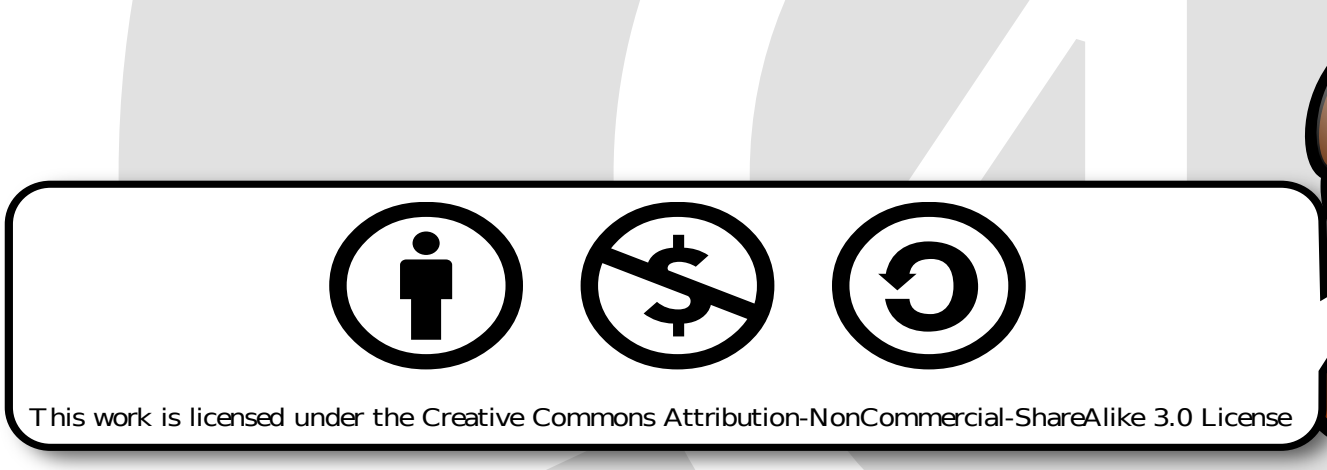
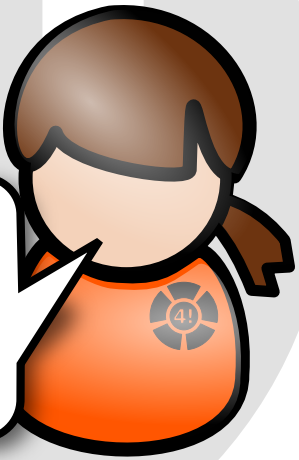
Ein- und Ausgabe

C - Kurs 2010

Mario Bodemann

15. September 2010


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

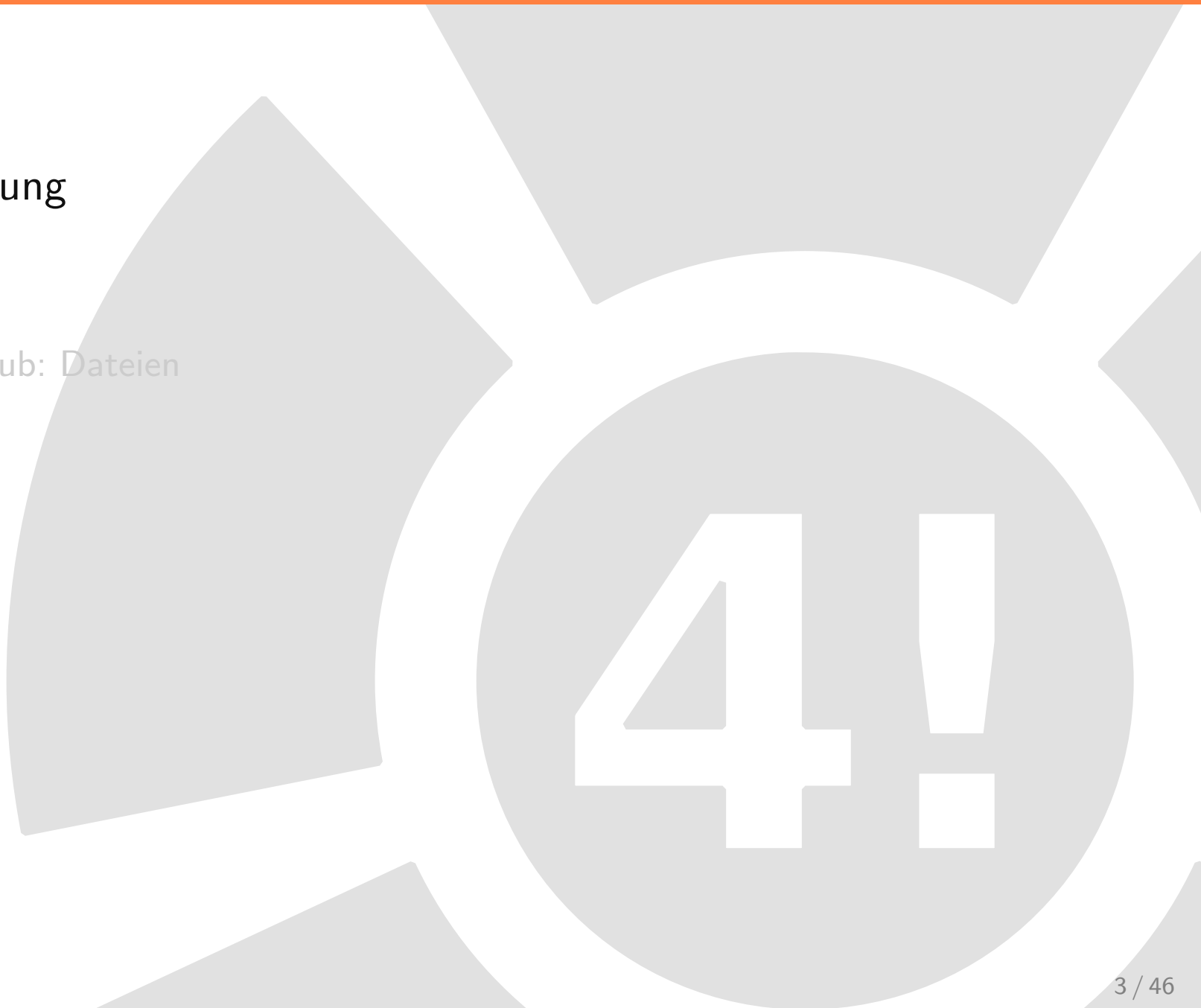


Inhaltsverzeichnis

1. Wiederholung
2. Ausgabe
   - Einschub: Dateien
3. Eingabe
4. Beispiel
5. Ausblick

Inhaltsverzeichnis

1. Wiederholung
2. Ausgabe
   - Einschub: Dateien
3. Eingabe
4. Beispiel
5. Ausblick



Wiederholung

- Was gelehrt wurde
  - Syntax von C
    - Typen
    - Operatoren
    - Kontrollfluss
    - Funktionen
  - Hello World

Wiederholung: Syntax

Listing 1: src/recap.c

```c
int a = 4;
int b = 4 + a;
int c = ++b + a++;
int d = (a == b ? c : --b + c);

if( 9 == d )
{
  d %= 8;
}
else
{
  d <<= 2;
}
```

Wiederholung: Hello World

Listing 2: src/hello.c

```c
#include <stdio.h>
int main(int argc, char **argv)
{
   printf("Hello World\n");
   return 0;
}
```

- Einfache Ausgabe
- Aber: Wo für steht das 'f' bei printf?

