

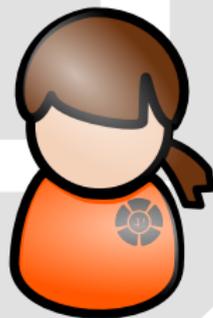
Ein- und Ausgabe

C - Kurs 2009

Mario Bodemann

www.freitagrunde.org

20. September 2009



Ein- und Ausgabe

C - Kurs 2009

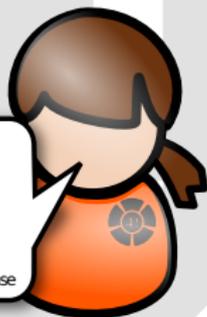
Mario Bodemann

www.freitagrunde.org

20. September 2009



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



- 1 Wiederholung
- 2 Ausgabe
 - Formate
 - Ziele der Ausgabe
- 3 Eingabe
- 4 Ausblick



4!

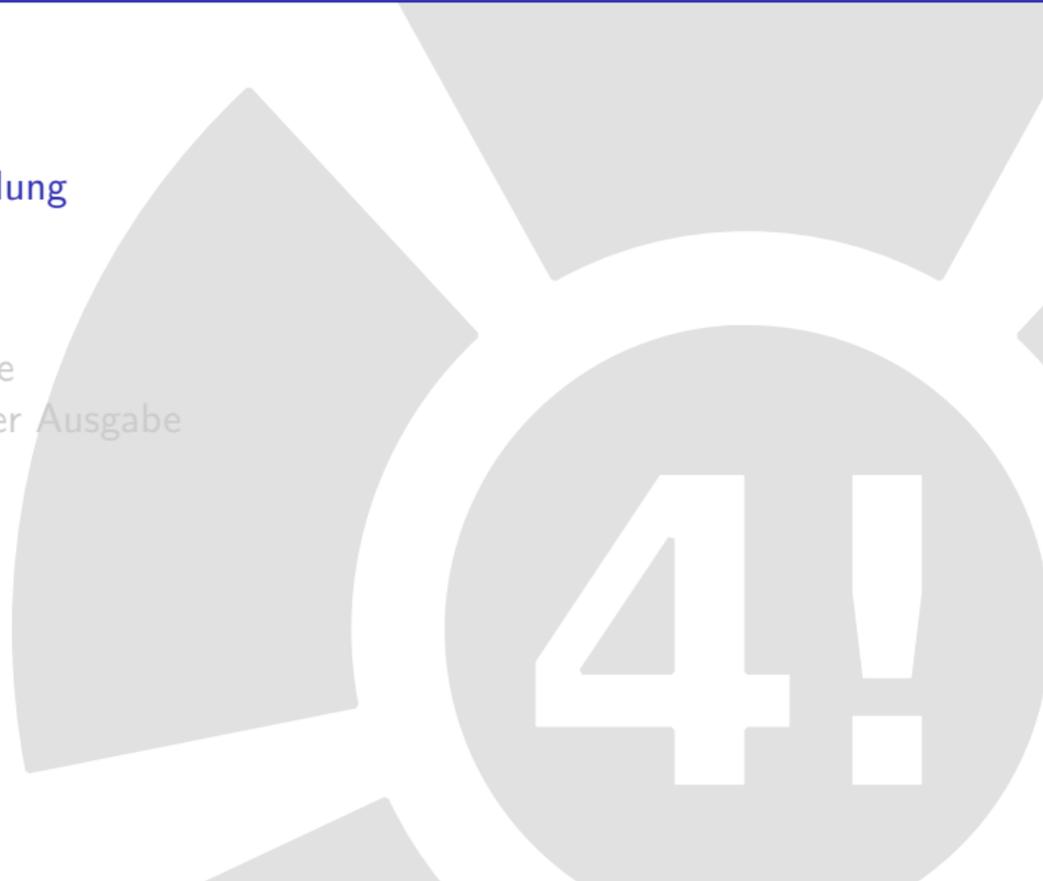
1 Wiederholung

2 Ausgabe

- Formate
- Ziele der Ausgabe

3 Eingabe

4 Ausblick



4!

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

A large, light gray circular graphic containing the number '4' and an exclamation mark '!' in white, positioned on the right side of the slide.

4!

Wiederholung: Syntax

```
1 int a = 4;
2 int b = 4 + a;
3 int c = ++b + a++;
4 int d = (a == b ? c : --b + c);
5
6 if( 9 == d ){
7     d %= 8;
8 } else {
9     d <<= 2;
10 }
```

- Datentypen/Variablen
- Operatoren
- Kontrollstrukturen
- Welchen Wert hat d?

Wiederholung: Hello World

```
1 #include <stdio.h>
2 int main(int argc, char **argv)
3 {
4     /* say hello to the world! */
5     printf("Hello World\n");
6     return 0;
7 }
```

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

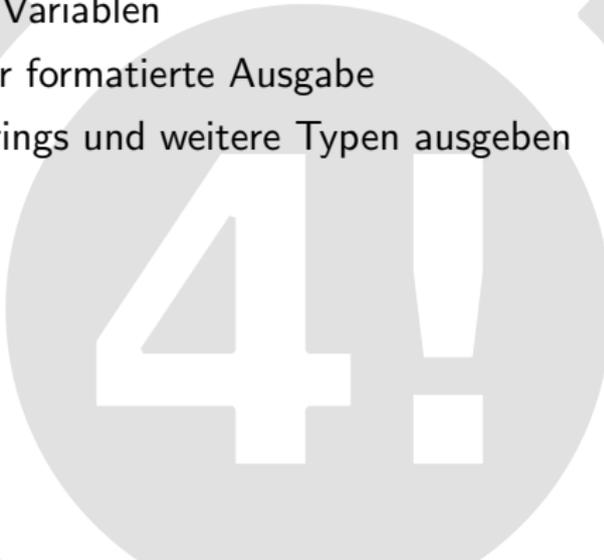
- 1 Wiederholung
- 2 Ausgabe
 - Formate
 - Ziele der Ausgabe
- 3 Eingabe
- 4 Ausblick



- Lerne, wie du Variablen ausgeben kannst
- Lerne, wie diese Ausgaben formatiert werden
- Speichere und lade Variablen in/aus Dateien

A large, light gray graphic of the number '4' followed by an exclamation mark, centered within a circular area. The background of the slide features abstract, overlapping geometric shapes in shades of gray and white.

