Algorithmen und Datenstrukturen
1. Algorithmisches Denken

Prof. Dr.-Ing. Marc Stamminger








FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

DEPARTMENT INFORMATIK
LEHRSTUHL FÜR PROGRAMMIERSYSTEME
(INFORMATIK 2)

Worum geht es in dieser Lehreinheit?

- Algorithmen und Datenstrukturen bilden das „Fundament“ für Analyse, Entwurf und Realisierung komplexer Softwaresysteme, zu denen Kompetenzen im Rahmen des Informatikstudiums vermittelt werden.
- Um *algorithmisch* zu denken ist es erforderlich, komplexe Vorgänge in Teilschritte gliedern und diese Teilschritte unter Verwendung so genannter *Kontrollstrukturen* (z. B. Sequenz, Fallunterscheidung, Wiederholung) strukturieren zu können.

  Dies erfolgt zunächst an Alltagsvorgängen und wird nachfolgend schrittweise formalisiert und auf die Programmiersprache *Java* übertragen.

Lernziele: Was sollen Sie am Ende dieser Lehreinheit können?

- ein lebensweltliches Problem abstrahieren und spezifizieren können
- einen gegebenen Algorithmus mit den Elementen Variable, Wertzuweisung, Alternative, Schleife, Block „von Hand“ ausführen können
- zu einer Problemspezifikation einen Algorithmus in Pseudocode-Form angeben können
- einen Algorithmus als Ablauf-Diagramm darstellen können

Gliederung der Lehreinheit

Einordnung der LV „Algorithmen und Datenstrukturen“ (I)

- Was hat ein Informatikstudium im Allgemeinen und diese Vorlesung im Speziellen mit dem Bauwesen gemein?
- Bauingenieure, Architekten, Innenarchitekten, Handwerker etc. planen und bauen gemeinsam komplexe Bauwerke.



- Im Informatikstudium erlernen Sie Planung und „Bau“ komplexer Hard- und Softwaresysteme.

Einordnung der LV „Algorithmen und Datenstrukturen“ (II)

- Damit komplexe Bauwerke entstehen können, bedarf es wichtiger handwerklicher Tätigkeiten: Wände bauen, Fenster einsetzen, Kabel verlegen, ...

...

- In der LV „Algorithmen und Datenstrukturen“ erlernen Sie vergleichbare Grundlagen für die Analyse, Entwurf und Realisierung komplexer Softwaresysteme.

Einordnung der LV „Algorithmen und Datenstrukturen“ (III)

- *LV „Konzeptionelle Modellierung“*
  - Bauwesen: Wie entwirft man Baupläne für große Bauvorhaben? Wie gestaltet man 3D-Modelle von Objekten? ...
  - Informatik: Wie entwirft man „Baupläne“ für große Softwaresysteme? Welche „Architektur“ soll ein Softwaresystem haben? Was soll im geplanten System in welcher Reihenfolge geschehen?
- *LV „Softwareentwicklung in Großprojekten“*
  - Bauwesen: Wie entwirft und organisiert man große Bauvorhaben? Planung, Entwurf, Umsetzung
  - Informatik: Wie entwirft und organisiert man große Softwareprojekte? Planung, Entwurf, Umsetzung
- *Weitere LV*: „Parallele und funktionale Programmierung“, „Implementierung von Datenbanksystemen“, ...

Gliederung der Lehreinheit

Was ist Informatik? (I)

- ***Informatik*** ist ein Kunstwort aus den 60er Jahren, das die Assoziationen Informatik gleich Information und Technik oder Information und Mathematik erwecken sollte. Bei der Begriffsbildung sollte durchaus bewusst ein Gegensatz zum amerikanischen Begriff „computer science“ aufgebaut werden, um zu verdeutlichen, dass die Wissenschaft Informatik nicht nur auf Computer beschränkt ist.

  Saake/Sattler

- In der ***Informatik*** geht es genauso wenig um Computer wie in der Astronomie um Teleskope.

