Debugging

C - Kurs 2009

Alexander Eichner

www.freitagsrunde.org

21. September 2009

This work is licensed under the *Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License*.

Inhaltsverzeichniss

1. Wiederholung
2. Ein wenig Theorie
3. Praxis

Inhaltsverzeichniss

1. Wiederholung
2. Ein wenig Theorie
3. Praxis

Was habt ihr gelernt

- (Fast) Alles was man zum Schreiben eines Programms in C wissen muss

Was habt ihr gelernt

- (Fast) Alles was man zum Schreiben eines Programms in C wissen muss - theoretisch

Was habt ihr gelernt

- (Fast) Alles was man zum Schreiben eines Programms in C wissen muss - theoretisch
- 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

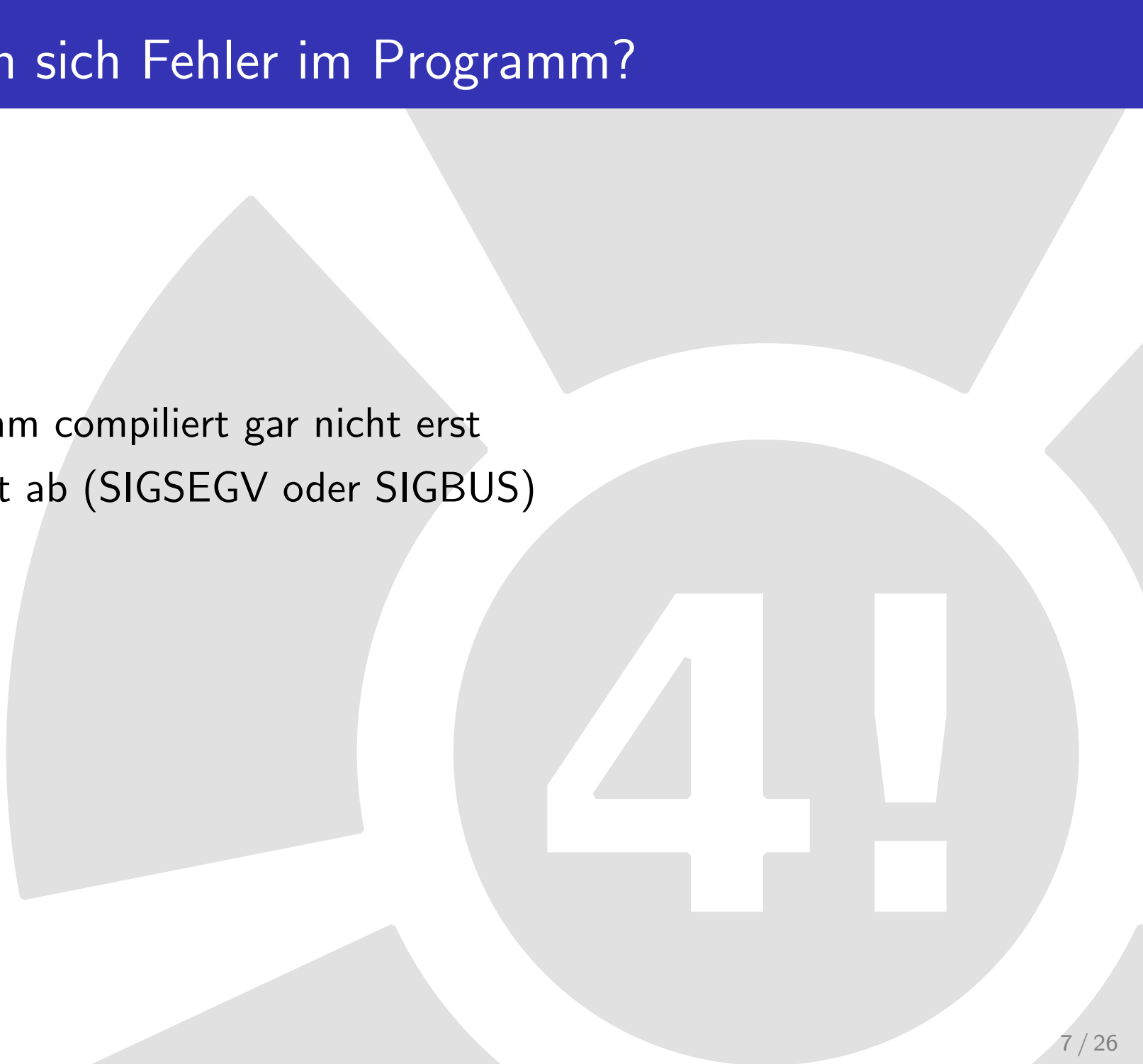
1. Wiederholung
2. Ein wenig Theorie
3. Praxis

Wie äußern sich Fehler im Programm?

- Programm compiliert gar nicht erst

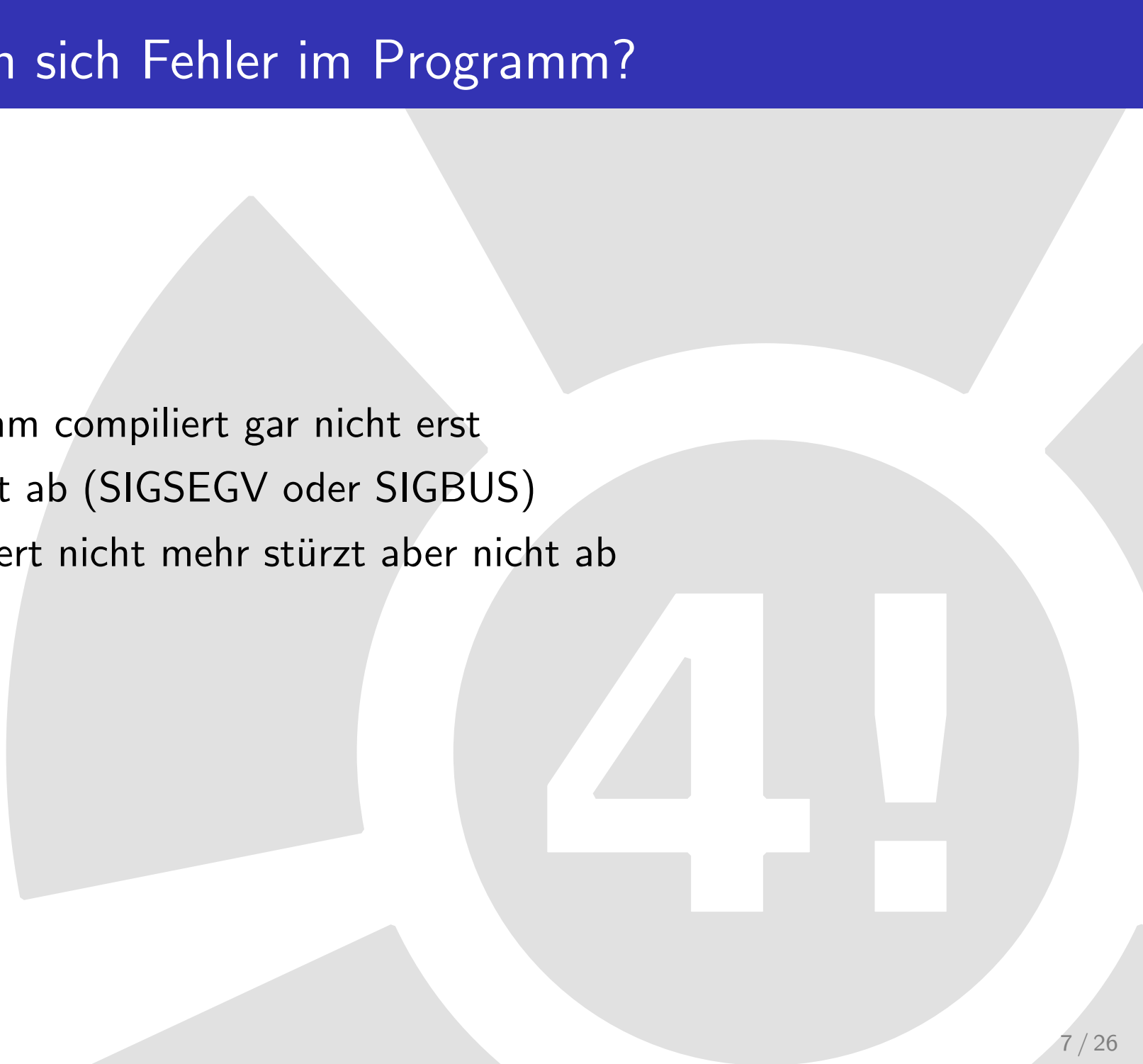
Wie äußern sich Fehler im Programm?

