Bitte ein Byte!
Grundlagen der Informatik: Von Bits und Bytes

Klausurrelevant sind die Themen (einschließlich untergeordnete Themen), die durch Ausrufezeichen (!!!) gekennzeichnet sind.

Literaturliste

Grundlagen und Nachschlagewerke

Friedrich L. Bauer, Gerhard Goos; Informatik : eine einfuehrende Uebersicht; Springer; Berlin 1991; 4. Aufl.
Roland Fahrion; Wirtschaftsinformatik : Grundlagen und Anwendungen; Physica-Verl.; Heidelberg 1989
Hans Robert Hansen; Wirtschaftsinformatik; Fischer Uni-Taschenbuecher; Stuttgart 1987; 3. Aufl.
Gregor Kuhlmann, u.a.; Computerwissen für Einsteiger; rowohlt; Reinbeck 1996
Der Computer, Wie funktioniert das?; Meyers Lexikonverlag; Mannheim 1994
dtv-Atlas zur Informatik; dtv; München 1995
Duden Informatik; Duden Verlag; Mannheim 1993
Werner u.a.; Taschenbuch der Informatik; Fachbuchverlag Leipzig; Leipzig 1995

Theoretische Informatik

Douglas R. Hofstadter; Gödel, Escher, Bach; dtv; 1992
Andrew Hodges; Alan Turing, Enigma; Kammerer & Unverzagt; Berlin 1989
A. M. Turing; On computable numbers, with an application to the Entscheidungsproblem

Was ist Informatik?

Informatik ist die Wissenschaft von der systematischen Verarbeitung von Informationen

-insbesondere mit Hilfe von Digitalrechnern.

Informatik befaßt sich mit

- der Struktur, der Wirkungsweise, den Fähigkeiten und dem Konstruktionsprinzip von Informationsverarbeitungssystemen.

- Strukturen, Eigenschaften und Beschreibungsmöglichkeiten von Informationen und von Informationsverarbeitungsprozessen.

- Möglichkeiten der Strukturierung, Formalisierung und Mathematisierung von Anwendungsgebieten sowie der Modellbildung und Simulation.

- Analyse und (Re-)Organisation der Arbeit mit Hilfe informationstechnischer Mittel, ihre maschinelle Unterstützung oder ihre Ersetzung durch Maschinen

Methoden der Informatik

Die Informatik untersucht mit abstrakten Zeichen, Objekten und Begriffen formale Strukturen (Daten-, Sprach- und Systemstrukturen). Sie beschreibt, analysiert und konstruiert auf verschiedenen Abstraktionsebenen.

Informatik = <u>Infor</u>mation + Auto<u>matik</u>

Einordnung der Informatik in das bestehende Wissenschaftsgefüge.
Informatik ist eine interdisziplinäre Wissenschaft, sie ist keine

- Naturwissenschaft, da ihre Objekte von Menschen geschaffene Systeme sind.
- klassische Ingenieurswissenschaft, da ihre Gegenstände meist immateriell sind
- Geisteswissenschaft, da sie ingenieursmäßig entwickelt und praktische Anwendersysteme erstellt.
- Informatik ist keine Strukturwissenschaft wie die Mathematik, da sie sich von jener erheblich in technischer und sozialer Wirksamkeit unterscheidet.

Schwerpunkte der Teilgebiete der Informatik !!!

Technische Informatik

- **Schaltungstechnologie**
- **Mikroprogrammierung**
- **Rechnerorganisation**
- **Prozeßrechner**
- **Spezialrechner**
- **Peripherie**
- **Sonstiges**

Praktische Informatik

- **Datenstrukturen, Datenorganisation**
- **Programmier- und Dialogsprachen**
- **Programmiertechnologie**
- **Übersetzerbau**
- **Betriebssysteme**
- **Informationssysteme, Kommunikationssysteme**
- **Graphische Datenverarbeitung**
- **Simulation**
- **Kognitive Verfahren und Systeme**
- **Sonstiges**

Theoretische Informatik (siehe Anhang)

- **Berechenbarkeitstheorie, Rekursive Funktionen**
- **Komplexitätstheorie**
- **Automatentheorie**
- **Theorie der Programmierung, Formale Sprachen**
- **Informationstheorie, Kommunikationstheorie, Codierungstheorie**
- **Kryptologie**
- **Mathematische Modelle für Rechensysteme**
- **Sonstiges**

	Informatik	
Kerninformatik		**Verwandte Gebiete**
Technische Informatik Hardware		**Elektrotechnik, Physik, Halbleitertechnik**
Praktische Informatik Software		**Angewandte Informatik** Wirtschaftsinformatik Ingenierinformatik Umwelt-Informatik Informatik in den Naturwissenschaften ...
Theoretische Informatik Mathematische Modelle von Computern und Programmierung		**Mathematik**

Was ist Information? !!!

