Debugging

C - Kurs 2010

Alexander Eichner

www.freitagsrunde.org

14. September 2010

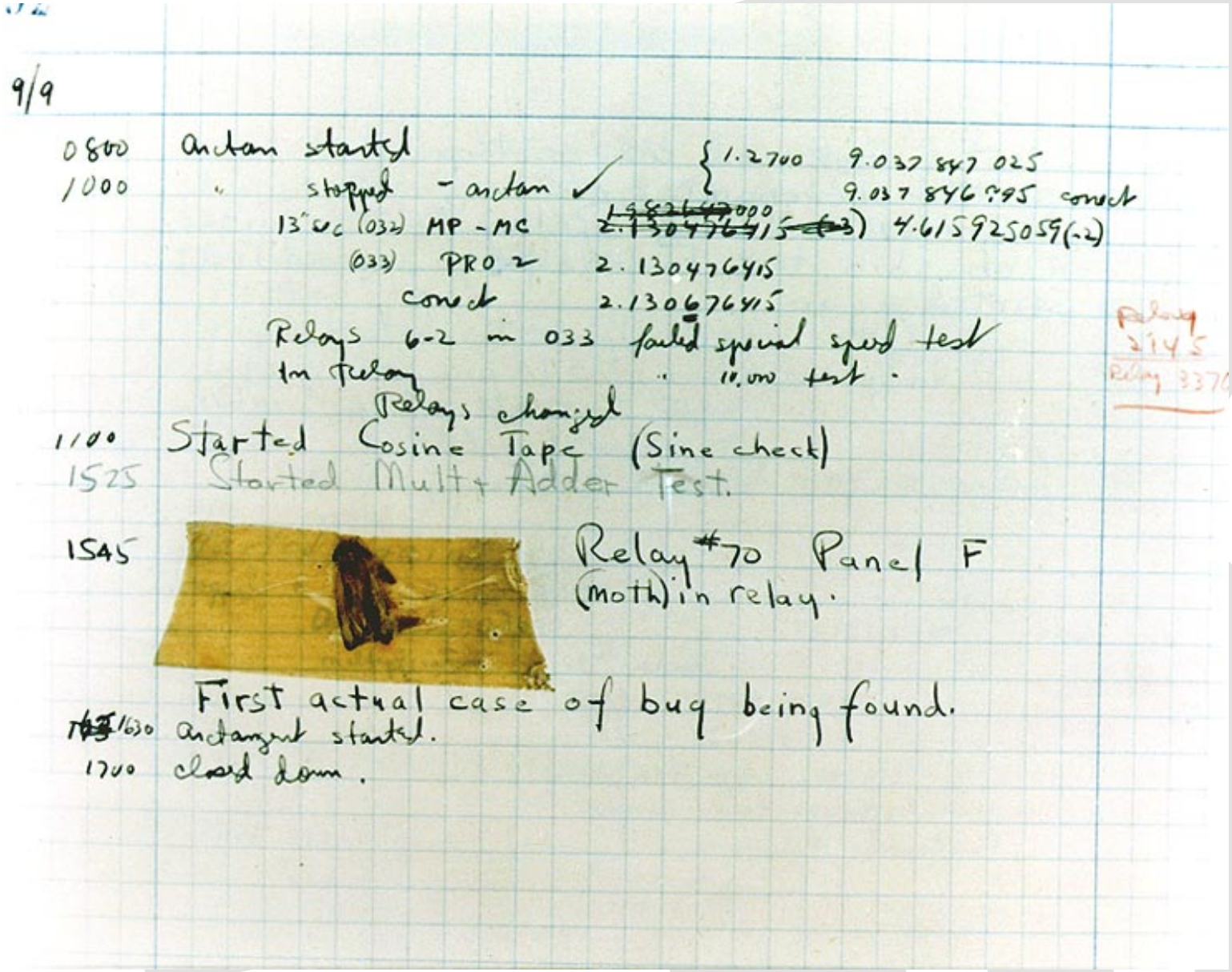


Abbildung: Erster gefundener Bug (1947).
Quelle: http://de.wikipedia.org/wiki/Datei:H96566k.jpg

Inhaltsverzeichniss

Inhaltsverzeichniss

Was habt ihr gelernt

- Theoretisch alles was man zum Schreiben eines Programms in C wissen muss

Was habt ihr gelernt

- Theoretisch alles was man zum Schreiben eines Programms in C wissen muss
- Es fehlt Übung und Erfahrung, um gewisse Fehler zu vermeiden

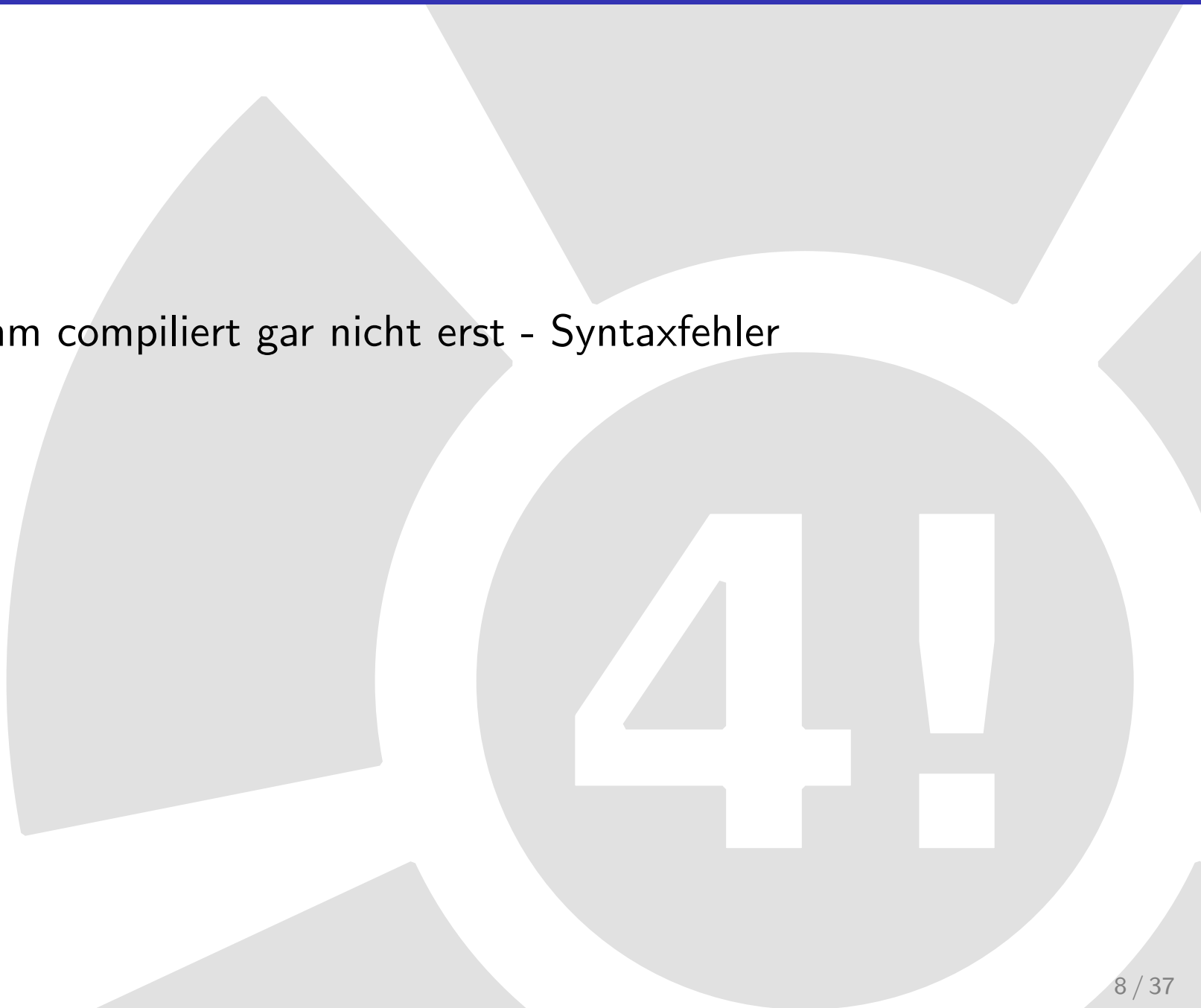
Ziele dieser Vorlesung

- Die Fallstricke von C kennenlernen
- Möglichkeiten des Debugging kennenlernen
- Bugs identifzieren
- einen Debugger effektiv einsetzen zu können
- euch das Leben einfacher zu machen ;)