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



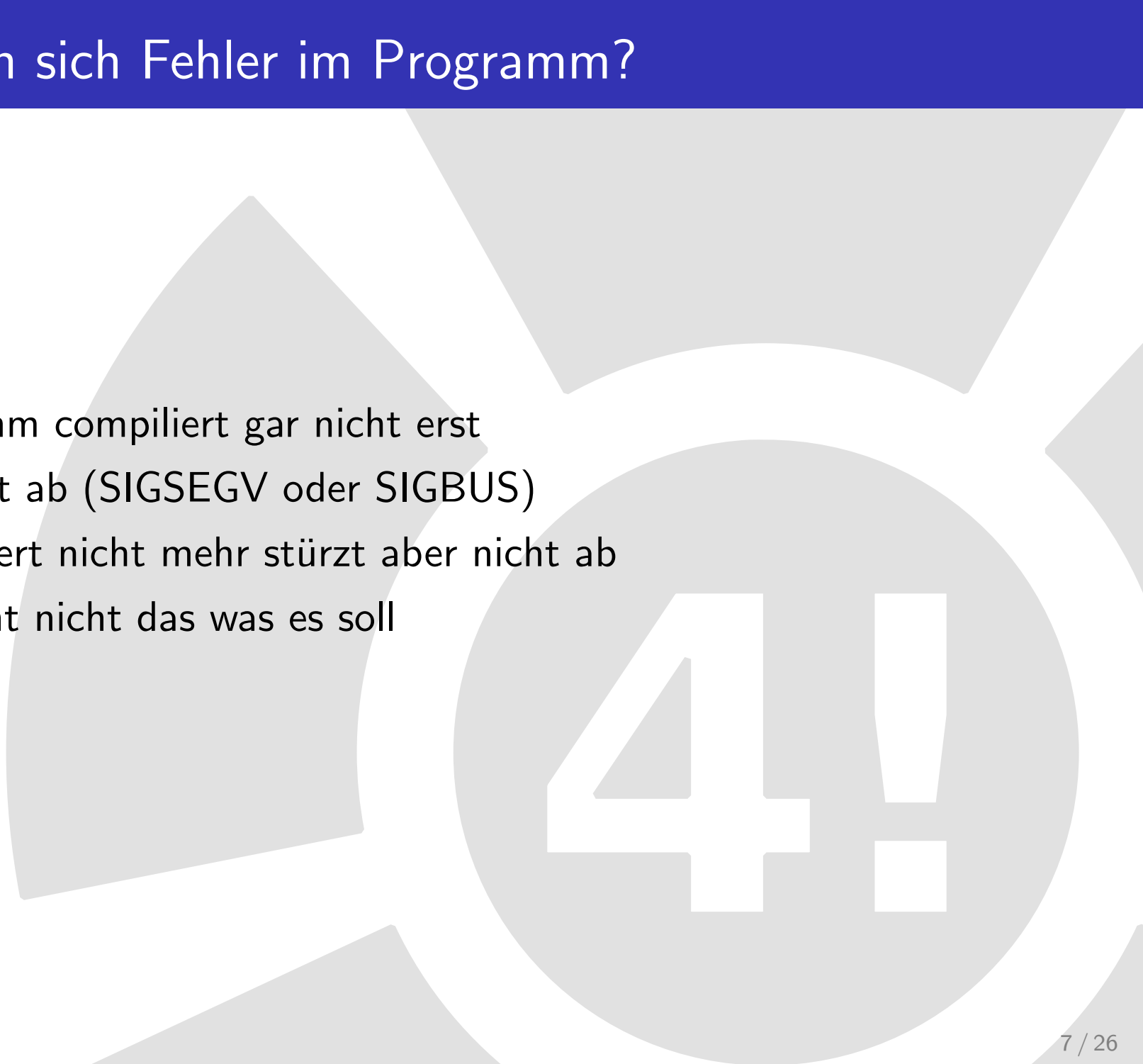
Wie äußern sich Fehler im Programm?

