

## Inhalt

Anhang A .....	2
Mindestschranken für die Wertebereiche .....	2
Anhang B.....	3
C-Operatoren (mit Rangfolge und Assoziativitätsrichtung) .....	3
Anhang C.....	4
Formatierte Eingabe .....	4
Anhang D .....	5
Formatierte Ausgabe .....	5
Anhang E.....	6
Übersicht über die Standardbibliothek .....	6

# Anhang A

## Mindestschranken für die Wertebereiche

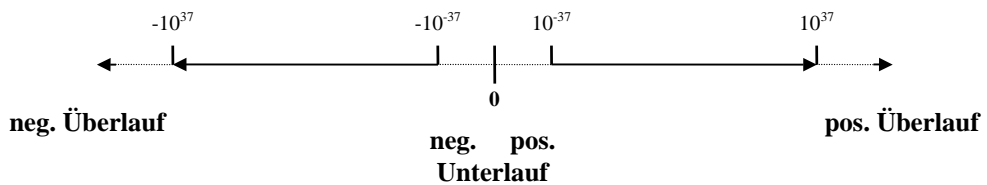
- Ein Byte muss 8 Bit lang sein.
- Wertebereiche der ganzzahligen Typen:

Typ	kleinster Wert	größter Wert	Byte
signed char	$-127 (= -2^7 + 1)$	$127 (= 2^7 - 1)$	1
unsigned char	0	$255 (= 2^8 - 1)$	1
signed short int	$-32767 (= -2^{15} + 1)$	$32767 (= 2^{15} - 1)$	2
unsigned short int	0	$65535 (= 2^{16} - 1)$	2
signed int	$-32767 (= -2^{15} + 1)$	$32767 (= 2^{15} - 1)$	2
unsigned int	0	$65535 (= 2^{16} - 1)$	2
signed long int	$-2147483647 (= -2^{31} + 1)$	$2147483647 (= 2^{31} - 1)$	4
unsigned long int	0	$4294967295 (= 2^{32} - 1)$	4

- Wertebereich für Gleitpunkttypen:  $[-10^{37}, -10^{-37}] \cup \{0\} \cup [10^{-37}, 10^{37}]$   
Genauigkeit (Dichte) für Gleitpunkttypen:

Typ	Genauigkeit	Stellen	Byte
float	$10^{-5}$	6	4
double	$10^{-9}$	10	6
long double	$10^{-9}$	10	6

Überlauf → Laufzeitfehler  
Unterlauf → Rundungsfehler



## Anhang B

### C-Operatoren (mit Rangfolge und Assoziativitätsrichtung)

15	( ) [ ] -> .	Ausdrucksgruppierung (Tiefe max. 32) Auswahl der Feldkomponenten Auswahl der Strukturkomponenten	=>
14	! ~ ++ -- + - ( Typ ) & * sizeof( Typ )	Negation (logisch, bitweise) Inkrementation, Dekrementation (Präfix oder Postfix) Vorzeichen Typumwandlung Adressbildung, Dereferenzierung Bestimmung des Speicherbedarfs	<=
13	* / %	Multiplikation, Division Rest bei ganzzahliger Division	=>
12	+ -	Summe, Differenz	=>
11	<< >>	bitweise Verschiebung nach links, rechts	=>
10	< <= > >=	Vergleich auf kleiner, kleiner oder gleich Vergleich auf größer, größer oder gleich	=>
9	== !=	Vergleich auf gleich, ungleich	=>
8	&	Und (bitweise)	=>
7	^	exklusives Oder (bitweise)	=>
6		inklusive Oder (bitweise)	=>
5	&&	Und (logisch)	=>
4		inklusive Oder (logisch)	=>
3	? :	bedingte Auswertung (paarweise)	<=
2	= °=	Wertzuweisung zusammengesetzte Wertzuweisung (* =, / =, % =, + =, - =, << =, >> =, & =, ^ =,   = )	<=
1	,	sequentielle Auswertung	=>

## Anhang C

### Formatierte Eingabe

**Standardeingabe:**            `int scanf ( const char * Format , Zeiger , ... ) ;`

*Zeiger:*

- Zeiger auf den Speicherbereich, in den der gelesene Wert geschrieben werden soll.

*Format:*

- Formatierungsstring, gibt die Interpretationsvorschrift für die Eingabewerte an.
- Folge von *Formatbeschreibern*.

### Grundaufbau eines Formatbeschreibers:

`% [ * ] [ Länge ] [ Typ ] Kennung`

**%:**            Anfangszeichen eines *Formatbeschreibers*.

**\*:**            Eingabe wird zwar interpretiert aber nicht abgespeichert, d.h. sie wird übergangen.

*Länge:*        Maximalzahl der zu interpretierenden Zeichen.