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

A large, light gray graphic in the background features a circle containing the number '4' followed by an exclamation mark '!', both in white. The circle is partially overlaid by other abstract shapes.

Beispiel: Ausgaben

```
1 printf(" a = %d\n" , 12);  
2 printf(" b = %f\n" , 1.3 f);  
3 printf(" c = %s\n" , " Hello" );
```



4!

Beispiel mit printf

Beispiel: Ausgaben

```
1 printf("a = %d\n", 12);  
2 printf("b = %f\n", 1.3f);  
3 printf("c = %s\n", "Hello");
```

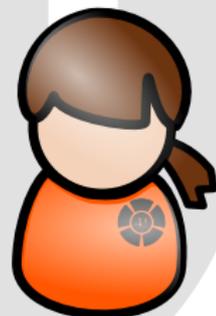
Ausgabe

```
1 a = 12  
2 b = 1.300000  
3 c = Hello
```

Formate für printf

Symbol	Beschreibung	Beispiel
%d	ganze Zahlen	3
%x	ganze Zahlen in Hex	affe1
%f	gebrochene Zahlen	3.1415927
%c	Buchstaben	p
%s	Strings	HalloWelt
%%	Prozentzeichen	%

4



Formate für printf

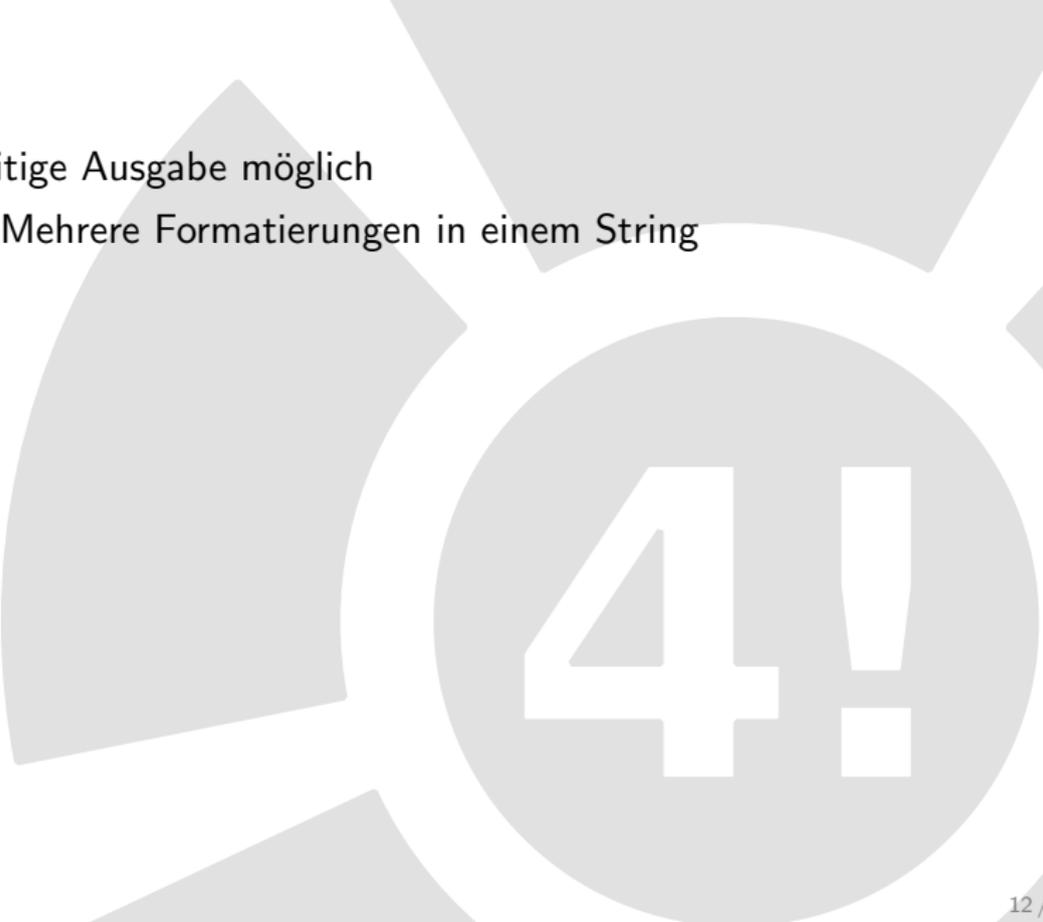
Symbol	Beschreibung	Beispiel
%d	ganze Zahlen	3
%x	ganze Zahlen in Hex	affe1
%f	gebrochene Zahlen	3.1415927
%c	Buchstaben	p
%s	Strings	HalloWelt
%%	Prozentzeichen	%



Für weitere Formate
siehe [Manpage](#).

Mehrere Werte gleichzeitig ausgeben

- Gleichzeitige Ausgabe möglich
- Lösung: Mehrere Formatierungen in einem String



4!