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



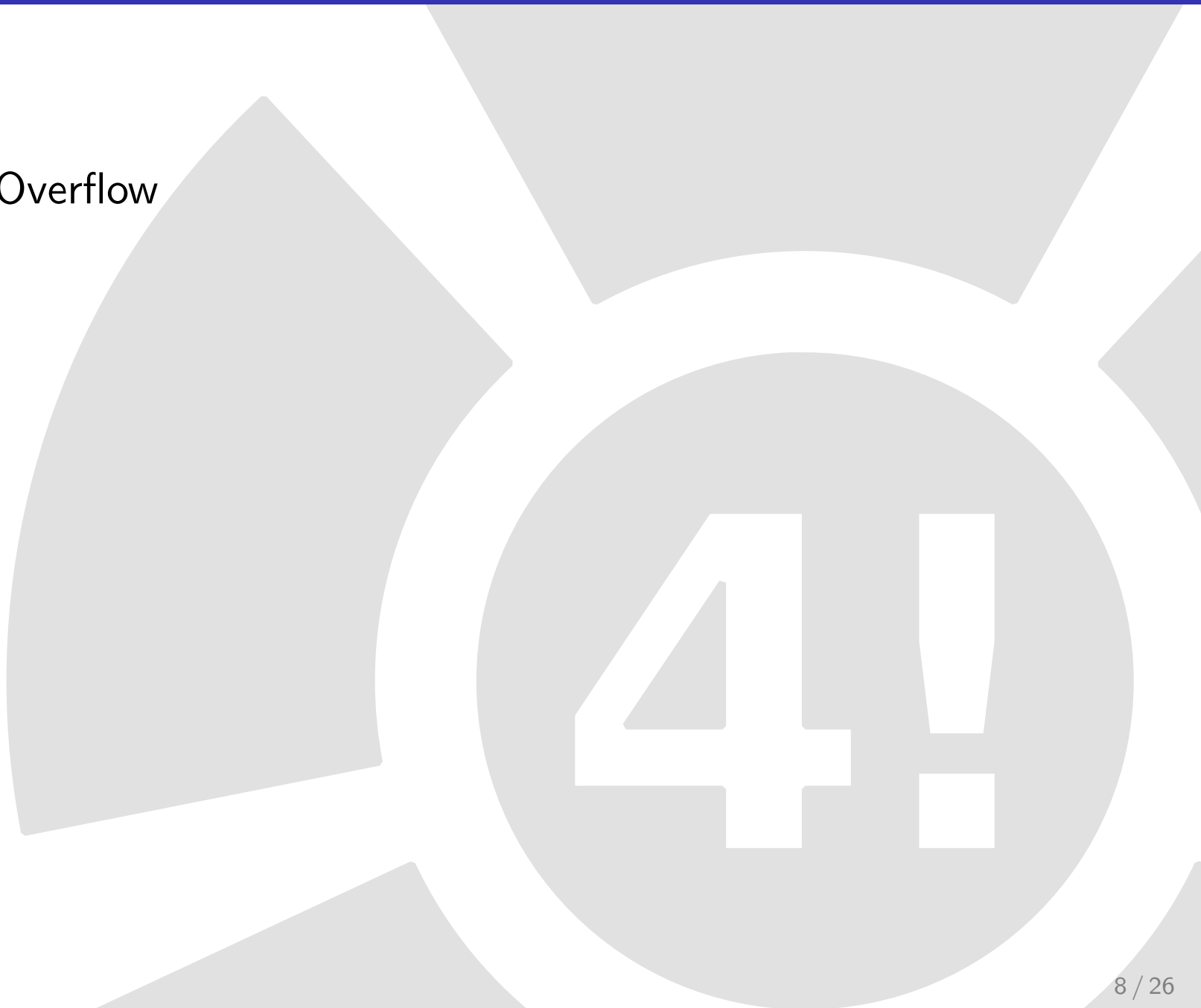
Wie äußern sich Fehler im Programm?

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



Arten von Bugs

Heap/Stack Overflow



Arten von Bugs

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 (core dumped)
```

Arten von Bugs

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 (core dumped)
```

Arten von Bugs

Unitialisierte Variablen

Beispiel

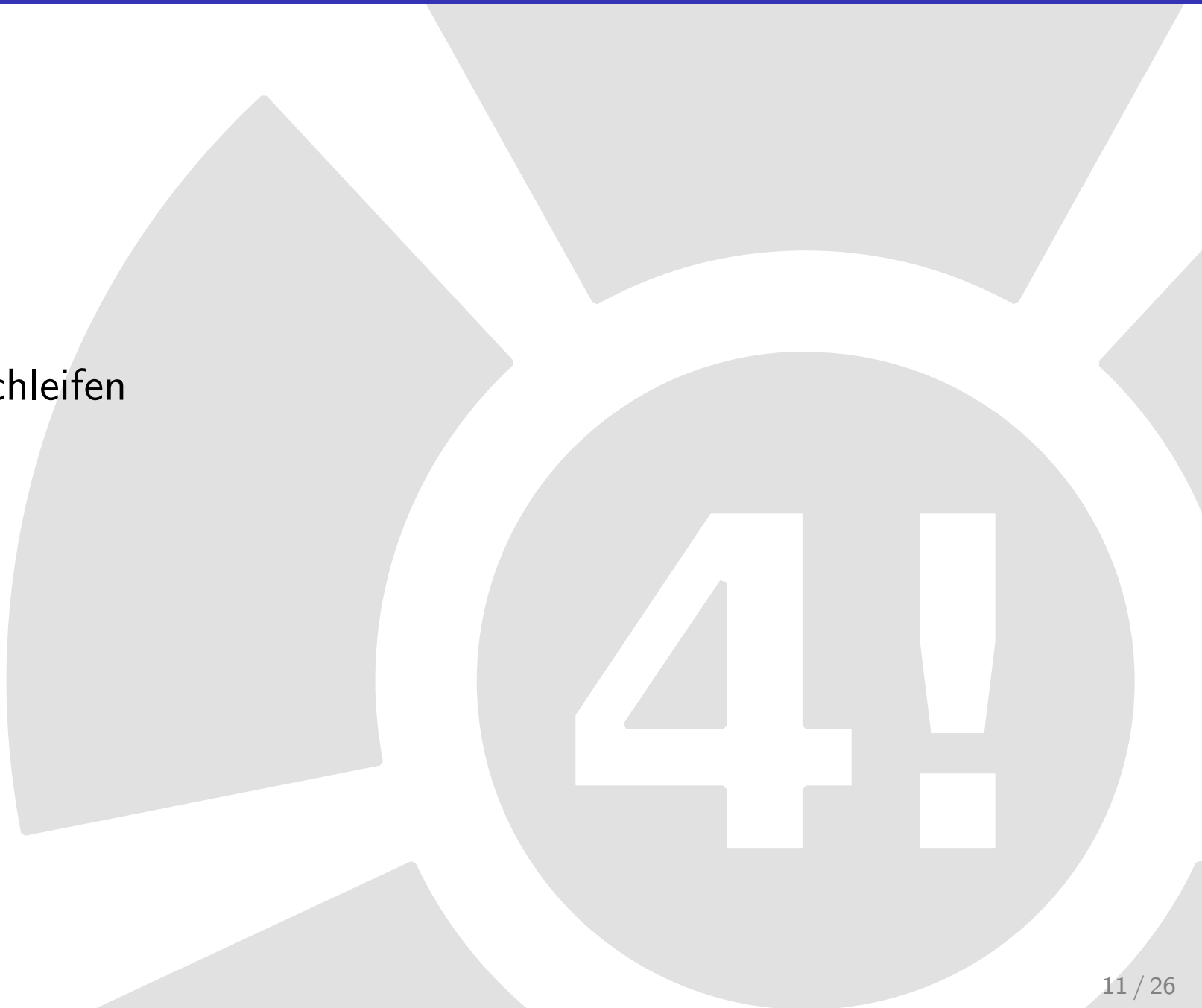
```
1 int iWert;
2 printf("wert = %d\n", iWert);
```