"␣" ist ein Leezeichen

Ziele dieser Vorlesung

- Lerne, wie du Variablen ausgeben kannst
- Lerne, wie diese Ausgaben formatiert werden
- Speichere und lade Variablen in/aus Dateien
- Alle Beispiele sind online[1] verfügbar

[1] www.freitagsrunde.org/Ckurs2010/Vortrag02

Inhaltsverzeichniss

1. Wiederholung
2. Ausgabe
   - Einschub: Dateien
3. Eingabe
4. Beispiel
5. Ausblick

Ausgabe

- Dient zur Ausgabe von Text und Variablen
- `printf` heißt die Funktion
- Das f steht für formatted, also für formatierte Ausgabe
- `printf` kann `int`, `float`, `double`, Strings und weitere Typen ausgeben

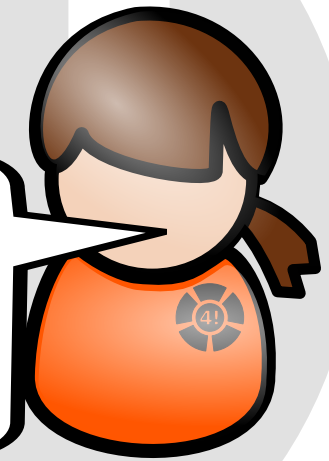
Beispiel *printf*

Beispiel: Ausgaben

Listing 3: src/output.c

```
printf("      Name | Anzahl |      Gewicht\n");
printf("  ------------+---------+-------------\n");
printf("  %s | %d | %f\n", "Samus", 12, 2.718 );
```

```
     Name | Anzahl |      Gewicht
----------+--------+-------------
Samus | 12 | 2.718000
```



Typen für printf

Symbol	Beschreibung	Besispielausgabe
%d	ganze Zahlen	3
%x	ganze Zahlen in Hex	0xD0DAAFFE
%f	gebrochene Zahlen	3.1415927
%s	Strings	HalloWelt
%%	Ausgabe eines %	100%

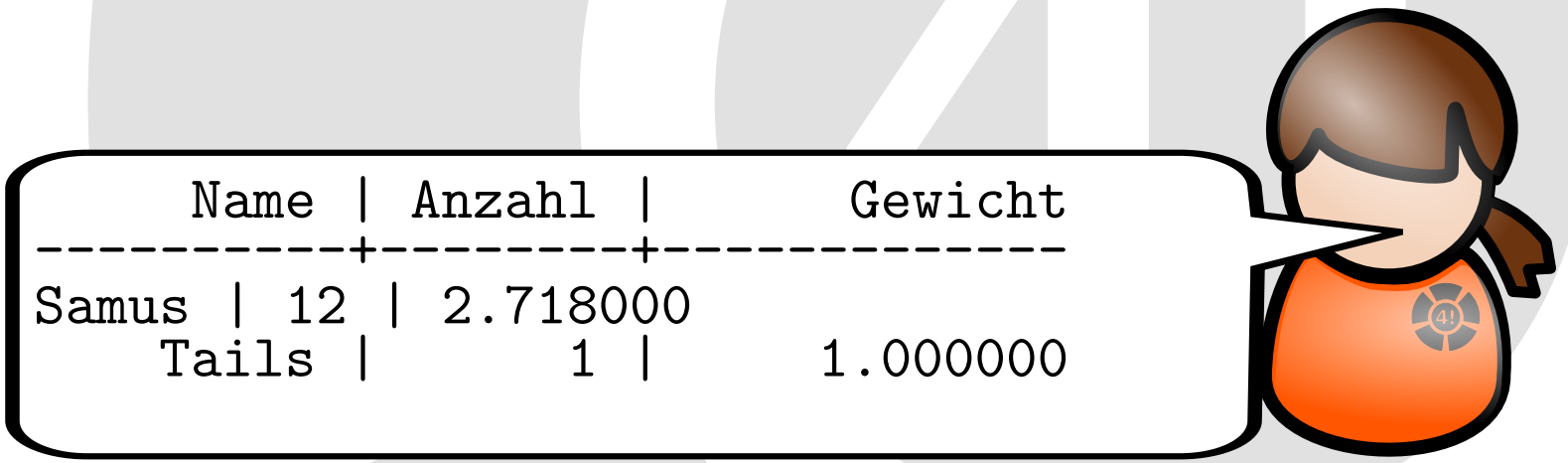
Rechtsbündig

- Wir wollen eine Tabelle ausgeben
- Ausgabe ist falsch
- Vorschlag: rechtsbündige Ausgabe

Beispiel: Ausgabe rechtsbündig

Listing 4: src/output.c

```
1  printf(" %s | %d | %f\n", "Samus", 12, 2.718 );
2  printf(" %9s | %6d | %12f\n", "Tails", 1, 1.0 );
```



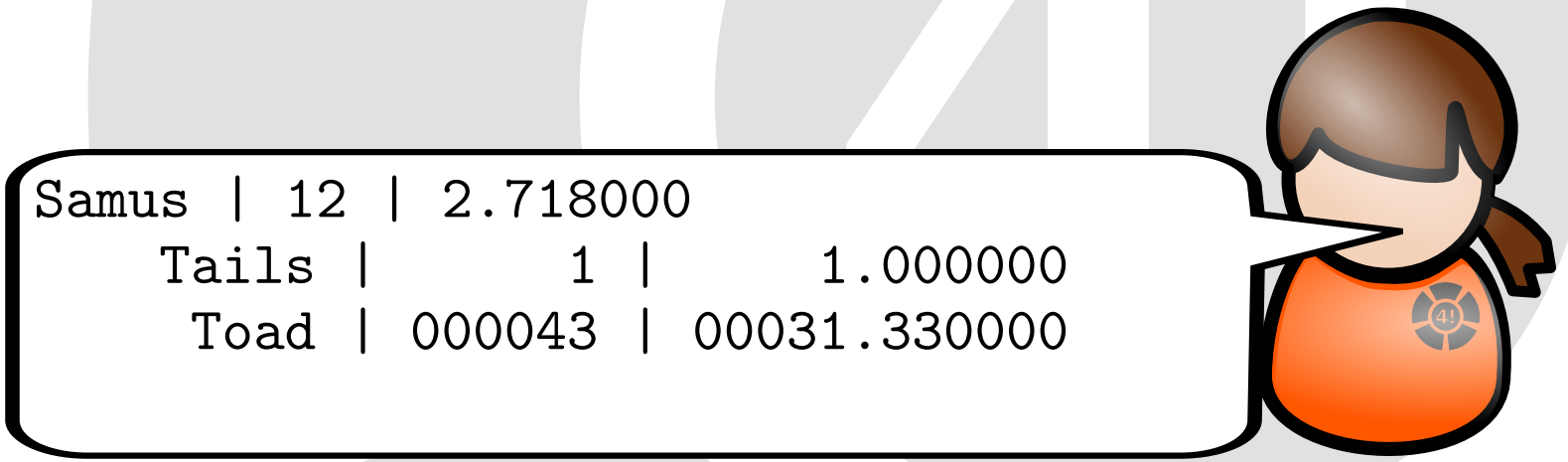
Auffüllung

- Ändert das Zeichen, mit dem aufgefüllt wird
- Sinnvoll bei rechtsbündiger Ausgabe
- Möglich: ' ' und '0' (Standard ist ' ')

Beispiel: Ausgeben von Werten

Listing 5: src/output.c

```
1  printf("␣%9s␣|␣%6d␣|␣%12f\n", "Tails", 1, 1.0 );
2  printf("␣%09s␣|␣%06d␣|␣%012f\n", "Toad", 43, 31.33 );
```



