

Ein C-Puzzler

Julian Fietkau

am 2. Dezember 2010
im KunterBuntenSeminar

```
1 #include <stdio.h>
2
3 void main(void)
4 {
5     double value = 10.0;
6     printf("%g", 1/value);
7 }
```

→ **0.1**

```
#define MACRO(x,y,z) 2*(x+y*z)
```

Funktionsartige Makros:

Können im Quelltext ähnlich wie Funktionen verwendet werden und bekommen Argumente übergeben.

Achtung:

Makros werden bereits zur Übersetzungszeit ausgewertet. Der Code des Makros wird an allen Stellen, an denen das Makro verwendet wird, vor dem eigentlichen Kompilieren einfach eingesetzt.

```
1 #include <stdio.h>
2
3 #define INVERSE(x) 1/x
4
5 void main(void)
6 {
7     double value = 10.0;
8     printf("%g", INVERSE(value));
9 }
```

→ **0.1**



Referenzierungs-Operator:

Liefert einen Zeiger auf eine Variable.



Dereferenzierungs-Operator:

Liefert den Inhalt der angegebenen Adresse.

```
1 #include <stdio.h>
2
3 void main(void)
4 {
5     double value = 10.0;
6     double* pValue = &value;
7     printf("%g", 1 / *pValue);
8 }
```

→ **0.1**

```
1 #include <stdio.h>
2
3 #define INVERSE(x) 1/x
4
5 void main(void)
6 {
7     double value = 10.0;
8     double* pValue = &value;
9     printf("%g", INVERSE(*pValue));
10 }
```

(A)

0.1

(B)

Übersetzungsfehler

(C)

Laufzeitfehler

(D)

noch etwas Anderes

```
1 #include <stdio.h>
2
3 #define INVERSE(x) 1/x
4
5 void main(void)
6 {
7     double value = 10.0;
8     double* pValue = &value;
9     printf("%g", INVERSE(*pValue));
10 }
```

(A)

0.1

(B)

Übersetzungsfehler

(C)

Laufzeitfehler

(D)

noch etwas Anderes

```
1 #include <stdio.h>
2
3
4
5 void main(void)
6 {
7     double value = 10.0;
8     double* pValue = &value;
9     printf("%g", 1/*pValue);
10 }
```

(A)

0.1

(B)

Übersetzungsfehler

(C)

Laufzeitfehler

(D)

noch etwas Anderes

```
1 #include <stdio.h>
2
3
4
5 void main(void)
6 {
7     double value = 10.0;
8     double* pValue = &value;
9     printf("%g", 1/*pValue);
10 }
```

(A)

0.1

(B)

Übersetzungsfehler

(C)

Laufzeitfehler

(D)

noch etwas Anderes

```
1 #include <stdio.h>
2
3 #define INVERSE(x) 1/x
4
5 void main(void)
6 {
7     double value = 10.0;
8     double* pValue = &value;
9     printf("%g", INVERSE(*pValue));
10 }
```

(A)

0.1

(B)

Übersetzungsfehler

(C)

Laufzeitfehler

(D)

noch etwas Anderes

```
gcc -std=c99 -E macro.c
```

```
687 (...)
688
689 void main(void)
690 {
691     double value = 10.0;
692     double* pValue = &value;
693     printf("%g", 1/_*pValue);
694 }
```

```
1 #define PREPEND_ASTERISK(x) *x
2 #define PREPEND_SLASH(x) /x
3
4 PREPEND_ASTERISK(literal)
5 PREPEND_SLASH(literal)
6 PREPEND_ASTERISK(*pointer)
7 PREPEND_SLASH(*pointer)
```

gcc -std=c99 -E test.c

```
7 (...)
8
9 *literal
10 /literal
11 **pointer
12 / *pointer
```

ISO/IEC 9899:TC3 (C99-Standard)

5.1.1.2 Translation phases:

3. The source file is decomposed into **preprocessing tokens**⁶⁾ and sequences of white-space characters (including comments). (...) **Each comment is replaced by one space character.** (...)
4. Preprocessing directives are executed, **macro invocations are expanded**, and `_Pragma` unary operator expressions are executed. (...)

(Hervorhebungen von mir.)

```
printf("%g", INVERSE(*pValue));
```

1

2

3

printf → (→ "%g" → , → [] → INVERSE → (→
* → pValue →) →) → ;

4

printf → (→ "%g" → , → [] → 1 → / → * →
pValue →) → ;

5

6

7

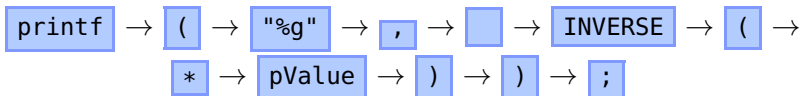
8

```
printf("%g", INVERSE(*pValue));
```

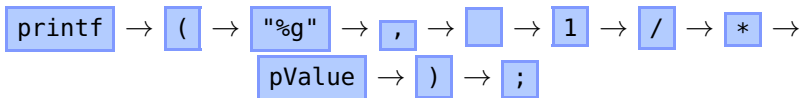
1

2

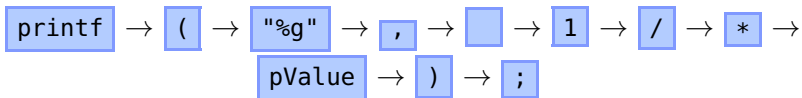
3



4



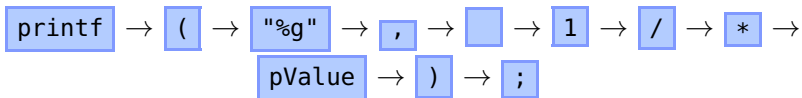
gcc -E



gcc -E

```
printf("%g", 1/*pValue);
```





gcc -E

```
printf("%g", 1/ *pValue);
```



Danke für die Aufmerksamkeit!



http://www.julian-fietkau.de/c_puzzler

