000090  6f 72 21 0a 55 6e 64 20  62 69 6e 20 73 6f 20 6b  |or!.Und bin so k|
000000a0  6c 75 67 20 61 6c 73 20  77 69 65 20 7a 75 76 6f  |lug als wie zuvo|
000000b0  72 3b 0a 48 65 69 df 65  20 4d 61 67 69 73 74 65  |r;.Heiße Magiste|
000000c0  72 2c 20 68 65 69 df 65  20 44 6f 6b 74 6f 72 20  |r, heiße Doktor |
000000d0  67 61 72 0a 55 6e 64 20  7a 69 65 68 65 20 73 63  |gar.Und ziehe sc|
000000e0  68 6f 6e 20 61 6e 20 64  69 65 20 7a 65 68 65 6e  |hon an die zehen|
000000f0  20 4a 61 68 72 0a 48 65  72 61 75 66 2c 20 68 65  | Jahr.Herauf, he|
00000100  72 61 62 20 75 6e 64 20  71 75 65 72 20 75 6e 64  |rab und quer und|
00000110  20 6b 72 75 6d 6d 0a 4d  65 69 6e 65 20 53 63 68  | krumm.Meine Sch|
00000120  fc 6c 65 72 20 61 6e 20  64 65 72 20 4e 61 73 65  |üler an der Nase|
00000130  20 68 65 72 75 6d 2d 0a  55 6e 64 20 73 65 68 65  | herum-.Und sehe|
00000140  2c 20 64 61 df 20 77 69  72 20 6e 69 63 68 74 73  |, daß wir nichts|
00000150  20 77 69 73 73 65 6e 20  6b f6 6e 6e 65 6e 21 0a  | wissen können!.|
```

Datenformate

Nahezu jede Anwendung bringt ihr eigenes Datenformat mit
(Word, Excel, OpenOffice, ...)
führt zu unendlichen Kompatibilitätsproblemen.

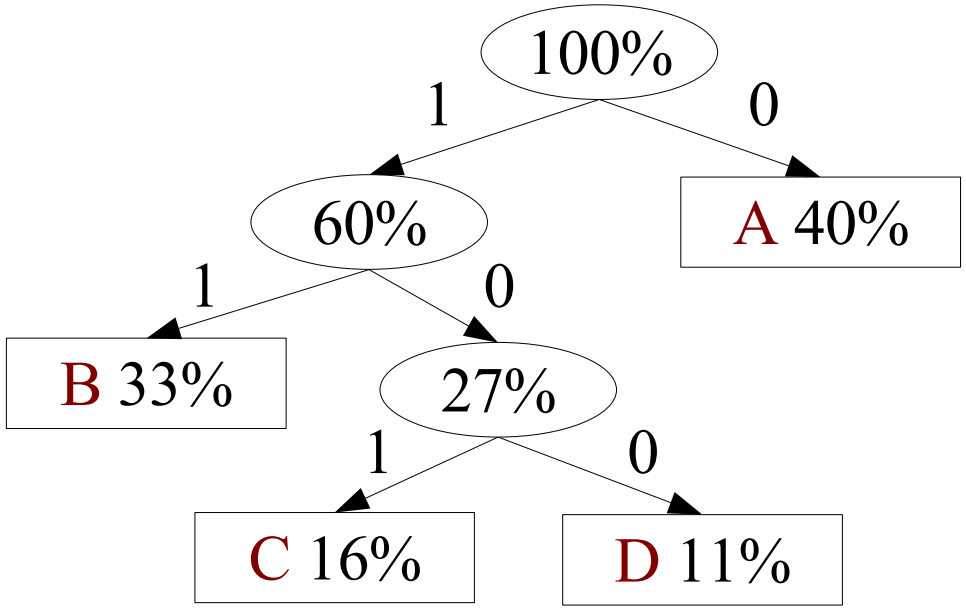
Jeder C++-Klasse entspricht im Prinzip einem eigenen „Datenformat“;
Methoden der Klasse sind für die Prozessierung der Daten
(Drucken, Speichern, Verarbeiten, Konvertieren, ...) zuständig

Viele moderne Datenformate bauen auf Ascii auf -
SGML, HTML, XML ...
recht gut lesbar, nahezu selbst-dokumentierend, aber Speicher-aufwändig
(die Ziffernfolge „12345678“ benötigt 8 Bytes, die dargestellte Zahl aber nur 3 !)
Komprimierungsverfahen (z.B. „ZIP“) helfen, den zusätzlichen
Speicherbedarf erträglich zu halten.

Datenformate – Huffman-Kodierung

Aufbau eines binärer Baumes:

1. Bestimmung der Häufigkeit der Zeichen in den Daten
2. Ordnung der Zeichen nach Häufigkeit
3. Zuordnung von Bits 1 und 0 zu den beiden Zeichen mit der geringsten Häufigkeit
4. Zusammenfassen dieser beiden Zeichen
5. Weiter mit 2. bis alle Zeichen zusammengefasst sind



A (40%) = 0
B (33%) = 11
C (16%) = 101
D (11%) = 100