Ausgabe:

```
1 wert = undefiniert
```

Arten von Bugs

- Endlosschleifen

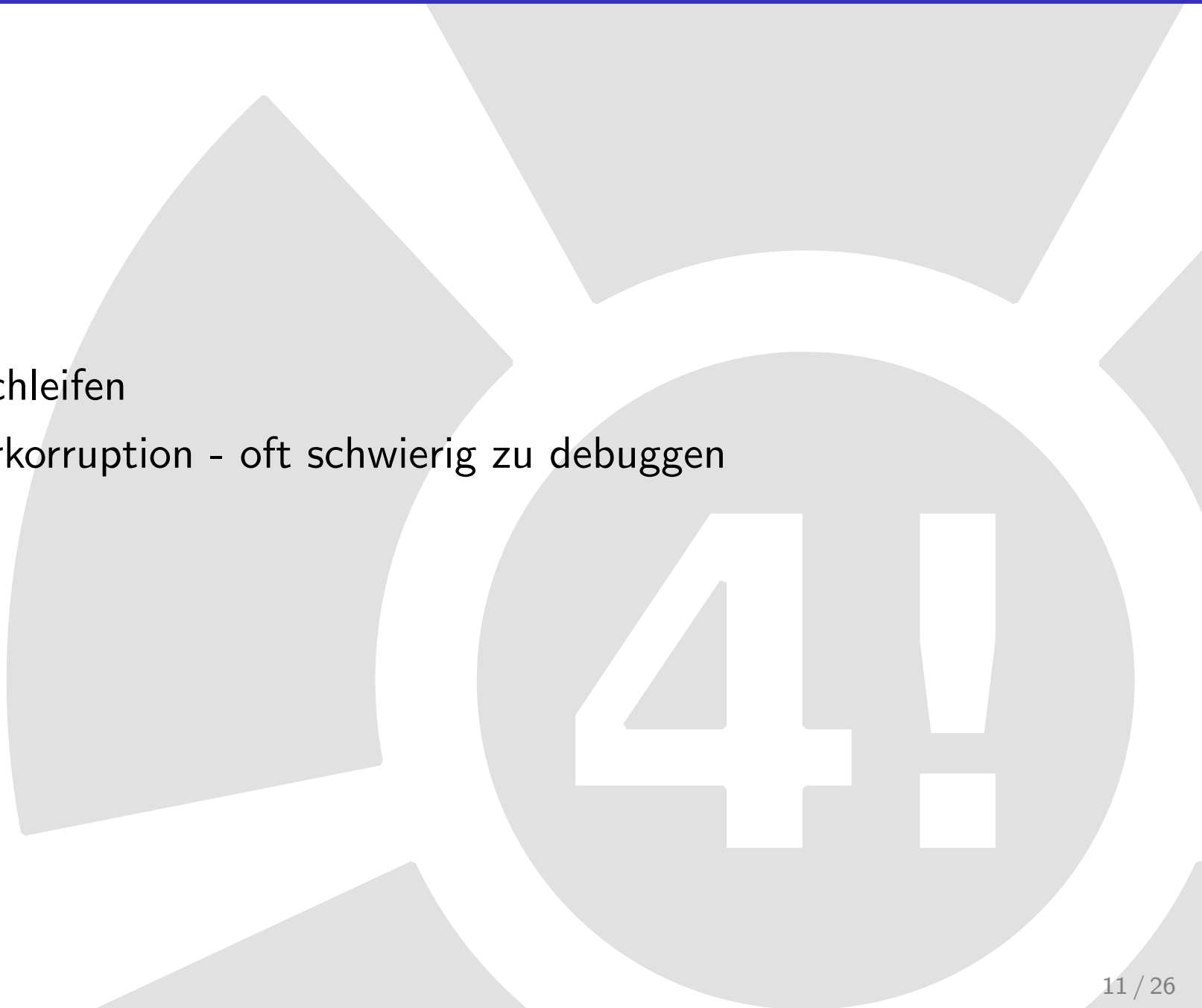


Arten von Bugs

- Endlosschleifen
- Speicherkorruption

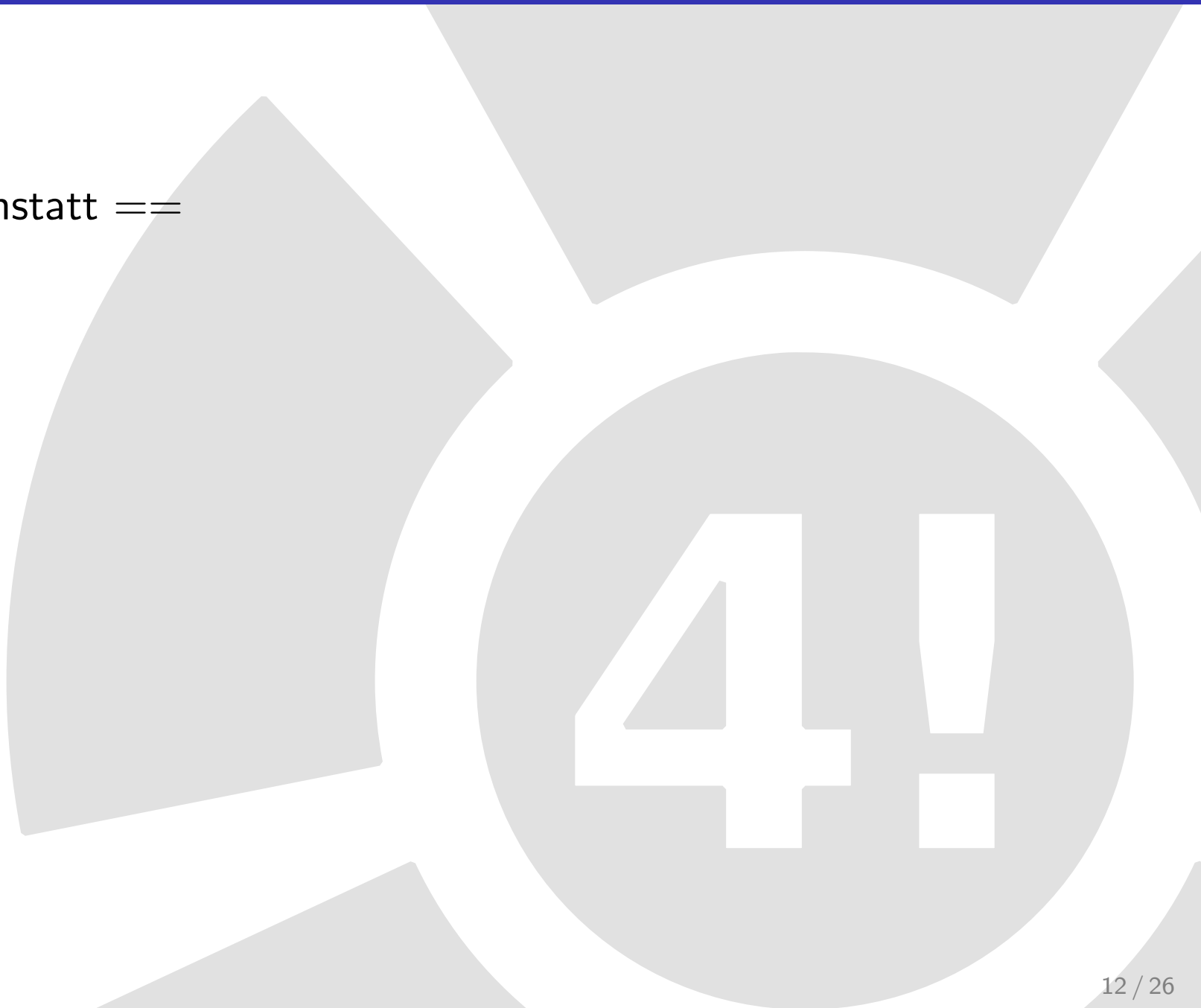
Arten von Bugs

- Endlosschleifen
- Speicherkorruption - oft schwierig zu debuggen



Hitliste der beliebtesten Stolpersteine in C

Platz 1: = anstatt ==



Hitliste der beliebtesten Stolpersteine in C

Platz 1: = anstatt ==

Beispiel

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

Ausgabe:

```
1         iWert ist 3
```

Hitliste der beliebtesten Stolpersteine in C

Platz 2: ; an der falschen Stelle

Hitliste der 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
```

Hitliste der beliebtesten Stolpersteine in C

Beispiel: Endlosschleife

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

Hitliste der 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:

Hitliste der beliebtesten Stolpersteine in C

