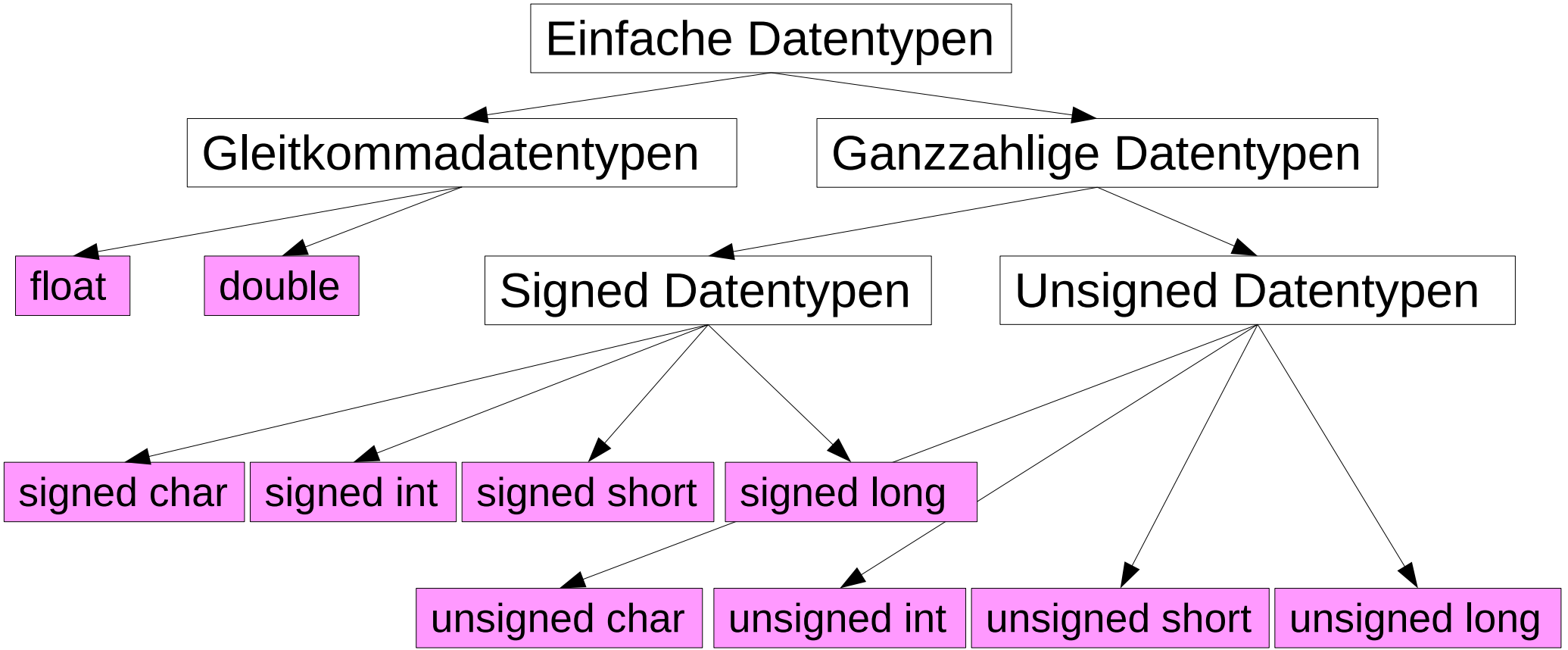
Einfache Datentypen

- Alle Daten (Zahlen und andere Daten) sind in Programmen von einem bestimmten Datentyp.
- Datentypen bestimmen
  - Verarbeitungsbreite bzw. Genauigkeit (typisch sind 1, 2, 4 oder 8 Byte
  - Interpretation des Bitmusters (Ganzzahlig, nur positiv oder Gleitpunkt)
  - Die damit ausführbaren Operationen und wie diese ausgeführt werden.

