

Wintersemester 2007/08

**Einführung in die Informatik für  
Naturwissenschaftler und Ingenieure  
(alias Einführung in die Programmierung)  
(Vorlesung)**

Prof. Dr. Günter Rudolph

Fachbereich Informatik

Lehrstuhl für Algorithm Engineering





# Kapitel 1: Einleitung

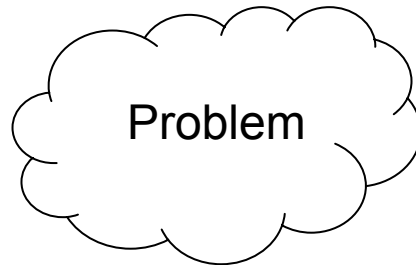
---

## Gliederung

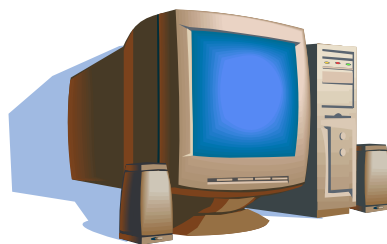
- Zum Begriff „Informatik“
- Zum Begriff „Algorithmus“
- Zum Begriff „Programmieren“



# Was ist Informatik?



?



Maschine

Erste Näherung:

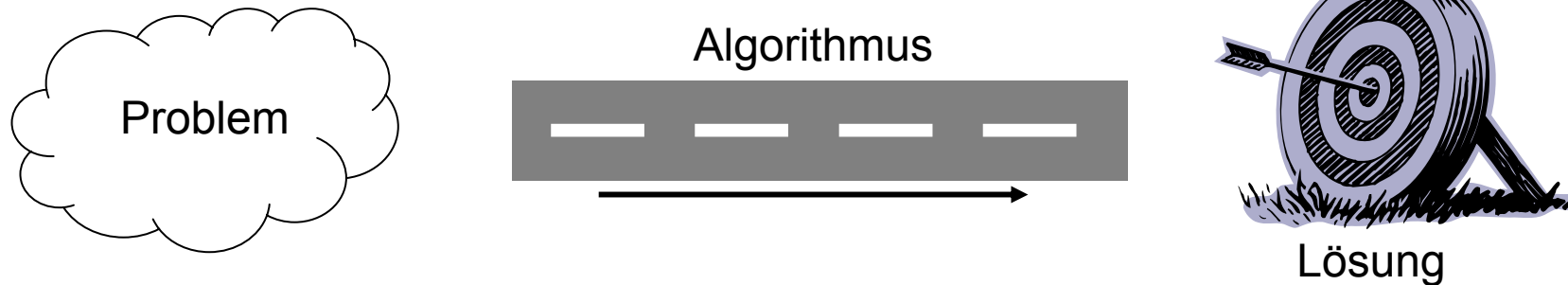
Die Informatik handelt vom **maschinellen Problemlösen.**

bzw.

Die Informatik ist die Wissenschaft von der **methodischen Beherrschung algorithmisch lösbarer Probleme.**



# Was ist ein Algorithmus?



## **Algorithmus:** (*anschaulich*)

Beschreibung eines Weges vom Problem zur Lösung.

Randbedingungen:

1. Der **Weg** muss **formal** so **präzise** definiert sein, dass er im Prinzip von einer Maschine (rein mechanisch) gegangen werden kann.
2. Problem und Lösung müssen vorher **formal spezifiziert** werden.



## Algorithmus: Beispiele

### Algorithmen-ähnlich:

- Kochrezepte
- Bastelanleitungen
- Partituren
- ...



- Selten exakt ausformuliert
- Interpretationsspielräume
- Unschärfe („fuzzy“), Vagheit

### Algorithmen aus der Schulzeit:

- „schriftliche“ Addition zweier Zahlen
- „schriftliche“ Multiplikation zweier Zahlen
- ...

$$\begin{array}{r} 2436 \\ +1383 \\ \hline 3819 \end{array}$$



## Algorithmus: Formalere Definition

---

Ein **Algorithmus** gibt an, wie Eingabedaten schrittweise in Ausgabedaten umgewandelt werden.

Er beschreibt also eine Abbildung

$$f: E \rightarrow A$$

von der Menge der Eingabedaten  $E$  in die Menge der Ausgabedaten  $A$  und wie die Abbildung zu „berechnen“ ist.

