Algorithmen und Datenstrukturen

Prof. Dr. Ralf Möller

Universität zu Lübeck

Institut für Informationssysteme

Stefan Werner (Übungen)

sowie viele Tutoren

Teilnehmerkreis und Voraussetzungen

Studiengänge

- Bachelor **Informatik**
- Bachelor/Master **Mathematik in Medizin und Lebenswissenschaften**
- Bachelor **Medieninformatik**
- Bachelor **Medizinische Ingenieurwissenschaft**
- Bachelor **Medizinische Informatik**

Voraussetzungen

- Einführung in die Programmierung
- Lineare Algebra und Diskrete Strukturen 1

Organisatorisches: Übungen

- **Start**: Montag, 20. April 2015
- **Übungen**: Montags 10-12, 12-14, 14-16, 16-18 Uhr, verschiedene Räume,
  Anmeldung über Moodle nach dieser Veranstaltung
- **Übungsaufgaben** stehen jeweils kurz nach der Vorlesung am Freitag über Moodle bereit
- Aufgaben sollen in einer **2-er Gruppe** bearbeitet werden
- **Abgabe der Lösungen** erfolgt bis Donnerstag in der jeweils folgenden Woche nach Ausgabe bis 12 Uhr in der IFIS-Teeküche (1 Kasten pro Gruppe)
- Bitte unbedingt Namen, Matrikelnummern und Übungsgruppennummern auf Abgaben vermerken

Organisatorisches: Prüfung

- Die **Eintragung in den Kurs** und in eine Übungsgruppe ist **Voraussetzung**, um an dem Modul Algorithmen und Datenstrukturen teilnehmen zu können und Zugriff auf die Unterlagen zu erhalten
- Am Ende des Semesters findet eine **Klausur** statt
- **Voraussetzung** zur Teilnahme an der Klausur sind mindestens **50% der gesamtmöglichen Punkte aller Übungszettel**

Literatur

Th. Cormen, C.E. Leiserson, R. Rivest, C. Stein,
Algorithmen: Eine Einführung,
4. Auflage, Oldenbourg, 2013

M. Dietzfelbinger, K. Mehlhorn, P. Sanders
Algorithmen und Datenstrukturen - Die Grundwerkzeuge,
Springer, 2014

R. Sedgewick, K. Wayne,
Algorithmen und Datenstrukturen,
4. Auflage, Pearson, 2014

Literatur

T. Ottmann, P. Widmayer,
Algorithmen und Datenstrukturen,
Spektrum 1997

U. Schöning,
Algorithmen - kurz gefasst,
Spektrum, 1997

Allgemeine Lernziele in diesem Kurs

- Weg **vom Problem zum Algorithmus** gehen können
  - **Auswahl** eines Algorithmus aus Alternativen unter Bezugnahme auf vorliegende Daten und deren Struktur
  - **Entwicklung** eines Algorithmus mitsamt geeigneter Datenstrukturen (Terminierung, Korrektheit, ...)
- **Analyse von Algorithmen** durchführen
  - Anwachsen der Laufzeit bei Vergrößerung der Eingabe
- Erste Schritte in Bezug auf die **Analyse von Problemen** gehen können
  - Ja, Probleme sind etwas anderes als Algorithmen!
  - Probleme können in gewisser Weise „schwer“ sein
  - Prüfung, ob Algorithmus optimal

Beispiel 1: Aufsummieren der Elemente eines Feldes A[1..n]

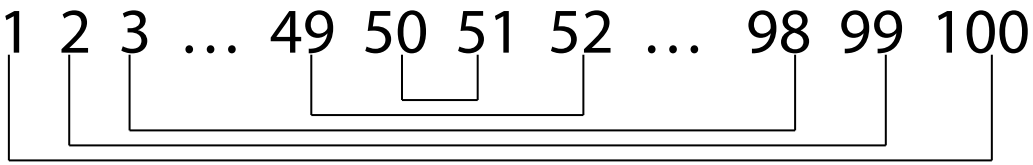
- Algorithmus?

- summe(A) = $\sum_{i=1}^{n} A[i]$
- Aufwand?
- Wenn A n Elemente hat, n Schritte!
- Der Aufwand wird *linear* genannt

Spezialisierung des Problems

- Vorwissen: A[i] = i
- Das Problem wird sehr viel einfacher!
- Entwurfsmuster: Ein-Schritt-Berechnung

Ausnutzen der Einschränkung



Summe jedes Paares: 101

50 Paare: $101 * 50 = 5050$

- Lösungsverfahren:

```
function summe-2(A)
    n ← length(A)
    return (n+1)*(n/2)
```

- Nach Carl Friedrich Gauß (ca. 1786)
- Aufwand?
- Konstant! (hängt nicht von n ab)

Algorithmen: Notation durch Programme

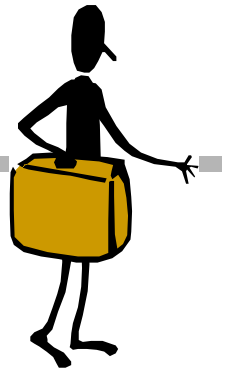
- Annahme: Serielle Ausführung
- Vgl. **Vorlesung „Einführung in die Programmierung“**
  - Variablen, Felder A[...]
  - Zuweisungen ← (manchmal auch **:=**)
  - Fallunterscheidungen **if then else**
    - Vergleich und Berechnungen für Bedingungstest
  - Schleifen **while do**
    - Vergleich und Berechnungen für Bedingungstest
  - **procedure**, **function**
  - Auf Folien wird der jeweilige Skopus durch Einrückung ausgedrückt

Ein erstes Problem: In-situ-Sortierproblem

- **Gegeben: A[1..n] : N**
  - Feld (Array) A von n Zahlen aus N (natürliche Zahlen)
- **Gesucht:**
  - Transformation S von A, so dass gilt: $\forall\, 1 \leq i < j \leq n$: $A[i] \leq A[j]$
  - Nebenbedingung: Es wird intern kein weiteres Feld gleicher (oder auch nur fast gleicher Größe) verwendet
  - Also: Gesucht ist ein Verfahren **S**, so dass **{ P } S { Q }** gilt (Notation nach Hoare)
    - Vorbedingung: **P = true** (keine Einschränkung)
    - Nachbedingung: **Q = $\forall\, 1 \leq i < j \leq n$: $A[i] \leq A[j]$**
    - Nebenbedingung: nur „konstant“ viel zusätzlicher Speicher (feste Anzahl von Hilfsvariablen)

C.A.R. Hoare, An axiomatic basis for computer programming. Communications of the ACM, Vol. 12, S. 567 - 583, **1969**

Zusammenfassung, Kernpunkte



- In dieser Vorlesungseinheit:
  - Ein-Schritt-Berechnung
- Nächste Vorlesung
  - Verkleinerungsprinzip
  - Teile und Herrsche
- „Später“:
  - Vollständige Suchverfahren
    (z.B. Rücksetzen, Verzweigen und Begrenzen)
  - Approximative Such- und Berechnungsverfahren
    (z.B. gierige Suche)
  - Schrittweise Annäherung
  - Dynamisches Programmieren (Berechnung von Teilen und deren Kombination, Wiederverwendung von Zwischenergebnissen)