Mehrere Werte gleichzeitig ausgeben

- Gleichzeitige Ausgabe möglich
- Lösung: Mehrere Formatierungen in einem String

Beispiel: Ausgabe von mehreren Werten

```
1 printf(" a = %d , b = %f , c = %s\n" , 12, 1.3f , " Hello" );
```

Mehrere Werte gleichzeitig ausgeben

- Gleichzeitige Ausgabe möglich
- Lösung: Mehrere Formatierungen in einem String

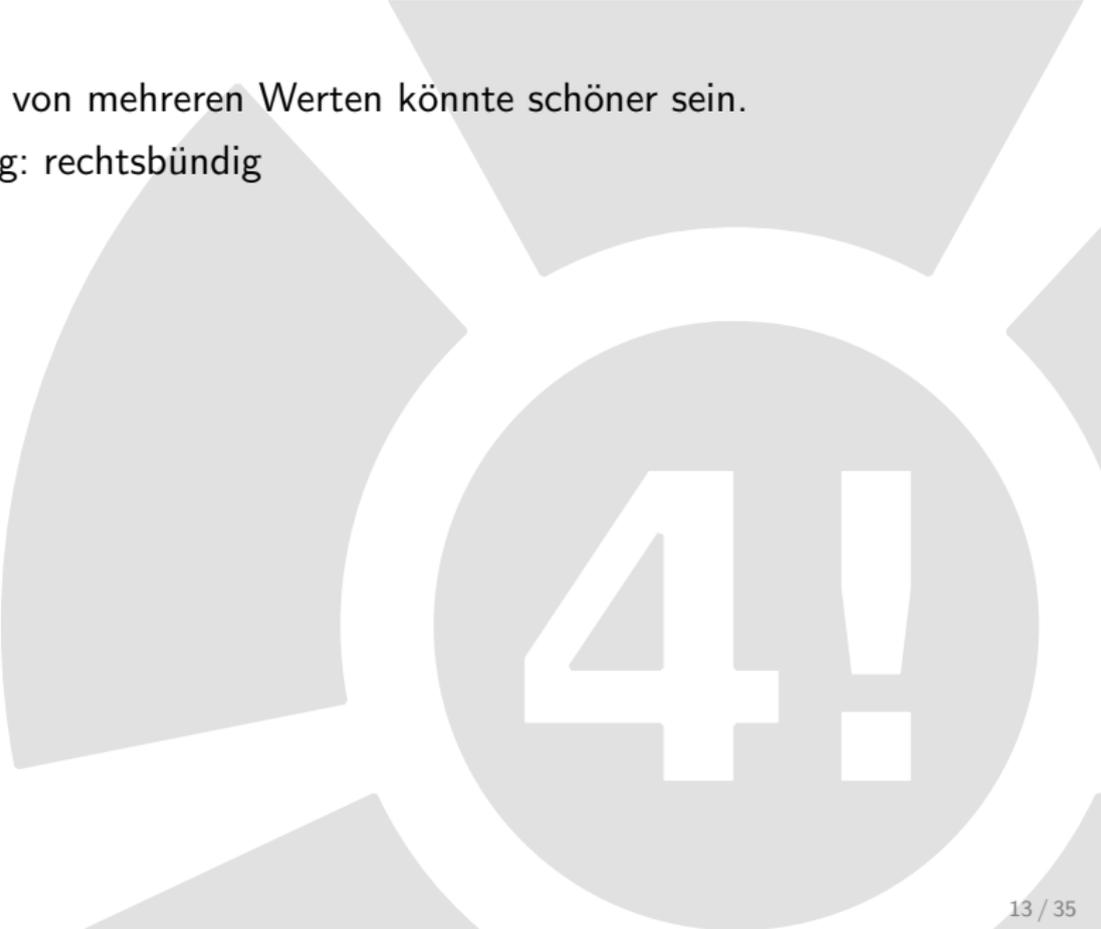
Beispiel: Ausgabe von mehreren Werten

```
1 printf("a = %d, b = %f, c = %s\n", 12, 1.3f, "Hello");
```

Ausgabe

```
1 a = 12, b = 1.300000, c = Hello
```

- Ausgabe von mehreren Werten könnte schöner sein.
- Vorschlag: rechtsbündig



4!

- Ausgabe von mehreren Werten könnte schöner sein.
- Vorschlag: rechtsbündig

Beispiel: Ausgeben von Werten

```
1 printf("a = %10d\n", 12);  
2 printf("b = %10f\n", 1.3f);  
3 printf("c = %10s\n", "Hello");
```

- Ausgabe von mehreren Werten könnte schöner sein.
- Vorschlag: rechtsbündig

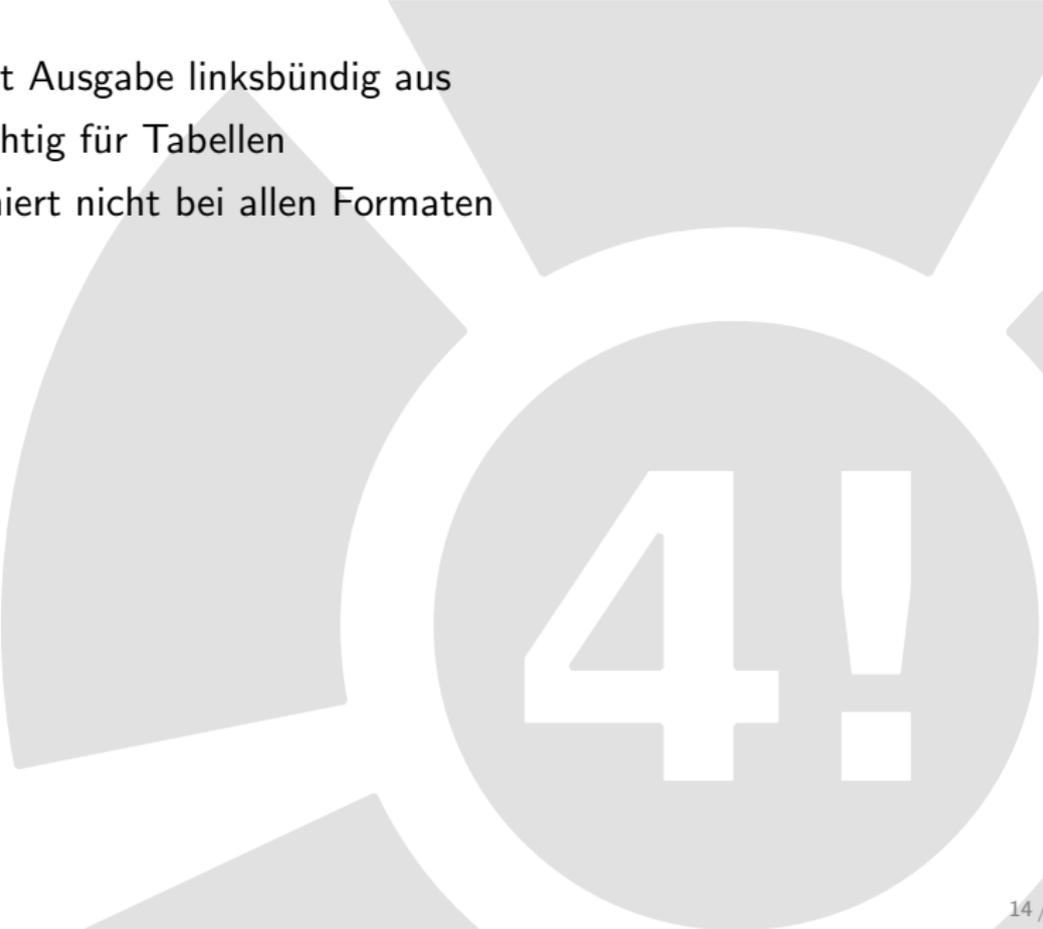
Beispiel: Ausgeben von Werten

```
1 printf("a = %10d\n", 12);  
2 printf("b = %10f\n", 1.3f);  
3 printf("c = %10s\n", "Hello");
```