Linksbündig

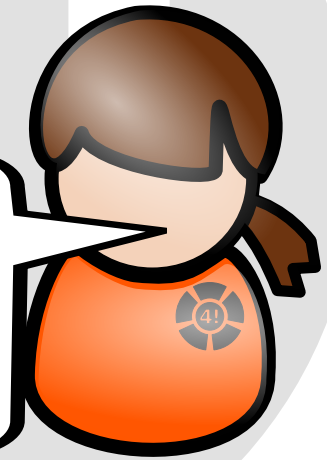
- '-' richtet Ausgabe linksbündig aus

Beispiel: Ausgeben von Werten

Listing 6: src/output.c

```
1  printf("␣%09s␣|␣%06d␣|␣%012f\n", "Toad", 43, 31.33 );
2  printf("␣%-9s␣|␣%-6d␣|␣%-12f\n", "GLaDOS", 87, 3.14 );
```

```
   Tails |      1 |     1.000000
    Toad | 000043 | 00031.330000
GLaDOS   | 87     | 3.140000
```



Nachkommastellen

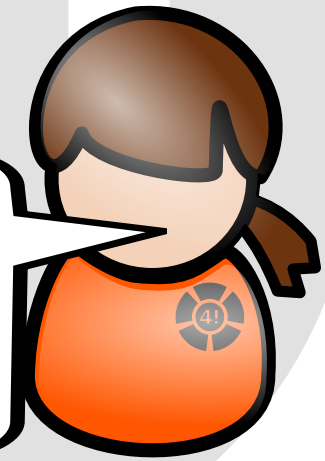
- Ausgabe wird normalerweise auf 6 Stellen gerundet
- Zuerst %, dann evtl nen Punkt, Anzahl der Stellen

Beispiel: Ausgeben von Werten

Listing 7: src/output.c

```
1  printf(" %-9s | %-6d | %-12f\n", "GLaDOS", 87, 3.14 );
2  printf(" %9s | %6d | %12.1f\n", "Kerrigan", 3, -0.16 );
```

```
    Toad | 000043 | 00031.330000
GLaDOS   | 87     | 3.140000
Kerrigan |      3 |         -0.2
```



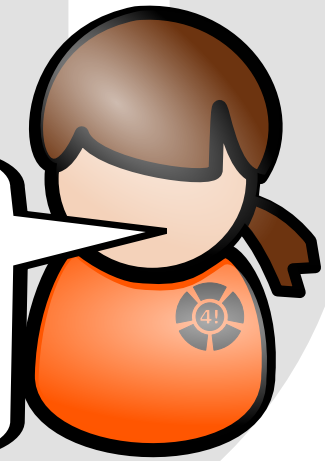
Gesamtausgabe

Beispiel: Ausgeben von Werten

Listing 8: src/output.c

```
1  printf(" %s | %d | %f\n", "Samus", 12, 2.718 );
2  printf(" %9s | %6d | %12f\n", "Tails", 1, 1.0 );
3  printf(" %09s | %06d | %012f\n", "Toad", 43, 31.33 );
4  printf(" %-9s | %-6d | %-12f\n", "GLaDOS", 87, 3.14 );
5  printf(" %9s | %6d | %12.1f\n", "Kerrigan", 3, -0.16 );
```

```
Samus | 12 | 2.718000
   Tails |      1 |     1.000000
    Toad | 000043 | 00031.330000
GLaDOS   | 87     | 3.140000
Kerrigan |      3 |         -0.2
```



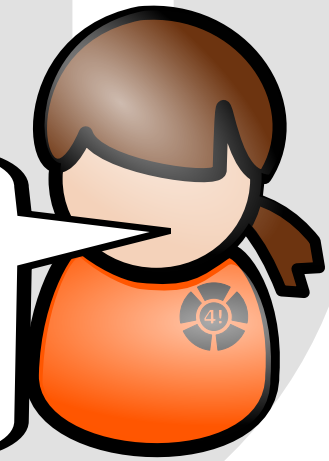
Aufgabe

Was geben folgende Zeilen aus:

Listing 9: src/exercise.c

```
1  float f = 3.14159265358979323846264338327795f;
2  printf("PI = %12f | \n", f);
3  printf("PI = %12.2f | \n", f);
4  printf("PI = %012.7f | \n", f);
5  printf("PI = %-012.7f | \n", f);
```

```
PI =     3.141593 |
PI =         3.14 |
PI = 0003.1415927 |
PI = 3.1415927    |
```



Wohin mit dem Ergebnis?

- Nicht nur die Konsole kann ein Ziel sein!
- Strings können befüllt werden
- Dateien können befüllt werden
- Name der Funktion ändert sich
  - Bei Strings benutzt man `sprintf` (s = string)
  - Und bei Dateien benutzt man `fprintf` (f = file)
- Es kommt jeweils ein neuer Parameter hinzu

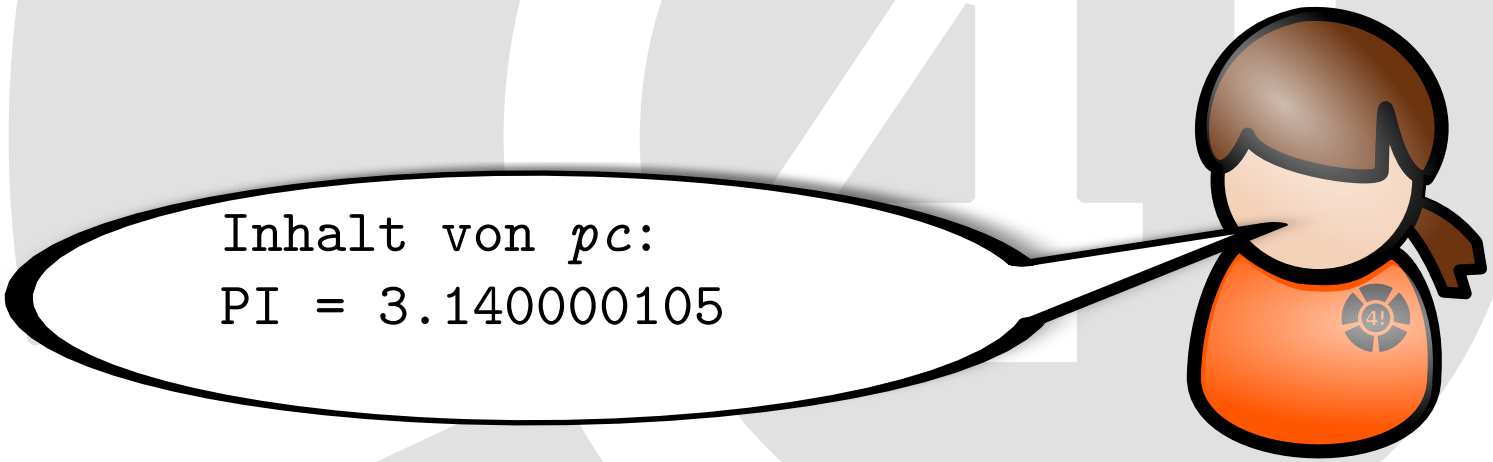
Ausgabe in einen String

- Strings werden hier nur kurz erwähnt
- Siehe dazu nächsten Vortrag

Ausgabe in String

Listing 10: src/sprintf.c

```
1  char pc[255];
2  sprintf(pc, "PI = %12.9f", 3.14f);
```