*Typ:*            Abweichungen vom *Standardtyp*

**h**            **short** bei ganzzahligen Werten

**l**            **long** bei ganzzahligen Werten

**double** bei Gleitkommawerten

**L**            **long double** bei Gleitkommawerten

*Kennung:*     Konvertierungsvorschrift

<i>Kennung</i>	erwartete Zeichenfolge	<i>Standardtyp</i>
<b>i d</b>	ganzzahlig dezimal, mit oder ohne Vorzeichen.	<b>signed int *</b>
<b>u</b>	ganzzahlig dezimal.	<b>unsigned int *</b>
<b>o</b>	ganzzahlig oktal.	
<b>x</b>	ganzzahlig hexadezimal.	
<b>X</b>	ganzzahlig hexadezimal.	
<b>f e E g G</b>	Gleitkommawert, mit oder ohne Vorzeichen, mit oder ohne Dezimalpunkt, mit oder ohne Exponentialteil.	<b>float *</b>
<b>c</b>	einzelnes Zeichen oder Zeichenfolge, auch „white space“.	<b>char *</b>
<b>s</b>	bel. Zeichenfolge, mit „white space“ beendet, \0 wird automatisch angehängt.	<b>char *</b>
[ <i>Zf</i> ]	wie <i>s</i> , wobei <i>Zf</i> die Zeichenfolge der zu akzeptierenden Zeichen ist, bei unzulässigen Zeichen stoppt die Übertragung.	<b>char *</b>
[ ^ <i>Zf</i> ]	wie <i>s</i> , wobei <i>Zf</i> die Zeichenfolge der nicht zu akzeptierenden Zeichen ist, die Übertragung stoppt.	<b>char *</b>

## Anhang D

### Formatierte Ausgabe

**Standardausgabe:** `int printf( const char *Format, Ausdruck, ... );`

*Ausdruck:*

- Der Wert des *Ausdrucks* wird berechnet und übertragen.

*Format:*

- Formatierungsstring, gibt die Interpretationsvorschrift für die Ausgabewerte an.
- Zeichenkettenkonstante mit ein oder mehreren *Formatbeschreibern*.

### Grundaufbau eines Formatbeschreibers:

`% [ Modus ] [ Länge ] [ .Stellen ][ Typ ] Kennung`

**%:** Anfangszeichen eines *Formatbeschreibers*.

**Modus:** 4 Steuerzeichen: -, +, **Leerzeichen**, # zur Druckgestaltung

**Länge:** Mindestzahl der zu übertragenden Zeichen, evtl. mit Leerzeichen aufgefüllt.

**Stellen:** Von der jeweiligen *Kennung* abhängig:

**i d u o x X** Mindestzahl der Ziffern, evtl. mit Nullen aufgefüllt.

**f e E g G** Stellen hinter dem Komma.

**c s** Höchstzahl der Zeichen.

**Typ:** Abweichungen vom Standardtyp

**h** **short** bei ganzzahligen Werten

**l** **long** bei ganzzahligen Werten

**L** **long** bei Gleitkommawerten

(**float**-Werte werden beim Aufruf in **double** umgewandelt.)

**Kennung:** Konvertierungsvorschrift

<b>Kennung</b>	<b>Darstellung</b>	<b>Standardtyp</b>
<b>i d</b>	ganzzahlig dezimal, mit oder ohne Vorzeichen.	<b>signed int</b>
<b>u</b>	ganzzahlig dezimal.	<b>unsigned int</b>
<b>o</b>	ganzzahlig okt.	
<b>x</b>	ganzzahlig hexadezimal, mit <b>x</b> .	
<b>X</b>	ganzzahlig hexadezimal, mit <b>X</b> .	
<b>f</b>	exponentenfrei mit oder ohne Dezimalpunkt, gerundet.	<b>double</b>
<b>e</b>	halblogarithmisch, mit <b>e</b> , gerundet.	
<b>E</b>	halblogarithmisch, mit <b>E</b> , gerundet.	
<b>g</b>	je nach Größe wie <b>f</b> oder <b>e</b> , Dezimalpunkt und Nullen werden ggf. weggelassen.	
<b>G</b>	je nach Größe wie <b>f</b> oder <b>E</b> , Dezimalpunkt und Nullen werden ggf. weggelassen.	
<b>c</b>	einzelnes Zeichen	<b>char</b>
<b>s</b>	Zeichenfolge	<b>char *</b>

## Anhang E

### *Übersicht über die Standardbibliothek*

<b>&lt;assert.h&gt;</b>	Testhilfen	
<b>&lt;ctype.h&gt;</b>	Zeichenverarbeitung	isdigit,...
<b>&lt;errno.h&gt;</b>	Fehlernummern	
<b>&lt;float.h&gt;</b>	Interne Datenformate (Gleitkommatypen)	FLT_MAX,...
<b>&lt;limits.h&gt;</b>	Interne Datenformate (Ganzzahlige Typen)	INT_MAX,...
<b>&lt;local.h&gt;</b>	Länderspezifische Darstellung von Zahlen	
<b>&lt;math.h&gt;</b>	Mathematische Funktionen	sin, cos,...
<b>&lt;setjmp.h&gt;</b>	Sprünge zwischen Funktionen	
<b>&lt;signal.h&gt;</b>	Behandlung von Signalen	
<b>&lt;stdarg.h&gt;</b>	Abarbeitung von Funktionen mit variabler Parameterzahl	
<b>&lt;stddef.h&gt;</b>	Elementare Typen	size_t, NULL,...
<b>&lt;stdio.h&gt;</b>	Ein-/ Ausgabe	printf, scanf,...
<b>&lt;stdlib.h&gt;</b>	Diverse Hilfsroutinen	malloc,...
<b>&lt;string.h&gt;</b>	Stringverarbeitung	strcpy, strcat,...
<b>&lt;time.h&gt;</b>	Termine und Zeiten	