

# Angewandte Informatik

Prof. Dr. rer. nat. Peter Gerwinski

12. November 2015

**Sie können diese Vortragsfolien einschließlich Beispielprogramme, Skript und sonstiger Lehrmaterialien unter**

<https://gitlab.cvh-server.de/pgerwinski/ainf.git>

**herunterladen.**

- **Files**  
einzelne Dateien herunterladen
- **Download zip/tar.gz/tar.bz2/tar**  
als Archiv herunterladen
- <https://gitlab.cvh-server.de/pgerwinski/ainf.git>  
mit GIT herunterladen und synchronisieren

# Angewandte Informatik

## 1 Einführung

## 2 Einführung in C

## 3 Bibliotheken

3.1 Der Präprozessor

3.2 Bibliotheken einbinden

3.3 Bibliothek verwenden (Beispiel: OpenGL)

3.4 Projekt organisieren: make

## 4 Algorithmen

4.1 Differentialgleichungen

4.2 Rekursion

4.3 Stack und FIFO

4.4 Aufwandsabschätzungen

## 5 Hardwarenahe Programmierung

...

## 3.4 Projekt organisieren: make

- Regeln
- Makros

## 3.4 Projekt organisieren: make

- Regeln

```
earth-6: earth-6.c opengl-magic.h opengl-magic-test.c \  
    textured-spheres.h textured-spheres.c  
gcc -Wall earth-6.c \  
    -lGL -lGLU -lglut opengl-magic-test.c textured-spheres.c \  
    -o earth-6
```

- Makros

## 3.4 Projekt organisieren: make

- Regeln
- Makros

SOURCES = opengl-magic-test.c textured-spheres.c

INCLUDES = opengl-magic.h textured-spheres.h

LIBS = -lGL -lGLU -lglut

earth-5: earth-5.c \$(SOURCES) \$(INCLUDES)

gcc -Wall earth-5.c \$(SOURCES) \$(LIBS) -o earth-5

earth-6: earth-6.c \$(SOURCES) \$(INCLUDES)

gcc -Wall earth-6.c \$(SOURCES) \$(LIBS) -o earth-6

## 3.4 Projekt organisieren: make

- Regeln
- Makros

SOURCES = opengl-magic.c textured-spheres.c

INCLUDES = opengl-magic.h textured-spheres.h

LIBS = -lGL -lGLU -lglut

%.c \$(SOURCES) \$(INCLUDES)

gcc -Wall \$< \$(SOURCES) \$(LIBS) -o \$@

## 3.4 Projekt organisieren: make

- Regeln
- Makros

SOURCES = opengl-magic.c textured-spheres.c

INCLUDES = opengl-magic.h textured-spheres.h

LIBS = -IGL -IGLU -lglut

%.c \$(SOURCES) \$(INCLUDES)

gcc -Wall \$< \$(SOURCES) \$(LIBS) -o \$@

→ 3 Sprachen: C, Präprozessor, make



# Angewandte Informatik

## 1 Einführung

## 2 Einführung in C

## 3 Bibliotheken

3.1 Der Präprozessor

3.2 Bibliotheken einbinden

3.3 Bibliothek verwenden (Beispiel: OpenGL)

3.4 Projekt organisieren: make

## 4 Algorithmen

4.1 Differentialgleichungen

4.2 Rekursion

4.3 Stack und FIFO

4.4 Aufwandsabschätzungen

## 5 Hardwarenahe Programmierung

...

## 4 Algorithmen

### 4.1 Differentialgleichungen

$$\varphi'(t) = \omega(t)$$

$$\omega'(t) = -\frac{g}{l} \cdot \sin \varphi(t)$$

- Von Hand (analytisch): Lösung raten (Ansatz), Parameter berechnen
- Mit Computer (numerisch): Eulersches Polygonzugverfahren

```
phi += dt * omega;  
omega += - dt * g / l * sin (phi);
```

Praktikumsaufgabe: Basketball

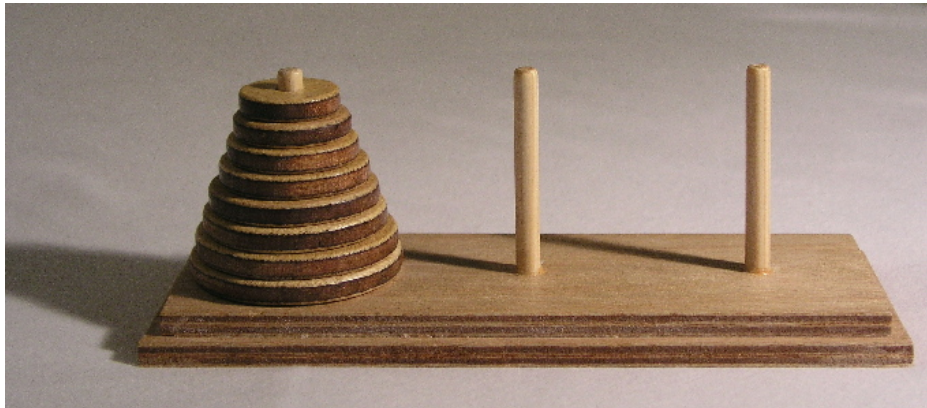
## 4.2 Rekursion

Vollständige Induktion:  $\left. \begin{array}{l} \text{Aussage gilt für } n = 1 \\ \text{Schluß von } n - 1 \text{ auf } n \end{array} \right\} \text{ Aussage gilt für alle } n \in \mathbb{N}$

