

# Angewandte Informatik

Prof. Dr. rer. nat. Peter Gerwinski

26. November 2015

# Angewandte Informatik

- 1 Einführung**
- 2 Einführung in C**
- 3 Bibliotheken**
- 4 Algorithmen**
  - 4.1** Differentialgleichungen
  - 4.2** Rekursion
  - 4.3** Stack und FIFO
  - 4.4** Aufwandsabschätzungen
- 5 Hardwarenahe Programmierung**
- ...

# Ergänzungen

## OpenGL

- **Doppelte Pufferung**

2 „Bildschirme“: einer zum Zeichnen; einer wird angezeigt  
[opengl-magic-double.c](#), [gluSwapBuffers\(\)](#)

- **Im Display-Handler ([draw\(\)](#)) wirklich nur zeichnen!**

Wir haben nicht unter Kontrolle, wann, wie oft oder ob überhaupt diese Funktion aufgerufen wird.

# Ergänzungen

## Umgang mit Bibliotheken

- **Separates Compilieren**

```
$ gcc -c -Wall opengl-magic.c
$ gcc -c -Wall textured-spheres.c
$ gcc -Wall orbit-x.c -lGL -lGLU -lglut \
    opengl-magic.o textured-spheres.o -o orbit-x
```

- Die **Link-Reihenfolge** kann eine Rolle spielen!

```
$ gcc -Wall orbit-x.c \
    opengl-magic.o textured-spheres.o \
    -lGL -lGLU -lglut -o orbit-x
```

Notfalls:

```
$ gcc -Wall orbit-x.c -lGL -lGLU -lglut \
    opengl-magic.o textured-spheres.o \
    -lGL -lGLU -lglut -o orbit-x
```

# Ergänzungen

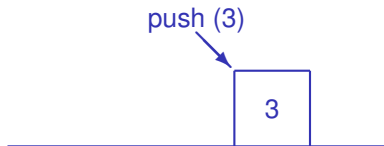
## Parameter des Hauptprogramms

```
#include <stdio.h>
```

```
int main (int argc, char **argv)
{
    printf ("argc=%d\n", argc);
    for (int i = 0; i < argc; i++)
        printf ("argv[%d]= \"%s\"\n", i, argv[i]);
    return 0;
}
```

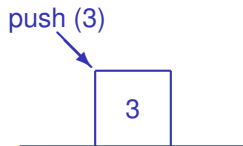
## 4.3 Stack und FIFO

„First In – First Out“



FIFO = Queue = Reihe

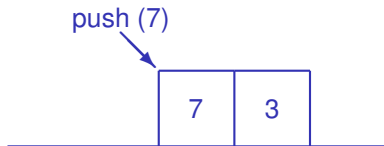
„Last In – First Out“



LIFO = Stack = Stapel

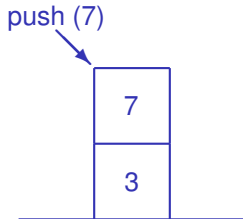
## 4.3 Stack und FIFO

„First In – First Out“



FIFO = Queue = Reihe

„Last In – First Out“



LIFO = Stack = Stapel

## 4.3 Stack und FIFO

„First In – First Out“

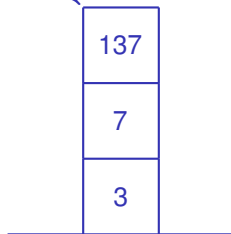
push (137)



FIFO = Queue = Reihe

„Last In – First Out“

push (137)



LIFO = Stack = Stapel



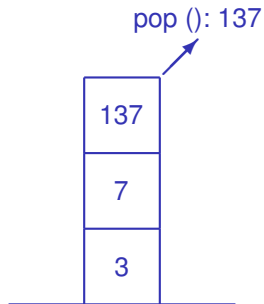
## 4.3 Stack und FIFO

„First In – First Out“



FIFO = Queue = Reihe

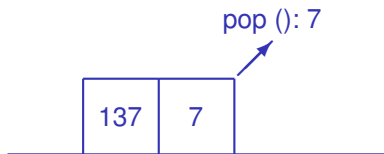
„Last In – First Out“



LIFO = Stack = Stapel

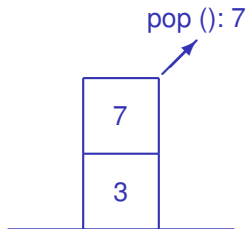
## 4.3 Stack und FIFO

„First In – First Out“



FIFO = Queue = Reihe

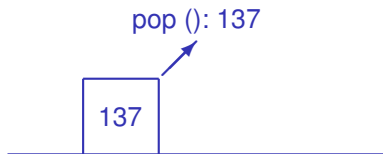
„Last In – First Out“



LIFO = Stack = Stapel

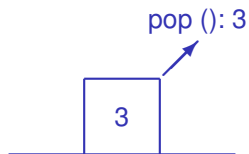
## 4.3 Stack und FIFO

„First In – First Out“



FIFO = Queue = Reihe

„Last In – First Out“

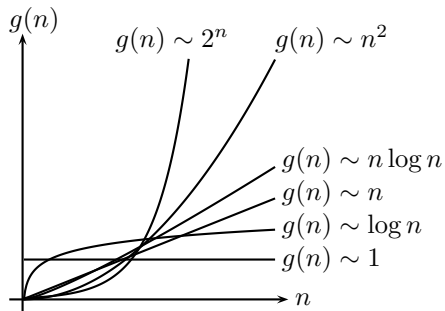


LIFO = Stack = Stapel

## 4.4 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen



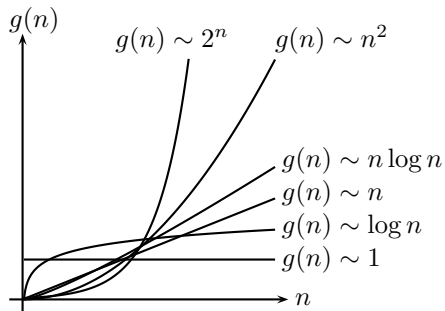
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 4.4 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$