Einfache Datentypen



Einfache Datentypen

Datentyp	Typischer Wertebereich	Typ. Breite	Kurzformen
signed char	0 .. 255	1	char
unsigned char	-128 .. 127	1	char
signed short int	-32.767 - 32.767	2	short, short int, signed short
unsigned short int	0.. 65535	2	unsigned short
signed int	-2.147.483.647 .. 2.147.483.647	4	int, signed
unsigned int	0 .. 4 294 967 295	4	unsigned
signed long int	−9223372036854775808 .. +9223372036854775807	8	long, long int
unsigned long int	0 .. 18446744073709551615	8	unsigned long
float	+/- 1.17E-38 .. 3.4E38	4	
double	+/- 2.2E-308 .. 1.8E308	8	

Anmerkungen zu int-Datentypen

- Die Sprachdefinition von c legt die Verarbeitungsbreite (2,4,8 Bytes) für die Integer Datentypen nicht fest. Sie sind plattformabhängig (Prozessor/Betriebssystem)
- Festgelegt ist lediglich, dass der Wertebereich von short kleiner oder gleich dem von int und der von int kleiner oder gleich dem von long sein muss.

Anmerkungen zu int-Datentypen

- Ein unsigned long-wert muss groß genug sein, um eine Adresse des Systems aufzunehmen (32-Bit-System: 4 Byte, 64-Bit-System 8 Byte)
- Die Angaben in der Tabelle beziehen sich auf 64-bit-PC-Systeme.
- Die plattformabhängigen Angaben zu größten und kleinsten Werten sind in einer Datei limits.h, die für jeden c-Compiler existiert, abgelegt.

Programmbeispiel sizeof

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  long i=99;
  printf("Sizeof long int: %ld\n",sizeof i);
  return 0;
}
```

Datentypen aus stdint.h

- Im Systemheaderfile stdint.h sind Datentypen mit fester Verarbeitungsbreite definiert.
- Sie können, genauso, wie die bislang behandelten primitiven Datentypen genutzt werden.
- Erforderlich ist
  `#include <stdint.h>`
  am Programmanfang

unsigned	signed
uint8_t	**int8_t**
uint16_t	**int16_t**
uint32_t	**int32_t**
uint64_t	**int64_t**

Anmerkungen zu char-Datentypen

- Der Datentyp char dient zum einen der Speicherung von Zeichen nach dem ascii-Code.
- Zum zweiten findet der Datentyp char auch Verwendung als int-Datentyp mit einer Verarbeitungsbreite von einem Byte.
- Ob char dabei als signed oder unsigned interpretiert wird, ist compilerabhängig und nicht festgelegt.

Die ASCII-Code-Tabelle

- Die Zeichen mit den Codes 0..31 sind spezielle Steuerzeichen, wie Zeilenende, Tabulator usw.
- Einzelne Zeichen werden in c-Programmen in Apostroph gesetzt. (‘a‘, ‘*‘ oder ‘3‘)

```
char x='*';  // x enthält den
             // Wert 0x2A oder 42
char c='a';  // c enthält den
             //Wert 0x41 oder 65
```

Das Apostroph liegt bei deutscher Tastatur über #, neben der Entertaste

dec	hex	Char	dec	hex	Char	dec	hex	Char	dec	hex	Char
0	0	.	32	20	.	64	40	@	96	60	`
1	1	.	33	21	!	65	41	A	97	61	a
2	2	.	34	22	"	66	42	B	98	62	b
3	3	.	35	23	#	67	43	C	99	63	c
4	4	.	36	24	$	68	44	D	100	64	d
5	5	.	37	25	%	69	45	E	101	65	e
6	6	.	38	26	&	70	46	F	102	66	f
7	7	.	39	27	'	71	47	G	103	67	g
8	8	.	40	28	(	72	48	H	104	68	h
9	9	.	41	29	)	73	49	I	105	69	i
10	A	.	42	2A	*	74	4A	J	106	6A	j
11	B	.	43	2B	+	75	4B	K	107	6B	k
12	C	.	44	2C	,	76	4C	L	108	6C	l
13	D	.	45	2D	-	77	4D	M	109	6D	m
14	E	.	46	2E	.	78	4E	N	110	6E	n
15	F	.	47	2F	/	79	4F	O	111	6F	o
16	10	.	48	30	0	80	50	P	112	70	p
17	11	.	49	31	1	81	51	Q	113	71	q
18	12	.	50	32	2	82	52	R	114	72	r
19	13	.	51	33	3	83	53	S	115	73	s
20	14	.	52	34	4	84	54	T	116	74	t
21	15	.	53	35	5	85	55	U	117	75	u
22	16	.	54	36	6	86	56	V	118	76	v
23	17	.	55	37	7	87	57	W	119	77	w
24	18	.	56	38	8	88	58	X	120	78	x
25	19	.	57	39	9	89	59	Y	121	79	y
26	1A	.	58	3A	:	90	5A	Z	122	7A	z
27	1B	.	59	3B	;	91	5B	[	123	7B	{
28	1C	.	60	3C	<	92	5C	\	124	7C	|
29	1D	.	61	3D	=	93	5D	]	125	7D	}
30	1E	.	62	3E	>	94	5E	^	126	7E	~
31	1F	.	63	3F	?	95	5F	_	127	7F	.

Das Wort C-Programmierung besteht aus den ASCII-Codes (Hexadezimal):

43	2D	50	72	6F	67	72	61	6D	6D	69	65	6F	75	6E	67	00
C	-	P	r	o	g	r	a	m	m	i	e	r	u	n	g	

Das Byte 00 ganz am Ende nennt man terminierende 0. Sie markiert das Ende einer Zeichenkette.
In c-Programmen werden Zeichenketten in Arrays vom Typ char gespeichert und in Gänsefüßchen geklammert.