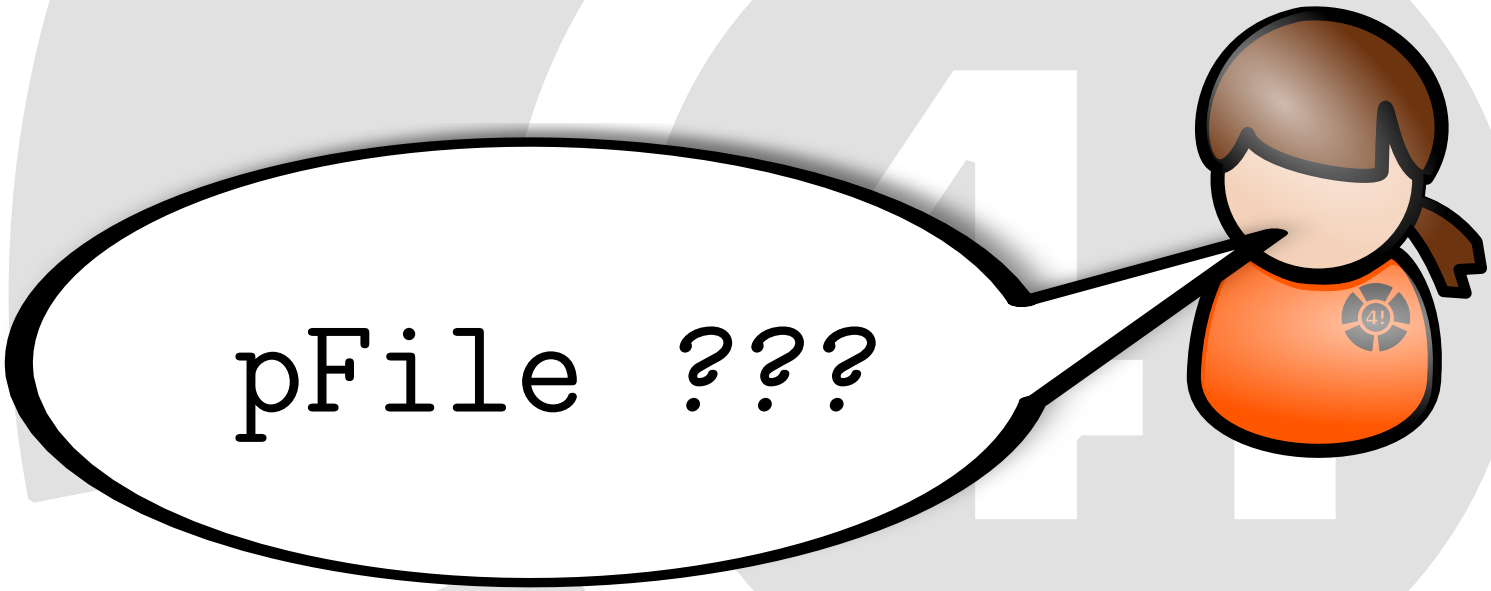


Ausgabe in eine Datei

Beispiel: Ausgabe in eine Datei

Listing 11: src/fprintf.c

```
1   fprintf(pFile, "HelloFile, %10.1f\n", 3.14f);
```



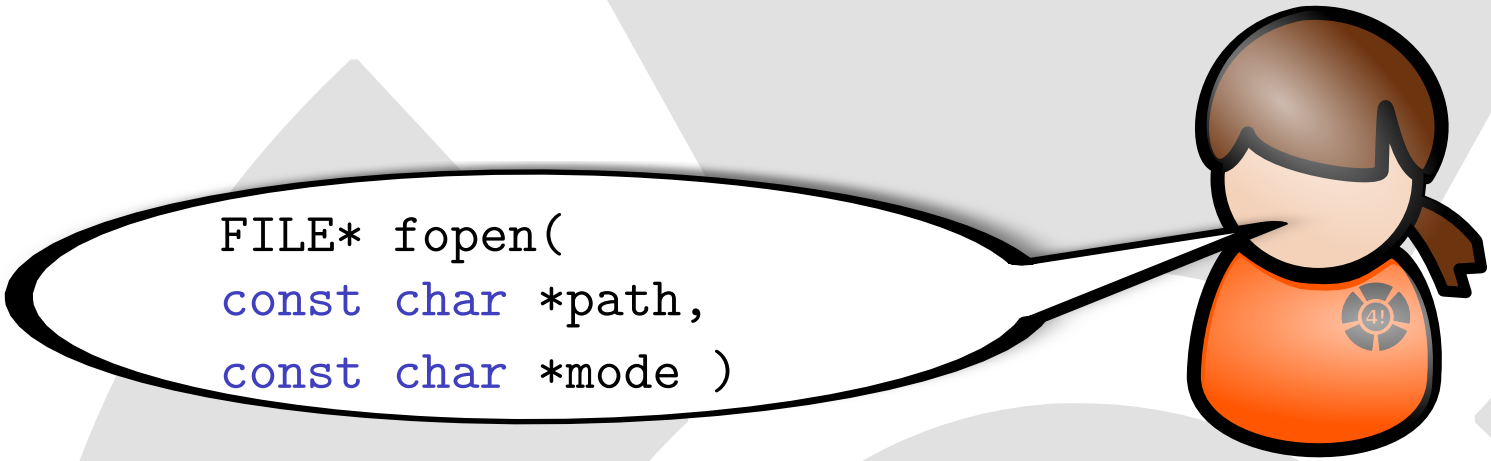
Inhaltsverzeichnis

1. Wiederholung
2. Ausgabe
   - Einschub: Dateien
3. Eingabe
4. Beispiel
5. Ausblick

Dateien

- Schema zum lesen/schreiben einer Datei immer gleich
  1. Öffnen der Datei mit *fopen*
  2. Überprüfung ob Öffnen erfolgreich
  3. Arbeiten auf der Datei mit *fprintf* oder *fscanf*
  4. Schließen der Datei mit *fclose*

Schritt 1: Öffnen einer Datei



- *FILE** ist der Rückgabetyp
  - Objekt, das die Datei repräsentiert
  - Kann definiert oder undefiniert (*NULL*) sein
- *path*
  - Name und Pfad der zu bearbeitenden Datei
- *mode*
  - "r": nur lesend (am Anfang der Datei)
  - "w": nur schreibend (erzeugt neue Datei)
  - "a": wie "w", nur schreibt ans Ende der Datei
  - "r+", "w+" oder "a+": lesend und schreibend

Schritt 1: Öffnen einer Datei - Beispiel

Beispiel: Ausgabe in eine Datei

Listing 12: src/fprintf.c

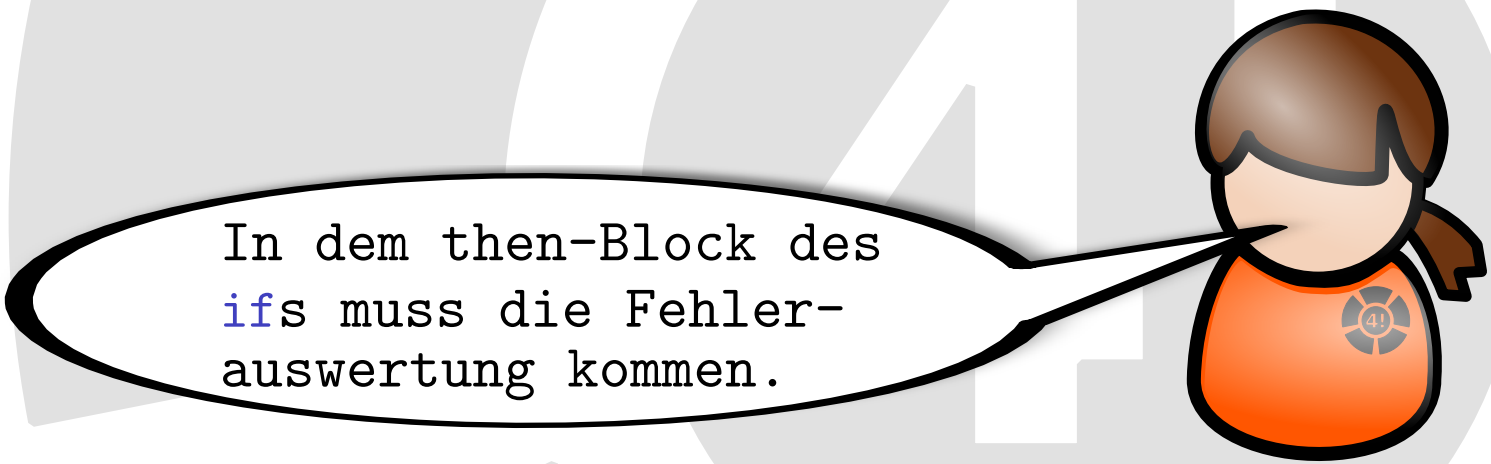
```
1  FILE *pFile = fopen("./test.dat", "w");
```