Ausgabe

```
1 a =          12  
2 b =      1.300000  
3 c =          Hello
```

- '-' richtet Ausgabe linksbündig aus
- Sehr wichtig für Tabellen
- Funktioniert nicht bei allen Formaten



4!

- '-' richtet Ausgabe linksbündig aus
- Sehr wichtig für Tabellen
- Funktioniert nicht bei allen Formaten

Beispiel: Ausgeben von Werten

```
1 printf(" a = %-10d | Next\n", 12);  
2 printf(" b = %-10f | Next\n", 1.3 f);  
3 printf(" c = %-10s | Next\n", " Hello" );
```

Linksbündig

- '-' richtet Ausgabe linksbündig aus
- Sehr wichtig für Tabellen
- Funktioniert nicht bei allen Formaten

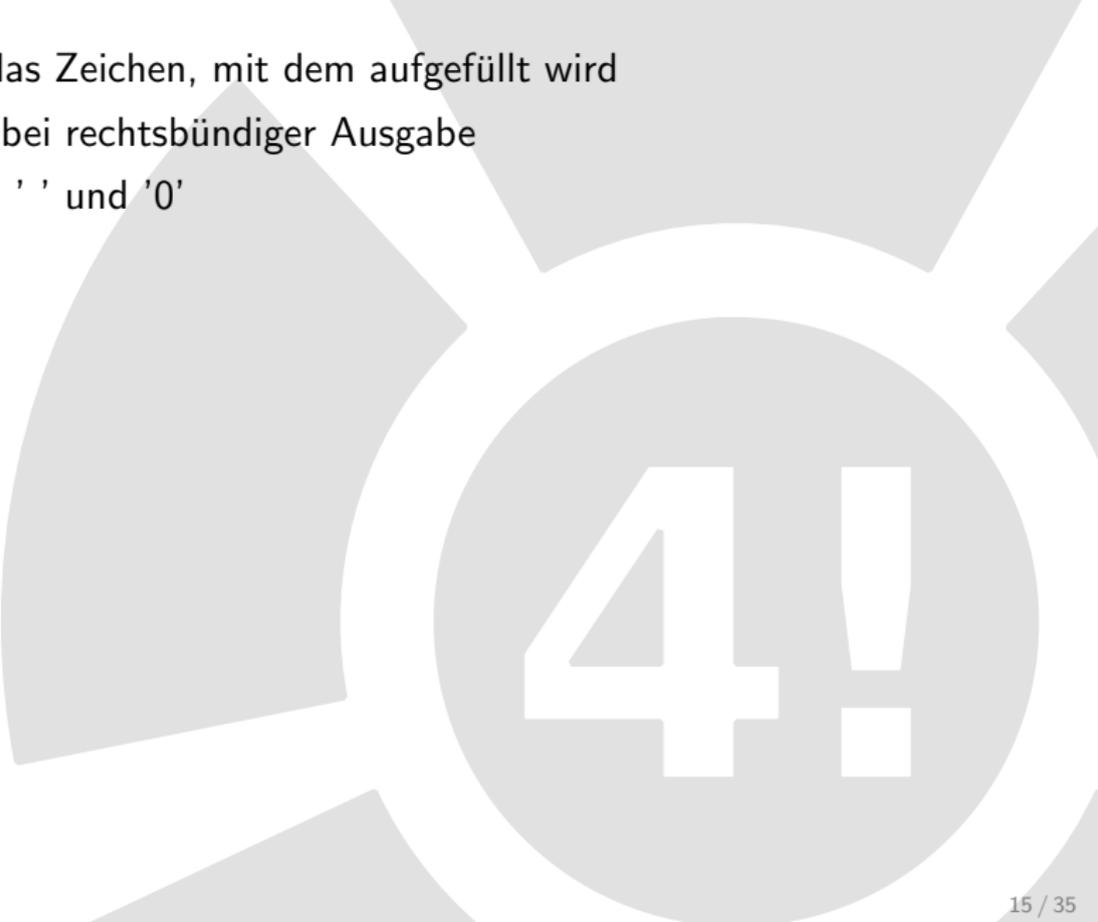
Beispiel: Ausgeben von Werten

```
1 printf("a = %-10d | Next\n", 12);  
2 printf("b = %-10f | Next\n", 1.3f);  
3 printf("c = %-10s | Next\n", "Hello");
```

Ausgabe

```
1 a = 12           | Next  
2 b = 1.300000    | Next  
3 c = Hello       | Next
```

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



4!