  Edsger Dijkstra

Teilgebiete der Informatik

Theoretische Informatik
- Formale Sprachen
- Automatentheorie
- Komplexitätstheorie
- …

Praktische Informatik
- Übersetzerbau
- Softwaretechnik
- Betriebssysteme
-

Kerninformatik

Technische Informatik
- Rechnerarchitektur
- Prozessdatenverarbeitung
- VLSI-Entwurf
- …

Anwendungen der Informatik
- Wirtschaftsinformatik
- Bioinformatik
- Rechtsinformatik
- …

„Bindestrich-Informatiken“

Ing.-Anteil

Didaktik der Informatik
- Verfügbarmachung von in der Fachwissenschaft gewonnenen Erkenntnissen für Schule oder allgemein für Aus-, Fort- und Weiterbildung von Kindern und Erwachsenen

Gesellschaftliche Bezüge der Informatik
- Informatik und Gesellschaft
- Untersuchung der gesellschaftlichen Auswirkungen

Was ist Informatik? (II)

- ***Informatik*** („computer science“) ist die Wissenschaft von der Modellierung, der Analyse, dem Entwurf und der Realisierung (technischer) Systeme zur systematischen (insb. automatischen) Informationsverarbeitung.

  - systematische Verarbeitung → ***Algorithmus***
  - digital repräsentierte Informationen → ***Daten***

Gliederung der Lehreinheit

Algorithmisches Denken: Algorithmen im Alltag (I)

Zutaten

300 g weiche Butter, 270 g Zucker,
1 Beutel Vanillezucker, 1 Rum-Aroma,
1 Prise Salz, 5 Eier, 375 g Weizen-
mehl, 12 g Backpulver, 3 EL Milch,
20 g Kakaopulver, 20 g Zucker,
3 EL Milch, Puderzucker

Zubereitung

Weiche Butter geschmeidig rühren, nach und nach Zucker, Vanillezucker, Rum-Aroma und Salz zugeben und solange rühren, bis eine gebundene Masse entstanden ist. Die Eier einzeln einrühren. Mehl und Backpulver vermischen und abwechselnd esslöffelweise mit der Milch einrühren (nur so viel Milch verwenden, dass der Teig schwer reißend von einem Löffel fällt). ...

Algorithmisches Denken: Algorithmen im Alltag (II)

... 2/3 des Teigs in eine Marmorkuchenform füllen. Kakao mit Zucker vermischen und Milch einrühren, das Ganze unter den restlichen Teig rühren. Den dunklen Teig auf dem hellen verteilen und mit einer Gabel spiralförmig durch die Teigschichten ziehen. Den Kuchen backen bei 190° C, ca. 60 Min. (Wenn er oben zu dunkel wird, nach der Hälfte der Backzeit mit einem Stück Alufolie abdecken). Den erkalteten Kuchen mit Puderzucker bestäuben. (Quelle: www.chefkoch.de)

Algorithmische Eigenschaften

- endliche Länge der Beschreibung
- Terminierung (Verfahren endet irgendwann)
- eindeutige Reihenfolge der Operationen
- eindeutige Wirkung der Anweisungsfolge (beim Backen nicht immer gegeben, Kuchen können leicht anders aussehen)

Arbeitsauftrag für Sie

Erklären Sie Ihrem Banknachbarn präzise, welche Schritte auf Ihrem jeweiligen Mobiltelefon erforderlich sind, um die Textnachricht „AuD ist super!“ als SMS an die Telefonnummer 0123456789 zu senden.

Mögliche Beobachtungen

- Erklärungen enthalten bestimmte Formulierungsmuster
- „Als erstes mache ich …, danach … und dann …“ (***Sequenz***)
- „Wenn … gilt, dann mache ich …, sonst …“ (***Fallunterscheidung/Alternative***)
- „Das mache ich solange, bis …“, „Solange … gilt, wiederhole ich folgende Schritte: …“ (***bedingte Wiederholung***)
- „Das wiederhole ich genau 10 mal.“ (***Wiederholung mit fester Anzahl***)

- Um sich einer präzisen schrittweisen „algorithmischen Beschreibung“ eines Vorgangs zu nähern, ist es erforderlich, diesen unter Verwendung solcher Sprachmittel zu beschreiben.

Algorithmus

- Ein ***Algorithmus*** ist eine Folge „einfacher“ Anweisungen, die folgende Eigenschaften aufweist:
  - ***Endlichkeit***:
    Die Beschreibung ist endlich lang.
  - ***Terminierung***:
    Nach Durchführung endlich vieler Operationen kommt das Verfahren zum Stillstand.
  - ***eindeutige Reihenfolge***:
    Die Reihenfolge, in der Operationen anzuwenden sind, ist festgelegt.
  - ***eindeutige Wirkung***:
    Die Wirkung jeder Anweisung der Anweisungsfolge und damit der gesamten Folge ist eindeutig festgelegt.
- **Hinweis:** die Eigenschaft der Terminierung wird manchmal auch weggelassen, um auch z. B. auch nicht abbrechende Server-Prozesse zu umfassen

Aufgabe

Wie können Sie alle miteinander möglichst effizient die in diesem Hörsaal jetzt gerade anwesenden Studierenden zählen?