Platz 3: case ohne break in switch

Hitliste der beliebtesten Stolpersteine in C

Platz 3: case ohne break in switch

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  }
```

Hitliste der beliebtesten Stolpersteine in C

Platz 3: case ohne break in switch

Beispiel

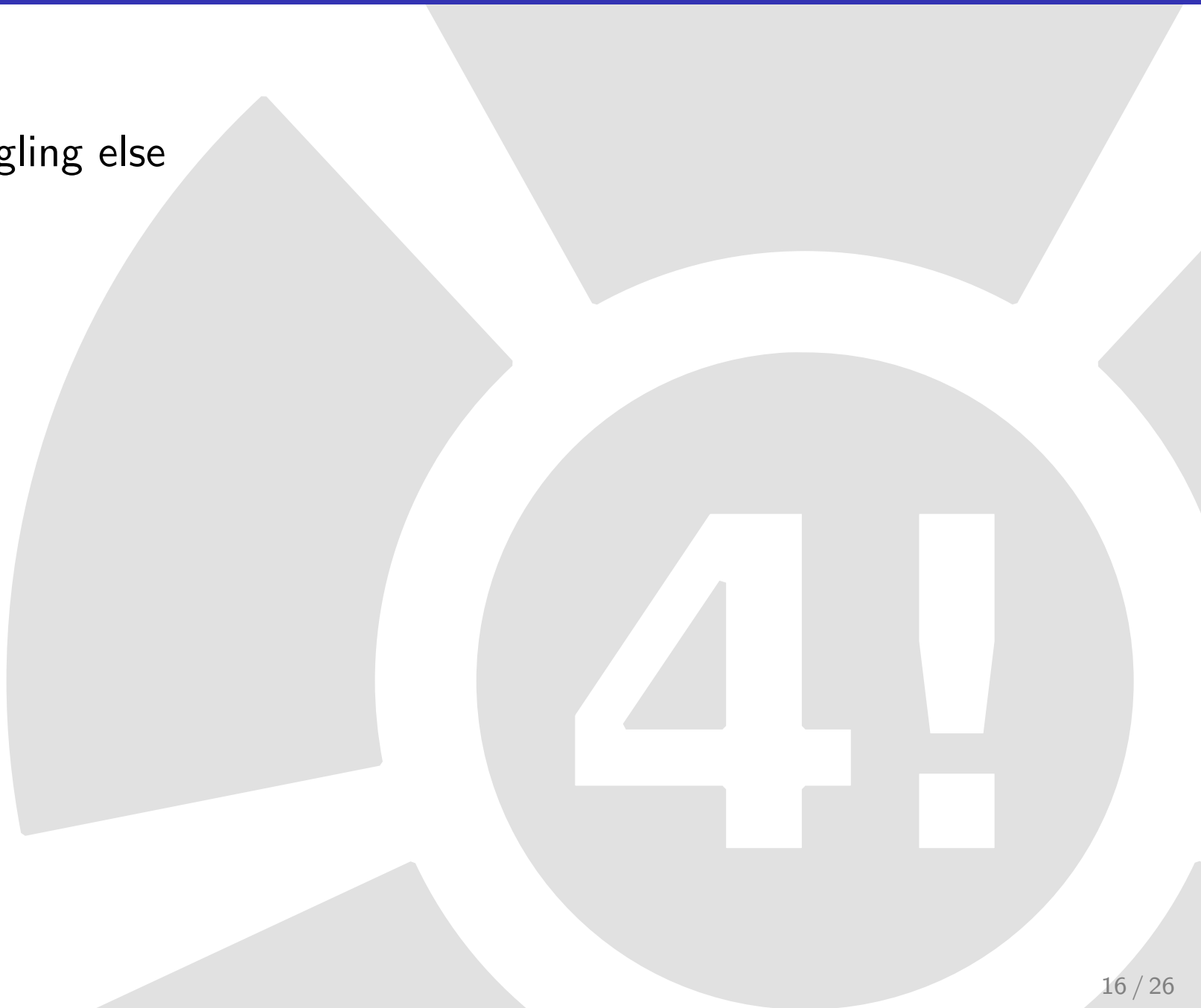
```
1   int iValue = 2;
2
3   switch (iValue)
4   {
5       case 1: printf("iValue = 1\n");
6       case 2: printf("iValue = 2\n");
7       case 3: printf("iValue = 3\n");
8       default:
9           printf("iValue = %d\n", iValue);
10  }
```

Ausgabe:

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

Hitliste der beliebtesten Stolpersteine in C

Platz 4: Dangling else



Hitliste der beliebtesten Stolpersteine in C

Platz 4: Dangling else

Beispiel: Wo gehört das else hin?

```
1 int iValue = 2;
2 int iVal2  = 1;
3
4 if (iValue == 1)
5     if (iVal2 == 1)
6         printf("Both 1\n");
7 else
8     printf("iValue=%d\n", iValue);
```