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

Beispiel: Ausgeben von Werten

```
1 printf(" a = %010d\n", 12);  
2 printf(" b = %_10f\n", 1.3f);  
3 printf(" c = %010s\n", "Hello");
```

Auffüllung

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

Beispiel: Ausgeben von Werten

```
1 printf(" a = %010d\n", 12);  
2 printf(" b = %_10f\n", 1.3f);  
3 printf(" c = %010s\n", "Hello");
```

Ausgabe

```
1 a = 0000000012  
2 b = 1.300000  
3 c = Hello
```

- Ausgabe wird auf 6 Stellen gerundet
- vgl. Ausgabe



4!

- Ausgabe wird auf 6 Stellen gerundet
- vgl. Ausgabe

Beispiel: PI

```
1 float f = 3.1415926535897932384626433832795 f;  
2 printf("PI = %f\n", f);
```

- Ausgabe wird auf 6 Stellen gerundet
- vgl. Ausgabe

Beispiel: PI

```
1 float f = 3.1415926535897932384626433832795 f;  
2 printf("PI = %f\n", f);
```

Ausgabe

```
1 PI = 3.141593
```

Beispiel zu Nachkommastellen

- Nach dem % kommt ein Punkt
- Nach dem Punkt die Anzahl der Nachkommastellen



4!

Beispiel zu Nachkommastellen

- Nach dem % kommt ein Punkt
- Nach dem Punkt die Anzahl der Nachkommastellen

Beispiel: PI begrenzt

```
1 float f = 3.1415926535897932384626433832795 f;  
2 printf("PI = %f\n", f);  
3 printf("PI = %.0f\n", f);  
4 printf("PI = %.2f\n", f);  
5 printf("PI = %.9f\n", f);
```

Beispiel zu Nachkommastellen

- Nach dem % kommt ein Punkt
- Nach dem Punkt die Anzahl der Nachkommastellen

Beispiel: PI begrenzt

```
1 float f = 3.1415926535897932384626433832795 f;  
2 printf("PI = %f\n", f);  
3 printf("PI = %.0f\n", f);  
4 printf("PI = %.2f\n", f);  
5 printf("PI = %.9f\n", f);
```

Ausgabe

```
1 PI = 3.141593  
2 PI = 3  
3 PI = 3.14  
4 PI = 3.141592741
```

Beispiel: PI begrenzt, Links- und Rechtsbündig

```
1 float f = 3.1415926535897932384626433832795 f;  
2 printf("PI = %12f| next\n", f);  
3 printf("PI = %12.2f| next\n", f);  
4 printf("PI = %012.7f| next\n", f);  
5 printf("PI = %-012.7f| next\n", f);
```



4!

Beispiel: PI begrenzt, Links- und Rechtsbündig

```
1 float f = 3.1415926535897932384626433832795 f;  
2 printf("PI = %12f| next\n", f);  
3 printf("PI = %12.2f| next\n", f);  
4 printf("PI = %012.7f| next\n", f);  
5 printf("PI = %-012.7f| next\n", f);
```

Ausgabe

```
1 PI =      3.141593| next
```

Beispiel: PI begrenzt, Links- und Rechtsbündig

```
1 float f = 3.1415926535897932384626433832795 f;  
2 printf("PI = %12f| next\n", f);  
3 printf("PI = %12.2f| next\n", f);  
4 printf("PI = %012.7f| next\n", f);  
5 printf("PI = %-012.7f| next\n", f);
```

Ausgabe

```
1 PI =      3.141593| next  
2 PI =           3.14| next
```

Kombination

Beispiel: PI begrenzt, Links- und Rechtsbündig

```
1 float f = 3.1415926535897932384626433832795 f;  
2 printf("PI = %12f| next\n", f);  
3 printf("PI = %12.2f| next\n", f);  
4 printf("PI = %012.7f| next\n", f);  
5 printf("PI = %-012.7f| next\n", f);
```

Ausgabe

```
1 PI =      3.141593| next  
2 PI =           3.14| next  
3 PI = 0003.1415927| next
```

Kombination

Beispiel: PI begrenzt, Links- und Rechtsbündig

```
1 float f = 3.1415926535897932384626433832795 f;  
2 printf("PI = %12f| next\n", f);  
3 printf("PI = %12.2f| next\n", f);  
4 printf("PI = %012.7f| next\n", f);  
5 printf("PI = %-012.7f| next\n", f);
```

Ausgabe

```
1 PI =      3.141593| next  
2 PI =           3.14| next  
3 PI = 0003.1415927| next  
4 PI = 3.1415927  | next
```



Kombination

Beispiel: PI begrenzt, Links- und Rechtsbündig

```
1 float f = 3.1415926535897932384626433832795 f;  
2 printf("PI = %12f| next\n", f);  
3 printf("PI = %12.2f| next\n", f);  
4 printf("PI = %012.7f| next\n", f);  
5 printf("PI = %-012.7f| next\n", f);
```

Ausgabe

```
1 PI =      3.141593| next  
2 PI =           3.14| next  
3 PI = 0003.1415927| next  
4 PI = 3.1415927  | next
```

Das Video / die Folien findest du auch online:
www.freitagrunde.org/Ckurs2009/Vortrag02



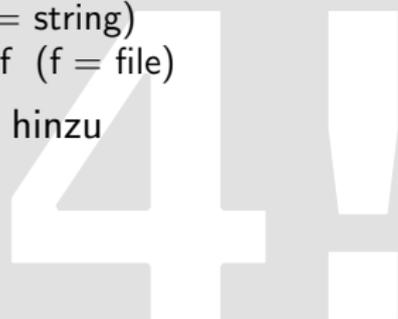
- 1 Wiederholung
- 2 Ausgabe
 - Formate
 - Ziele der Ausgabe
- 3 Eingabe
- 4 Ausblick



4!

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

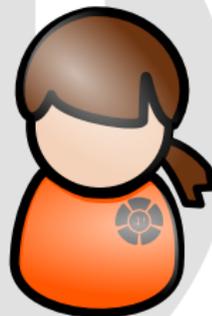
A large, light gray graphic in the background of the slide, consisting of a circle containing a large white number '4' followed by a large white exclamation mark '!'.