Inhaltsverzeichniss

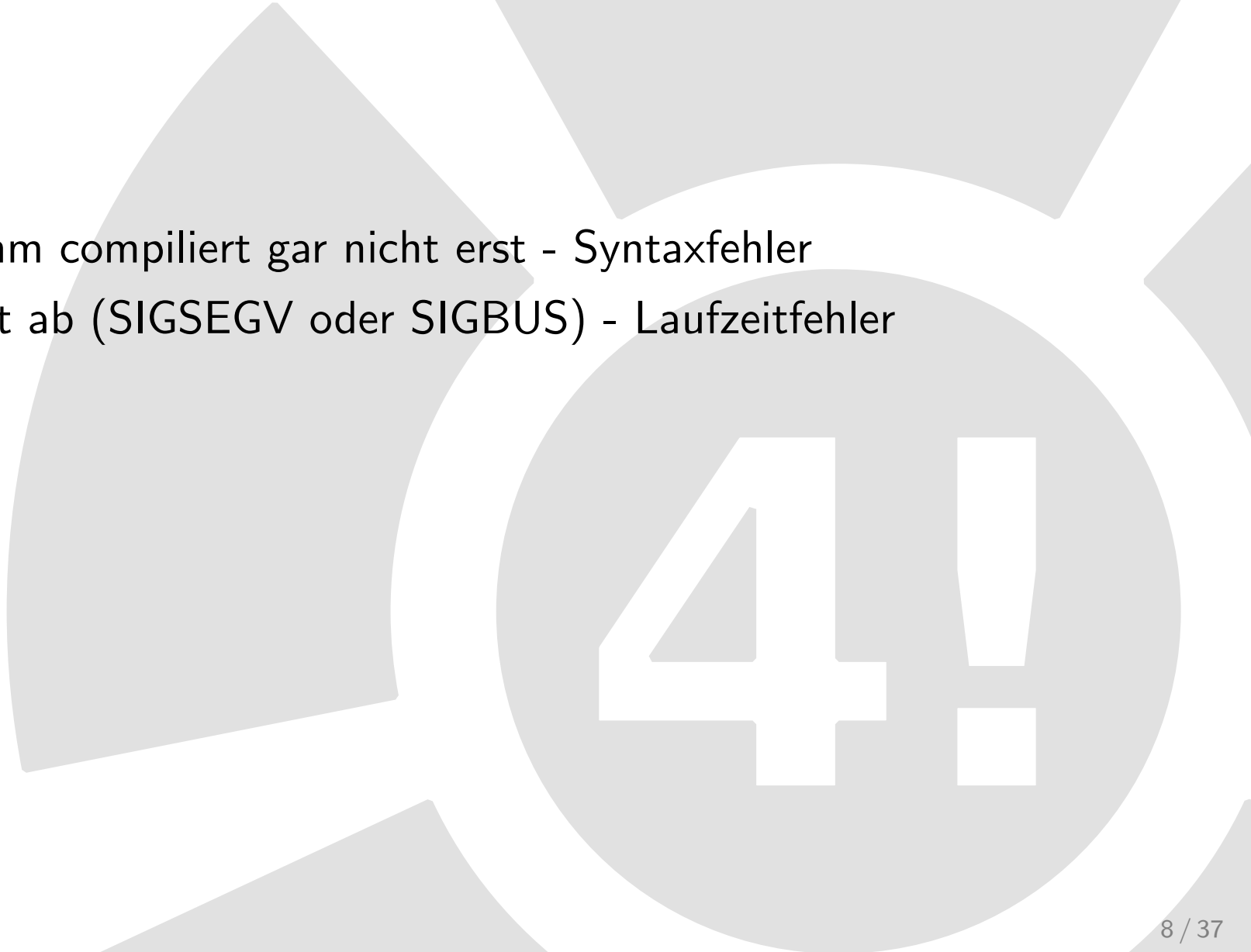
Welche Arten von Fehlern gibt es?

- Programm compiliert gar nicht erst - Syntaxfehler



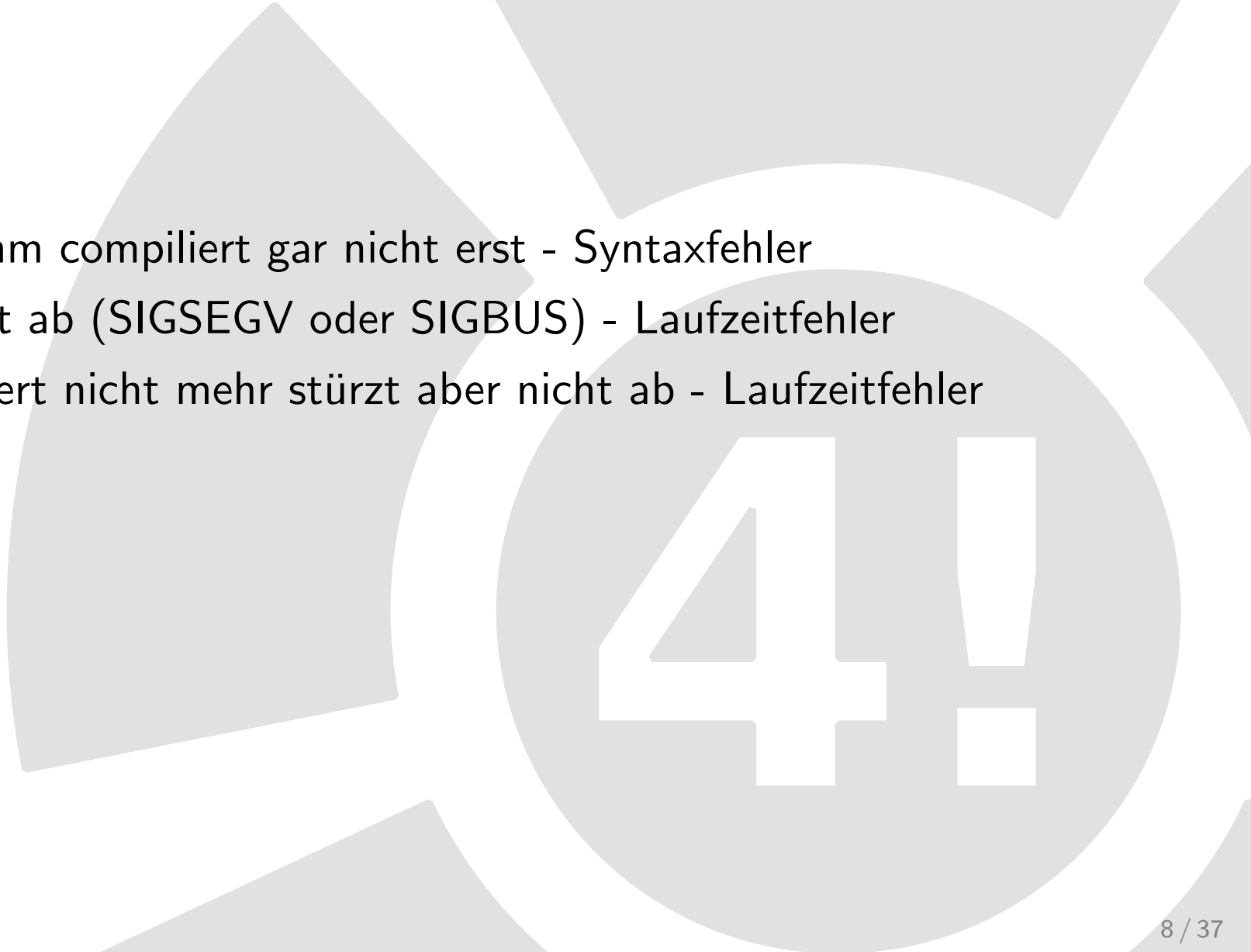
Welche Arten von Fehlern gibt es?

- Programm compiliert gar nicht erst - Syntaxfehler
- Es stürzt ab (SIGSEGV oder SIGBUS) - Laufzeitfehler



Welche Arten von Fehlern gibt es?

- Programm compiliert gar nicht erst - Syntaxfehler
- Es stürzt ab (SIGSEGV oder SIGBUS) - Laufzeitfehler
- Es reagiert nicht mehr stürzt aber nicht ab - Laufzeitfehler

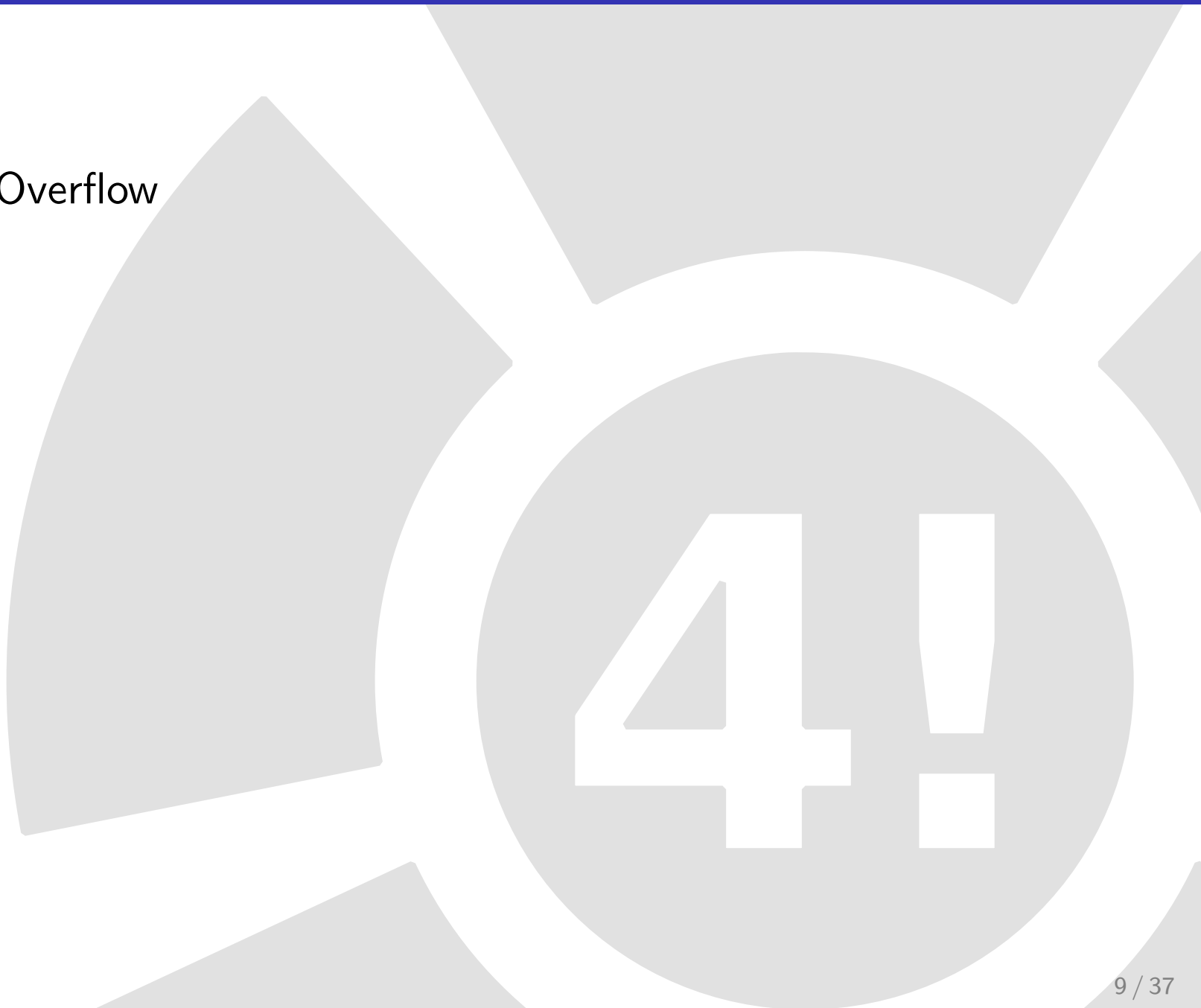


Welche Arten von Fehlern gibt es?