Hitliste der beliebtesten Stolpersteine in C

Platz 4: Dangling else

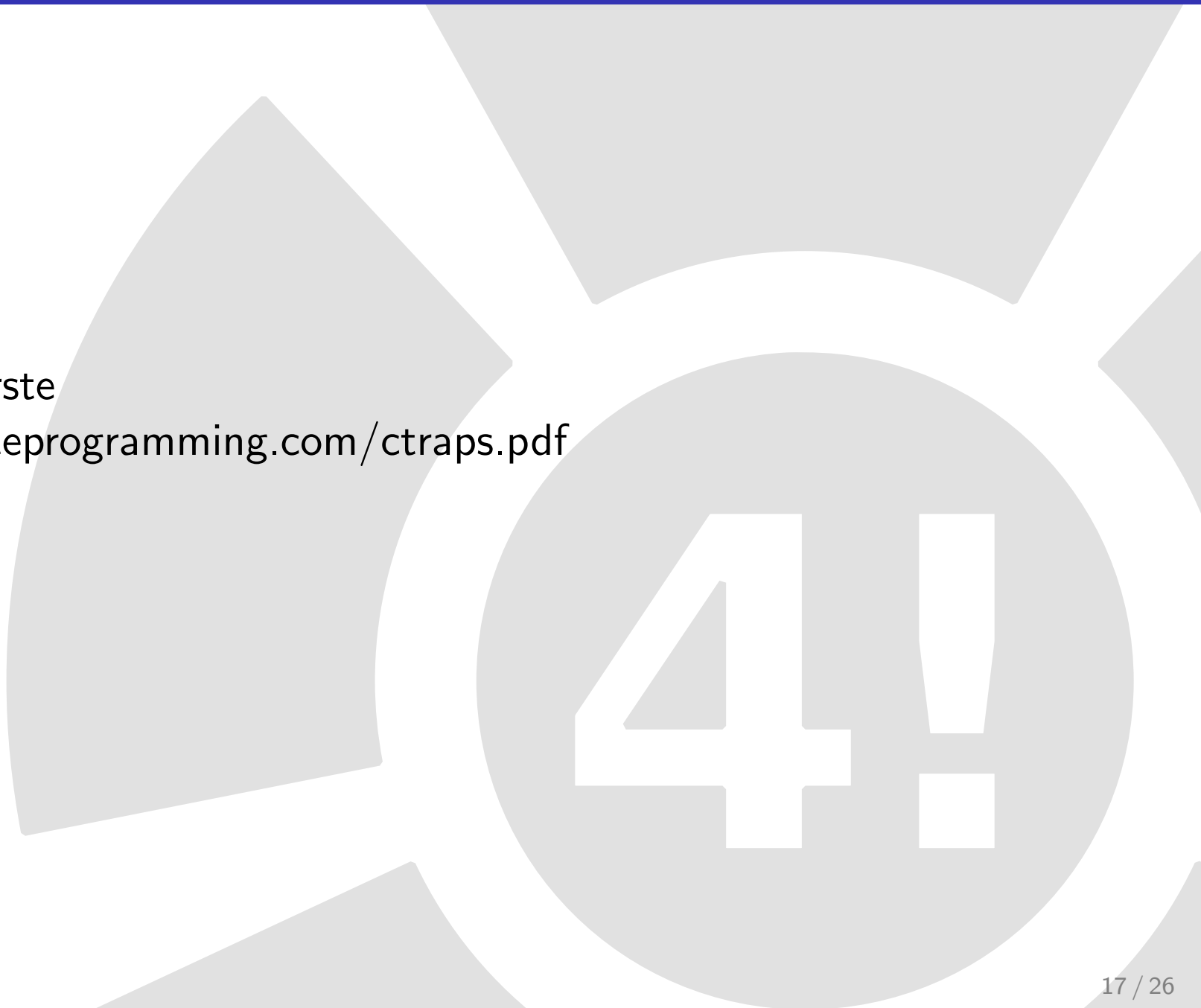
Beispiel: Wo gehört das else hin?

```
1 int iValue = 2;
2 int iVal2  = 1;
3
4 if (iValue == 1)
5     if (iVal2 == 1)
6         printf("Both 1\n");
7 else
8     printf("iValue=%d\n", iValue);
```

Ausgabe:

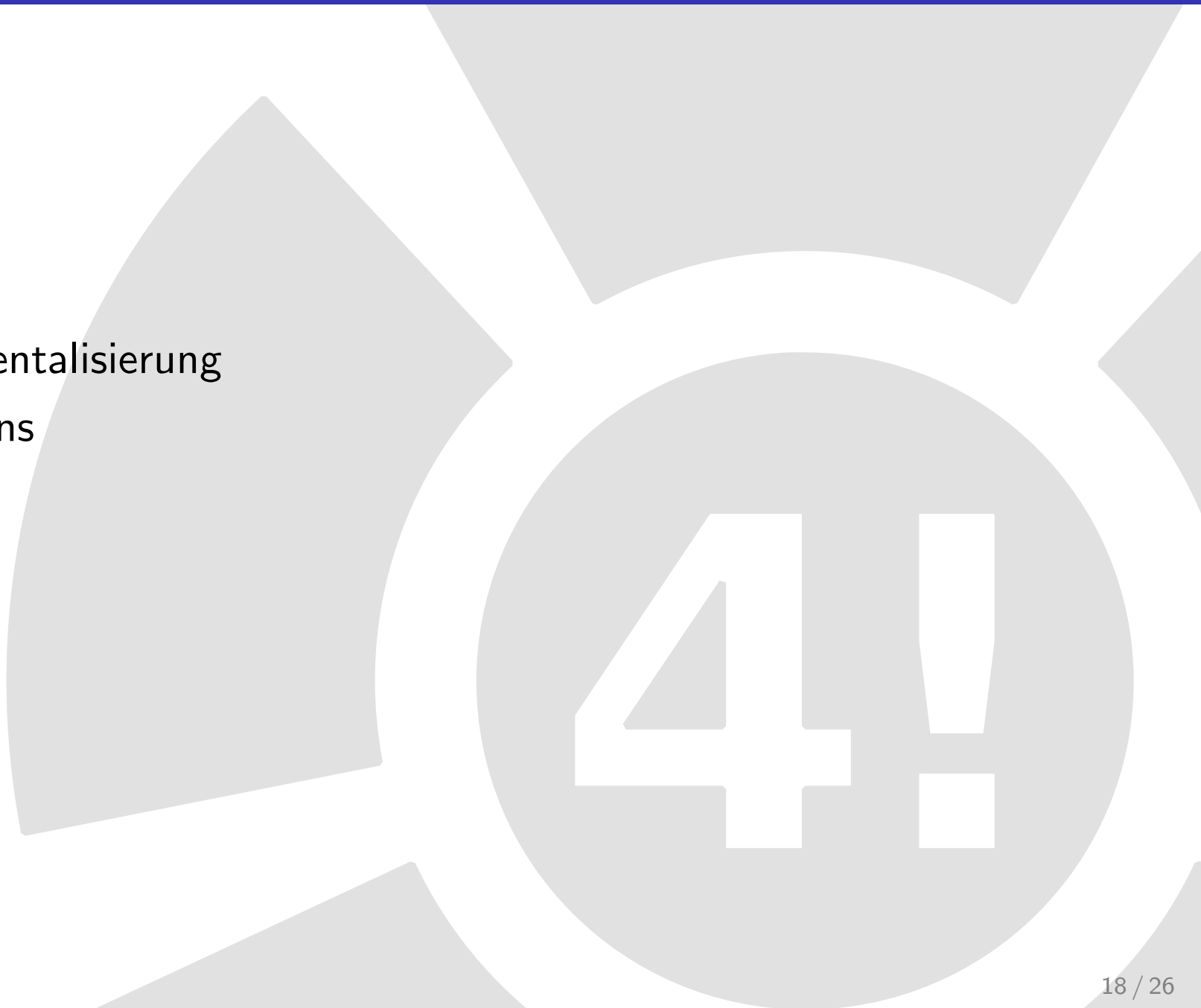
Hitliste der beliebtesten Stolpersteine in C