Ausgabe in einen String

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

Beispiel: Ausgabe in einen String

```
1 char pcString[256];  
2 sprintf(pcString, "PI = %12.9f\n", 3.14f);
```



Ausgabe in einen String

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

Beispiel: Ausgabe in einen String

```
1 char pcString[256];  
2 sprintf(pcString, "PI = %12.9f\n", 3.14f);
```

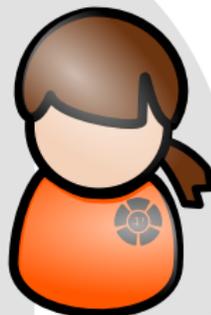
Inhalt von `pcString`:
"PI = 3.14"



Beispiel: Ausgabe in eine Datei

```
1 FILE *pFile = fopen("./test.dat", "w");  
2 fprintf(pFile, "HelloFile, %10.1f\n", 3.14f);  
3 fclose(pFile);
```

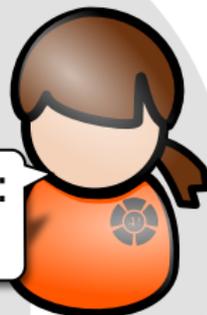
4



Beispiel: Ausgabe in eine Datei

```
1 FILE *pFile = fopen("./test.dat", "w");  
2 fprintf(pFile, "HelloFile, %10.1f\n", 3.14f);  
3 fclose(pFile);
```

Inhalt von ./test.dat:
"HelloFile, 3.1"



Ausgabe in Dateien

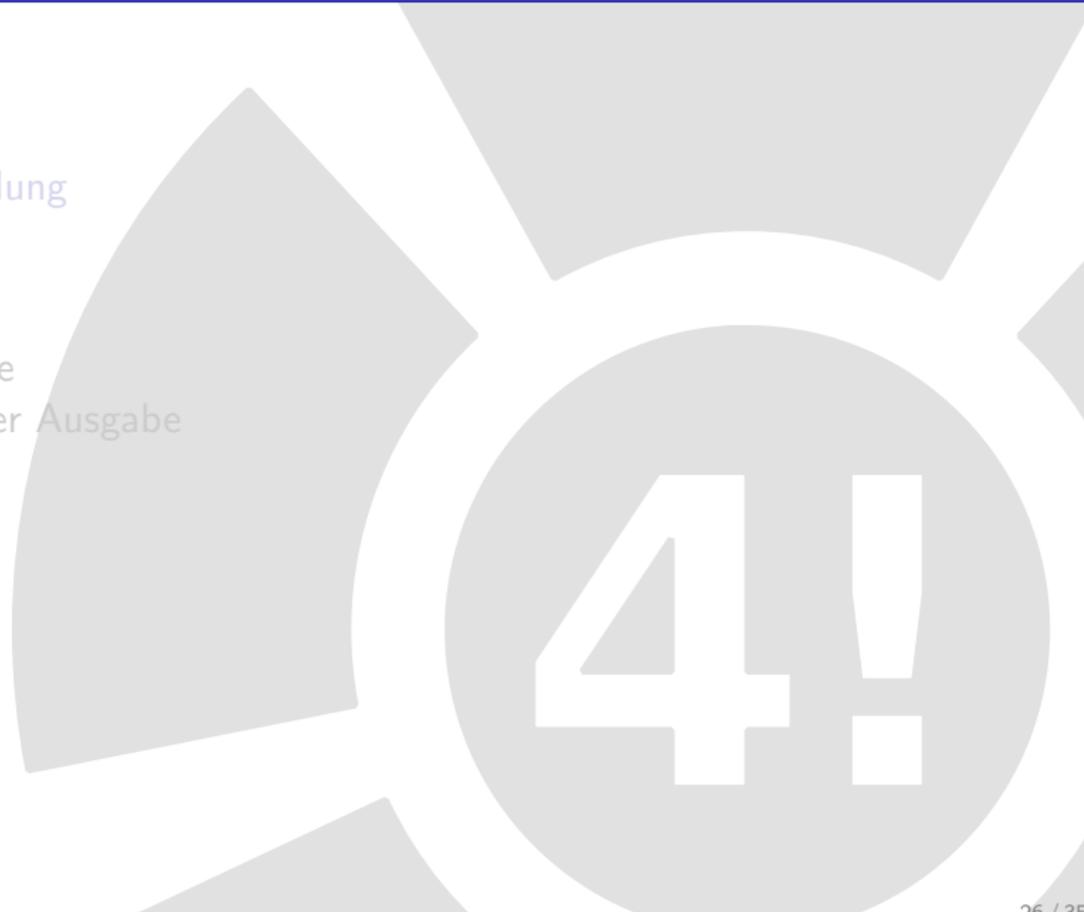
- `fopen` öffnet eine Datei
 - Erster Parameter: Name der zu öffnenden Datei
 - Zweiter Parameter: Modus der Datei (lesen: "r", schreiben: "w")
 - Rückgabewert: Pointer zur FILE-Struktur
 - Rückgabewert kann aber auch leer sein
 - Leer bedeutet, Rückgabewert ist NULL
 - Kann auftreten, wenn zu öffnende Datei nicht existiert
- `fprintf` braucht den Rückgabewert von `fopen`
- Nur so kann `fprintf` in die gerade geöffnete Datei schreiben.
- `fclose` schließt eine geöffnete Datei
 - Bekommt als Parameter ein FILE*
 - Speichert alle Änderungen
 - Auf einigen System kann eine Datei nur von einem Benutzer geöffnet werden
 - Dateien kurz öffnen, schließen so früh wie möglich
- `fflush` speichert Änderungen, ohne Datei zu schließen
 - Bekommt als Parameter nur die Rückgabe von `fopen`

Beispiel: Ausgabe in Dateien

```
1 /* try to open file */
2 FILE *pFile = fopen("./test.dat", "w");
3 if( NULL == pFile )
4 {
5     /* inform user about failure */
6     printf("Error: Could not read file.\n");
7 }
8 else
9 {
10    /* write sth. to this file */
11    fprintf( pFile ,
12            "Hello World\nPI = %10.4f\n4 = %10d\n" ,
13            3.14159265358979f ,
14            (2 + 1 + 1) );
15
16    /* done */
17    fclose( pFile );
18 }
```

