

Inhalt

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

Anhang A

Mindestschranken für die Wertebereiche

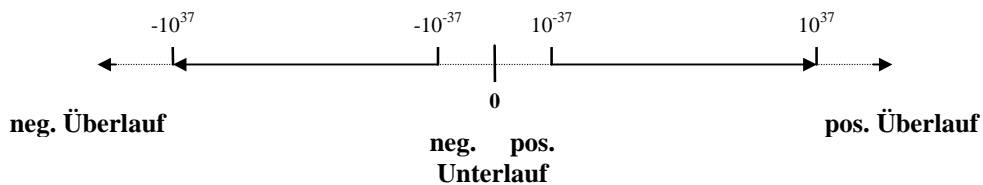
- Ein Byte muss 8 Bit lang sein.
- Wertebereiche der ganzzahligen Typen:

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

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

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

Anhang E

Übersicht über die Standardbibliothek

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