```
char text[]="C-Programmierung";
```

Dabei wird die Größe dieses Arrays automatisch berechnet.

Merke:
ein einzelnes Zeichen: einfache Hochkommas,
mehrere Zeichen: doppelte Hochkommas

Grundelemente von C

- **Großbuchstaben : A...Z**
- **Kleinbuchstaben : a...z**
- **Ziffern : 0...9**
- **Sonderzeichen : ( ) [ ] { } < > + - * / % ^ ~ & |**
  _ = ! ? # \ , . ; : ' "
- **Ersatzsymbolfolgen**

#	=	??=
[	=	??(
]	=	??)
^	=	??'
{	=	??<
}	=	??>
|	=	??!
~	=	??-

- **white spaces :**

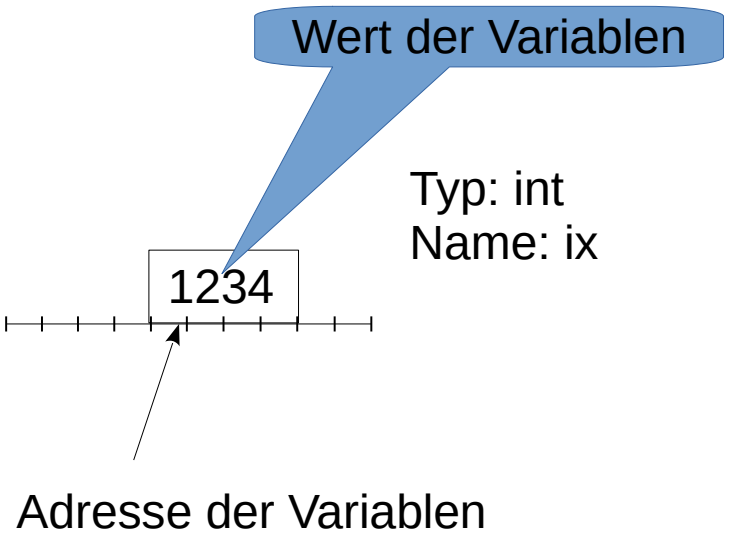
Leerzeichen, Zeilenende(\n), Tabulator(\t), vertikaler Tab(\v), Seitenvorschub(\f), Bell(\a), Backspace(\b), Carriage return(\r)

Variablen

- Variablen sind Speicherplätze, an denen Werte abgelegt werden.
- Den Speicher hatten wir bereits als ein langes Band von durchnummerierten Bytes betrachtet. Die laufende Nummer eines jeden Bytes ist seine Adresse.
- Eine Variable wird in c definiert, in dem man Typ und Namen der Variable in folgender Form angibt:

```
int ix;
```

Variablen



- Variablen sind gekennzeichnet durch:
  - Wert (Bitmuster, Byteorder)
  - Typ (Anzahl der Bytes, Interpretation des Bitmusters)
  - Adresse des ersten Bytes im Speicher
  - Name

Variablenvereinbarung

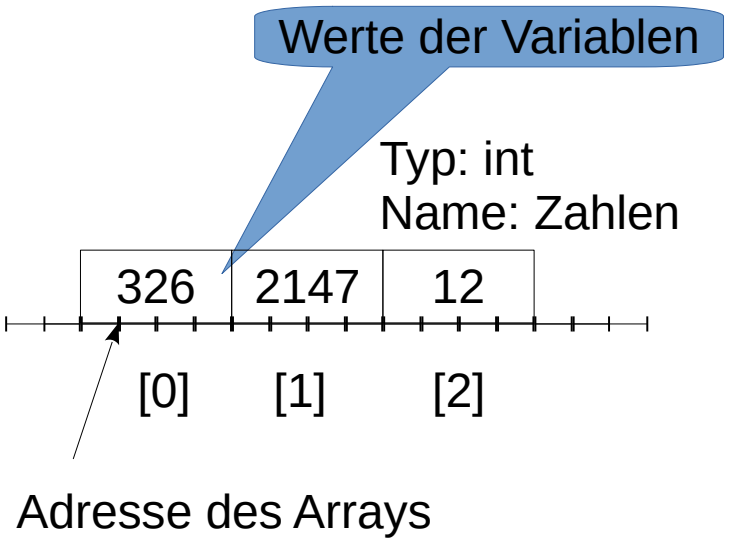
```
int ix;
int j,k=1;
unsigned long = 0L;
double x,y;
double factor= 1.0;
char ziffer = '1';
c3har Operator= '+';
```

- Variablenvereinbarungen werden mit einem Semikolon abgeschlossen.
- Mehrere Variablen des selben Typs können durch Komma getrennt definiert werden.
- Variablen können initialisiert werden (bekommen einen Anfangswert)

Variablenvereinbarung

- Variablen ohne Angabe einer Initialisierung haben den Wert 0 oder einen undefinieren Wert.
  - Variablen innerhalb von geschweiften Klammern (lokale Variable) die nicht explizit initialisiert wurden, haben einen undefinierten (zufälligen) Wert. Er ergibt sich aus dem Typ der Variablen und dem Bitmuster an der Adresse der Variablen, das da gerade steht.
  - Variablen außerhalb aller geschweiften Klammern werden mit 0 initialisiert, man bezeichnet sie als externe, statische oder globale Variablen.

Arrays / Vektoren



- Arrays fassen mehrere Variablen eines Typs, die hintereinander angeordnet sind, unter einem Namen zusammen.
- Sie werden mit 0 beginnend durchnummeriert.
- Durch Angabe eines Index wird eine einzelne Variable selektiert (Zahlen[1]).

Arrayvereinbarung

```
int Werte[10];
int Zahlen[]={9,4,5,6};
char name1[]="Hans";
char name2[5]="Otto";
```

- Arrays werden vereinbart, indem man hinter dem Namen eckige Klammern angibt.

- In den Klammern wird die Anzahl der Arrayelemente angegeben.
- Wird das Array initialisiert, so kann die Angabe der Anzahl der Werte entfallen, die Anzahl wird vom Compiler berechnet.

Arrayvereinbarung

- Ist die Anzahl angegeben und es gibt weniger Werte in der Initialsierungsliste, so wird zum Ende mit Nullen aufgefüllt.
- Ist die Anzahl angegeben und es gibt mehr Werte in der Initialsierungsliste, so wird ein Fehler beim Compilieren ausgegeben.
- char-Arrays können mit Zeichenketten initialisert werden.
- Gibt man bei char-Arrays die Länge an, muss immer mindestens ein Byte mehr reserviert werden, als die Zeichenkette Zeichen enthält. Es wird für die terminierende 0 benötigt.

Namen/Bezeichner

- Bezeichner werden in Programmen zur Benennung von Programmkonstrukten, unter anderem von Variablen, verwendet.
- Bezeichner müssen mit einem Buchstaben oder dem Unterstrich _ beginnen.
- Bezeichner können aus Buchstaben, Ziffern und dem Unterstrich bestehen.
- Groß- und Kleinbuchstaben werden unterschieden, c ist casesensitiv.

Namen/Bezeichner

- Bezeichner sind frei wählbar bis auf die reservierten Schlüsselwörter.
- Bezeichner sollen gut dokumentieren. Sie sollen sinnvoll gewählt werden, damit das Programm gut les- und verstehbar ist.
- Zu lange Bezeichner sollten aber dennoch vermieden werden.

Reservierte Schlüsselwörter

auto	default	float	long	sizeof	union
break	do	for	register	static	unsigned
case	double	goto	return	struct	void
char	else	if	short	switch	volatile
const	enum	int	signed	typedef	while
continue	extern				

Ausgabe von Text und Zahlen

- Die schönste Berechnung nützt nichts, wenn man das Ergebnis oder die Ergebnisse nicht visualisieren kann.
- In c verwendet man zur Ausgabe Funktionen der Standardbibliothek, unter anderem die Funktion printf.
- Um printf benutzen zu können, muss ein c-Programm am Anfang nachfolgende Zeile enthalten:

```
#include <stdio.h>
```

Ausgabe von Text und Zahlen

- Weiterhin benötigt jedes c-Programm als Programmstartpunkt eine main-Funktion
- Die Zeile return 0; nehmen wir erst mal hin.
- Somit haben wir das Grundgerüst:

```
#include <stdio.h>
int main()
{
  //hier koennen wir nun etwas programmieren
  return 0
}
```

Ausgabe von Text

- Zur Ausgabe von Text kann man printf benutzen. Die Ausgabe sollte einen Zeilenwechsel (‘\n‘)am Ende enthalten. ‘\n‘ wird vom Compiler als ein Zeichen (1 Byte) interpretiert.
- Schreiben Sie das unten stehende Programm ab und testen Sie es mit und ohne ‘\n‘

```c
#include <stdio.h>
int main()
{
  printf("Willkommen zu Programmierung I\n");
  return 0;
}
```

Ausgabe von Zahlen

- Um Werte aus Variablen auszugeben, fügt man in die Zeichenkette, die nun auch Formatsteuerkette genannt wird, Platzhalter ein.
- Ein Platzhalter beginnt mit einem % gefolgt von einem oder mehreren Zeichen zur Ausgabeformatierung
- Für jeden Platzhalter muss hinter der Formatsteuerkette durch Komma getrennt ein Wert angegeben werden.
- Der Wert und das Formatsteuerzeichn müssen „zueinander passen“.

Ausgabe von Text und Zahlen

```c
#include <stdio.h>
int main()
{
  int  i=95;
  printf("i: %d (dezimal) %04x (hexadezimal)\n",i,i);
  return 0;
}
```

- Das Programm gibt den Wert der Variablen i 2 mal aus.
- Mit %d wird der Zahlenwert dezimal in seiner Länge ausgegeben.
- Mit %04x wird der Zahlenwert hexadezimal (%x) ausgegeben, die 4 steht für vierstellig und die 0 steht für die Ausgabe führender Nullen.
- Schreiben Sie das Programm ab und testen Sie mit %d, %i, %x, %X, %6d, %06d, %4x, %04x, %08x, ...
- In einem 2. Test geben Sie statt 95 eine negative Zahl an.
- Testen Sie mit verschiedenen positiven und negativen Zahlen und betrachten Sie die Ergebnisse

Formatsteuerzeichen

Datentyp	Formatsteuerzeichen
signed char	%c (Zeichen), %d oder %i (dezimall) oder %x hexadezimal
unsigned char	%c (Zeichen), %u (dezimal) oder %x hexadezimal
signed short int	%d oder %i oder %x
unsigned short int	%u oder %x
signed int	%d oder %i oder %x
unsigned int	%u oder %x hexadezimal
signed long int	%ld oder %lx hexadezimal
unsigned long int	%lu oder %lx hexadezimal
float	%f
double	%lf

Ausführlich: **`man 3 printf`**

Beispiel

```c
#include <stdio.h>
int main()
{
  double array[5];
  int i;
  for(i=0; i<5; i++)
      array[i]=1.0/(i+1);

  for(i=0; i<5; i++)
      printf("array[%d]: %6.4lf\n",i,array[i]);
  return 0;
}
```

Array mit 5 Elementen, die zunächst undefinierte Werte haben.