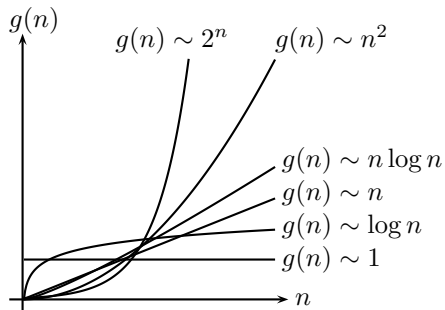
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 4.4 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort



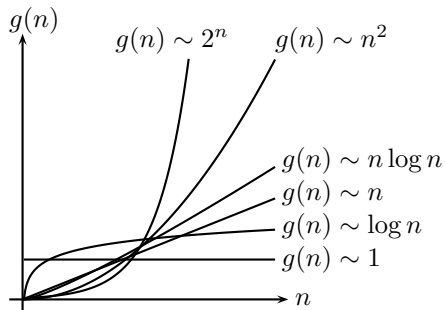
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 4.4 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort:  $\mathcal{O}(n^2)$



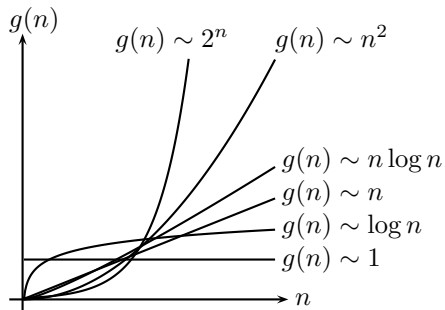
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 4.4 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort:  $\mathcal{O}(n^2)$
- Während Maximumsuche prüfen,  
abbrechen, falls schon sortiert  
→ Bubblesort



$n$ : Eingabedaten

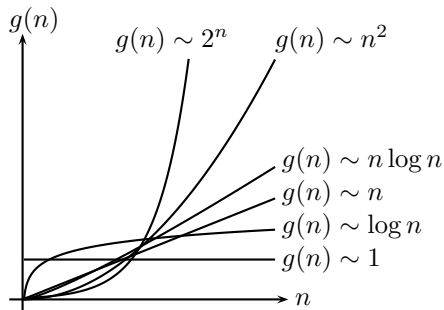
$g(n)$ : Rechenzeit



## 4.4 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort:  $\mathcal{O}(n^2)$
- Während Maximumsuche prüfen, abbrechen, falls schon sortiert  
→ Bubblesort:  $\mathcal{O}(n)$  bis  $\mathcal{O}(n^2)$



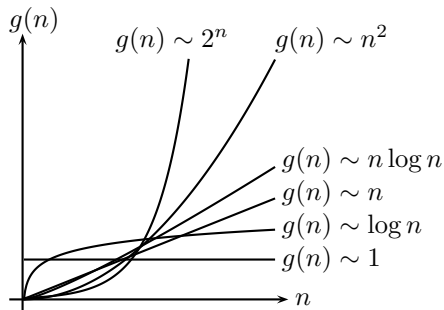
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 4.4 Aufwandsabschätzungen

Beispiel: Sortieralgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort:  $\mathcal{O}(n^2)$
- Während Maximumsuche prüfen, abbrechen, falls schon sortiert  
→ Bubblesort:  $\mathcal{O}(n)$  bis  $\mathcal{O}(n^2)$
- Rekursiv sortieren  
→ Quicksort



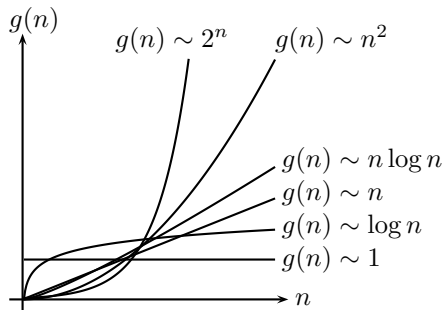
$n$ : Eingabedaten

$g(n)$ : Rechenzeit

## 4.4 Aufwandsabschätzungen

Beispiel: Sortialgorithmen

- Maximum suchen:  $\mathcal{O}(n)$
- Maximum ans Ende tauschen  
→ Selectionsort:  $\mathcal{O}(n^2)$
- Während Maximumsuche prüfen, abbrechen, falls schon sortiert  
→ Bubblesort:  $\mathcal{O}(n)$  bis  $\mathcal{O}(n^2)$
- Rekursiv sortieren  
→ Quicksort:  $\mathcal{O}(n \log n)$  bis  $\mathcal{O}(n^2)$



$n$ : Eingabedaten

$g(n)$ : Rechenzeit

# Angewandte Informatik

**1 Einführung**

**2 Einführung in C**

**3 Bibliotheken**

**4 Algorithmen**

**4.1** Differentialgleichungen

**4.2** Rekursion

**4.3** Stack und FIFO

**4.4** Aufwandsabschätzungen

**5 Hardwarenahe Programmierung**

...