Information wird mit Hilfe von Nachrichten übertragen. Nachrichten bestehen aus Folgen von Zeichen, die durch Signale dargestellt werden.

Die Größe der Information wird bestimmt durch die Neuigkeit und damit durch die Unwahrscheinlichkeit der übertragenen Zeichen. Von Shannon stammt die formale Definition

$$I(X) = -\log_2 p(X)$$

Dabei ist X eine Nachricht und p(X) die Wahrscheinlichkeit ihres Auftretens.

Bei binärer Darstellung ist die Wahrscheinlichkeit der 0 und der 1 gleich 0,5. Deshalb definiert man I(0)=I(1)=1 Bit. Weitere Größen:

1 Nibble	=4 Bit	
1 Byte	=8 Bit	
1 Word	=16 Bit	
1 KiloByte	=1024 Byte	$=2^{10}$ Byte
1 MegaByte	=1024 KiloByte	$=2^{20}$ Byte
1 GigaByte	=1024 MegaByte	$=2^{30}$ Byte
1 TerraByte	=1024 GigaByte	$=2^{40}$ Byte

Zahlensysteme !!!

Aufbau eines Zahlensystems

Stellenwertschreibung:
Durch die Stellenwertschreibung erhält eine Zahl ihren Wert durch die Stellung der Ziffern. Dabei ist die Basis des Zahlensystems maßgeblich, denn die Stellenwerte berechnen sich als Potenzen mit der Basis des Zahlensystems.

Polyadisches Zahlensystem:

$$Zahl = \sum_{i=0}^{n} a_i \cdot b^i$$

Das Dualsystem

Das Dualsystem besteht aus den beiden Ziffern 0 und 1, kennt somit nur zwei Zustände, die sich technisch extrem leicht realisieren lassen.

Strom fließt	↔	Strom fließt nicht
Licht an	↔	Licht aus
Spannung da	↔	Spannung weg
Relaiskontakt angezogen	↔	Relaiskontakt abgefallen etc.

Dezimalsystem

Das Dezimalsystem hat sich über die Jahrtausende entwickelt und findet sich in vielen Kulturen wieder. Ausschlaggebend für die Wahl der Basis war die Anatomie der menschlichen Hand, was sich auch in den frühen Zahlzeichen widerspiegelt. Die Null wurde um 800 nach Christus von den Indern "erfunden".

Hexa- oder Sedezimalsystem

Die verbreiteteste Wortlänge in der Digitalelektronik ist 8 Bit (1 Byte). Für die Halbbytes bietet sich das Sedezimalsystem zur Berechnung an.

hexa	=	sechs (griechisch),
decem	=	zehn (lateinisch),
sedecem	=	sechzehn (lateinisch).

Im Hexadezimalsystem braucht man zur Darstellung der Zahl genau 16 Ziffern, muß also zu den zehn des Dezimalsystems noch neue schaffen. Damit nicht neue Zahlensymbole geschaffen werden (und behalten!) werden müssen, hat man auf die ersten Buchstaben des Alphabets zurückgegriffen, also A, B, C, D, E, F.

Zeichenvorrat und Zählen

Übung: Zählen

Zählen Sie in den drei vorgestellten Zahlensysteme von 1 bis 40!
Wie sieht 50, 70, 100 im Dual- bzw. im Hexadezimalsystem aus?

Dezimal	Dual	Hexadezimal
0	**0**	**0**
1	**1**	**1**
2	**10**	**2**
3	**11**	**3**
4	**100**	**4**
5	**101**	**5**
6	**110**	**6**
7	**111**	**7**
8	**1000**	**8**
9	**1001**	**9**
10	**1010**	**A**
11	**1011**	**B**
12	**1100**	**C**
13	**1101**	**D**
14	**1110**	**E**
15	**1111**	**F**
16	**10000**	**10**
17	**10001**	**11**
18	**10010**	**12**
19	**10011**	**13**
20	**10100**	**14**

Wertebereiche

Die Menge der darstellbaren Zahlen hängt von der Anzahl der zur Verfügung stehenden Stellen und vom verwendeten Zahlensystem ab.
Beispiel: 8-Bit Wortbreite, 16-Bit Wortbreite etc.
Durch den technischen Aufbau digitalelektronischer Bauteile entstehen häufig 16 Bit breite Worte (=zwei Byte), wie z.B. zur Adressberechnung.