- Programm compiliert gar nicht erst - Syntaxfehler
- Es stürzt ab (SIGSEGV oder SIGBUS) - Laufzeitfehler
- Es reagiert nicht mehr stürzt aber nicht ab - Laufzeitfehler
- Es macht nicht das was es soll - Logikfehler oder Designfehler

Fehlerarten

Heap/Stack Overflow



Fehlerarten

Heap/Stack Overflow - Programm stürzt ab

- bedeutet das das Programm auf nicht reservierten Speicher zugreift

Beispiel

```
1 char *pStr = 0xdeadbeef;
2 printf("%s\n", pStr);
```

Ausgabe:

```
1 Segmentation fault
```

Fehlerarten

Uninitialisierte Variablen

Beispiel

```
1 int iWert;
2 printf("wert = %d\n", iWert);
```

Ausgabe:

```
1 wert = 1337
```

Fehlerarten

Integeroverflow

Beispiel

```
1 unsigned char byte = 255;
2 printf("byte = %d\n", byte);
3 byte++;
4 printf("byte = %d\n", byte);
```

Ausgabe:

```
1 byte = 255
2 byte = 0
```

Fehlerarten

Integeroverflow

Beispiel

```
1 unsigned char byte = 255;
2 printf("byte = %d\n", byte);
3 byte++;
4 printf("byte = %d\n", byte);
5 printf("%d\n", 10/byte);
```

Ausgabe:

```
1 byte = 255
2 byte = 0
```

Fehlerarten

Integeroverflow

Beispiel

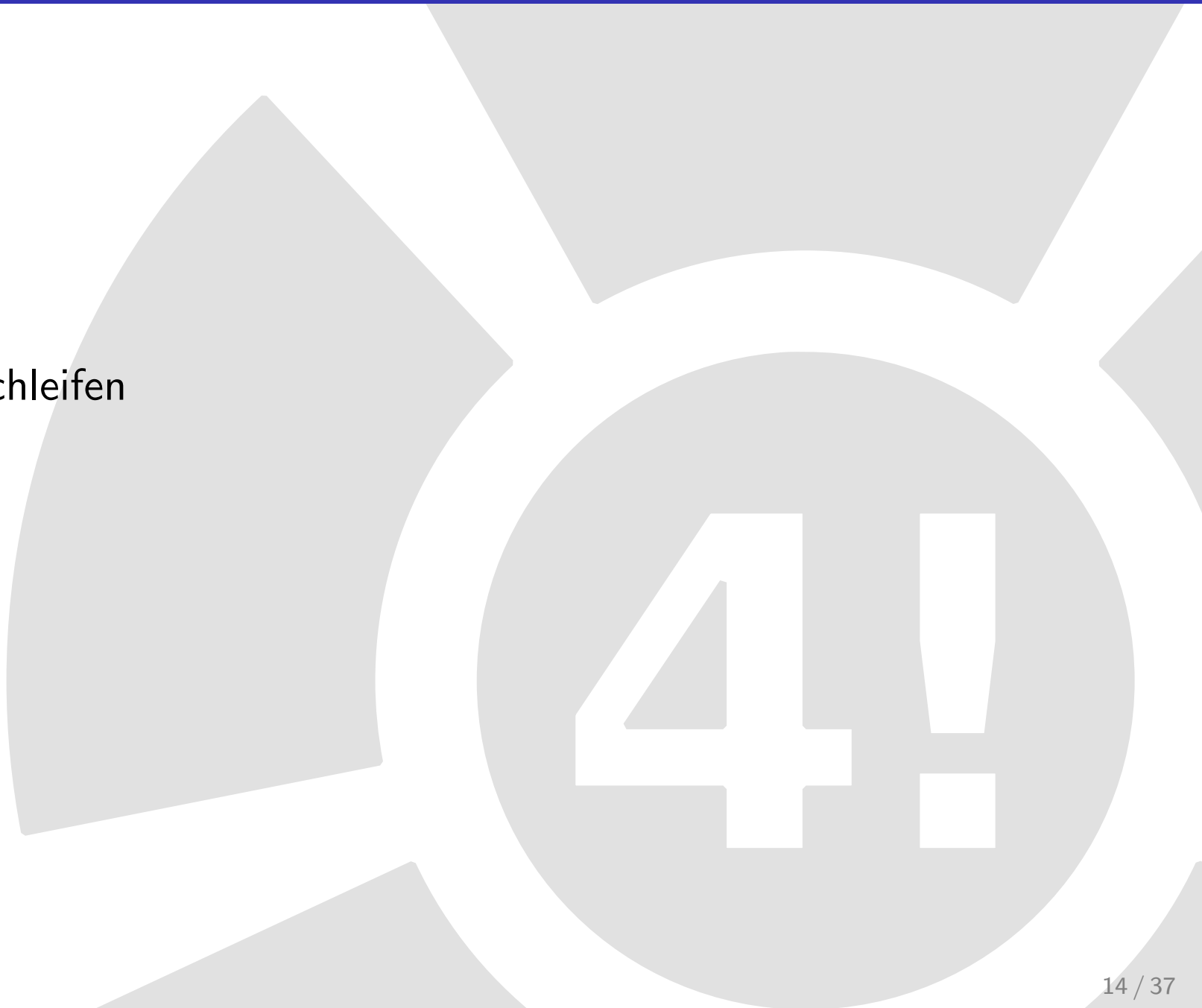
```
1 unsigned char byte = 255;
2 printf("byte = %d\n", byte);
3 byte++;
4 printf("byte = %d\n", byte);
5 printf("%d\n", 10/byte);
```