reicht fürs Erste
http://literateprogramming.com/ctraps.pdf



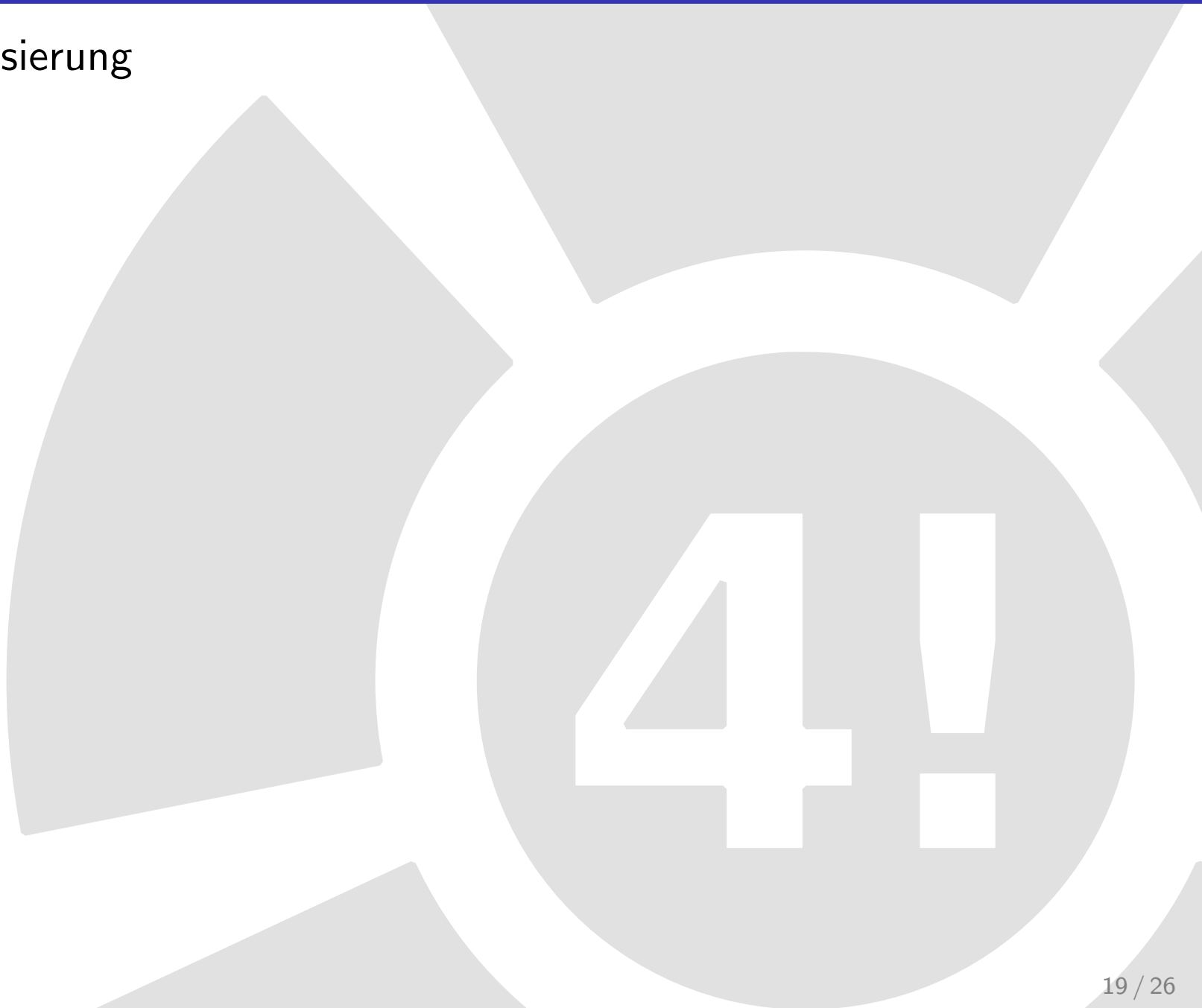
Und wie debugge ich jetzt?

- Instrumentalisierung
- Assertions
- gdb



Arten des Debugging

Instrumentalisierung



Arten des Debugging

Instrumentalisierung

Beispiel: printf

```
1  int iValue = 2
2
3  printf("iValue is %d now\n", iValue);
4  if (iValue == 2) {
5      printf("Branch 1 taken\n");
6      ...
7  } else {
8      printf("Branch 2 taken\n");
9      ...
10 }
```

