

D Datentypen, einfache Programmstruktur, Operatoren, Ausdrücke

- Datentypen →
 - ◆ Ganze Zahlen
 - ◆ Fließkommazahlen
 - ◆ Zeichen
 - ◆ Zeichenketten
- Aufbau eines C-Programms — erste Hinweise
 - ◆ Hauptprogramm
 - ◆ Bedingte Anweisung
- Ausdrücke
 - ◆ Operatoren
 - ◆ Typumwandlung in Ausdrücken
 - ◆ Vorrangregeln bei Operatoren

D.2 Standardtypen in C

- Eine Reihe häufig benötigter Datentypen ist in C vordefiniert
 - char** Zeichen (im ASCII-Code dargestellt, 8 Bit)
 - int** ganze Zahl (16 oder 32 Bit)
 - float** Gleitkommazahl (32 Bit)
etwa auf 6 Stellen genau
 - double** doppelt genaue Gleitkommazahl (64 Bit)
etwa auf 12 Stellen genau
 - void** ohne Wert

D.1 Datentypen

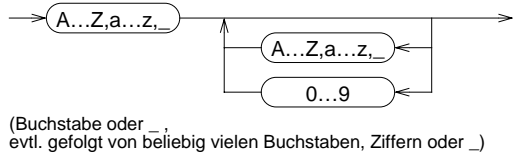
- Menge von Werten + Menge von Operationen auf den Werten
 - ◆ **Konstanten** stellen einen Wert dar
 - ◆ **Variablen** sind Namen für Speicherplätze, die einen Wert aufnehmen können
 - Konstanten und Variablen besitzen einen **Typ**
- Datentypen legen fest:
 - ◆ Repräsentation der Werte im Rechner
 - ◆ Größe des Speicherplatzes für Variablen
 - ◆ erlaubte Operationen
- Festlegung des Datentyps
 - ◆ implizit durch Verwendung und Schreibweise (Zahlen, Zeichen)
 - ◆ explizit durch **Deklaration** (Variablen)

D.2 Standardtypen in C (2)

- Die Bedeutung der Basistypen kann durch vorangestellte **Typ-Modifier** verändert werden
 - short, long** legt für den Datentyp **int** die Darstellungsbreite (i. a. 16 oder 32 Bit) fest.
Das Schlüsselwort **int** kann auch weggelassen werden
 - long double** **double**-Wert mit erweiterter Genauigkeit (je nach Implementierung) –
mindestens so genau wie **double**
 - signed, unsigned** legt für die Datentypen **char**, **short**, **long** und **int** fest, ob das erste Bit als Vorzeichenbit interpretiert wird oder nicht

D.3 Variablen

- Variablen besitzen
 - ◆ **Namen** (Bezeichner)
 - ◆ **Typ**
 - ◆ zugeordneten Speicherbereich für einen Wert des Typs
Inhalt des Speichers (= **aktueller Wert** der Variablen) ist veränderbar!
 - ◆ **Lebensdauer**
- Bezeichner



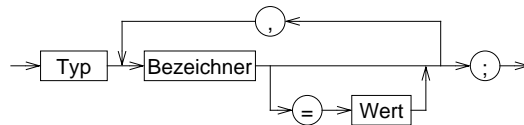
D.3 Variablen (3)

- Variablen-Definition: Beispiele

```
int a1;
float a, b, c, dis;
int anzahl_zeilen=5;
char Trennzeichen;
```
- ◆ Position im Programm:
 - nach jeder "{"
 - außerhalb von Funktionen
- Wert kann bei der Definition initialisiert werden
- Wert ist durch Wertzuweisung und spezielle Operatoren veränderbar
- Lebensdauer ergibt sich aus der Programmstruktur

D.3 Variablen (2)

- Typ und Bezeichner werden durch eine **Variablen-Deklaration** festgelegt
 - ◆ reine Deklarationen werden erst in einem späteren Kapitel benötigt
 - ◆ vorerst beschränken wir uns auf Deklarationen in **Variablen-Definitionen**
- eine **Variablen-Definition** deklariert eine Variable und reserviert den benötigten Speicherbereich



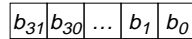
D.4 Ganze Zahlen

- Definition
-
- Speicherbedarf(short int) ≤ Speicherbedarf(int) ≤ Speicherbedarf(long int)
 - Speicherbedarf(int): heute meist 32 Bit
 - Konstanten (Beispiele):

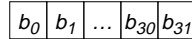
```
42, -117
035      (oktal = 2910)
0x10     (hexadezimal = 1610)
0x1d     (hexadezimal = 2910)
```

D.4 Ganze Zahlen (2)

- Repräsentation (meistens): (Motorola, HP, SPARC, ...)



bei Intel-Prozessoren:



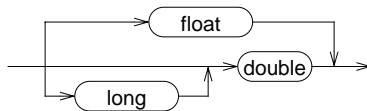
- Wertebereich:
 - unsigned:** $0 \dots 2^{32} - 1$
 - signed:** $-2^{31} \dots 2^{31} - 1$
 - b_{31} ist Vorzeichenbit

D.5 Fließkommazahlen (2)

- Repräsentation: durch Mantisse m und Exponent e
 - Wert = $m \cdot 2^e$
- Genauigkeit
 - hängt von der Anzahl der Bits für die Mantisse ab
- Wertebereich
 - hängt von der Anzahl der Bits für den Exponent ab

D.5 Fließkommazahlen

- Definition



- Speicherbedarf(float) ≤ Speicherbedarf(double) ≤ Speicherbedarf(long double)
- Speicherbedarf(float): 32 Bit
- Konstanten (Beispiele):
 - normale Dezimalpunkt-Schreibweise
 $3.14, -2.718, 368.345, 0.003$
 aber nicht einfach 1 (wäre eine int-Konstante!)
 - 10er-Potenz Schreibweise ($368.345 = 3.68345 \cdot 10^2, 0.003 = 3.0 \cdot 10^{-3}$)
 $3.68345e2, 3.0e-3$

D.5 Fließkommazahlen (3)

- Repräsentation abhängig vom jeweiligen Prozessor (häufig verwendetes Format: IEEE-Standard)

	Mant. (Bit)	Exp. (Bit)	kleinste darstellbare positive Zahl größte darstellbare Zahl prozentuale Genauigkeit
float	24	7	$1.175494351 \cdot 10^{-38}$ $3.402823466 \cdot 10^{+38}$ $1.192092896 \cdot 10^{-07}$
double	33	10	$2.2250738585072014 \cdot 10^{-308}$ $1.7976931348623157 \cdot 10^{+308}$ $2.2204460492503131 \cdot 10^{-16}$
long double 80 Bit (Intel-Architektur)	64	14	$3.3621031431120935062627 \cdot 10^{-4932}$ $1.1897314953572317650213 \cdot 10^{+4932}$ $1.0842021724855044340075 \cdot 10^{-19}$
long double 128 Bit	113	14	$3.36210314311209350626267781732175260 \cdot 10^{-4932}$ $1.189731495357231765085759326628007016 \cdot 10^{+4932}$ $1.925929944387235853055977942584927319 \cdot 10^{-34}$

D.6 Zeichen

- Bezeichnung: **char**
- Speicherbedarf: 1 Byte
- Repräsentation: ASCII-Code
zählt damit zu den ganzen Zahlen
- Konstanten: Zeichen durch ' ' geklammert
 - ◆ Beispiele: 'a', 'x'
 - ◆ Sonderzeichen werden durch **Escape-Sequenzen** beschrieben
 - Tabulator: '\t' Backslash: '\\'
 - Zeilentrenner: '\n' Backspace: '\b'
 - Apostroph: '\''

D.7 Zeichenketten

- Bezeichnung: **char ***
- Speicherbedarf: (Länge + 1) Bytes
- Repräsentation: Folge von Einzelzeichen,
letztes Zeichen: 0-Byte (ASCII-Wert 0)
- Werte: alle endlichen Folgen von **char**-Werten
- Konstanten: Zeichenkette durch " " geklammert
 - ◆ Beispiel: "Dies ist eine Zeichenkette"
 - ◆ Sonderzeichen wie bei char, " wird durch \" dargestellt
- Beispiel für eine Definition einer Zeichenkette:
char *Mitteilung = "Dies ist eine Mitteilung\n";

D.6 Zeichen (2)

American Standard Code for Information Interchange (ASCII)

NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL
00	01	02	03	04	05	06	07
BS	HT	NL	VT	NP	CR	SO	SI
08	09	0A	0B	0C	0D	0E	0F
DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB
10	11	12	13	14	15	16	17
CAN	EM	SUB	ESC	FS	GS	RS	US
18	19	1A	1B	1C	1D	1E	1F
SP	"	#	\$	%	&	'	(
20	21	22	23	24	25	26	27
()	*	+	,	-	.	/
28	29	2A	2B	2C	2D	2E	2F
0	1	2	3	4	5	6	7
30	31	32	33	34	35	36	37
8	9	:	;	<	=	>	?
38	39	3A	3B	3C	3D	3E	3F
@	A	B	C	D	E	F	G
40	41	42	43	44	45	46	47
H	I	J	K	L	M	N	O
48	49	4A	4B	4C	4D	4E	4F
P	Q	R	S	T	U	V	W
50	51	52	53	54	55	56	57
X	Y	Z	[\]	^	_
58	59	5A	5B	5C	5D	5E	5F
`	a	b	c	d	e	f	g
60	61	62	63	64	65	66	67
h	i	j	k	l	m	n	o
68	69	6A	6B	6C	6D	6E	6F
p	q	r	s	t	u	v	w
70	71	72	73	74	75	76	77
x	y	z	{		~	DEL	
78	79	7A	7B	7C	7D	7E	7F