Ausgabe:

```
1 byte = 255
2 byte = 0
3 Floating point exception
```

Fehlerarten

- Endlosschleifen

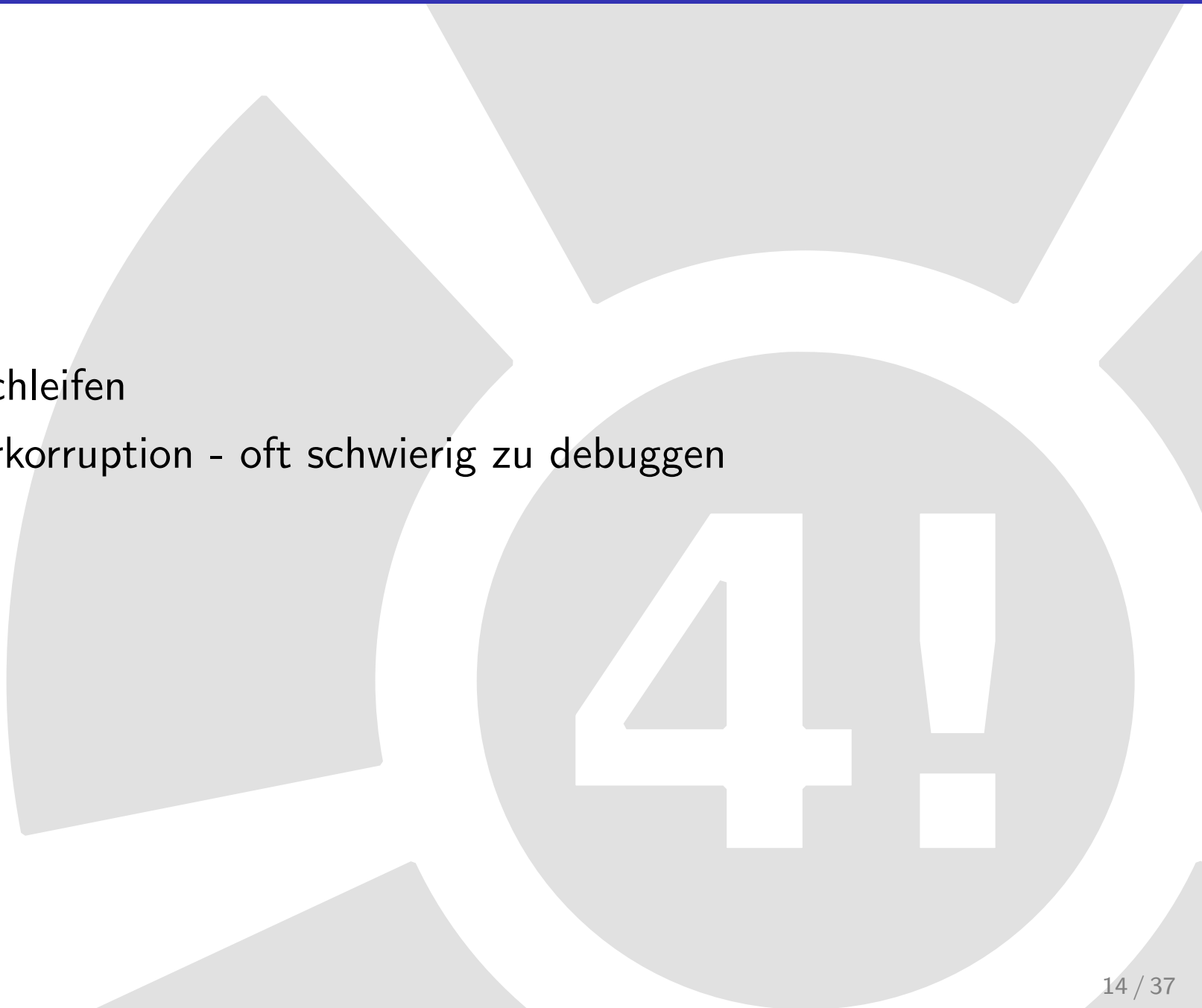


Fehlerarten

- Endlosschleifen
- Speicherkorruption

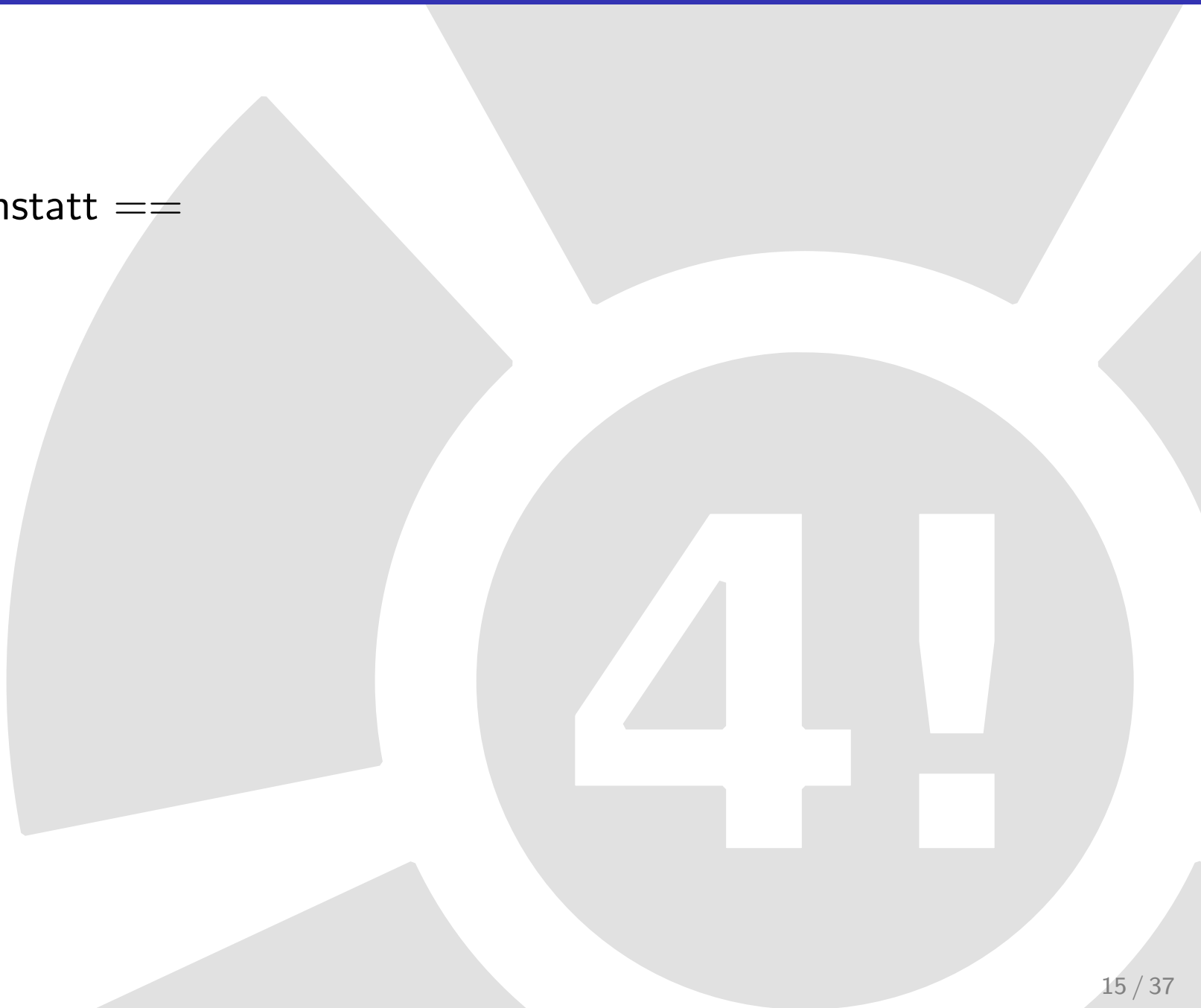
Fehlerarten

- Endlosschleifen
- Speicherkorruption - oft schwierig zu debuggen



Die 4 beliebtesten Stolpersteine in C

Platz 1: = anstatt ==



Die 4 beliebtesten Stolpersteine in C

Platz 1: = anstatt ==

Beispiel

```
1     int iWert = 2;
2     if (iWert = 3)
3         printf("iWert ist %d\n", iWert);
```

Die 4 beliebtesten Stolpersteine in C

Platz 1: = anstatt ==

Beispiel

```
1     int iWert = 2;
2     if (iWert = 3)
3         printf("iWert ist %d\n", iWert);
```

Ausgabe:

```
1         iWert ist 3
```

Die 4 beliebtesten Stolpersteine in C

Platz 2: ; an der falschen Stelle

Die 4 beliebtesten Stolpersteine in C

Platz 2: ; an der falschen Stelle

Beispiel

```
1 int iWert = 2;
2 if (iWert == 3);
3     printf("iWert ist 3\n");
```

Die 4 beliebtesten Stolpersteine in C

Platz 2: ; an der falschen Stelle

Beispiel

```
1 int iWert = 2;
2 if (iWert == 3);
3     printf("iWert ist 3\n");
```

Ausgabe:

```
1 iWert ist 3
```

Die 4 beliebtesten Stolpersteine in C

Beispiel: Endlosschleife

```
1 int iCountdown = 10;
2
3 while (iCountdown > 0);
4 {
5     printf("%d\n", iCountdown);
6     iCountdown--;
7 }
```

Die 4 beliebtesten Stolpersteine in C

Beispiel: Endlosschleife

```
1 int iCountdown = 10;
2
3 while (iCountdown > 0);
4 {
5     printf("%d\n", iCountdown);
6     iCountdown--;
7 }
```

Ausgabe:

Die 4 beliebtesten Stolpersteine in C

Platz 3: switch case ohne break

Die 4 beliebtesten Stolpersteine in C

Platz 3: switch case ohne break

Beispiel

```
1   int iValue = 2;
2
3   switch (iValue)
4   {
5       case 1: printf("iValue = 1\n");
6       case 2: printf("iValue = 2\n");
7       case 3: printf("iValue = 3\n");
8       default:
9           printf("iValue = %d\n", iValue);
10  }
```

Die 4 beliebtesten Stolpersteine in C

Platz 3: switch case ohne break

Beispiel

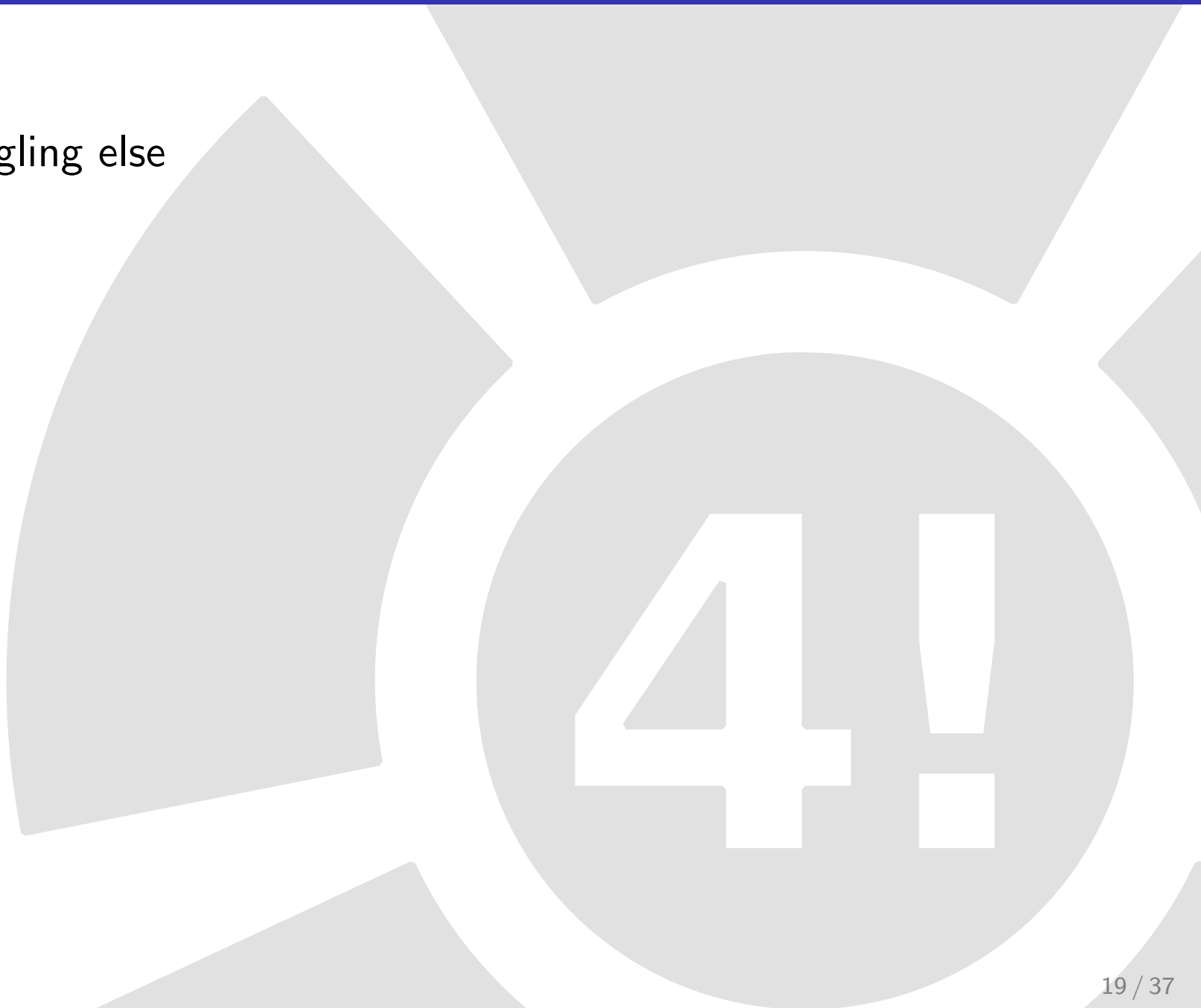
```
1   int iValue = 2;
2
3   switch (iValue)
4   {
5       case 1: printf("iValue = 1\n");
6       case 2: printf("iValue = 2\n");
7       case 3: printf("iValue = 3\n");
8       default:
9           printf("iValue = %d\n", iValue);
10  }
```

