C-OPERATOREN

Op	Prio	Grupp	1. Op	2. Op	3.Op	Ergebnis	Bedeutung
()	1	lr	Funktionsref			Alle 1)	Funktionsaufruf
[]	1	lr	Arrayref	integral		Alle 2)	Arraykomponente
->	1	lr	Strukturpointer	Name		Alle 2)	Strukturkomponente
.	1	lr	Strukturref	Name		Alle 2)	Strukturkomponente
()	1	lr					Klammerung
+	2	rl	Arithmetisch			Arithmetisch	Vorzeichen +
-	2	rl	Arithmetisch			Arithemtisch	Vorzeichen -
~	2	rl	integral			integral	Bitweises NOT
!	2	rl	integral			0 oder 1	Logisches NOT
++	2	rl	Skalare Var.			Skalar	Inkrement
--	2	rl	Skalare Var.			Skalar	Dekrement
*	2	rl	Pointer			Alle 2)	Dereferenz
&	2	rl	Objekt			int	Adresse
sizeof	2	rl	beliebig			int	Größe in Bytes
() (Cast)	3	rl	Typ	Ausdruck		Typ	Typumwandlung
*	4	lr	Arithmetisch	Arithmetisch		Arithmetisch	Multiplikation
/	4	lr	Arithmetisch	Arithmetisch		Arithmetisch	Division
%	4	lr	integral	integral		integral	Rest
+	5	lr	Arithmetisch	Arithmetisch		Arithmetisch	Addition
-	5	lr	Arithmetisch	Arithmetisch		Arithmetisch	Subtraktion
<<	6	lr	integral	integral		wie 1. Operand	Bitschieben links
>>	6	lr	integral	integral		wie 1. Operand	Bitschieben rechts
<	7	lr	Skalar	Skalar		0 oder 1	Kleiner
<=	7	lr	Skalar	Skalar		0 oder 1	Kleiner gleich
>=	7	lr	Skalar	Skalar		0 oder 1	Größer gleich
>	7	lr	Skalar	Skalar		0 oder 1	Größer
==	8	lr	Skalar	Skalar		0 oder 1	Gleich
!=	8	lr	Skalar	Skalar		0 oder 1	Unleich
&	9	lr	integral	integral		integral	Bitweises AND
^	10	lr	integral	integral		integral	Bitweises XOR
|	11	lr	integral	integral		integral	Bitweises OR
&&	12	lr	Skalar	Skalar		0 oder 1	Logisches AND
||	13	lr	Skalar	Skalar		0 oder 1	Logisches OR
?:	14	rl	Skalar	Arithmetisch Pointer	Arithmetisch Pointer	Arithmetisch Pointer	Bedingung
=	15	rl	Variablenref	Arithmetisch Struktur Pointer		wie 1. Operand	Zuweisung
*= /= %= += -= <<= >>= &= ^= |=	15	rl	Variablenref	s.o.		wie 1. Operand	Zuweisung mit Op
,	16	lr	beliebig	beliebig		wie 2. Operand	Wert des 2. Operanden

1) außer array
2) außer void

Übliche arithmetische Umwandlungen (Usual Arithmetic Conversions)

```
Ist ein Operand vom Typ long double?
 JA: Umwandlung des anderen Operanden in long double
 SONST:
  Ist ein Operand vom Typ double?
   JA: Umwandlung des anderen Operanden in double
   SONST:
    Ist ein Operand vom Typ float?
     JA: Umwandlung des anderen Operanden in float
     SONST:
      Integral Promotion durchführen: [unsigned] char und [unsigned] short int
      werden nach int konvertiert.
      Ist ein Operand vom Typ unsigned long int ?
       JA: Umwandlung des anderen Operanden in unsigned long int
       SONST:
        Ist ein Operand vom Typ long int und der andere vom Typ unsigned int?
         JA: Umwandlung beider Operanden in unsigned long int
         SONST:
          Ist ein Operand vom Typ long int?
           JA: Umwandlung des anderen Operanden in long int
           SONST:
            Ist ein Operand vom Typ unsigned int?
             JA: Umwandlung des anderen Operanden in unsigned int
             SONST:
              Beide Operanden sind vom Typ int
```

Wertumwandlungen:

\VON NACH\	sc	uc	ss	us	si	ui	sl	ul	fl	db	ld
sc	-	1	3	3	3	3	3	3	5	5	5
uc	1	-	3	3	3	3	3	3	5	5	5
ss	2	2	-	1	3	3	3	3	5	5	5
us	2	2	1	-	3	3	3	3	5	5	5
si	2	2	2	2	-	1	3	3	5	5	5
ui	2	2	2	2	1	-	3	3	5	5	5
sl	2	2	2	2	2	2	-	1	5	5	5
ul	2	2	2	2	2	2	1	-	5	5	5
fl	4	4	4	4	4	4	4	4	-	6	6
db	4	4	4	4	4	4	4	4	4	-	6
ld	4	4	4	4	4	4	4	4	4	4	-

```
sc: char
uc: unsigned char
ss: short
us: unsigned short
si: int
ui: unsigned int
sl: long
ul: unsigned long
fl: float
db: double
ld: long double
```

-: keine Umwandlung
1: Bitmuster bleibt unverändert
2: Auffüllen mit linksseitigen Nullen bzw. Vorzeichenerweiterung
3: Abschneiden der höherwertigen Bits (ohne Overflowerkennung)
4: Nähester Kommawert
5: Ganzzahliger Anteil oder undefiniert
6: Nähester Kommawert oder undefiniert