Diese kleinsten Einheiten werden zusammengefaßt zu:

1 Nibble	=	4 Bit	
1 Byte	=	8 Bit	
1 Word	=	16 Bit	
1 Longword	=	32 Bit	
1 Kilobyte	=	1024 Byte	2^{10} = 1.024 Bytes
1 Megabyte	=	1024 Kilobyte	2^{20} = 1.048.576 Bytes
1 Gigabyte	=	1024 MegaByte	2^{30} = 1.073.741.824 Bytes
1 Terrabyte	=	1024 GigaByte	2^{40} = 1.099.511.627.776 Bytes

Übung: Wieviele verschiedene Zeichen können mit 8-Bit codiert werden?
Wieviele Speicherzellen lassen sich mit 16 Bit adressieren?

Konvertierung zwischen verschiedenen Zahlensystemen

Dual → Dezimal
Beispiel: 1111101=1*1+0*2+1*4+1*8+1*16+1*32+1*64

Dezimal → Dual
Um das binäre Äquivalent einer Dezimalzahl zu suchen, dividiert man die Dezimalzahl solange durch 2, bis das Endergebnis Null ist. Der jeweilige Rest, der ja nur Null oder Eins sein kann, wird notiert und ergibt dann -rückwärts gelesen- die gesuchte Dualzahl.

Beispiel:	125:2	=	62	Rest 1
	62:2	=	31	Rest 0
	31:2	=	15	Rest 1
	15:2	=	7	Rest 1
	7:2	=	3	Rest 1
	3:2	=	1	Rest 1
	1:2	=	0	Rest 1

Also ist die gesuchte Zahl 1111101 (von unten nach oben gelesen).

Übung: Konvertierung
Wandeln Sie folgende Zahlen um:

	=	1010101
	=	1101111101
65	=	
255	=	

Binärarithmetik

Im dualen Zahlensystem kann man genauso rechnen wie im dezimalen, vieles ist sogar erheblich einfacher.

Addition
Für die Addition im Dualsystem braucht man sich nur zu merken, daß 0+0=0 und 0+1=1+0=1 und 1+1=10.

1253	101101
+3586	+110110
= 4839	= 1100011

Aufgabe: Addieren Sie:
1010101 +10101011 = 100000000

Subtraktion
Auch die Subtraktion im Binärsystem verläuft analog zur bekannten Subtraktion.

4839	1100011
+3586	+ 110110
1253	101101

Aufgabe: Subtrahieren Sie:

```
  100000000
- 10101011
=   1010101
```

Multiplikation

```
312*576          10110*101
 1560              10110
  2184              00000
   1872              10110
 179712            1101110
```

Auch die Multiplikation kann durch mehrfaches Addieren ersetzt werden, so daß mit dem Aufbau eines Addierwerkes die Grundrechenarten ausgeführt werden können.

Zweierkomplement

Man kann die Subtraktion allerdings auch auf eine Addition zurückführen und kann dann nach erfolgreicher technischer Realisierung eines Addierwerkes auch subtrahieren. Die dazu verwendete Methode heißt Einer- bzw. Zweier-komplementbildung.

Beispiel: 101101 = 1100011-110110 =

```
  1100011
+ 001001
+       1
= (1)101101
```

Hierbei wird die zweite Zahl negiert. Diese Zahl heißt Einerkomplement. Dann wird dazu eine 1 addiert. Dieses Ergebnis nennt man Zweierkomplement. Diese Zweierkomplement der zweiten Zahl wird zur ersten Zahl addiert. Das Ergebnis ist dann die Differenz beider Zahlen, wobei die führende 1 gestrichen werden muß.

Beispiel:
Das gleiche Verfahren funktioniert auch im Dezimalsystem. Hier verwendet man das 9er- bzw. 10er-Komplement.

Beispiel: 93-17=76.

Neunerkomplement von 17 ist 82.
Zehnerkomplement von 17 ist 83.
93+83=176.
Streichen der führenden 1 ergibt 76!

Codierungen !!!

Man will nicht nur natürliche Zahlen durch Bitfolgen darstellen, sondern auch Gleitkommazahlen, Texte, Bilder, Sounds, Filme usw. Hier einige wichtige Kodierungen:

BCD = **B**inary **C**oded **D**ecimal

4 bit stellen jeweils eine Dezimalziffer dar, also wird z.B. 14_{10} dargestellt durch 0001 0110.