Ausgabe:

```
1 iValue is 2 now
2 Branch 1 taken
```

Arten des Debugging - Assertions

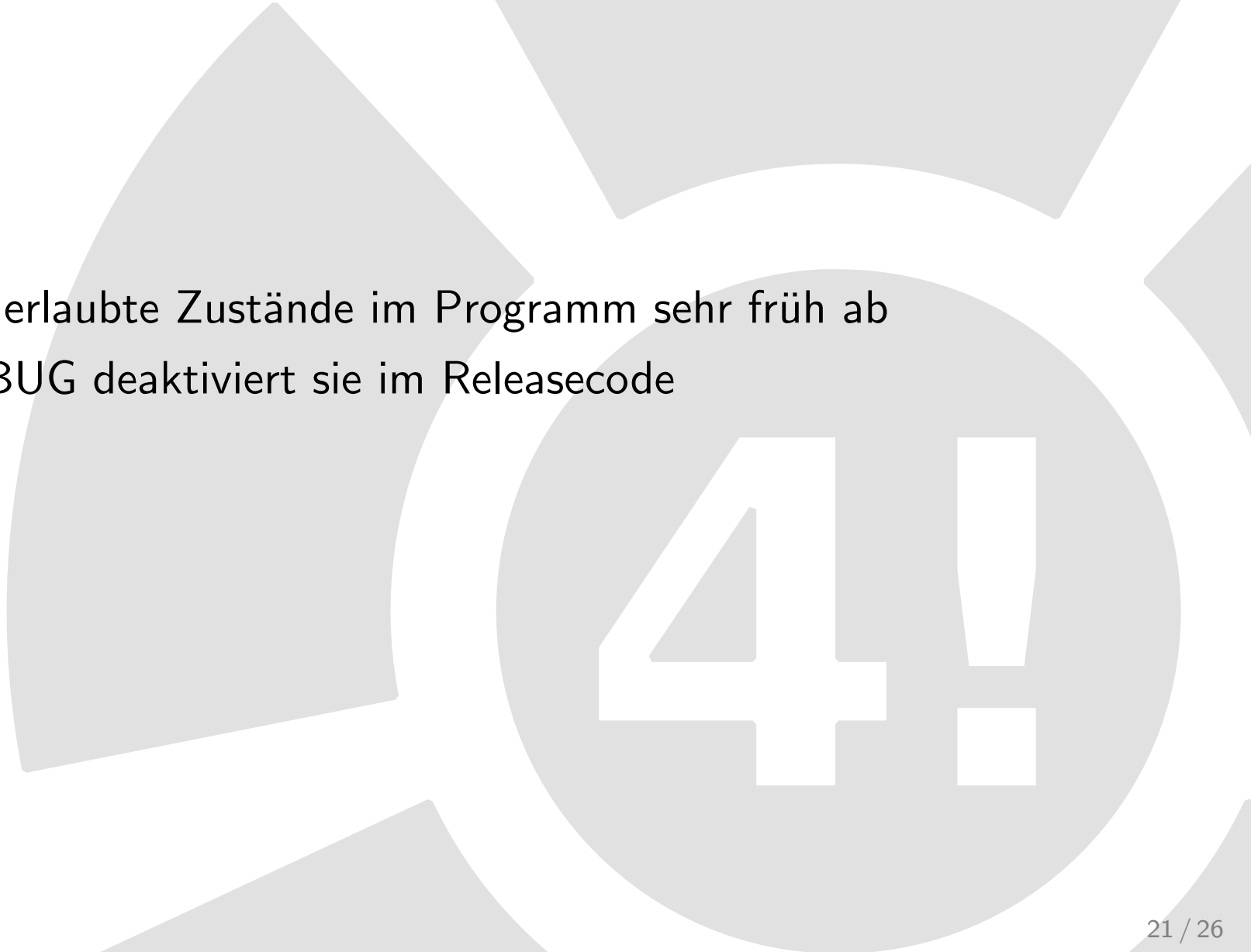
Assertions

Beispiel: assert

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

Arten des Debugging - Assertions

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



Arten des Debugging - 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

Arten des Debugging - gdb

Vorbereitung

Arten des Debugging - gdb

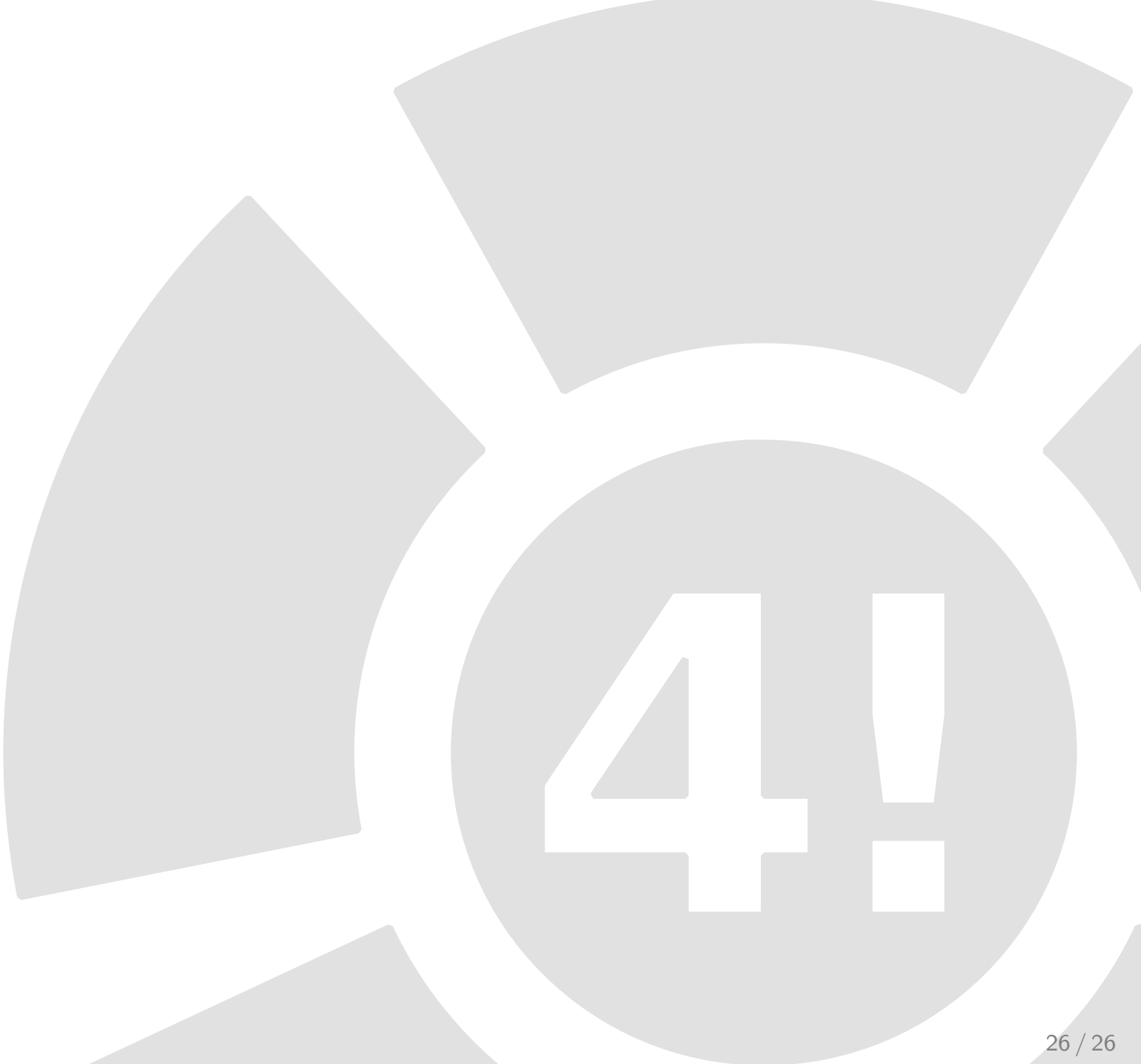
Vorbereitung
Die richtigen Compilerflags: gcc -g -O0 -o prog prog.c

Inhaltsverzeichniss

1. Wiederholung
2. Ein wenig Theorie
3. Praxis

Danke

Fragen?


4!