Einführung in die Informatik

Algorithms

Vom Problem zum Algorithmus und zum Programm

Wolfram Burgard
Cyrill Stachniss

Motivation und Einleitung

- In der Informatik sucht man im Normalfall nach **Verfahren zur Lösung von Problemen**.
- Eine zentrale Fragestellung ist wie man solche Verfahren beschreibt.
- Man ist meist daran interessiert **von einer konkreten Programmiersprache zu abstrahieren**,
- weil es **für ein- und dasselbe Problem unterschiedliche Programme mit verschiedenen Eigenschaften** (z.B. bezüglich Laufzeit) geben kann.
- Dabei stellt man an diese Verfahren noch bestimmte Anforderungen und bezeichnet sie als **Algorithmen**.
- **Programme** sind dann nur noch die **Umsetzung dieser Algorithmen in einer speziellen Programmiersprache**.

Der Begriff „Algorithmus“

- Der Begriff „Algorithmus“ geht auf den persischen Mathematiker und Astronomen Ibn Musa Al-Chwarismi zurück.
- Im 9. Jahrhundert hat er das Lehrbuch „Kitab al jabr w’almuqabala“ („Regeln der Wiedereinsetzung und Reduktion“) geschrieben.
- Im folgenden werden wir verschiedene Fragestellungen untersuchen:
    1. Was sind Algorithmen?
    2. Wie erstellt man Algorithmen?
    3. Wie untersucht man Algorithmen?
    4. Wie beschreibt man Algorithmen?

Handlungsanweisungen

Im täglichen Leben begegnen uns **Handlungsanweisungen** aller Art, wie zum Beispiel die folgenden:

- Ärztliche Verordnung: Nimm dreimal täglich 15 Tropfen Asperix vor den Mahlzeiten.
- Waschanleitung: Bei 60 Grad waschen; Waschmittelzugabe in Abhängigkeit von der Wasserhärte nach Angaben des Herstellers.
- Einfahrvorschrift für Autos: Im 2. Gang nicht über 50 km/h, im 3. Gang nicht über 80 km/h, im 4. Gang nicht über 120 km/h; nach 1000 gefahrenen km Motor- und Getriebe Ölwechsel.
- Spielregel: . . . bei einer 6 darf noch einmal gewürfelt werden . . .
- Koch- oder Backrezepte
- ...

Aspekte von Handlungsanweisungen

Wir unterscheiden drei verschiedene Aspekte:

1. Der **Text einer Handlungsanweisung**,
2. der **Ausführende** und
3. die **Ausführung**.

Im Kontext der Informatik sind dies

1. der **Algorithmus**,
2. der **Prozessor** und
3. der **Prozess**.

Eigenschaften von Handlungsanweisungen

- Einzelne Anweisungen werden stets in bestimmter Reihenfolge ausgeführt.
- Diese kann mit der textuellen Reihenfolge der Beschreibung der Handlungsanweisungen übereinstimmen. Sie kann aber auch von Bedingungen abhängig gemacht werden.
- Bisweilen ist es auch erlaubt, Handlungsanweisungen *nebenläufig*, d.h. nicht sequentiell oder nacheinander, sondern parallel oder gleichzeitig, auszuführen, d.h. die zeitliche Reihenfolge wird dann nicht festgelegt.
- Schließlich wird bei allen, auch bei Alltagsanweisungen, ein Unterschied zwischen ihrer Beschreibung im Text und den Daten gemacht.

Dies findet sich auch bei Algorithmen wieder.

Problematische Handlungsanweisungen (1)

1. Starte mit der Zahl 3.
2. Addiere 0,1.
3. Addiere 0,04.
4. Addiere 0,001.
5. Addiere 0,0005.
6. Addiere 0,00009.
7. ...

Diese Handlungsanweisung hat **keine endliche Länge** .

Problematische Handlungsanweisungen (2)

Zur Berechnung der dritten Wurzel einer Zahl x verfahre wie folgt:

1. Erfrage x.

2. Setze r auf 1.

3. Wiederhole

```
r := r-(r*r*r -x)/(3 *r *r);
```

Die **Ausführung** dieser Handlungsanweisung **hält nicht an**.

Präzision

Aus einer Zubereitungsanleitung:

> „Die Suppe aus der Dose nach Vorschrift zubereiten. Sie können zum Schluss noch einige Spargelstückchen hinzugeben. Den Schinken in Streifen schneiden ...“

Aus der Spielanleitung zu „Hugo, das Schlossgespenst“:

> „... Der mutigste Spieler setzt zunächst einen seiner Gäste auf ein beliebiges freies Feld ... Wer von Euch jetzt schon zittert, darf beginnen.“

Anleitungen für Spiele oder Kochrezepte sind häufig **unpräzise**. Die **Ausführungsreihenfolge** einzelner Anweisungen ist nicht genau **festgelegt**. Auch **steht häufig nicht fest, wie begonnen wird**.

Allgemeinheit

Betrachten Sie die folgende Handlungsanweisung:

Um in das Glottertal zu kommen,

- verlassen Sie die Georges-Köhler-Allee und biegen Sie links ab.
- ...
- Fahren Sie dann geradeaus auf die B3 Richtung Emmendingen und
- biegen Sie hinter Gundelfingen auf die B294 Richtung Waldkirch ab.
- Nehmen Sie dann die erste Ausfahrt und biegen Sie rechts ab.
- ...

Diese **Handlungsanweisung** ist präzise, führt aber nur dann zum Ziel, wenn man in Freiburg an der Fakultät für Angewandte Wissenschaften startet.