ASCII-Tabelle = **A**merican **S**tandard **C**ode of **I**nformation **I**nterchange

7bit- bzw. 8bit-Code zur Darstellung der Zeichen des Alphabets. Der 7bit-Code enthält nur die amerikanischen Zeichen, Sonderzeichen und Steuerzeichen für Datenübertragung. Das 8. Bit erweitert den Code um weitere 128 Character. Diese werden für spezielle Zeichen verschiedener Länder und kleine Grafiken verwendet. Aber vor der Verwendung muss immer der Ländercode angegeben und passende Programme geladen werden.

EBCDIC = Extended **B**inary-**C**oded **D**ezimal **I**nterchange **C**ode

Unicode

16bit-Code zur Darstellung der wichtigsten Zeichen der Welt (einschließlich kyrillisch, arabisch, chinesisch, japanisch, Sanskrit, Hindi, griechisch ...)

RTF (**R**ich **T**ext **F**ormat)

codiert über die Buchstaben hinaus auch einige Formatierungen

Zeichencodes in Computern

hex	dez.	ASCII	hex	dez.	ASCII	hex	dez.	ASCII	hex	dez.	ASCII
00	0	NUL	20	32	Space	40	64	@	60	96	'
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	S	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	'	47	71	G	67	103	g
08	8	BS	28	40	(	48	72	H	68	104	h
09	9	HT	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	:
1D	29	GS	3D	61	=	5D	93	]	7D	125	;
1E	30	RS	3E	62	>	5E	94	↑	7E	126	~
1F	31	US	3F	63	?	5F	95	_	7F	127	DEL

Bedeutung der wichtigsten Steuerzeichen

NUL = ohne Wirkung, SOH = start of header, STX = start of text, ETX = end of text, EOT = end of transmission, ENQ = enquiry (Aufforderung der Gegenstation zum Senden), ACK = acknowledge (Bestätigung, Rückmeldung), DLE = data link escape (Umschalten auf eine andere Steuerzeichengruppe), DC = Device Control (DC1 = XON, DC3 = XOFF), NAK = negative acknowledge, SYN = Synchronisationszeichen, ETB = end of transmission block, VT = vertical tabulating (Cursor nach oben), HT = horizontal tab. (Cursor nach rechts), BS = Back Space (Cursor nach links), LF = Line feed (Cursor nach unten), BEL = bell (akustisches Zeichen, Klingel), CR = Carriage Return, FS = field seperator, GS = group seperator, US = unit separator, Space = Leerraum, DEL = Delete/Rub out.

Deutsche Sonderzeichen

Zeichen	IBM-PC u. ä. hex	IBM-PC u. ä. dez.	ISO-Norm hex	ISO-Norm dez.
§	15	21	40	64
Ä	8E	142	5B	91
Ö	99	153	5C	92
Ü	9A	154	5D	93
ä	84	132	7B	123
ö	94	148	7C	124
ü	81	129	7D	125
ß	E1	225	7E	126

ASCII-Code (7 bit)
Darstellung der ersten 128 Zeichen

Erweiterte ASCII Tabelle

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
Ä	128	200	80	†	160	240	A0	¿	192	300	C0	‡	224	340	E0
Å	129	201	81	°	161	241	A1	¡	193	301	C1	·	225	341	E1
Ç	130	202	82	¢	162	242	A2	¬	194	302	C2	‚	226	342	E2
É	131	203	83	£	163	243	A3	√	195	303	C3	„	227	343	E3
Ñ	132	204	84	§	164	244	A4	ƒ	196	304	C4	‰	228	344	E4
Ö	133	205	85	•	165	245	A5	≈	197	305	C5	Â	229	345	E5
Ü	134	206	86	¶	166	246	A6	Δ	198	306	C6	Ê	230	346	E6
á	135	207	87	ß	167	247	A7	«	199	307	C7	Á	231	347	E7
à	136	210	88	®	168	250	A8	»	200	310	C8	Ë	232	350	E8
â	137	211	89	©	169	251	A9	…	201	311	C9	È	233	351	E9
ä	138	212	8A	™	170	252	AA		202	312	CA	Í	234	352	EA
ã	139	213	8B	´	171	253	AB	À	203	313	CB	Î	235	353	EB
å	140	214	8C	¨	172	254	AC	Ã	204	314	CC	Ï	236	354	EC
ç	141	215	8D	≠	173	255	AD	Õ	205	315	CD	Ì	237	355	ED
é	142	216	8E	Æ	174	256	AE	Œ	206	316	CE	Ó	238	356	EE
è	143	217	8F	Ø	175	257	AF	œ	207	317	CF	Ô	239	357	EF
ê	144	220	90	∞	176	260	B0	–	208	318	D0	Ò	240	360	F0
ë	145	221	91	±	177	261	B1	—	209	321	D1	Ú	241	361	F1
í	146	222	92	≤	178	262	B2	"	210	322	D2	Û	242	362	F2
ì	147	223	93	≥	179	263	B3	"	211	323	D3	Ù	243	363	F3
î	148	224	94	¥	180	264	B4	'	212	324	D4	ı	244	364	F4
ï	149	225	95	µ	181	265	B5	'	213	325	D5	ˆ	245	365	F5
ñ	150	226	96	∂	182	266	B6	÷	214	326	D6	˜	246	366	F6
ó	151	227	97	∑	183	267	B7	◊	215	327	D7	¯	247	367	F7
ò	152	230	98	∏	184	270	B8	ÿ	216	330	D8	˘	248	370	F8
ô	153	231	99	π	185	271	B9	Ÿ	217	331	D9	˙	249	371	F9
ö	154	232	9A	∫	186	272	BA	/	218	332	DA	˚	250	372	FA
õ	155	233	9B	ª	187	273	BB	¤	219	333	DB	¸	251	373	FB
ú	156	234	9C	º	188	274	BC	‹	220	334	DC		252	374	FC
ù	157	235	9D	Ω	189	275	BD	›	221	335	DD	˝	253	375	FD
û	158	236	9E	æ	190	276	BE	fi	222	336	DE	˛	254	376	FE
ü	159	237	9F	ø	191	277	BF	fl	223	337	DF	ˇ	255	377	FF