- Können überall benutzt werden wo FILE* erwartet wird.
- stdout
 - Standardausgabe
 - `fprintf (stdout, [...]); == printf ([.]);`
 - Ausgabe erfolgt sobald `/n` oder `fflush(stdout)`; angegeben wurde
 - Daher wurden bisher alle Ausgaben mit `/n` beendet
 - Falls Zeile nicht beendet werden soll, bitte `fflush (stdout)`; benutzen
 - Wichtig bei Eingaben
- stderr
 - Fehlerausgabe
 - Standard für Fehlermeldungen
 - Ausgabe erfolgt sofort (Kein `/n` oder `fflush()` nötig, aber erwünscht)
- stdin
 - Standardeingabe
 - Nur lesender Zugriff, nächster Abschnitt

- 1 Wiederholung
- 2 Ausgabe
 - Formate
 - Ziele der Ausgabe
- 3 Eingabe**
- 4 Ausblick



4!

- Eingabe ist analog zu Ausgabe:
- Statt `printf` benutzt man nun `scanf`
- Alle Formate von `printf` gelten auch für `scanf`

A large, light gray graphic in the background consists of several overlapping circular and semi-circular shapes. In the center-right, there is a prominent white circle containing the number '4' and an exclamation mark '!' in a bold, sans-serif font.

- Eingabe ist analog zu Ausgabe:
- Statt printf benutzt man nun scanf
- Alle Formate von printf gelten auch für scanf

Beispiel: Eingabe von Variablen

```
1 int iFirstValue = 0;
2 int iSecondValue = 0;
3
4 int iValues = scanf("%d %d", &iFirstValue ,
5                       &iSecondValue );
```



- Eingabe ist analog zu Ausgabe:
- Statt printf benutzt man nun scanf
- Alle Formate von printf gelten auch für scanf

Beispiel: Eingabe von Variablen

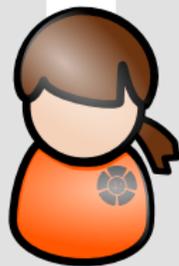
```
1 int iFirstValue = 0;
2 int iSecondValue = 0;
3
4 int iValues = scanf("%d %d", &iFirstValue ,
5                       &iSecondValue );
```

Inhalt von `iFirstValue`
und `iSecondValue` ist
abhängig vom `Benutzer`.



- 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 kommendes CSV-Beispiel)
- Rückabewert beinhaltet Anzahl der gelesenen Daten.
- Daher: Immer den Rückgabewert überprüfen

4



Was tut 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 kommendes CSV-Beispiel)
- Rückabewert beinhaltet Anzahl der gelesenen Daten.
- Daher: Immer den Rückgabewert überprüfen

Im **Beispiel** bekommen wir also **2** als Rückgabe, wenn alles geklappt hat



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.

A large, light gray graphic of a gear or sunburst is centered on the slide. Inside the gear, there is a white circle containing the number '4' followed by an exclamation mark '!' in a bold, sans-serif font.

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.

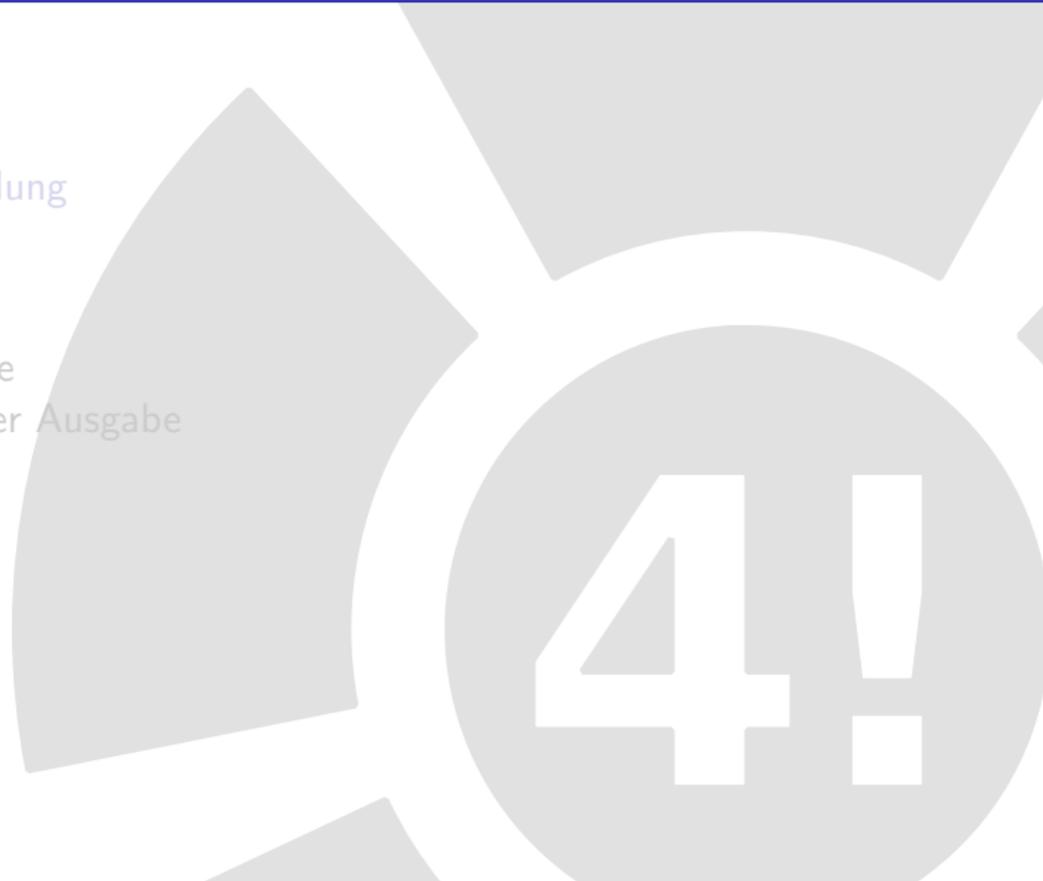
Beispiel: Einlesen einer Datei

```
1 float fValue1 = 0;
2 float fValue2 = 0;
3 /* we want to read a file , so use "r" */
4 FILE *pFile = fopen("./test.dat", "r");
5 if( NULL == pFile)
6 {
7     fprintf(stderr, "Cannot find file.\n");
8     return -1; /* do not continue execution*/
9 } else {
10     int i = fscanf(pFile, "%f %f", &fValue1, &fValue2);
11     if( 2 != i )
12         fprintf(stderr, "Cannot read values.\n");
13 }
```