Bedenken Sie bei Ihren Verfahren jeweils: Endlichkeit, Terminierung, eindeutige Reihenfolge, eindeutige Wirkung

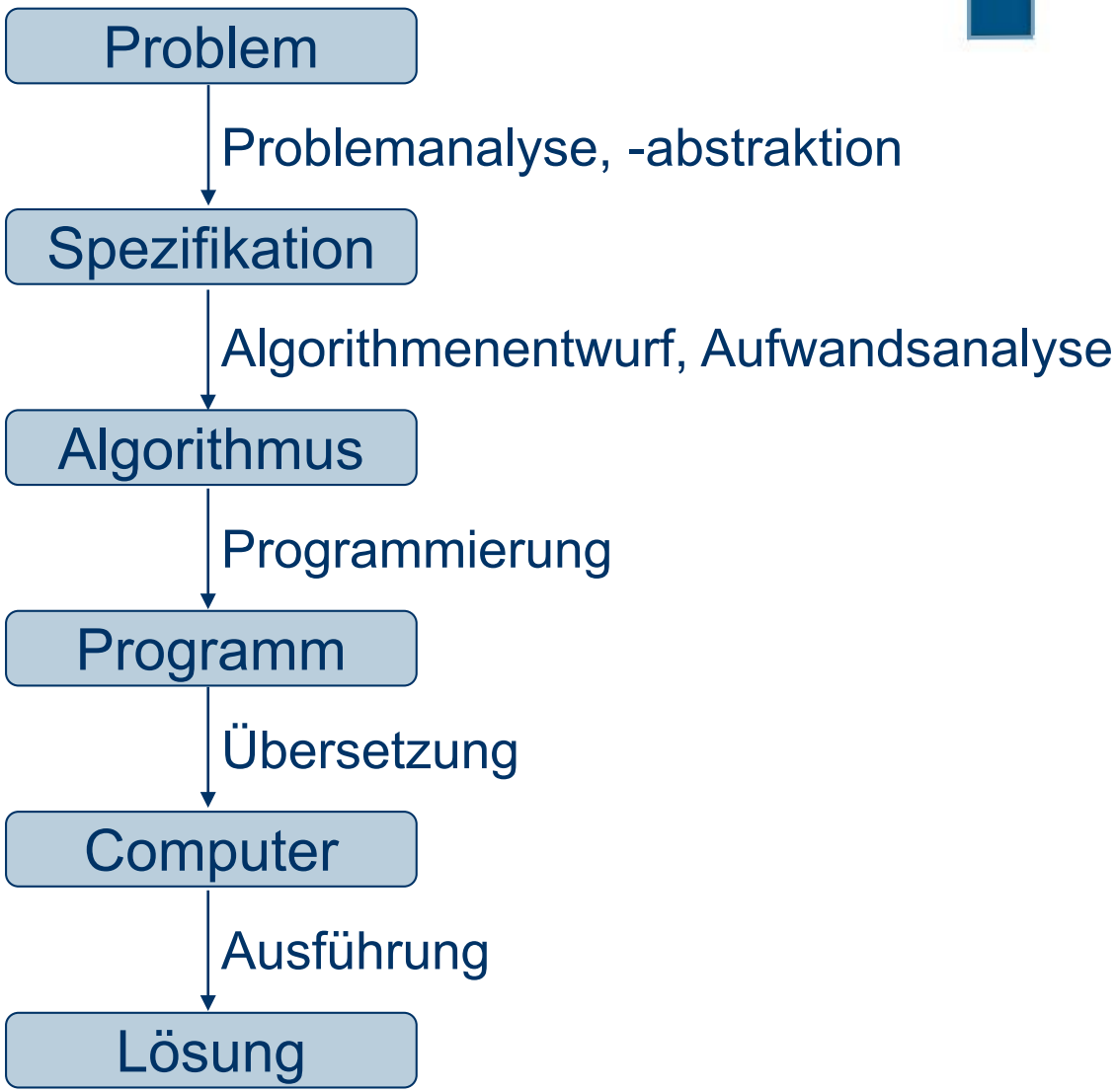
Vorgehensweise bei der Lösung von Programmierproblemen (I)

Problem verstehen und präzise beschreiben

Problemlösung in algorithmischer Form beschreiben

Algorithmus in einer Programmiersprache implementieren

Übersetzung des Programms in Maschinencode



Vorgehensweise bei der Lösung von Programmierproblemen (II)

Überprüfung: beschreibt die Spezifikation wirklich das Problem und nicht mehr und nicht weniger?