Erweiterter ASCII-Code (8 bit)
Darstellung der letzten 128 Zeichen

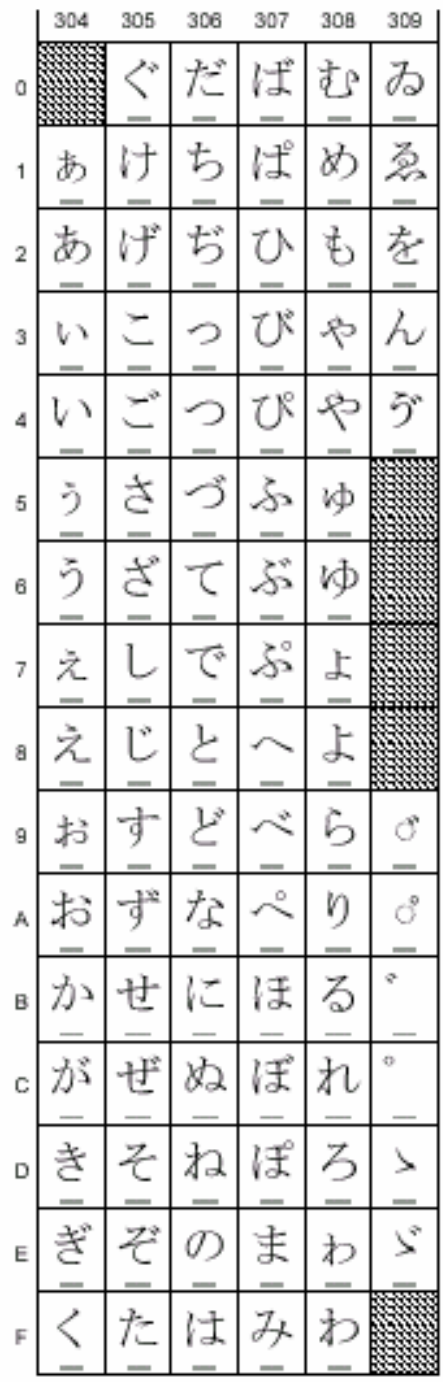
	00	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**
00	NU	SH	SX	EX	ST	HT	SA	DT	EG	RI	S2	VT	FF	CR	SO	SI
01	DL	D1	D2	D3	OC	NL	BS	ES	CN	EM	P2	S3	FS	GS	RS	US
02	PA	HO	BH	NH	IN	LF	EB	EC	HS	HJ	VS	PD	PU	EQ	AK	BL
03	DC	P1	SY	TC	CC	MW	SG	ET	SS	GC	SC	CI	D4	NK	PM	SB
04	SP										[	.	<	(	+	!
05	&										]	$	*	)	;	^
06	-	/									¦	,	%	_	>	?
07										`	:	#	@	'	=	"
08		a	b	c	d	e	f	g	h	i						
09		j	k	l	m	n	o	p	q	r						
10		~	s	t	u	v	w	x	y	z						
11																
12		A	B	C	D	E	F	G	H	I						
13		J	K	L	M	N	O	P	Q	R						
14	\		S	T	U	V	W	X	Y	Z						
15	0	1	2	3	4	5	6	7	8	9						AC