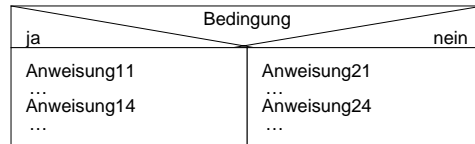
D.8 Aufbau eines C-Programms

- frei formulierbar - **Zwischenräume** (*Leerstellen, Tabulatoren, Zeilenvorschub und Kommentare*) werden i. a. ignoriert - sind aber zur eindeutigen Trennung direkt benachbarter Worte erforderlich
- **Kommentar** wird durch /* und */ geklammert,
keine Schachtelung möglich
- Hauptprogramm


```
main()
{
    Variablendefinitionen
    Anweisungen
}
```
- Anweisungen werden generell durch ; abgeschlossen

D.8 Aufbau eines C-Programms (2)

Bedingte Anweisung



```
if ( Bedingung ) {  
    Anweisung11  
    ...  
    Anweisung14  
    ...  
}  
else {  
    Anweisung21  
    ...  
    Anweisung24  
    ...  
}
```

- Den **else**-Teil kann man auch weglassen!

D.10 Operatoren

1 Zuweisungsoperator =

→ Zuweisung eines Werts an eine Variable

- Beispiel:

```
int a;  
a = 20;
```

D.9 Ausdrücke

- Ausdruck = gültige Kombination von **Operatoren, Konstanten und Variablen**
- Reihenfolge der Auswertung
 - ◆ Die Vorrangregeln für Operatoren legen die Reihenfolge fest, in der Ausdrücke abgearbeitet werden
 - ◆ Geben die Vorrangregeln keine eindeutige Aussage, ist die Reihenfolge undefiniert
 - ◆ Mit Klammern () können die Vorrangregeln überstimmt werden
 - ◆ Es bleibt dem Compiler freigestellt, Teilausdrücke in möglichst effizienter Folge auszuwerten

2 Arithmetische Operatoren

[D.10 Operatoren](#)

→ für alle **int** und **float** Werte erlaubt

+	Addition
-	Subtraktion
*	Multiplikation
/	Division
%	Rest bei Division, (modulo)
unäres -	negatives Vorzeichen (z. B. -3)
unäres +	positives Vorzeichen (z. B. +3)

- Beispiel:

```
a = -5 + 7 * 20 - 8;
```

3 spezielle Zuweisungsoperatoren D.10 Operatoren

↳ Verkürzte Schreibweise für Operationen auf einer Variablen

$a \text{ op} = b \equiv a = a \text{ op } b$
 mit $\text{op} \in \{+, -, *, /, \%, <<, >>, \&, \wedge, | \}$

■ Beispiele:

```
a = -8;
a += 24;
a /= 2;

/* -> a: 16 */
/* -> a: 8 */
```

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Logische Operatoren D.10 Operatoren

↳ Verknüpfung von Wahrheitswerten (wahr / falsch)

"nicht"		"und"		"oder"	
!		&&			
f	w	f	f	f	w
w	f	w	f	w	w

◆ Wahrheitswerte (Boole'sche Werte) werden in C generell durch int-Werte dargestellt:

- ▶ Operanden in einem Ausdruck: Operand = 0: falsch
Operand ≠ 0: wahr
- ▶ Ergebnis eines Ausdrucks: falsch: 0
wahr: 1

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Vergleichsoperatoren D.10 Operatoren

< kleiner
 <= kleiner gleich
 > größer
 >= größer gleich
 == gleich
 != ungleich

■ **Beachte!** Ergebnistyp int: wahr (true) = 1
falsch (false) = 0

■ Beispiele:

```
a > 3
a <= 5
a == 0
if ( a >= 3 ) { ...
```

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Logische Operatoren (2) D.10 Operatoren

■ Beispiel:

```
a = 5; b = 3; c = 7;
a > b && a > c
  1   und  0
  0
```

■ Die Bewertung solcher Ausdrücke wird abgebrochen, sobald das Ergebnis feststeht!

```
(a > c) && ((d=a) > b)
  0           wird nicht ausgewertet
  ↓
Gesamtergebnis=falsch → (d=a) wird nicht ausgeführt
```

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Bitweise logische Operatoren

D.10 Operatoren

→ Operation auf jedem Bit einzeln (Bit 1 = wahr, Bit 0 = falsch)