Schritt 2: Überprüfung

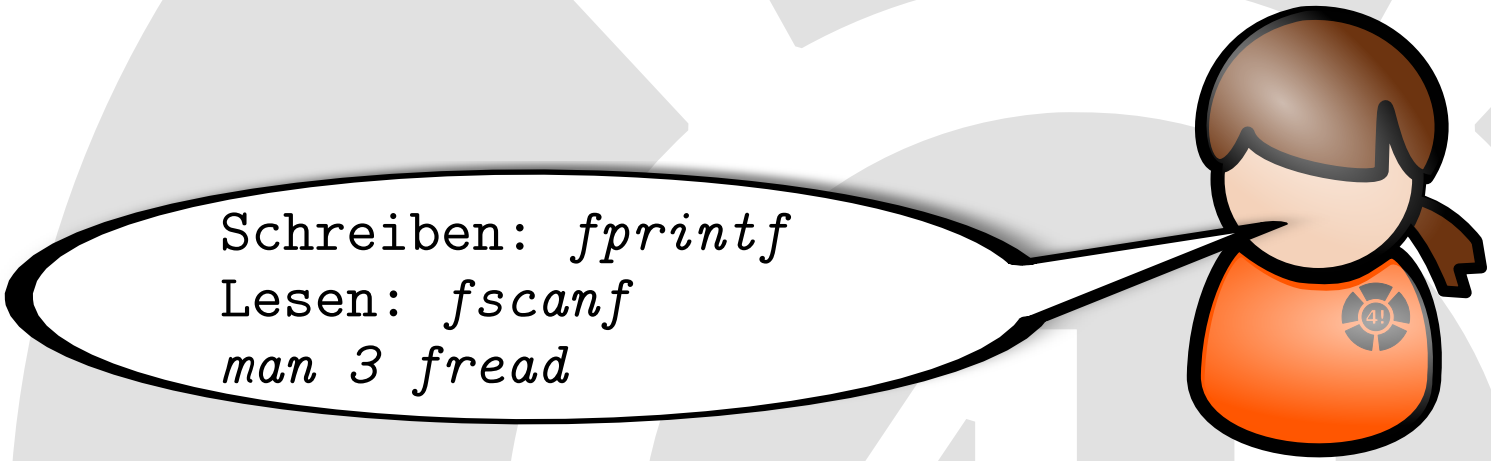
Beispiel: Ausgabe in eine Datei

Listing 13: src/fprintf.c

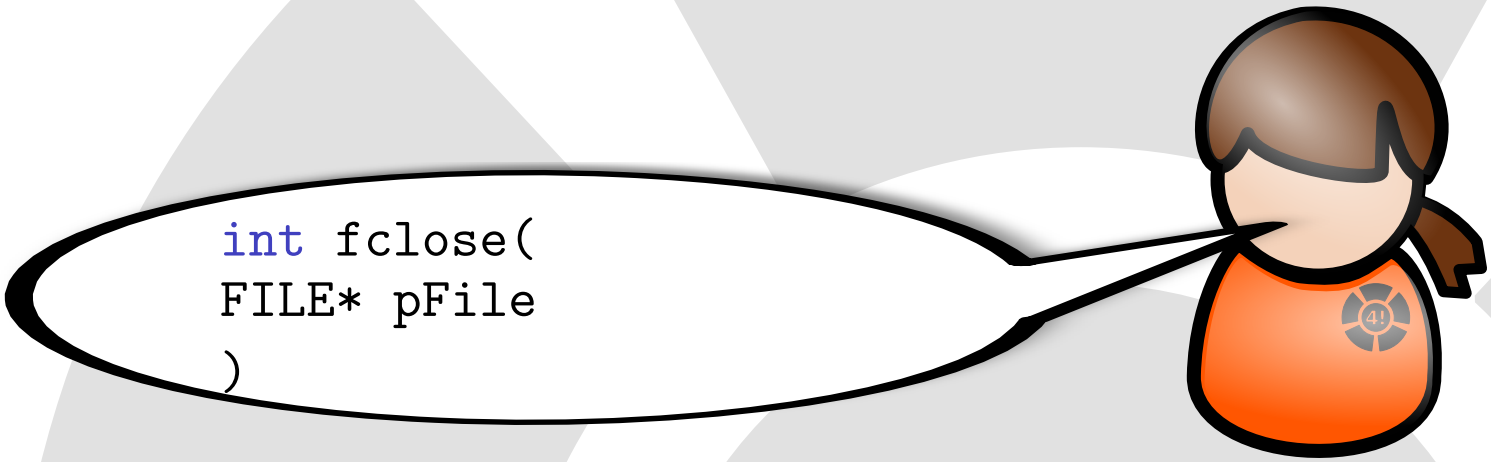
```
1  if( NULL == pFile )
2  {
3    // error handling ...
```



Schritt 3: Arbeit mit der Datei



Schritt 4: Schließen einer Datei



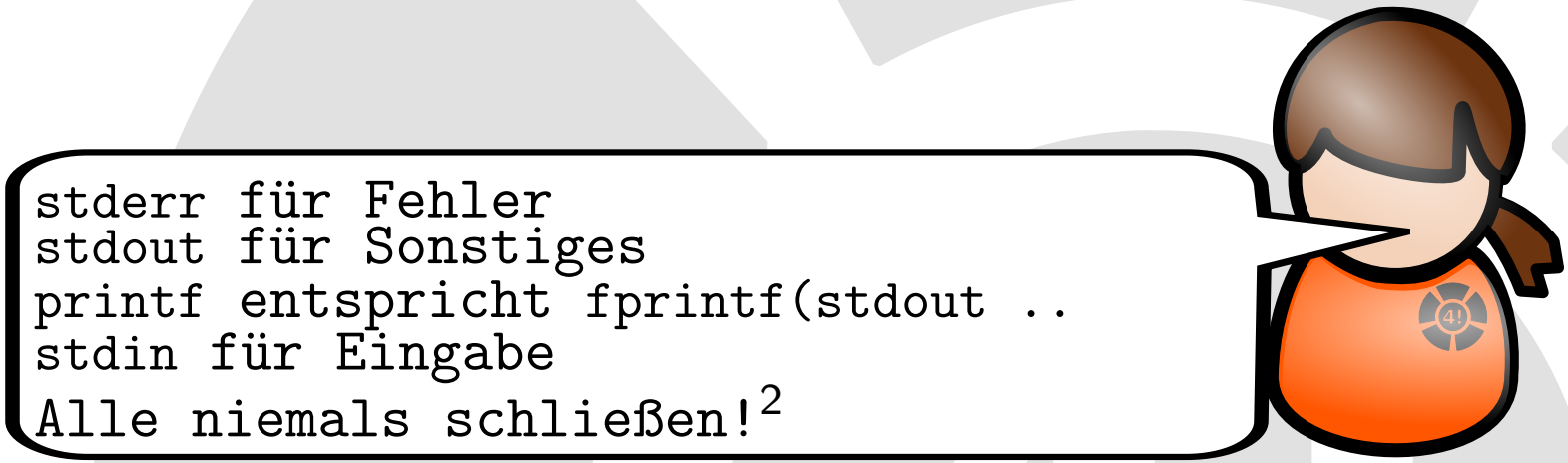
- Fehlercode (int) als Rückgabe
- *pFile* Dateiobjekt, welches geschlossen werden soll
- Datei muss wieder geschlossen werden
- Danach kann das Dateiobjekt nicht mehr benutzt werden