EBCDIC nach IBM International Character Set

Ausschnitt aus der Unicode-Tabelle:

	090	091	092	093	094	095	096	097
0		ऐ	ठ	र	ी	ॐ	ॠ	॰
1	ँ	ऑ	ड	ऱ	ु	॑	ॡ	
2	ं	ओ	ढ	ल	ू	॒	ॢ	
3	ः	ओ	ण	ळ	ृ	॓	ॣ	
4		औ	त	ऴ	ॄ	॔	।	
5	अ	क	थ	व	ॅ		॥	
6	आ	ख	द	श	ॆ		०	
7	इ	ग	ध	ष	े		१	
8	ई	घ	न	स	ै	क़	२	
9	उ	ङ	ऩ	ह	ॉ	ख़	३	
A	ऊ	च	प		ॊ	ग़	४	
B	ऋ	छ	फ		ो	ज़	५	
C	ऌ	ज	ब	़	ौ	ड़	६	
D	ऍ	झ	भ	ऽ	्	ढ़	७	
E	ऎ	ञ	म	ा		फ़	८	
F	ए	ट	य	ि		य़	९	

Devanagari (Sanskrit)



	304	305	306	307	308	309
0		ぐ	だ	ば	む	ゐ
1	ぁ	け	ぢ	ぱ	め	ゑ
2	あ	げ	ぢ	ひ	も	を
3	ぃ	こ	っ	び	ゃ	ん
4	い	ご	つ	ぴ	や	ゔ
5	ぅ	さ	づ	ふ	ゅ	
6	う	ざ	て	ぶ	ゆ	
7	ぇ	し	で	ぷ	ょ	
8	え	じ	と	へ	よ	
9	ぉ	す	ど	べ	ら	゙
A	お	ず	な	ぺ	り	゚
B	か	せ	に	ほ	る	゛
C	が	ぜ	ぬ	ぼ	れ	゜
D	き	そ	ね	ぽ	ろ	ゝ
E	ぎ	ぞ	の	ま	ゎ	ゞ
F	く	た	は	み	わ	

Hiragana (Japanisch)

Anhang: Was können Computer leisten?

Berechenbarkeit

Angesichts der fortschreitenden Möglichkeiten der Computer fragt es sich, ob der Computer nicht praktisch alle Fragen und Probleme lösen kann, zumindest diejenigen, die man in klarer und eindeutiger Weise formulieren kann.

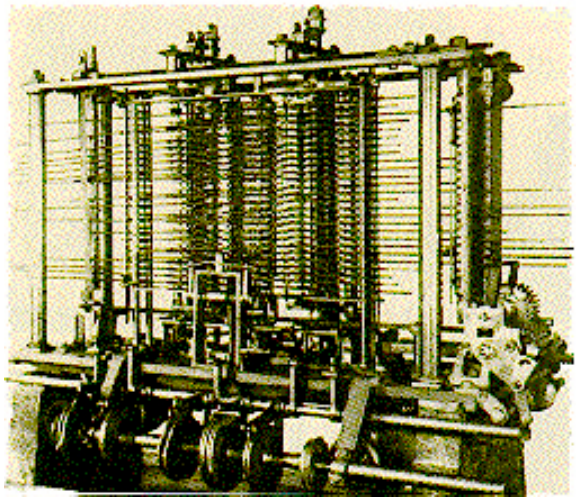
So hat ein Computersystem (Deep Blue von IBM) bei einer Meisterschaft den Weltmeister Kasparow geschlagen, und viele Schachexperten hielten manche der Spielzüge der Maschine für überraschend gut, teilweise genial.

Schon Leibniz hat vorgeschlagen, Maschinen zu bauen, die rechnerisch denken können und alle wichtigen Fragen, auch die der Philosophie und der Theologie, endgültig lösen sollten.

Leibniz

Charles Babbage entwickelte den Plan einer ersten Programmgesteuerten Rechenmaschine, die Analytical Engine, welche aber mit den damaligen Mitteln nicht realisiert werden konnte.