Ein Algorithmus wird **korrekt** genannt, wenn er

1. den spezifizierten Zusammenhang zwischen  $E$  und  $A$  für alle Eingaben aus  $E$  erfüllt und wenn er
2. terminiert.



## Algorithmus: Beispiel „Finde jüngste Person hier im Raum“

### Problemanalyse

- Annahme: Es sind  $n \geq 1$  Personen im Raum
- Formulierung „jüngste Person“ eindeutig?  $\Rightarrow$  Nein!
  - a) Genauigkeit der Altersangabe in Sekunden oder Tage oder Jahre?
  - b) Es könnten  $\geq 2$  Personen gleichen Alters im Raum sein!

zu a) Annahme: Jahre

zu b) Reihenfolge der Personen festlegen; wähle 1. Person, die minimales Alter hat

### Spezifikation

Gegeben: Folge von  $n$  Altersangaben  $a_1, a_2, \dots, a_n$  in Jahren,  $n \geq 1$

Gesucht:  $a_j = \min(a_1, a_2, \dots, a_n)$ , wobei  $j$  die erste Stelle in der Folge sei, an der das Minimum auftritt



## Algorithmus: Beispiel „Finde jüngste Person hier im Raum“ (2)

### Algorithmmentwurf

Gegeben: Folge von  $n$  Altersangaben  $a_1, a_2, \dots, a_n$  in Jahren,  $n \geq 1$

Gesucht:  $a_j = \min(a_1, a_2, \dots, a_n)$ , wobei  $j$  die erste Stelle in der Folge sei, an der das Minimum auftritt

(1) [ Wähle 1. Kandidat ] Setze  $j = 1$  und  $x = a_j$ .

(2) [ Suchlauf ]

Setze  $i = 2$ .

Solange  $i \leq n$  gilt,

falls  $a_i < x$ , dann setze  $j = i$  und  $x = a_j$ .

[ jetzt gilt  $a_j = \min(a_1, \dots, a_i)$  ]

erhöhe  $i$  um 1

(3) [ Ausgabe ] Person  $j$  mit Alter  $x$  ist eine jüngste Person





## Algorithmus: Beispiel „Finde jüngste Person hier im Raum“ (3)

### Beispiel

Gegeben: Folge von 8 Altersangaben 20, 21, 20, 19, 18, 19, 18, 20

|     | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | j |
|-----|----|----|----|----|----|----|----|----|---|
| (1) | 20 |    |    |    |    |    |    |    | 1 |
| (2) | 20 | 21 |    |    |    |    |    |    | 1 |
| (2) | 20 | 21 | 20 |    |    |    |    |    | 1 |
| (2) | 20 | 21 | 20 | 19 |    |    |    |    | 4 |
| (2) | 20 | 21 | 20 | 19 | 18 |    |    |    | 5 |
| (2) | 20 | 21 | 20 | 19 | 18 | 19 |    |    | 5 |
| (2) | 20 | 21 | 20 | 19 | 18 | 19 | 18 |    | 5 |
| (2) | 20 | 21 | 20 | 19 | 18 | 19 | 18 | 20 | 5 |



## Algorithmus: Beispiel „Finde jüngste Person hier im Raum“ (4)

### Korrektheit

Behauptung: Der Algorithmus ist korrekt.

Beweis: Wenn der Algorithmus anhält, dann ist

a)  $a_j = \min(a_1, \dots, a_i)$  mit  $1 \leq i \leq n$ .

Das gilt für  $i = 1$  nach Schritt (1) und während des Suchlaufs invariant für alle  $i$  an der angegebenen Stelle.

b)  $j$  ist die erste Stelle, an der (a) gilt, weil im Fall  $a_i = x$  kein Austausch mehr stattfindet, sondern nur bei  $a_i < x$ .

Der Algorithmus hält an, nachdem  $i = n$  war.

q.e.d.



## Algorithmus: Beispiel „Finde jüngste Person hier im Raum“ (5)

### Effizienz

Wir messen den Zeitaufwand in Einheiten E.

| Aktion                 | Aufwand | Häufigkeit der Aktion |
|------------------------|---------|-----------------------|
| Setze $j = 1, x = a_j$ | 2 E     | 1                     |
| Setze $i = 2$          | 1 E     | 1                     |
| Test $i \leq n$        | 1 E     | n                     |
| Test $a_i < x$         | 1 E     | n - 1                 |
| Setze $j = i, x = a_j$ | 2 E     | A                     |
| Erhöhe i               | 1 E     | n - 1                 |