Gesamtbeispiel

Listing 14: src/fprintf.c

```
1  // open a file
2  FILE *pFile = fopen("./test.dat", "w");
3
4  // check for errors
5  if( NULL == pFile )
6  {
7    // error handling ...
8    fprintf(stderr,"Konnte Datei nicht finden ...\n");
9    return -1;
10 }
11
12 // write something to a file
13 fprintf(pFile, "HelloFile, %10.1f\n", 3.14f);
14
15 // now close it again
16 fclose(pFile);
```

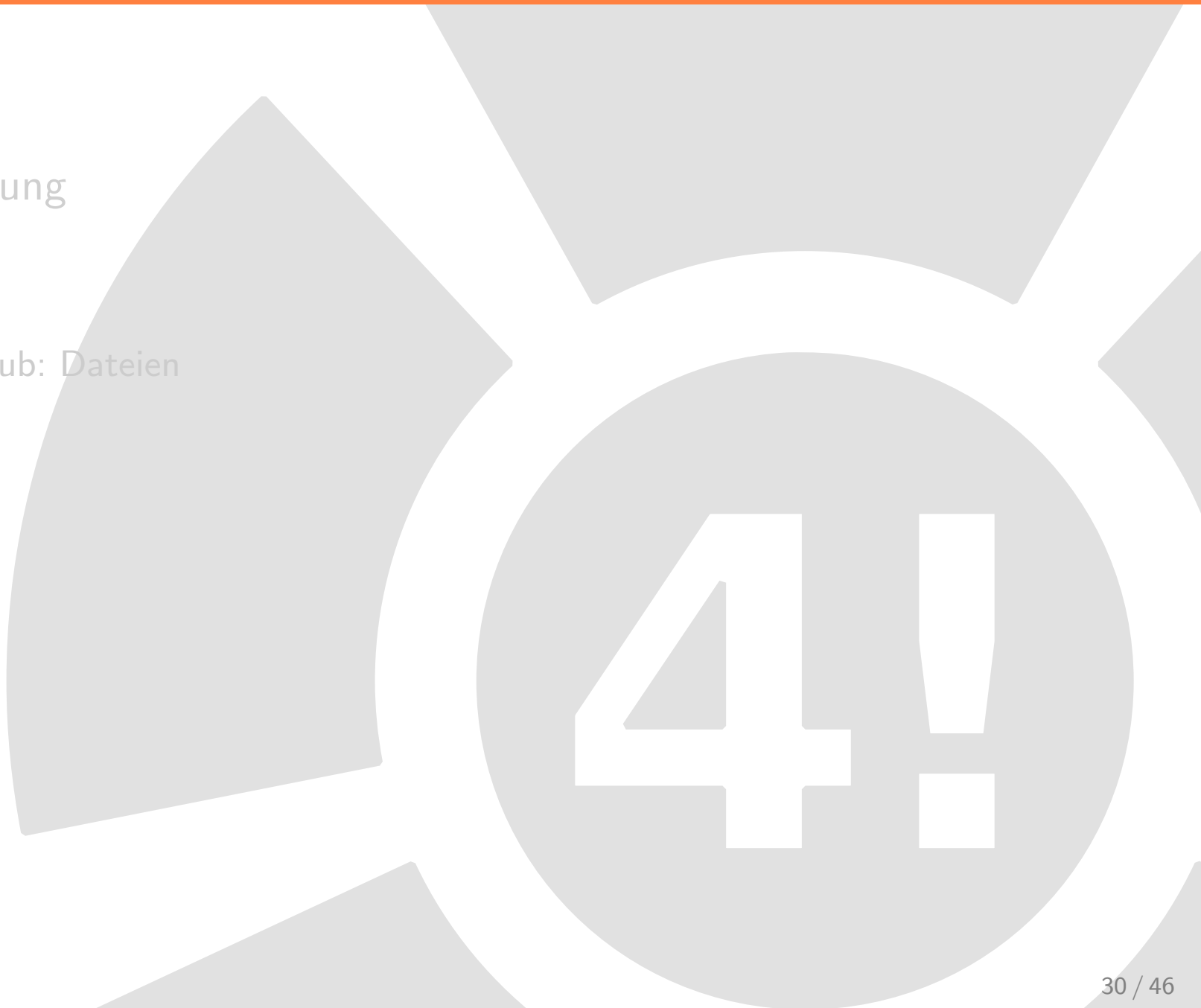
Besondere Dateien



[2] Außer es gibt einen wichtigen Grund!

Inhaltsverzeichnis

1. Wiederholung
2. Ausgabe
   - Einschub: Dateien
3. Eingabe
4. Beispiel
5. Ausblick



Analogie zur Ausgabe

- Eingabe ist analog zu Ausgabe
- Statt `printf` benutzt man nun `scanf`
- Alle Typen von `printf` gelten auch für `scanf`
- Auch die Formatangaben (Linksbündigkeit, Stellenzahl, etc.) gelten

Eingabe Beispiel

Beispiel: Eingabe zweier Werte

Listing 15: src/input.c

```
1  int a = 0;
2  int b = 0;
3  int iValues = scanf("%d %d", &a, &b);
4  printf("a + b = %d\n", a+b);
5  printf("iValues = %d\n", iValues);
```

Funktionsweise von *scanf*

- Es speichert die Eingabe des Benutzer
- Überprüft, ob die Eingabe zu den angegebenen Typen passt.
- Ignoriert Leerzeichen, Tabulatoren und Zeilenumbrüche
- Nicht Formatangaben müssen vorhanden sein (siehe Beispiel)
- Rückabewert beinhaltet Anzahl der erfolgreich gelesenen Werte.
- Daher: *Immer* den Rückgabewert überprüfen