Charles Babbage



Analytical Engine

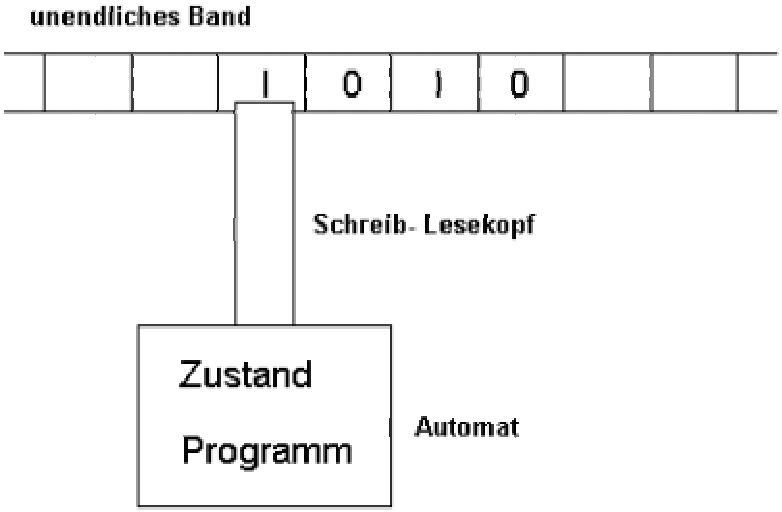
1936 stellte sich der Endländer Alan C. Turing die Frage, ob man jedes mathematische Problem mit einer festgelegten und sicheren Methode entscheiden kann.

Alan Matthew Turing

Er überlegte sich, daß man eine feste Methode am besten mit einer Maschine realisieren könnte. Die Maschine sollte lesen und rechnen und ihre Ergebnisse wieder aufschreiben können. Um die Maschine möglichst einfach zu halten, sollte das Speichermedium ein unendlich langes Band mit Zellen sein. Jede Zelle darf leer sein oder ein Zeichen eines festgelegten Alphabets enthalten.

Man beschreibt das Band zu Anfang mit einer oder mehreren Zeichenketten. Dann startet die Maschine. Sie arbeitet in Schritten. In jedem Schritt befindet sie sich in einem von endlich vielen Zuständen und liest das Zeichen, das sich unter dem Schreiblesekopf befindet. In Abhängigkeit von den beiden Größen führt sie eine von drei möglichen Operationen aus. Sie kann ein Zeichen auf das Band schreiben oder die Zelle löschen, sie kann den Schreiblesekopf nach rechts oder nach links bewegen, und sie kann anhalten. Danach geht sie in einen neuen Zustand über oder behält den alten bei.

Turingmaschine



Was können Turingmaschinen?

Antwort: Alles was andere Computer auch können, sogar das, was Computer in der Zukunft zu leisten imstande sind. (Churchsche These)

Was können Turingmaschinen nicht?

Es gibt Probleme, die sich nicht lösen lassen.

Beispiel: Halteproblem.

Problem: Es soll festgestellt werden, ob ein beliebiges Programm, das eine Eingabe erwartet, hält oder nicht.

Annahme: Wir können ein Programm H erstellen, das überprüft, ob ein beliebiges Programm P hält.

Programm H

Eingabe von P
Test, ob P hält
Wenn ja, Ausgabe „JA“, sonst „NEIN“

Dann kann man aber auch folgendes Programm schreiben:

Programm H2

Eingabe von P
Test, ob P hält
Wenn ja, Endlosschleife, sonst Programmende

Dann kann man aber auch das Programm auf sich selbst anwenden:

Programm H2

Eingabe von H2
Test, ob H2 hält
Wenn ja, Endlosschleife, sonst Programmende

Das ist ein Widerspruch.
Wenn H2 hält, geht H2 in eine Endlosschleife über und hält also nicht. Wenn aber H2 nicht hält, so wird das beim Test entdeckt, und H2 wird beendet, hält also.

Wie kommt es zu diesem Widerspruch? Wir haben eine falsche Annahme in die Überlegung gesteckt, nämlich, dass wir über einen Test verfügen, der haltende von nicht haltenden Programmen unterscheidet. Solch ein Test kann also nicht in einem Programm realisiert werden.

Gibt es weitere nicht berechenbare Probleme?

Ja, viele.

Beispiel: Machen zwei verschiedene Programme dasselbe (Äquivalenzproblem)?

Dies läßt sich nicht testen. Wenn es solch einen Test gäbe, könnten wir ein Programm P mit einem Programm vergleichen, das garantiert nicht anhält. So könnten wir ein Testprogramm für haltende Programme schreiben.

Gleichheit von Programmen

Eingabe von P1, P2 (nicht haltend)
Test, ob P1 und P2 das gleiche tun
Wenn ja, P1 hält nicht, sonst P1 hält

Beispiel: Tut ein Programm das richtige (Korrektheitsproblem)?

Das läßt sich ebenfalls nicht testen.

Komplexität von Problemen

Manche Probleme lassen sich lösen, aber mit einem hohen Aufwand an Rechenzeit oder Speicherplatz.

Der Aufwand oder die Komplexität wird dabei mit der Größe der Eingabe oder des Problems verglichen.