Beispiel: Einlesen und Ausgeben von 3er Tupeln

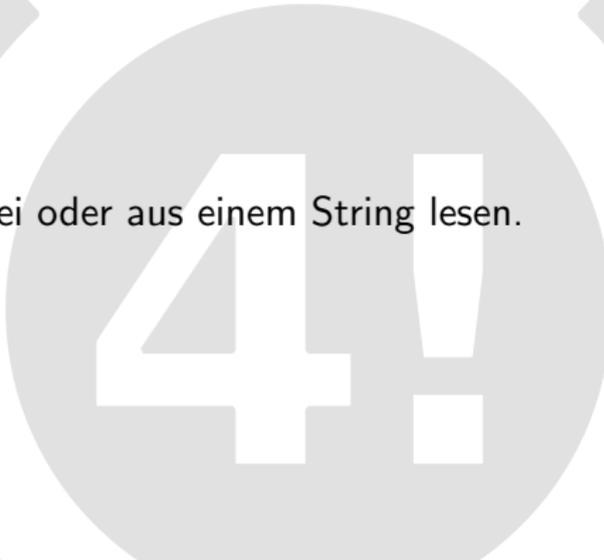
```
1 /* again: we want to read a file */
2 FILE *pFile = fopen("./test.csv", "r");
3 if( NULL == pFile){
4     fprintf(stderr, "Cannot_find_file.\n");
5 }
6 /* read csv file */
7 float a = 0.0f, b = 0.0f, c = 0.0f;
8 int i = 0;
9
10 /* loop to end of file or read error */
11 do{
12     i = fscanf( pFile, "%f %f %f\n",&a, &b, &c);
13     if( 3 == i){
14         printf("%f + %f + %f = %.2f\n",a,b,c, a+b+c);
15     }
16 }while( i == 3 );
17
18 fclose( pFile );
```

- 1 Wiederholung
- 2 Ausgabe
 - Formate
 - Ziele der Ausgabe
- 3 Eingabe
- 4 **Ausblick**



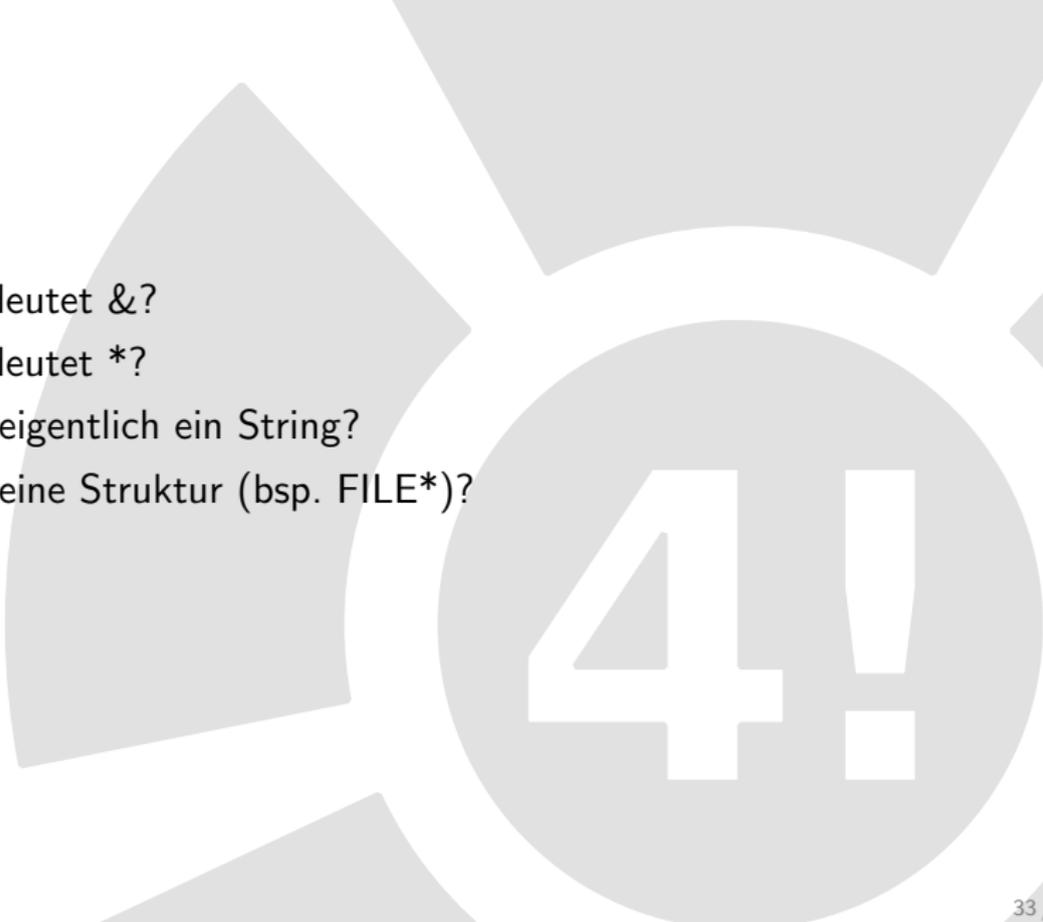
4!

- 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.



4!

- Was bedeutet &?
- Was bedeutet *?
- Was ist eigentlich ein String?
- Was ist eine Struktur (bsp. FILE*)?



4!

Danke

Fragen?





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