Probleme beim Einlesen

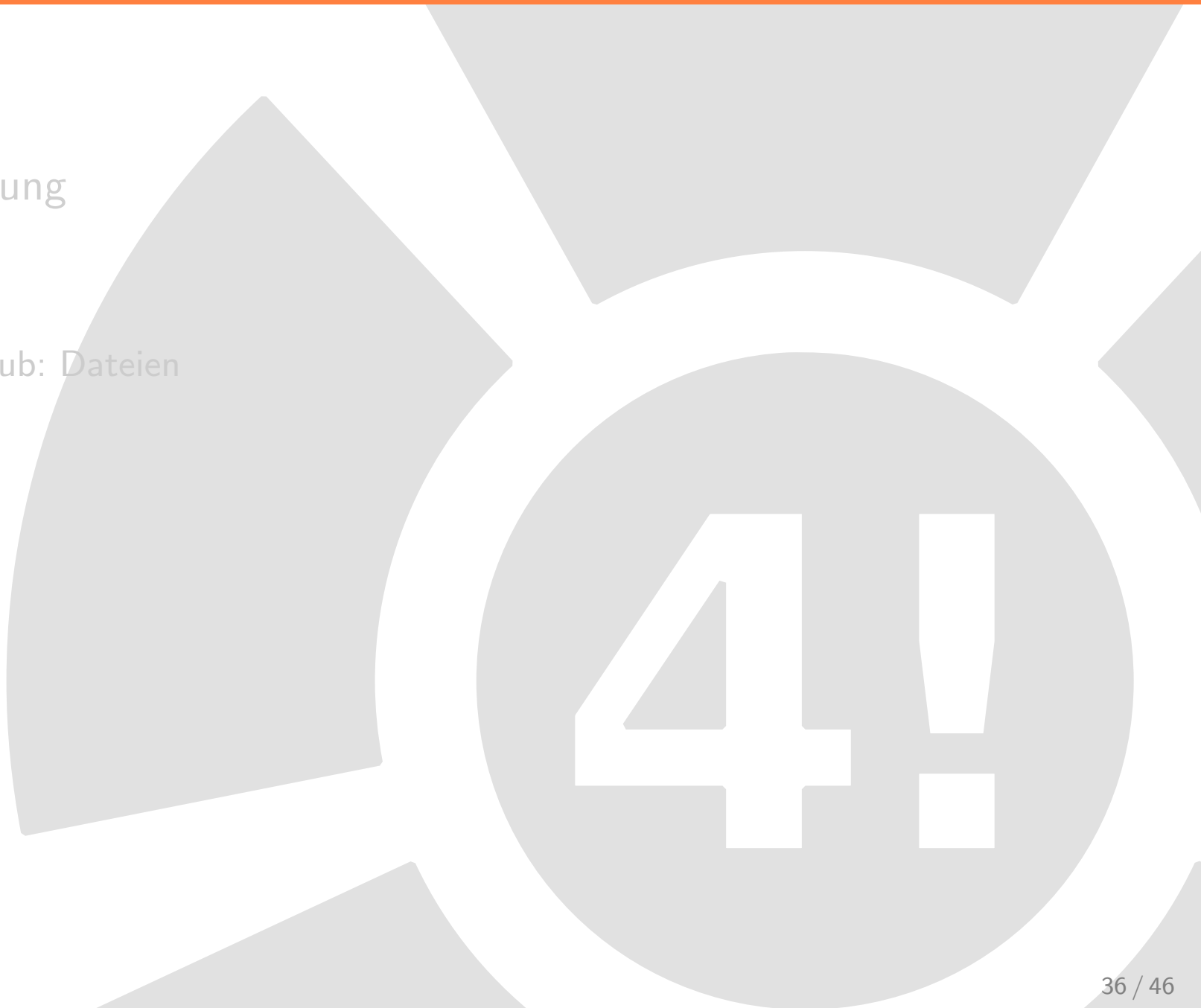
- "%s" liest alle Eingaben bis zur neuen Zeile
  - Auch andere Formatangaben werden überlesen ("%d", "%f")
  - Daher: Genauer formulieren mit bspw. "%[A−Z]"
  - Registriert alle Buchstaben zwischen A und Z
  - Für alle Zeichen: "%[A−Za−z]"
- "%d" und Buchstaben werden eingegeben?
  - Rückgabewert ist kleiner als erwartet
  - "Rest"der Eingabe liegt noch im Speicher, muss gelöscht werden
  - scanf("%s",pcTemp); liest den Rest ein

Eingabe aus Strings und Dateien

- Genau wie bei `printf` gibt es auch bei `scanf` weitere Varianten
- `sscanf` analog zu `sprintf`: liest aus einem String
- `fscanf` analog zu `fprintf`: liest aus einer Datei
  - *mode* muss dann mindestens `"r"` oder `"+"` sein
  - Reihenfolge von oben beachten

Inhaltsverzeichnis

1. Wiederholung
2. Ausgabe
   - Einschub: Dateien
3. Eingabe
4. Beispiel
5. Ausblick



Beispiel

- Einlesen einer *CSV*[3]-Datei
- Benutzt als einfaches Datenbankformat
- Beispiel umfasst Datei lesen, schreiben und Ausgabe von Werten

[3] coma seperated values

Datei lesen

Beispiel: Lesen der CSV-Datei

Listing 16: src/csv.c

```
1   do
2   {
3      // read values
4      iItemsRead = fscanf(pFile, "%[A-Za-z];␣%d;␣%f;␣", &
           pcName, &iAmount, &fWeight );
5
6      // check values read
7      if( iItemsRead == 3 )
8      {
9         printf("%12s␣|␣%12d␣|␣%012.3f\n", pcName, iAmount,
              fWeight );
10     }
11  } while( iItemsRead == 3 );
```

Benutzer überprüfen

Beispiel: Benutzereingaben überprüfen

Listing 17: src/csv.c

```
1       printf("Anzahl: ");
2       if( 1 != scanf("%d", &iAmount) )
3       {
4          fprintf(stderr, "Konnte Anzahl nicht lesen\n");
5          scanf("%s", &pcName);
6          continue;
7       }
```

Benutzer schreiben

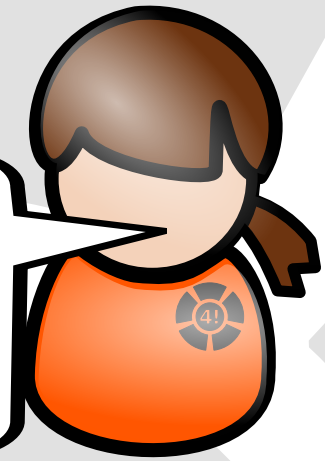
Beispiel: Benutzereingaben schreiben

Listing 18: src/csv.c

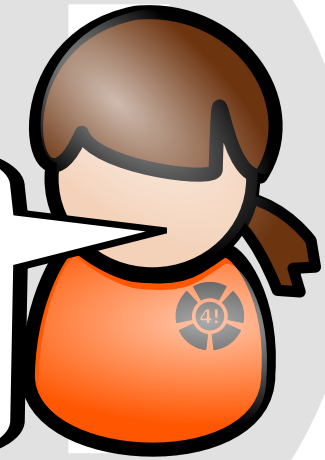
```
1       // everything read successfully
2       int iCharsWritten = fprintf( pFile, "%s; %d; %f;\n",
            pcName, iAmount, fWeight );
3       if(iCharsWritten <= 0)
4       {
5          fprintf(stderr, "Konnte Daten nicht schreiben!\n");
6       }
```

Ausgabe des CSV Beispiels

```
Pigmin; 738; 0.0;
Mii; 98; 10.20;
Thaddaeus; 9999; 999;
```