Insgesamt also:

$$T(n) = 2 + 1 + n + (n-1) + 2A + (n-1) E = (3n + 2A + 1) E$$



## Algorithmus: Beispiel „Finde jüngste Person hier im Raum“ (6)

### Effizienz

$$T(n) = (3n + 2A + 1) E \quad \Rightarrow \text{Welche Werte kann } A \text{ annehmen?}$$

Hier: zwei Szenarien

1. Schlimmster Fall (engl. *worst case*):  $A = n - 1$   
d.h., das Alter aller Personen ist paarweise verschieden und es ist in der Aufzählung absteigend sortiert

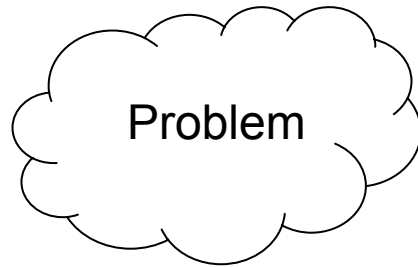
$$\Rightarrow T_{\max}(n) = (5n - 1) E$$

2. Bester Fall (engl. *best case*):  $A = 0$   
d.h., erste Person in der Aufzählung ist bereits eine jüngste Person

$$\Rightarrow T_{\min}(n) = (3n + 1) E$$



# Vom Problem zur Maschine ...

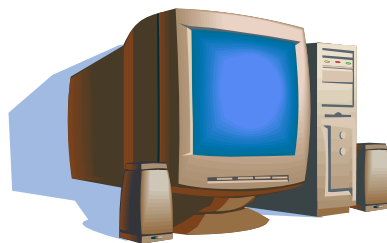


Algorithmus



```
#include <iostream>
using namespace std;
int main() {
    cout << "Hi!";
    return 0;
}
```

Programm



Maschine



## Schwerpunkte der Veranstaltung EINI

- |                              |   |                  |
|------------------------------|---|------------------|
| ● Problemanalyse             | } | Übungen +        |
| ● Spezifikation              |   | Praktikum +      |
| ● Algorithmenentwurf         |   | Vorlesung        |
| ● Korrektheit / Verifikation |   | → BSc Informatik |
| ● Effizienzuntersuchungen    |   | → DAP 2          |
| ● Programmieren (kodieren)   |   | Vorlesung        |
| ● Testen / Fehlerbeseitigung |   | Übung, Praktikum |
| ● Wartung / Pflege           |   | —                |



# Programmiersprachen

---

## Vorbemerkungen

- Denken  $\Leftrightarrow$  Sprache
- Fachsprachen
  - für komplexe Sachverhalte mit akzeptablem Aufwand
  - für Fachleute
- Programmiersprache
  - syntaktische Form, um Problemlösungen zu beschreiben
  - muss von Maschine interpretiert werden können

$\Rightarrow$  der Programmierer muss genau wissen,  
wie die Maschine die vereinbarte Sprache interpretiert!



# Programmiersprachen

---

## Historisches in Kürze:

- In den 1930er Jahren:  
Präzisierung von berechenbaren Funktionen, Algorithmus, Turing-Maschine,  $\mu$ -rekursive Funktion,  $\lambda$ -Kalkül, ...
- In den 1940er Jahren:  
reale technische Realisierung von Rechenmaschinen (von-Neumann)  
→ Konrad Zuse (Z3), Howard Aiken (Mark I), Eckert/Mauchly (ENIAC), ...
- zuerst: Programmierung in Maschinensprache (oder mit Kabeln)
- dann: Assemblersprachen  
→ Ersetzung von Zahlen (Maschinencode) durch mnemonische Bezeichnungen  
→ leichter zu merken, z.B. ADD, JMP, BNE, ...
- darauf aufbauend: höhere Programmiersprachen  
→ sind abstrakter, ermöglichen komplexe Sachverhalten einfacher auszudrücken  
→ Übersetzungsalgorithmen erlauben Rückführung auf niedrigere Sprachen  
→ Compiler, Assembler, ...





## Klassifikation nach Denkweisen (Paradigmen)

- Imperativ / prozedural  
→ FORTRAN, BASIC, PASCAL, **C**, ...
- Funktional  
→ LISP, SCHEME, HASKELL, ...
- Relationen- oder Logik-bezogen  
→ PROLOG, ...
- Objektorientiert  
→ Smalltalk, **C++**, Java, C#, ...

Mehr dazu am Semesterende, wenn Sie C und C++ kennen gelernt haben!