Konstanter Aufwand:	1
Linearer Aufwand:	n
Quadratischer Aufwand:	n^2
Exponentieller Aufwand:	2^n

	1	n	n*log(n)	n^2	2^n	n!
1	1	1	0	1	2	1
10	1	10	33,21928095	100	1024	3628800
100	1	100	664,385619	10000	1,26765E+30	9,333E+157
1000	1	1000	9965,784285	1000000	1,0715E+301	#ZAHL!
1000000	1	1000000	19931568,57	1E+12	#ZAHL!	#ZAHL!
1000000000	1	1000000000	29897352854	1E+18	#ZAHL!	#ZAHL!

Beispiel:

Es sollen die Daten aller Bundesbürger (ca. 80.000.000) mit dem Computer sortiert werden. Ein naheliegender, einfacher Sortieralgorithmus (z.B. Bubblesort) arbeitet mit dem Aufwand n*n/2, ein ausgefeilter Algorithmus (z.B. Quicksort) dagegen mit dem Aufwand n*log(n).

Bei 100 Millionen Vergleichen pro Sekunde ergiebt sich für die Zeit zum Sortieren:

Bubblesort:

Zeit = 80.000.000*80.000.000 / 2 / 100.000.000 sek
= 80.000.000*40.000.000 / 100.000.000 sek
= 80.000.000*40 / 100 sek
= 3.200.000.000 / 100 sek
= 32.000.000 sek
= 32.000.000 / 3.600 Std
≈ 9000 Std ≈ 1 Jahr

Quicksort:

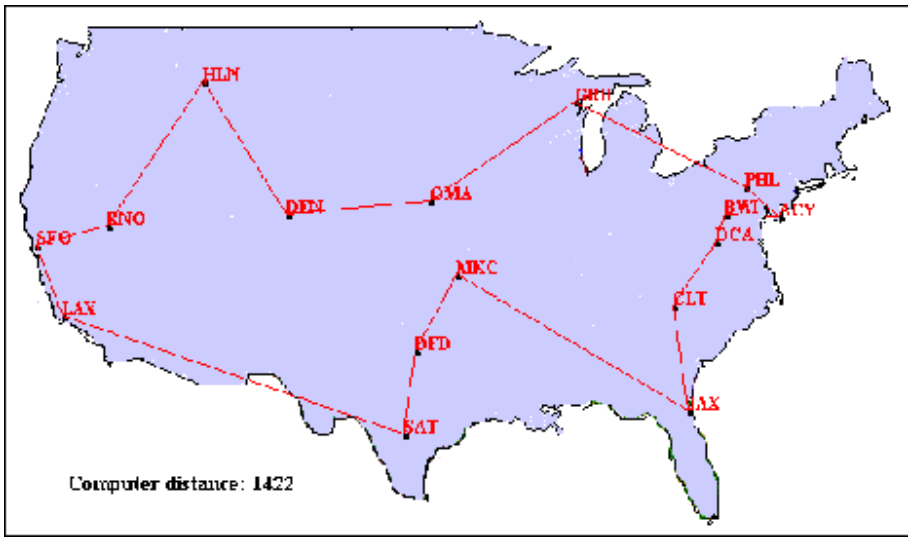
Zeit = 80.000.000*log(80.000.000) / 100.000.000 sek
≈ 80.000.000* 26,253496664211536435092236006426 / 100.000.000 sek
≈ 2100279733,1369229148073788805136 / 100.000.000 sek
≈ 21 sek

Der erste Algorithmus benötigt also auf dem (recht schnellen) Computer 1 Jahr, der zweite auf demselben Rechner nur 21 Sekunden.

Die Komplexitätstheorie beschäftigt sich mit der Frage, ob es zu Problemen schnelle Algorithmen gibt. Für viele Probleme gibt es keine schnellen Programme, d.h. jedes denkbare Programm zur Lösung des Problems ist zu langsam für praktische Zwecke. Für viele Problemstellungen ist es allerdings bis heute unbekannt, ob es schnelle Algorithmen gibt. Dazu gehören viele praktisch interessante Probleme, z.B. aus der Betriebswirtschaft, dort vor allem im Operations Research.

Wenn die Berechnung einer exakten oder optimalen Lösung zu aufwendig ist, weicht man oft auf heuristische Lösungsverfahren aus. Mit diesen Verfahren wird nur eine Näherung an die exakte Lösung oder eine Antwort berechnet, die dem Optimum möglichst nahe kommt.

Beispiel: Planung von günstigen Reiserouten.



http://www.cacr.caltech.edu/~manu/tsp.html