Nachweis, dass Algorithmus das spezifizierte Problem korrekt löst

Setzt das Programm den Algorithmus korrekt um?

Vom Problem über den Algorithmus zum Programm (I)

1. *Problem formulieren*
   - Ziel: jüngsten Studierenden in dieser Vorlesung finden

2. *Problemanalyse, Problemabstraktion, Problemspezifikation*
   - Problemanalyse
     - Gibt es überhaupt eine Lösung? Gibt es genau eine Lösung?
     - Es könnte mehrere jüngste Studierende geben.
   - Problemabstraktion
     - Was ist „Alter“?
       a. Alter in Jahren? Messbar durch positive, ganze Zahl
       b. Alter in Jahren, Monaten und Tagen?

       Wir entscheiden uns für Variante a.
     - Was ist „jüngster“?
       - erforderlich: Ordnungsrelation
       - positive Zahlen: „<“ – Relation

Vom Problem über den Algorithmus zum Programm (II)

- Müssen Personenangaben über die Studierenden vorhanden sein?
  a. Studierende in Vorlesung anwesend
  b. Liste mit Personendaten (u. a. Alter)

  Annahme: Variante b
- Interessiert uns tatsächlich der Name oder nur das Alter des jüngsten Studierenden?
  - Überarbeitung der Problemformulierung:
    Finde das Alter des jüngsten Studierenden.

□ Abstrakte Spezifikation

- Problem: Minimum einer Menge von Zahlen
- gegeben: Folge $a_0, ..., a_{n-1}$ von positiven ganzen Zahlen, $n > 0$
- gesucht: kleinster Wert a der gegebenen Zahlen, d. h. $a = \min(a_0, ..., a_{n-1})$

Vom Problem über den Algorithmus zum Programm (III)

3. *Algorithmenentwurf*
   - Top-Down-Strategie (auch: „Divide and Conquer“-Strategie):
     - Gesamtproblem in Teilprobleme zerlegen und diese einzeln lösen
     - Gesamtlösung aus Teillösungen zusammensetzen
   - Bottom-Up-Strategie
     - Gesamtproblem besteht aus Teilproblemen
     - Gesamtlösung aus Teillösungen zusammensetzen
   - hier (einfacher):
     - Menge elementweise durchgehen und bisher kleinstes Element merken
     - „Menge elementweise durchgehen“: Folge $a_0, a_1, \ldots a_{n-1}$ von Elementen
     - „elementweise durchgehen“: Index i des aktuellen Elements a_i merken
     - „bisher kleinstes Element merken“: *merker* für bislang kleinsten Wert
     - „bisher kleinstes“: aktuelles Element a_i mit dem gemerkten vergleichen
     - führt zu Pseudocode

Vom Problem über den Algorithmus zum Programm (IV)

Algorithmus Minimumsuche
(sog. ***Pseudocode***)

Programmierung

Algorithmus Minimumsuche
(Ausschnitt eines Java-Programms)

