Expressions 1

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

k=(i+j)
```

Expressions 2

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

z=(x+y)
```

Expressions 3

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

i=j;
```

Expressions 4

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

k=x+y;
```

Expressions 5

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

k=c;
```

Expressions 6

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

z=i/j
```

Expressions 7

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;
```

a=b=d

Es ha ndelt sich um sog. Mehrfachzuweisung. Die Zuweisung ist rechtsassoziativ. Es wird also zunächt der Wert von d gebildet und b zugewiesen, danach wird der Wert von b auf a zugewiesen.

Expressions 8

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

i=j=1.1
```

Expressions 9

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

z=k=x
```

Expressions 10

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

k=z=x
```

Expressions 11

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

i+=2
```

Expressions 12

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

y-=x
```

Expressions 13

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

x*=j-2

x=x*(j-2)
```

Expressions 14

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

i/=j+1

i=i/(j+1)
```

Expressions 15

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

i%=j


i=i%j
```

Expressions 16

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

k=(j==5)?i: j
```

```
if (j==5) tmp=i;
else      tmp=j;
k=tmp;
```

Expressions 17

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

k=j>5?i:j
```

```
if (j>5) tmp=i;
else     tmp=j;
k=tmp;
```

Expressions 18

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;
```

```
z=x>0?x:0
```

```
if (x>0) tmp=x;
else     tmp=0;
z=tmp;
```

Expressions 19

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;
```

```
z=y>=0?y:0
```

```
if (y>=0) tmp=y;
else      tmp=0;
z=tmp;
```

Expressions 20

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;
```

a=c<d=c:d

```
if (c<d) tmp=c;
else     tmp=d;
a=tmp;
```

Expressions 21

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

i-=j>0?j:0
```

```
if (j>0) tmp=j;
else     tmp=0;
i-=tmp;
```

Expressions 22

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

(3*i-2*j) % (2*d-c)
(  14   ) % ( 101 )
```

Expressions 23

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

2*( (i/5) + (4*(j-3) ) % (i+j-2) )
2*( (  1) + (       8) % (   11) )
2*(    1  +          8          )
```

Expressions 24

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

(i-3*j) % (c+2*d) / (x-y)
(   -7) % (  299) / (0.015)
```

Expressions 25

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

k=0xf0, k&=j
 11110000
&00000101
 00000000
```

Expressions 26

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

k=0xf0, k&&=j
 true
&true
 true
```

Expressions 27

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

k=0xaa, k^=8
 10101010
^00001000
 10100010
```

Expressions 28

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

c==99
```

Expressions 29

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

k=i=!6?1:0
```

Expressions 30

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

2*x+y==0
```

float / double nicht auf 0 testen!

Expressions 31

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

(i>0)||(j<5)
```

Expressions 32

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

!(i>0 && j<5)
```

!(i>0) || !(j<5)

i<=0 || j>=5

Expressions 33

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

(x>y)&&(i>0)||(j>5)
```

j>5 wird nicht bewertet

Expressions 34

```
int   i=8, j=5, k;
char  a, b, c='c';  d='d';
float x=0.005, y=-0.01, z;

(x>y)&&(i>0)&&(j>5)
```

Nr	Wert des Ausdrucks	Typ	Anmerkungen
1	13	int	k hat den Wert 13
2	-0.005	float/double	z hat den Wert -0.005
3	5	int	i hat den Wert 5
4	0	int	Der gesamte Ausdruck hat wg. der Zuweisung auf den int-Wert k, den Wert 0.
5	99 'c' 0x63 0143	int	Die Spalte Wert zeigt unterschiedliche Darstellungen des Wertes.
6	1	float/double	Die Zuweisung ist rechtsassoziativ, i/j wird ganzzahlig dividiert
7	100 0x64 'd'	char	a, b und d haben den selben Wert
8	1	int	j erhält den Wert von 1.1, dabei wird abgeschnitten, i und j bekommen den Wert 1.
9	0.0	float&/ double	Bei k=x wird abgeschnitten → 0, danach erfolgt erst die Zuweisung auf z.
10	0	int	Hier erhält z den Wert von x (0.005)
11	10	int	Entspricht dem Ausdruck i=i+2
12	-0.015	float/double	Entspricht y=y-x (-0,01-0,005)
13	0.015	float/double	Entspricht x=x*(j-2)
14	1	int	Entspricht i=i/(j+1)
15	3	int	Entspricht i=i%j, i modulo j liefert den ganzzahliugen Rest der Division 8/5
16	8	int	Hat k den Wert 5, so hat die rechte Seite den Wert von i, ansonsten den von j. Dieser Wert wird k dann zugewiesen.
17	5	int	Ist j Größer als 5, so hat die rechte Seite den Wert von i, ansonsten den von j. Dieser Wert wird k dann zugewiesen. Die Bedingung ist false, da j nicht größer als 5 ist.

Nr	Wert des Ausdrucks	Typ	Anmerkungen
18	0.005	float>/ double	
19	0.0	float/double	Da y negativ ist, ist es nicht größer oder gleich 0.
20	'c' / 0x63 / 99	char	Der ascii-code von c (99) ist kleiner als der von d (100)
21	3	int	Zuerst wird der bedingte Ausdruck bewertet, dann das Ergebnis von i subtrahiert.
22	14	int	Erst werden die geklammerten Ausdrücke berechnet, danach der ganzzahlige Rest einer Division
23	18	int	Die Klammern werden von innen nach außen ausgerechnet,
24	-466,6	Float/ double	Taschenrechner verwenden
25	0	int	Notieren Sie die Werte bitweise und veknüpfen Sie bitweise mit and
26	1	int	Wahrheitwert! Es werden logische Werte verknüpft (0: false, nicht 0: true)
27	10100010 / 0xA2 /162	int	Bitweise xor-Verknüpfung (gleiche bits→ 0, ungleiche bits→ 1)
28	1 (true)	int	Vergleich auf Gleichheit
29	0	int	i=!6 heit nicht i ungleich 6, sondern i bekommt den Wert von logisch !6? 1:0 (also 0, weil !6 false ergibt) zugewiesen, danach wird dieser Wert k zugewiesen.
30	(1)	int	Gleitpunkt werte sollten nicht aus (Un-)Gleichheit getestet werden, da die Werte systembedingt fehlerbehaftet sind.
31	1	int	Wahrheitswert
32	1	int	
33	1	int	J<5 wird nicht bewertet, weil (x>y)&&(i>0) bereits true liefert.
34	0	false	