Ausgabe:

```
1       iValue = 2
2       iValue = 3
3       iValue = 2
```

Die 4 beliebtesten Stolpersteine in C

Platz 4: Dangling else



Die 4 beliebtesten Stolpersteine in C

Platz 4: Dangling else

Beispiel: Wo gehört das else hin?

```
1 int i = 2;
2 int j = 1;
3
4 if (i == 1)
5     if (j == 1)
6         printf("Both 1\n");
7 else
8     printf("iValue=%d\n", iValue);
```

Die 4 beliebtesten Stolpersteine in C

Platz 4: Dangling else

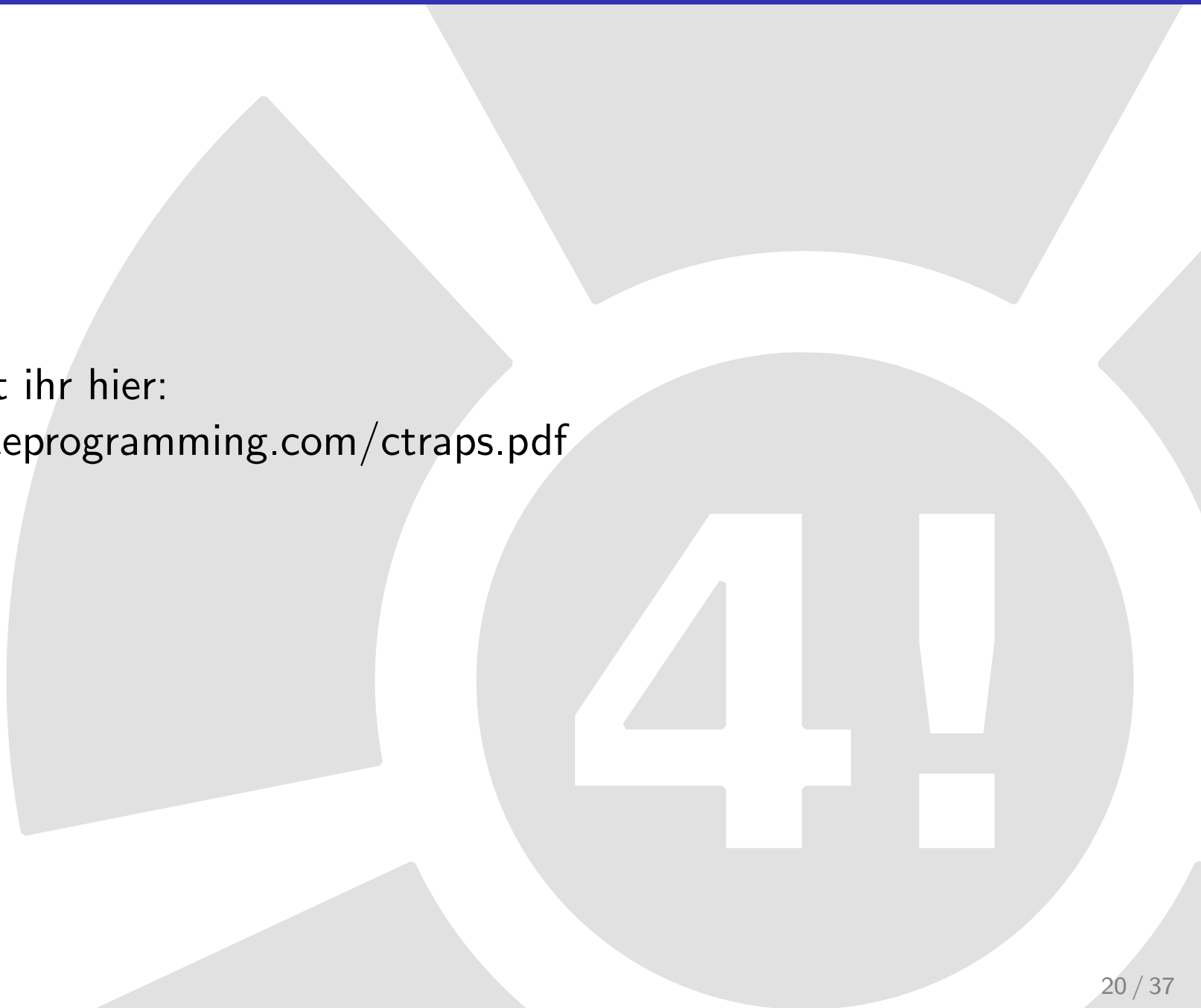
Beispiel: Wo gehört das else hin?

```
1 int i = 2;
2 int j = 1;
3
4 if (i == 1)
5     if (j == 1)
6         printf("Both 1\n");
7 else
8     printf("iValue=%d\n", iValue);
```

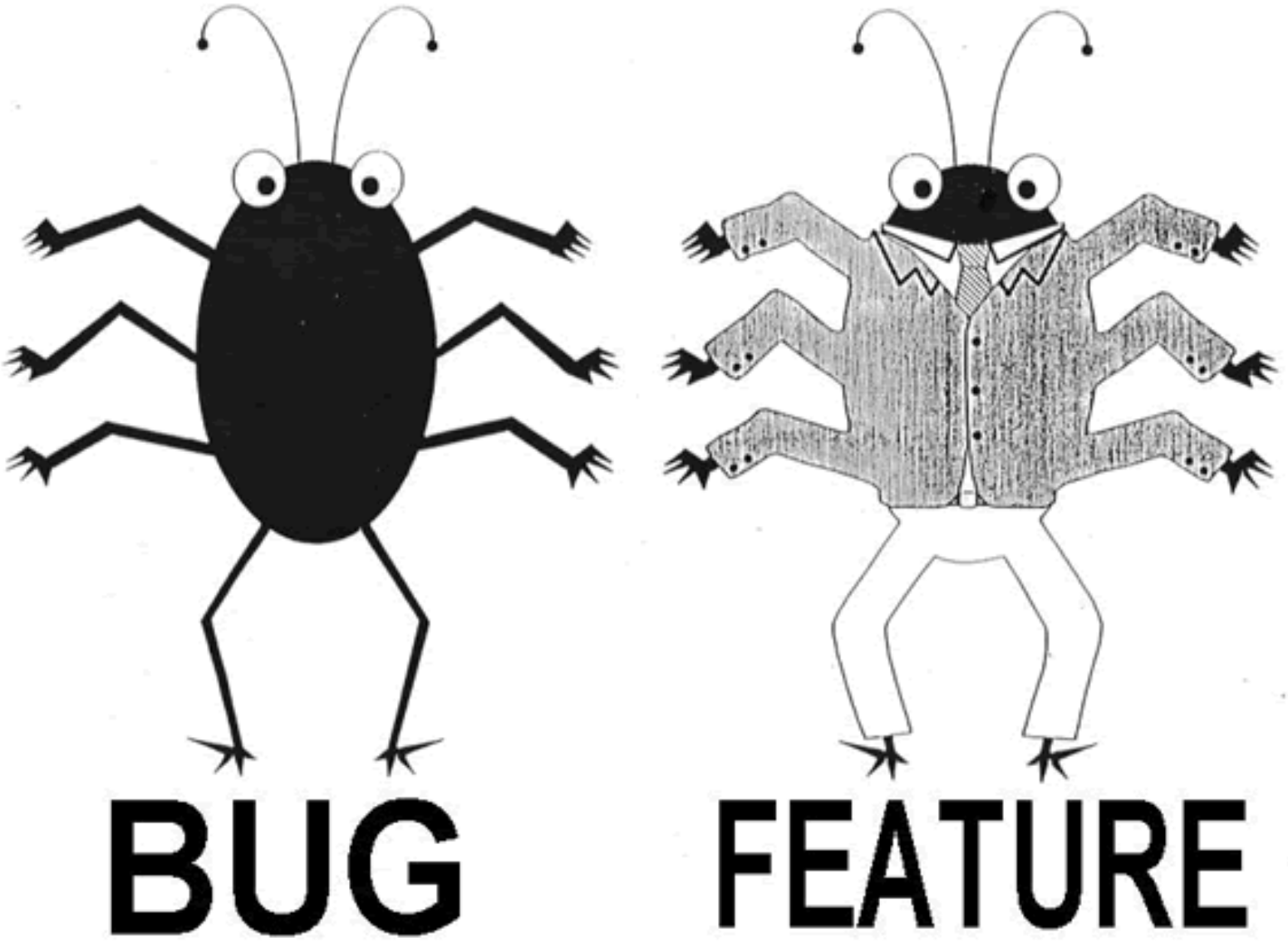
Ausgabe:

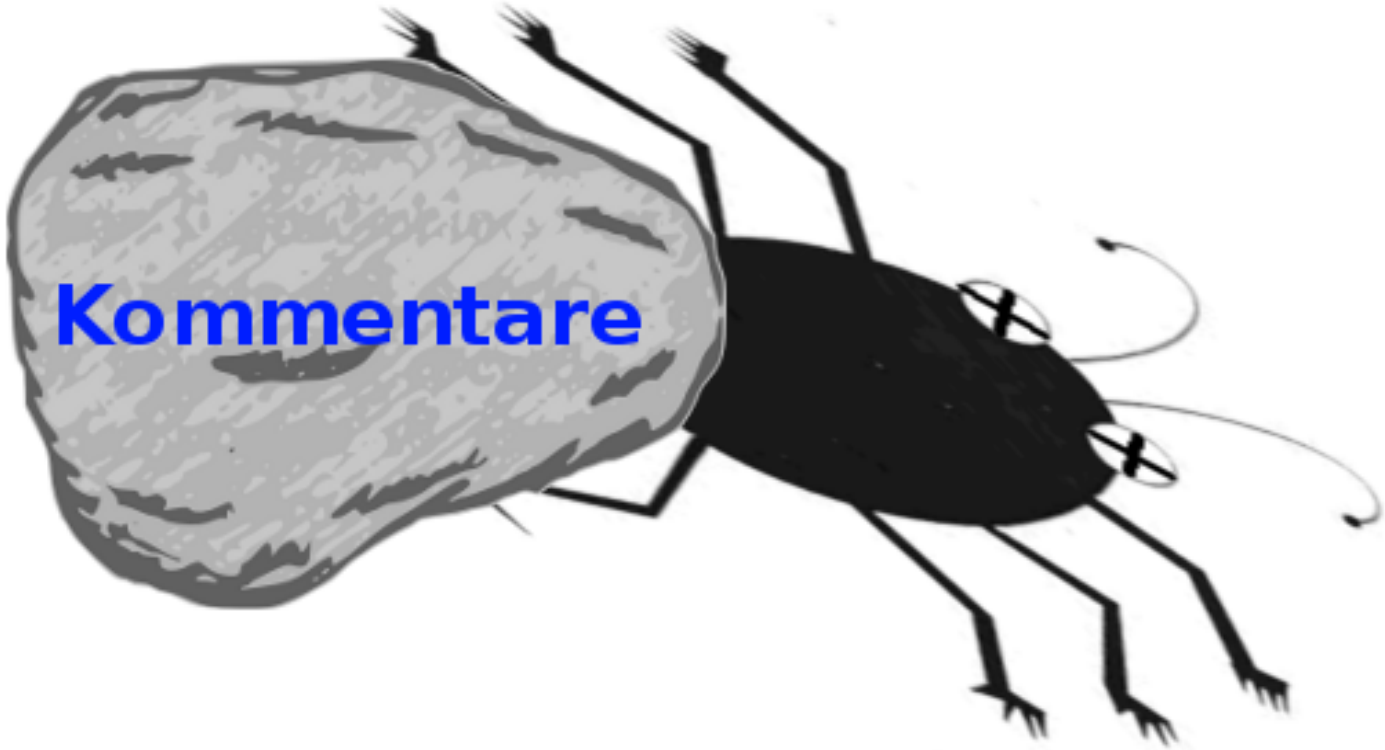
Die beliebtesten Stolpersteine in C

weitere findet ihr hier:
http://literateprogramming.com/ctraps.pdf



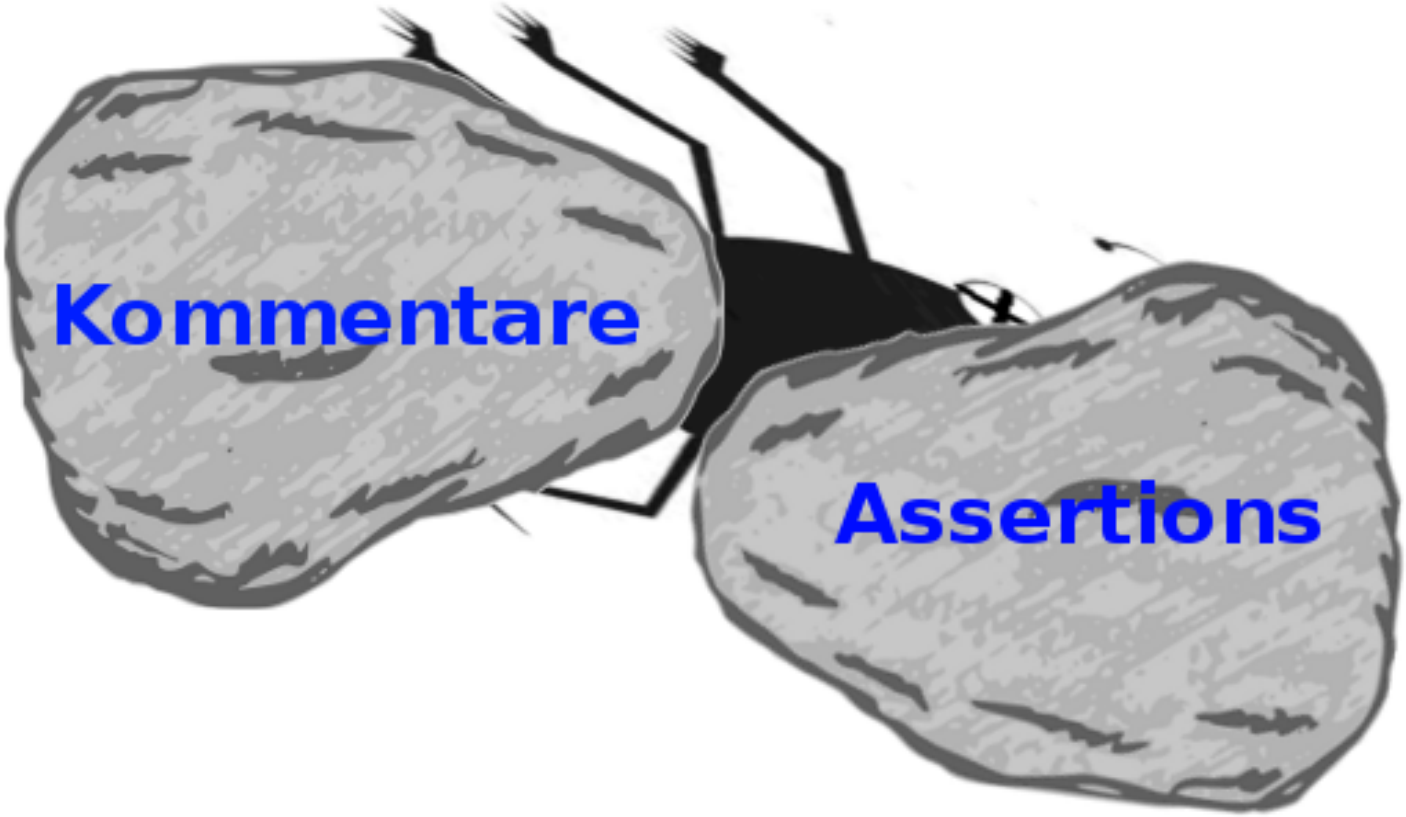
Bugs beseitigen




Kommentare

Kommentare - Beispiel

```
1  /**
2   * Millisecond granular sleep function.
3   *
4   * @returns IPRT status code.
5   * @retval  VINF_SUCCESS on success.
6   * @retval  VERR_INTERRUPTED if a signal or other
7   *          asynchronous stuff happend
8   *          which interrupt the peaceful sleep.
9   * @param   cMillies    Number of milliseconds to sleep.
10  *                      0 milliseconds means yielding the
11  *                      timeslice - deprecated!
12  * @remark  See RTThreadNanoSleep() for sleeping for
13  *          smaller periods of time.
14  */
15 int RTThreadSleep(RTMSINTERVAL cMillies);
```