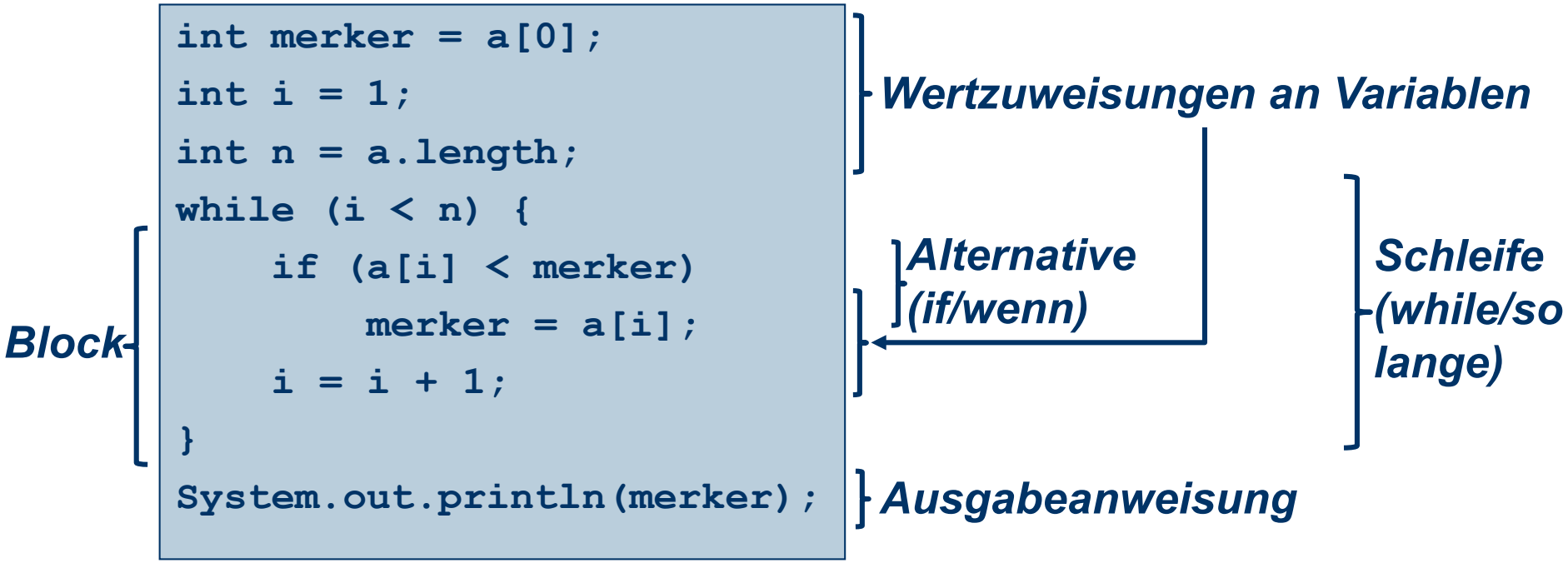
```
setze merker auf a_0;
setze i auf 1;

solange i < n ist, fuehre aus:
    wenn a_i < merker, dann
        setze merker auf a_i;
    erhöhe i um 1;

gib merker aus;
```

```
int merker = a[0];
int i = 1;
int n = a.length;
while (i < n) {
    if (a[i] < merker)
        merker = a[i];
    i = i + 1;
}
System.out.println(merker);
```

Zentrale Begriffe von Algorithmen



Schreibtischlauf zum Algorithmus Minimumsuche

- Gegebene Folge **a**:

15,	23,	22,	21,	22,	18,	19,	17,	14,	16
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

```
int merker = a[0];
int i = 1;
int n = a.length;
while (i < n) {
    if (a[i] < merker)
        merker = a[i];
    i = i + 1;
}
System.out.println(merker);
```

i	n	merker
1	10	15
2	10	15
3	10	15
4	10	15
5	10	15
6	10	15
7	10	15
8	10	15
9	10	14
10	10	14

Vom Problem über den Algorithmus zum Programm (V)

4. *Nachweis der Korrektheit (Semantik, Verifikation)*
   - Terminiert der Algorithmus, d. h. hört er irgendwann einmal auf zu arbeiten?
   - Liefert er das richtige Ergebnis?
   - Vorgehensweise: Wirkung von Einzelschritten des Algorithmus spezifizieren (z. B. über Aussagen, die Zustand einer Berechnung nach Ausführung eines Einzelschrittes in Abhängigkeit des Zustandes vor der Ausführung des Schrittes beschreiben; dann: Zusammensetzen der Aussagen entlang der Einzelschritte)

5. *Aufwandsanalyse*
   - Welche Rechnerressourcen (Speicher zur Ablage von Daten, Ausführungszeit) werden bei der Abarbeitung eines Algorithmus benötigt?
   - Anzahl der Schritte der Minimumssuche auf n gegebenen Zahlen ist proportional zu n (jedes Element muss einmal angesehen werden, für jedes Element konstante Anzahl von Schritten) (optimal, wenn 1 Prozessor vorh.)
   - Speicherbedarf: proportional zu n, da neben den n Elementen nur Minimum und i gemerkt werden müssen