```
beck@pc:~/serv/TESTC/Hello$ gcc h4.c
beck@pc:~/serv/TESTC/Hello$ ./a.out
array[0]: 1.0000
array[1]: 0.5000
array[2]: 0.3333
array[3]: 0.2500
array[4]: 0.2000
```

Zeile	Erläuterung
4	Ein Array vom Typ double mit 5 Elementen wird angelegt, die Werte des Arrays bleiben undefiniert (zufällige Werte, die gerade dort im Speicher stehen)
5	Eine Variable i wird angelegt. Die Bezeichner i, j, und k werden gern für Indexvariablen verwendet.
6	Es handelt sich um eine ‚for-schleife‘. i bekommt den Wert 0. Danach wird geprüft, ob i<5 wahr ist und ist das der Fall, wird die nachfolgende Zeile ausgeführt und dann mit i++ die Variable i um 1 erhöht. Ist die Bedingung falsch, wird in Zeile 9 fortgefahren.
7	Dem Array array wird an der Stelle i der Wert 1.0/(i+1) zugewiesen. Die Formulierung 1.0 ist wichtig, damit die Division mit Gleitkommaarithmetik erfolgt. Der Ausdruck 1/(i+1) liefert, außer bei i=0, immer 0! (Ganzzahlige Division)
9	Die Schleife wird ein 2. Mal zur Ausgabe ausgeführt
10	Ausgabe. Die Formatierung %6.4lf besagt, dass ein Double-wert 6-stellig mit 4 Kommastellen ausgegeben werden soll.

Eingabe von Zahlen

- Für die Eingabe numerischer Werte gibt es verschiedene Möglichkeiten.
- Sehr häufig werden Eingaben in c mit der Funktion scanf bewerkstelligt. Diese Funktion führt aber häufig zu Programmfehlern.
- Wir verwenden eine andere Methode zur Eingabe von Zahlen

Eingabe von Zahlen

- Als erstes definieren wir uns einen genügend großen Speicherbereich von chars, vielleicht 128.

```
char buf[128];
```

- Mit fgets lesen wir die Zeichen der einzugebenden Zahl nach buf ein.

```
fgets(buf,128,stdin);
```

Dort kommen die eingebenen Zeichen rein

Es werden max. 128-1 Zeichen übernommen

Eingabe von der Tastatur

Eingabe von Zahlen

- In buf stehen nun die Zeichen der eingegebenen Zahl als Zeichenkette.
- Um daraus eine Zahl zu bekommen, muss sie ‚konvertiert' - in eine Zahl umgewandelt – werden.
- Soll es eine ganze Zahl werden, verwenden wir atoi (ascii to integer) oder atol für long.
- Soll es eine Gleitpunktzahl werden, verwenden wir atof.

```
beck@PC:./a.out
Eingabe Dividend:10
Eingabe Divisor :3
Quotient(int)    10/3: 3
Qutient(double) 10.000000/3: 3.333333
beck@PC:
```

```c
#include <stdio.h>
#include <stdlib.h>

char buf[128];

int main()
{
  int Dividend, Divisor;
  double DividendDouble;
  printf("Eingabe Dividend:");
  fgets(buf,128,stdin);     // Eingabe Text
  Dividend=atoi(buf);       // Konvertierung nach int
  DividendDouble=atof(buf); // Konvertierung nach double
  printf("Eingabe Divisor: ");
  fgets(buf,128,stdin);     // Eingabe Text
  Divisor=atoi(buf);        // Konvertierung nach int
  printf("Quotient(int)   %d/%d: %d\nQutient(double) %lf/%d: %lf\n",
      Dividend, Divisor, Dividend/Divisor,                // ganzzahlige Rechnung
      DividendDouble,Divisor, DividendDouble/Divisor);  // gebrochenzahlige Rech.
  return 0;
}
```

Erläuterung dazu

Zeile 4	Definition eine Eingabepuffers
Zeile 11	Eingabe der Zahl als Zeichenkette. Im Puffer stehen die Ascii-Zeichen der einzelnen Ziffern. Ggf. des Dezimalpunktes und am Ende eine terminierende 0.
Zeile 12	Der Text in buf wird in eine int-Zahl umgewandelt. Die int-Zahl wird in der Variablen Dividend gespeichert.
Zeile 13	Der Text in buf wird in eine double-Zahl umgewandelt. Die double-Zahl wird in der Variablen DividendDouble gespeichert
Zeile 17	Ausgabe mit printf, die Formatsteuerkette
	Ausgabe mit printf, Ausgabe der Ganzahldivision
	Ausgabe mit printf, Ausgabe der gebochenzahligen Division.

Aufgabe:

- Schreiben Sie die Beispiele ab,
- speichern Sie sie
- Compilieren Sie sie
- Probieren Sie die Programme aus.