"nicht"	~		^	f	w
"und"	&	Antivalenz	f	f	w
"oder"		"exklusives oder"	w	w	f

■ Beispiele:

x	1 0 0 1 1 1 0 0
~x	0 1 1 0 0 0 1 1
7	0 0 0 0 0 1 1 1
x 7	1 0 0 1 1 1 1 1
x & 7	0 0 0 0 0 1 0 0
x ^ 7	1 0 0 1 1 0 1 1

C-Ing

Einführung in die Programmierung für Ingenieure — C

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

D-TypenOperatorenAusdruecke.doc 1998-05-12 16.37

D.25

Reproduktion jeder Art oder Vervielfachung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Inkrement / Dekrement Operatoren

D.10 Operatoren

++	inkrement
--	dekrement

- **linksseitiger Operator:** ++x bzw. --x
 - es wird der Inhalt von x inkrementiert bzw. dekrementiert
 - das Resultat wird als Ergebnis geliefert
- **rechtsseitiger Operator:** x++ bzw. x--
 - es wird der Inhalt von x als Ergebnis geliefert
 - anschließend wird x inkrementiert bzw. dekrementiert.

■ Beispiele:

```
a = 10;
b = a++; /* -> b: 10 und a: 11 */
c = ++a; /* -> c: 12 und a: 12 */
```

C-Ing

Einführung in die Programmierung für Ingenieure — C

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

D-TypenOperatorenAusdruecke.doc 1998-05-12 16.37

D.27

Reproduktion jeder Art oder Vervielfachung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Logische Shiftoperatoren

D.10 Operatoren

→ Bits werden im Wort verschoben

<<	Links-Shift
>>	Rechts-Shift

■ Beispiel:

x	1 0 0 1 1 1 0 0
x << 2	0 1 1 1 0 0 0 0

C-Ing

Einführung in die Programmierung für Ingenieure — C

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

D-TypenOperatorenAusdruecke.doc 1998-05-12 16.37

D.26

Reproduktion jeder Art oder Vervielfachung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Bedingte Bewertung

D.10 Operatoren

A ? B : C

- der Operator dient zur Formulierung von Bedingungen in Ausdrücken
- zuerst wird Ausdruck **A** bewertet
- ist **A ungleich 0**, so hat der gesamte Ausdruck als Wert den Wert des Ausdrucks **B**,
- sonst den Wert des Ausdrucks **C**

■ Beispiel:

```
c = a>b ? a : b; /* z = max(a,b) */
besser:
c = (a>b) ? a : b;
```

C-Ing

Einführung in die Programmierung für Ingenieure — C

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

D-TypenOperatorenAusdruecke.doc 1998-05-12 16.37

D.28

Reproduktion jeder Art oder Vervielfachung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

9 Komma-Operator

D.10 Operatoren

,

- der Komma-Operator erlaubt die Aneinanderreihung mehrerer Ausdrücke
- ein so gebildeter Ausdruck hat als Wert den Wert des letzten Teil-Ausdrucks

C-Ing

Einführung in die Programmierung für Ingenieure — C

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

D-TypenOperatorenAusdruecke.doc 1998-05-12 16.37

D.29

Reproduktion jeder Art oder Vervielfachung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.12 Vorrangregeln bei Operatoren

Operatorklasse	Operatoren	Assoziativität
unär	! ~ ++ -- + -	von rechts nach links
multiplikativ	* / %	von links nach rechts
additiv	+ -	von links nach rechts
shift	<< >>	von links nach rechts
relational	< <= > >=	von links nach rechts
Gleichheit	== !=	von links nach rechts
bitweise	&	von links nach rechts
bitweise	^	von links nach rechts
bitweise		von links nach rechts
logisch	&&	von links nach rechts
logisch		von links nach rechts
Bedingte Bewertung	?:	von rechts nach links
Zuweisung	= op=	von rechts nach links
Reihung	,	von links nach rechts

C-Ing

Einführung in die Programmierung für Ingenieure — C

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

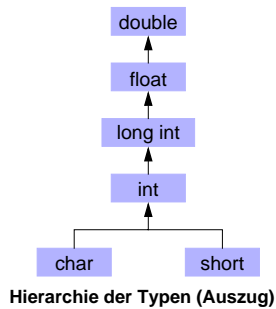
D-TypenOperatorenAusdruecke.doc 1998-05-12 16.37

D.31

Reproduktion jeder Art oder Vervielfachung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

D.11 Typumwandlung in Ausdrücken

- Enthält ein Ausdruck Operanden unterschiedlichen Typs, erfolgt eine automatische Umwandlung in den Typ des in der **Hierarchie der Typen** am höchsten stehenden Operanden. (*Arithmetische Umwandlungen*)



C-Ing

Einführung in die Programmierung für Ingenieure — C

© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

D-TypenOperatorenAusdruecke.doc 1998-05-12 16.37

D.30

Reproduktion jeder Art oder Vervielfachung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.