Sie ist **so spezifisch**, dass sie **nur ein einziges Problem löst**. Für einen Algorithmus **fehlt** ihr **die notwendige Allgemeinheit**.

Eine intuitive Definition des Algorithmenbegriffs

Definition: *Ein Algorithmus ist eine präzise, endliche Verarbeitungsvorschrift, die genau festlegt, wie die Instanzen einer Klasse von Problemen gelöst werden. Ein Algorithmus liefert eine Funktion (Abbildung), die festlegt, wie aus einer zulässigen Eingabe die Ausgabe ermittelt werden kann.*

Eigenschaften von Algorithmen (1)

Finitheit: Die Beschreibung des Verfahrens ist von endlicher Länge (*statische* Finitheit) und zu jedem Zeitpunkt der Abarbeitung des Algorithmus hat der Algorithmus nur endlich viele Ressourcen belegt (*dynamische* Finitheit).

Terminierung: Algorithmen, die nach Durchführung endlich vieler Schritte (Operationen) zum Stillstand kommen, heißen *terminierend*. In der Informatik spielen aber auch viele nichtterminierende Programme eine große Rolle. Sie werden beispielsweise zur Prozessteuerung, Datenübertragung in Netzen und Mensch-Maschine Kommunikation benutzt. (Man spricht in diesem Kontext auch von reaktiven Systemen.)

Eigenschaften von Algorithmen (2)

Determinismus: Liegt die Reihenfolge, in der die einzelnen Schritte eines Algorithmus ausgeführt werden, eindeutig fest, hängt sie also nur von den Eingabedaten ab, so spricht man von *determinierten* Algorithmen. Daneben spielen in der Theorie auch *nichtdeterminierte* und in der Praxis zunehmend auch *stochastische*, d.h. von einem zufälligen Ereignis abhängige Algorithmen eine Rolle.

Effektivität: Die Wirkung einer einzelnen Anweisung eines Algorithmus ist eindeutig festgelegt.

Wie beschreibt man Algorithmen?

- Es gibt eine Vielzahl von Techniken, Algorithmen zu beschreiben.
- Hierzu gehören beispielsweise umgangssprachliche Formulierungen, spezielle, abstrakte Maschinenmodelle, wie z.B. Register- oder Turingmaschinen, aber auch spezielle Sprachen.
- Im folgenden wollen wir zunächst ausgehen von einer Formulierung durch so genannte **`while`-Programme**.

`while`-Programme (1)

- While Programme erlauben es **Variablen** zu verwenden und darin Werte oder Zwischenergebnisse abzuspeichern.
- Wir verwenden beispielsweise `a, b, x, wert,...` zur Bezeichnung von Speicherzellen, die beliebige (ganze, reelle, ...) Zahlen aufnehmen können.
- Neben Variablen die Zahlen aufnehmen können gibt es auch so genannte **boolesche Variablen**. Eine boolesche Variable hat nur die **beiden Zustände wahr und falsch** (true and false). Meist benutzt man boolesche Ausdrücke um Bedingungen zu modellieren.

while-Programme (2)

Elementare Anweisungen: Sie sind entweder die **leere Anweisung** (`skip`) oder die **Zuweisung**, die es erlaubt, eine Speicherzelle `x` mit dem Wert eines arithmetischen Ausdrucks zu füllen:

```
x := t
```

Dabei ist also `t` ein aus Variablen, Operatoren (wie t, *, /,...), Funktionszeichen und Klammern zusammengesetzter Ausdruck.

```
x := (y + 17) * 3
```

```
u := (u + x) / 2 * x
```

```
B := (x > 5)
```

Eine Wertzuweisung `u := t` setzt u auf den Wert von `t` und terminiert anschließend.

`while`-Programme (3)

Komposition: Bei der (sequentiellen) **Komposition** `s1;s2` zweier Programme `s1` und `s2` wird zunächst `s1` und nach der Terminierung von `s1` dann `s2` ausgeführt.

Selektion: Die Selektion ist eine **bedingte Anweisung** der Form

```
if B then s1 else s2 end
```

Zunächst wird der boolesche Ausdruck `B` ausgewertet. Wenn `B` wahr ist, wird `s1` ausgeführt, andernfalls `s2` ausgeführt.

Iteration: Die Ausführung einer **Schleife**

```
while B do s1 end
```

beginnt mit der Auswertung des booleschen Ausdrucks B. Wenn `B` falsch ist, terminiert die Schleife sofort. Ist `B` wahr, wird `s1` ausgeführt. Nachdem `s1` terminiert, wird der ganze Vorgang wiederholt.

Beispiel (1)

```
x := 23;
y := 17;
while x != 0 do
   x := x - 1;
   y := y + 1;
end
```

Am Ende der Ausführung dieses Algorithmus steht in der Variablen y der Wert `40 = 23+17`.

Beispiel (2)

```
wert := 1;
while n > 0 do
    wert := 2 * wert;
    n := n - 1;
end
```

Das Programmstück berechnet offenbar 2^n, falls anfangs $n > 0$ war.

Wie wir im Laufe dieser Vorlesung sehen werden gibt es zwischen `while`-Programmen und `Java`-Programmen eine starke Ähnlichkeit gibt. Genau genommen sind **`while`-Programme** eine **Untermenge von Java**.

Zusammenfassung

- Ein **Algorithmus ist ein allgemeines Verfahren zur Lösung einer Klasse von Einzelproblemen**.
- Algorithmen haben eine **endliche Länge** und sie sind präzise formuliert.
- Eine **typische Form von Algorithmen** sind **`while`-Programme**.
- **Java ist eine Programmiersprache zur Implementierung von Algorithmen.**