## 4.2 Rekursion

Vollständige Induktion:  $\left. \begin{array}{l} \text{Aussage gilt für } n = 1 \\ \text{Schluß von } n - 1 \text{ auf } n \end{array} \right\} \text{ Aussage gilt für alle } n \in \mathbb{N}$

Türme von Hanoi

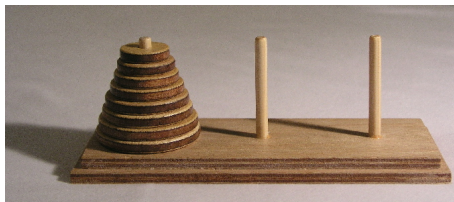


## 4.2 Rekursion

Vollständige Induktion:  $\left. \begin{array}{l} \text{Aussage gilt für } n = 1 \\ \text{Schluß von } n - 1 \text{ auf } n \end{array} \right\} \text{Aussage gilt für alle } n \in \mathbb{N}$

### Türme von Hanoi

- 64 Scheiben, 3 Plätze, immer 1 Scheibe verschieben
- Ziel: Turm verschieben
- Es dürfen nur kleinere Scheiben auf größeren liegen.

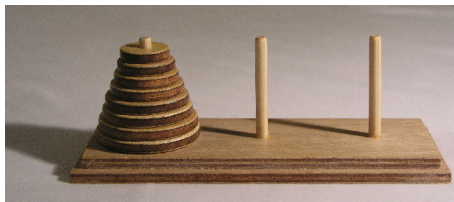


## 4.2 Rekursion

Vollständige Induktion:  $\left. \begin{array}{l} \text{Aussage gilt für } n = 1 \\ \text{Schluß von } n - 1 \text{ auf } n \end{array} \right\} \text{ Aussage gilt für alle } n \in \mathbb{N}$

### Türme von Hanoi

- 64 Scheiben, 3 Plätze, immer 1 Scheibe verschieben
- Ziel: Turm verschieben
- Es dürfen nur kleinere Scheiben auf größeren liegen.
- $n = 1$  Scheibe: fertig
- Wenn  $n - 1$  Scheiben verschiebbar: schiebe  $n - 1$  Scheiben auf Hilfsplatz, verschiebe die darunterliegende, hole  $n - 1$  Scheiben von Hilfsplatz



## 4.2 Rekursion

Vollständige Induktion: 
$$\left. \begin{array}{l} \text{Aussage gilt für } n = 1 \\ \text{Schluß von } n - 1 \text{ auf } n \end{array} \right\} \text{ Aussage gilt für alle } n \in \mathbb{N}$$

### Türme von Hanoi

- 64 Scheiben, 3 Plätze, immer 1 Scheibe verschieben
- Ziel: Turm verschieben
- Es dürfen nur kleinere Scheiben auf größeren liegen.
- $n = 1$  Scheibe: fertig
- Wenn  $n - 1$  Scheiben verschiebbar: schiebe  $n - 1$  Scheiben auf Hilfsplatz, verschiebe die darunterliegende, hole  $n - 1$  Scheiben von Hilfsplatz

```
void verschiebe (int n, int start, int ziel)
{
    if (n == 1)
        verschiebe_1_scheibe (start, ziel);
    else
    {
        verschiebe (1, start, hilfsplatz);
        verschiebe (n - 1, start, ziel);
        verschiebe (1, hilfsplatz, ziel);
    }
}
```

## 4.2 Rekursion

### Floodfill

```
void fill (int x, int y, char c, char o)
{
    if (get_point (x, y) == o)
    {
        put_point (x, y, c);
        fill (x + 1, y, c, o);
        fill (x - 1, y, c, o);
        fill (x, y + 1, c, o);
        fill (x, y - 1, c, o);
    }
}
```



## 4.2 Rekursion

### Floodfill

```
void fill (int x, int y, char c, char o)
{
    if (get_point (x, y) == o)
    {
        put_point (x, y, c);
        fill (x + 1, y, c, o);
        fill (x - 1, y, c, o);
        fill (x, y + 1, c, o);
        fill (x, y - 1, c, o);
    }
}
```

Aufgabe: Schreiben Sie eine Bibliothek für „Text-Grafik“ mit folgenden Funktionen:

- **void clear (char c)**  
Bildschirm auf Zeichen **c** löschen
- **void put\_point (int x, int y, char c)**  
Punkt setzen
- **char get\_point (int x, int y)**  
Punkt lesen
- **void fill (int x, int y, char c, char o)**  
Fläche in der „Farbe“ **o**,  
die den Punkt **(x, y)** enthält,  
mit der „Farbe“ **c** ausmalen
- **void display (void)**  
Inhalt des Arrays auf dem  
Bildschirm ausgeben

Hinweis: Verwenden Sie ein Array als „Bildschirm“.

# Angewandte Informatik

## 1 Einführung

## 2 Einführung in C

## 3 Bibliotheken

3.1 Der Präprozessor

3.2 Bibliotheken einbinden

3.3 Bibliothek verwenden (Beispiel: OpenGL)

3.4 Projekt organisieren: make

## 4 Algorithmen

4.1 Differentialgleichungen

4.2 Rekursion

4.3 Stack und FIFO

4.4 Aufwandsabschätzungen

## 5 Hardwarenahe Programmierung

...