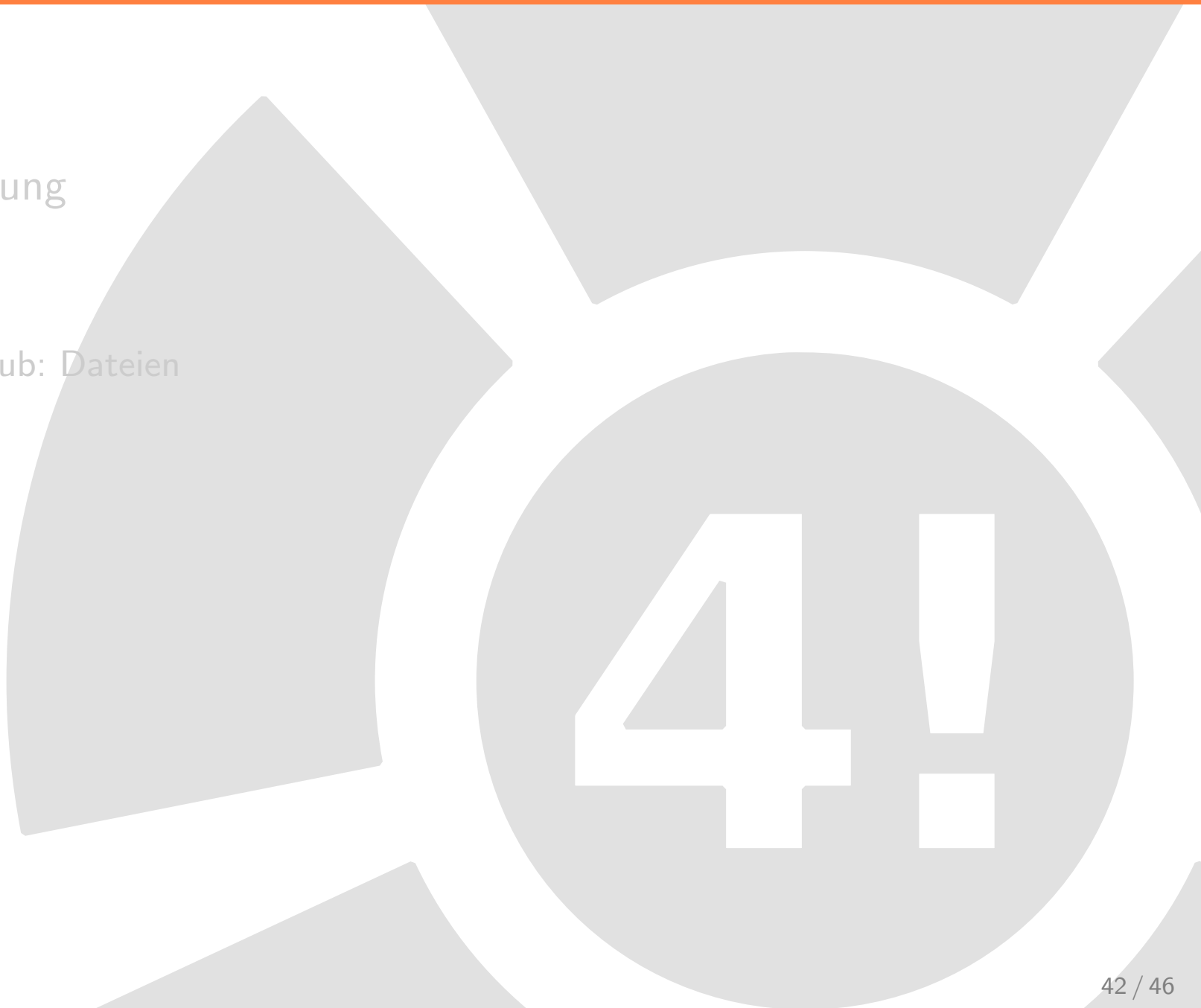


```
        Name | Anzahl |       Gewicht
-------------+--------+--------------
      Pigmin |    738 | 00000000.000
         Mii |     98 | 00000010.200
   Thaddaeus |   9999 | 00000999.000
```



Inhaltsverzeichnis

1. Wiederholung
2. Ausgabe
   - Einschub: Dateien
3. Eingabe
4. Beispiel
5. Ausblick

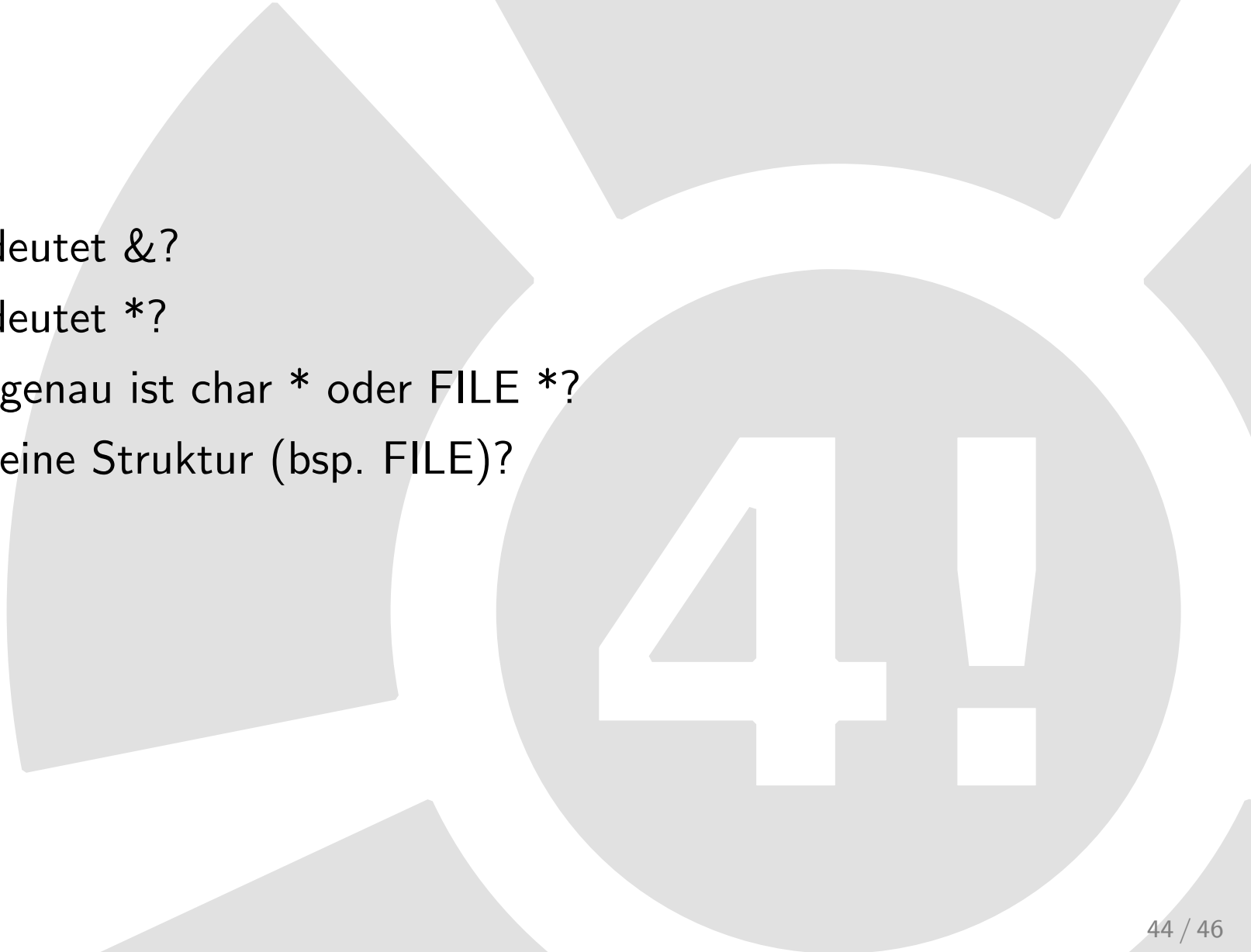


Zusammenfassung: Ein- und Ausgabe

- Ausgabe von Strings, ganzen Zahlen, gebrochenen Zahlen
- Ausgabe in Datei umleiten
- Ausgabe in String umleiten
- Eingabe = -Ausgabe
- Eingabe kann auch aus einer Datei oder aus einem String lesen.

Ausblick

- Was bedeutet &?
- Was bedeutet *?
- Was ist genau ist char * oder FILE *?
- Was ist eine Struktur (bsp. FILE)?



Danke

Quellen

- *man 3 printf*
- *man 3 scanf*
- *man 3 fopen*
- *man 3 fclose*
- *man 3 fread*

Alle Grafiken stammen aud der openCliparts Library und wurden von Mario Bodemann bearbeitet. Ausnahme bildet hierbei nur das Logo der Freitagsrunde. Dies wurde in einer Coproduktion von Mario Bodemann und Sebastian D. aus dem alten Logo der Freitagsrunde erstellt. Die Präsentationsvorlage darf weiter benutzt werden, jedoch besteht dabei keinerlei Garantie oder Gewährleistung.