Kommentare
Assertions

Assertions - Beispiel

Beispiel: assert

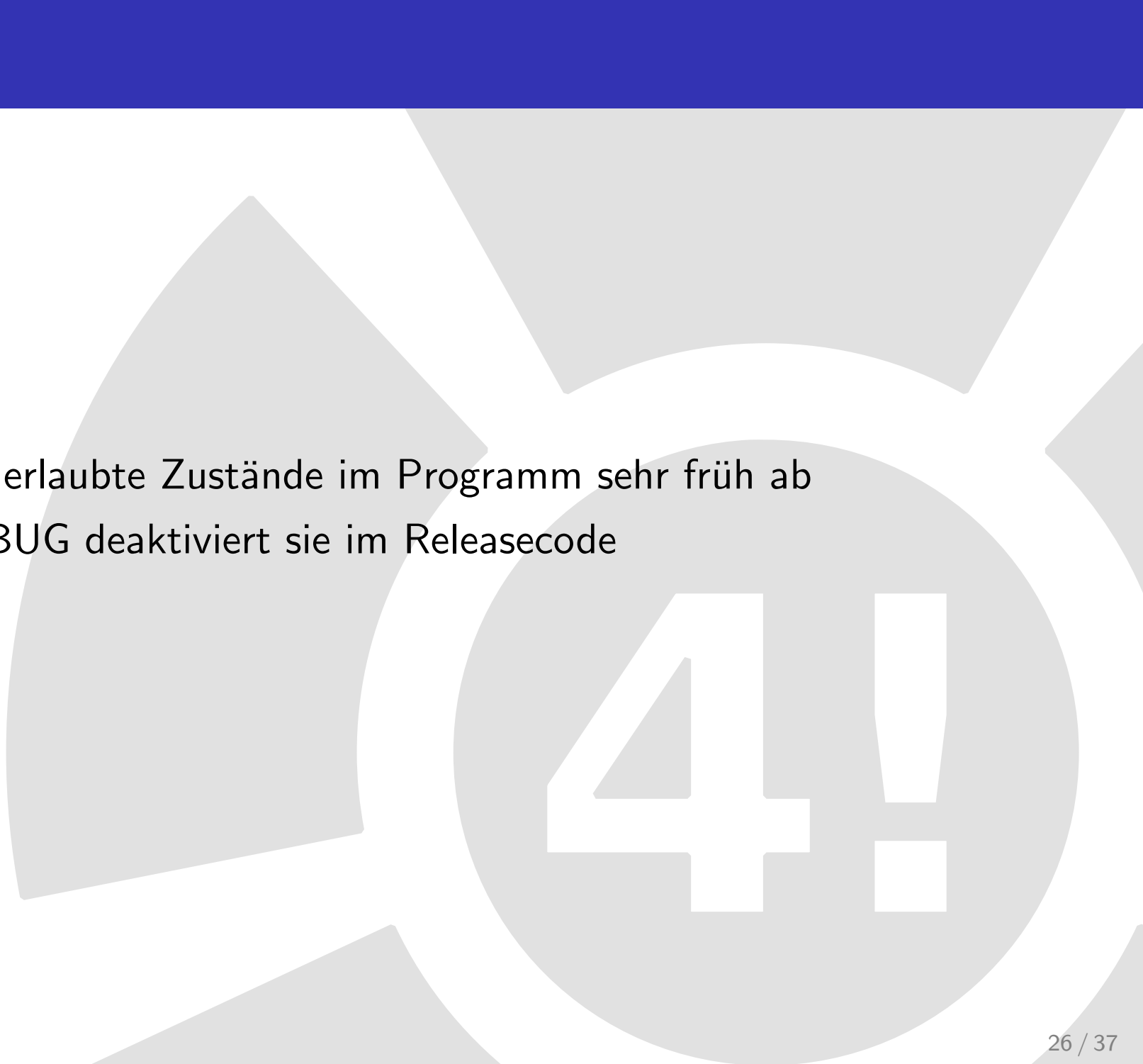
```
1  #include <assert.h>
2
3  int strlen(const char *pszStr) {
4      assert(pszStr != NULL);
5      ...
6  }
7
8  void main(void) {
9      strlen(NULL);
10 }
```

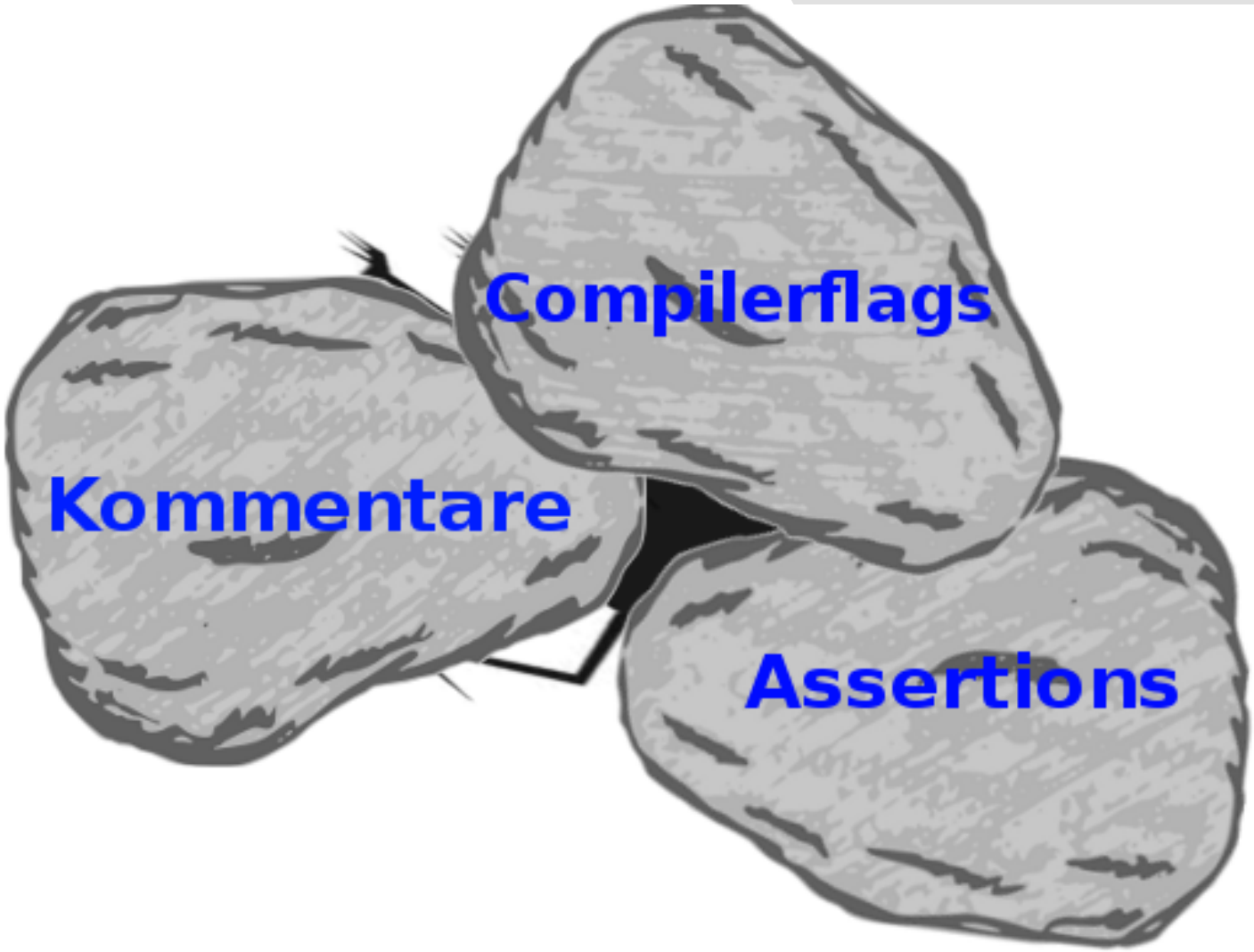
Ausgabe:

```
1  test_assert: test_assert.c:6: my_strlen: Assertion
2  pszStr != ((void *)0) failed.
3  Aborted
```

Assertions

- fängt unerlaubte Zustände im Programm sehr früh ab
- -DNDEBUG deaktiviert sie im Releasecode




Compilerflags
Kommentare
Assertions

Compiler flags

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    int a, b;

    a = b;
    if (a = 42);
        printf("Answer to the Ultimate Question of Life ,"
               "the Universe , and Everything!\n");
}
```

Compiler flags

Ausgabe:

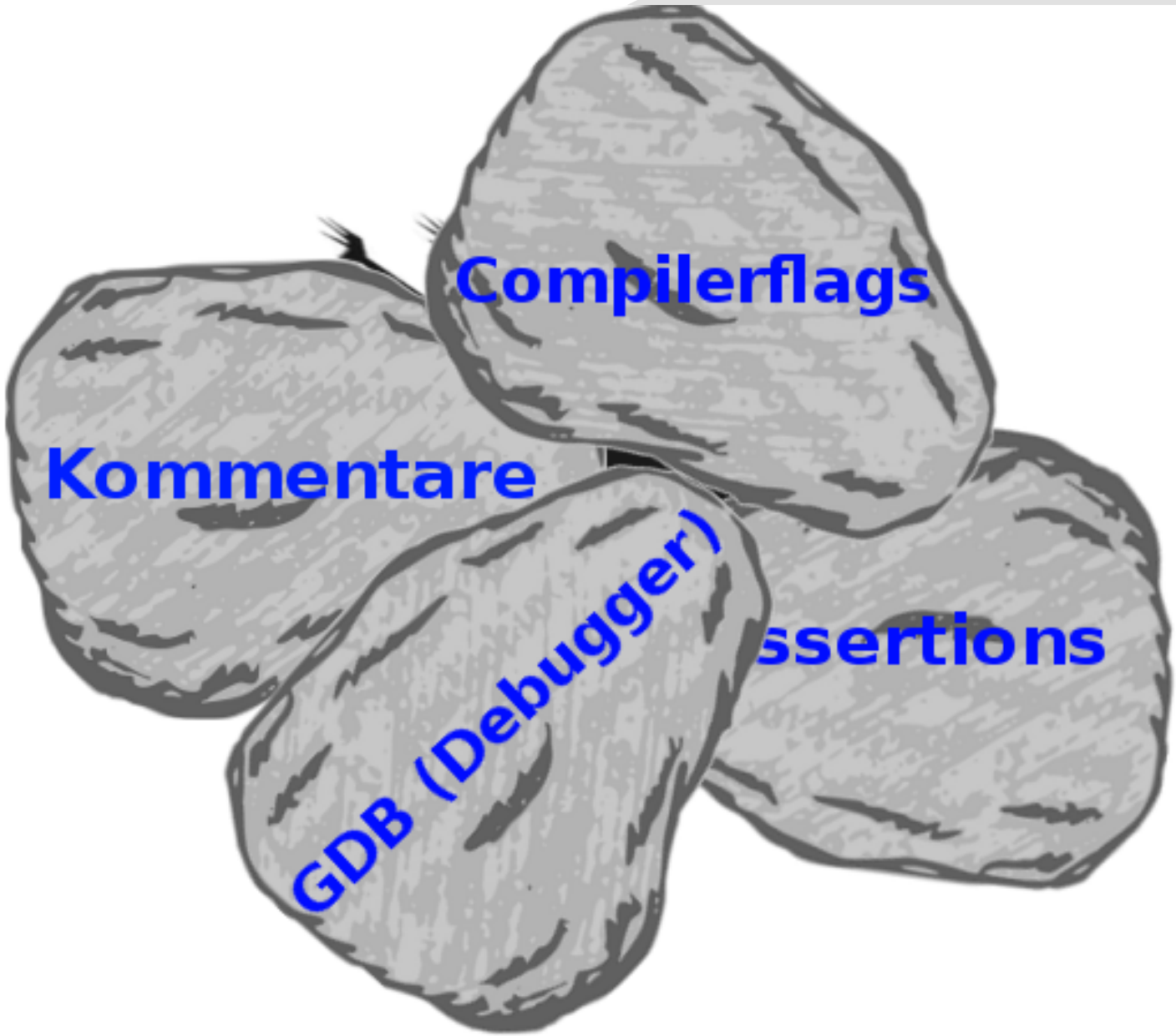
```
1 #gcc -o warnings warnings.c
2 #
```

Compiler flags

Ausgabe:

```
#gcc -Wall -Wextra -pedantic -o warnings warnings.c
warnings.c: In function ‘main’:
warnings.c:8: warning: suggest parentheses around assignment
                       used as truth value
warnings.c:8: warning: suggest braces around empty body in
                       an ‘if’ statement
warnings.c:3: warning: unused parameter ‘argc’
warnings.c:3: warning: unused parameter ‘argv’
warnings.c:11: warning: control reaches end of non-void
                        function
warnings.c:7: warning: ‘b’ is used uninitialized in this
                       function
```

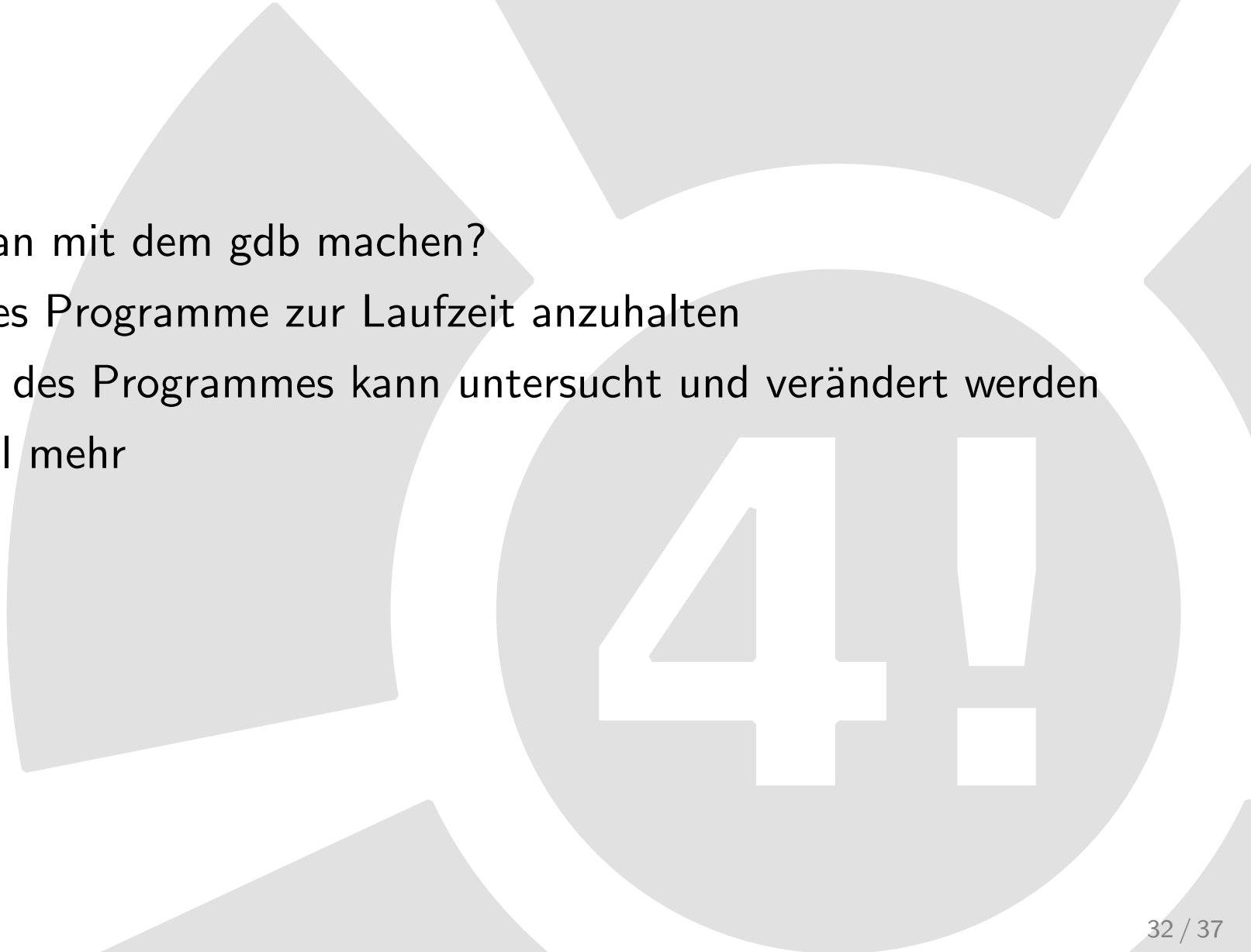
Dokumentation zu den Optionen:
http://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html


Compilerflags
Kommentare
GDB (Debugger)
ssertions

GDB

Was kann man mit dem gdb machen?

- erlaubt es Programme zur Laufzeit anzuhalten
- Zustand des Programmes kann untersucht und verändert werden
- noch viel mehr



Vorbereitung

GDB

Vorbereitung
Die richtigen Compilerflags:

```
gcc -g -O0 -o prog prog.c
```

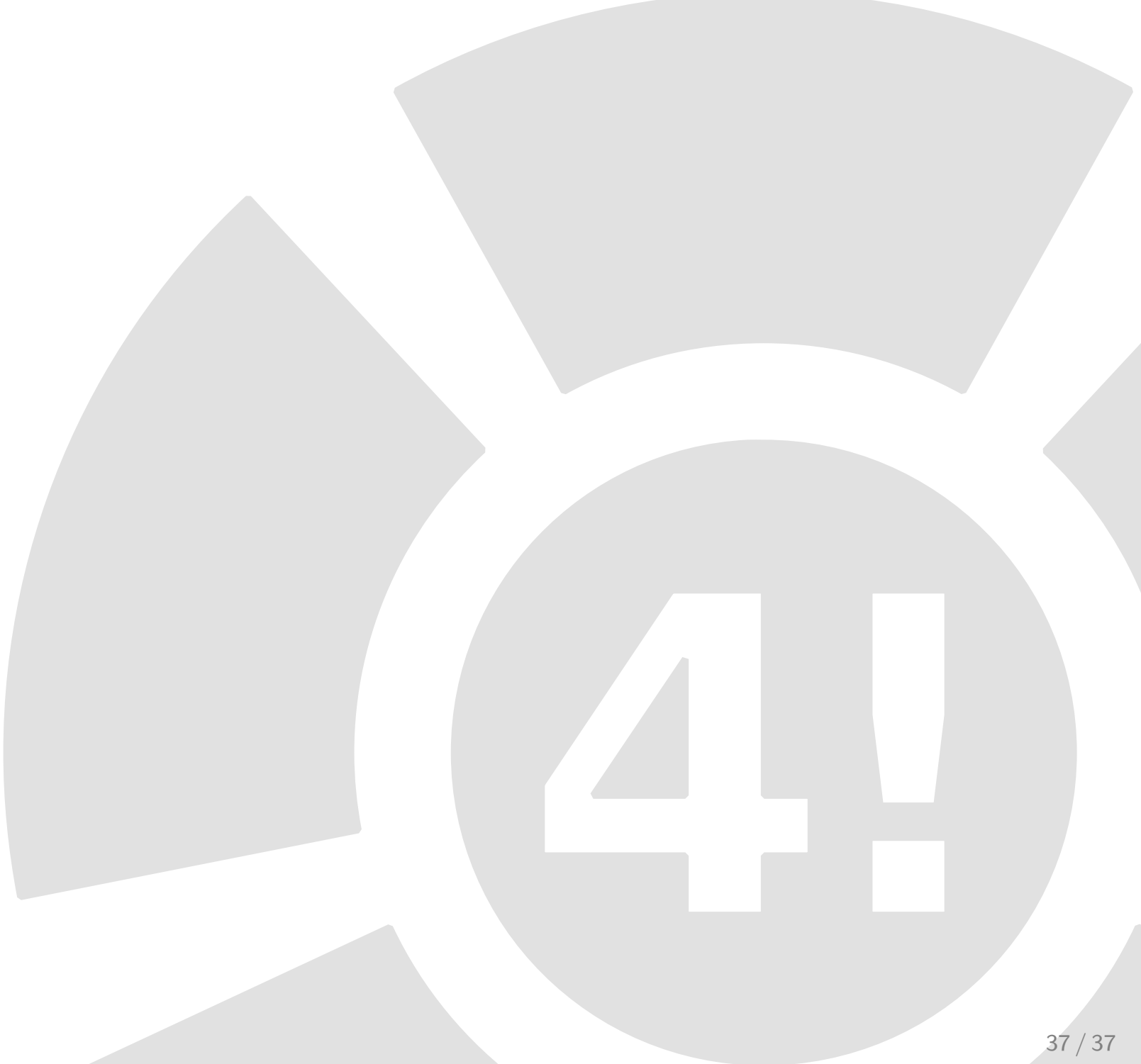
Inhaltsverzeichniss

Danke

Fragen?

Liste einiger gdb Kommandos

run	Ausführen eines Programms
quit	gdb beenden
kill	Programm beenden
cont	Fortfahren mit der Ausführung eines Programms
bt	Stacktrace
frame nr	Zum Frame im Stacktrace wechseln
list	Quellcode anzeigen
print	Variableninhalte anzeigen
printf	print formatted
break	Breakpoint setzen
condition	Bedingung für Breakpoint setzen
delete	Breakpoint löschen
next	Zeile im Quellcode ausführen (überspringt Funktionen)
step	Zeile im Quellcode ausführen (geht in Funktionen rein)
set print pretty on	übersichtliches Ausgeben von struct


4!