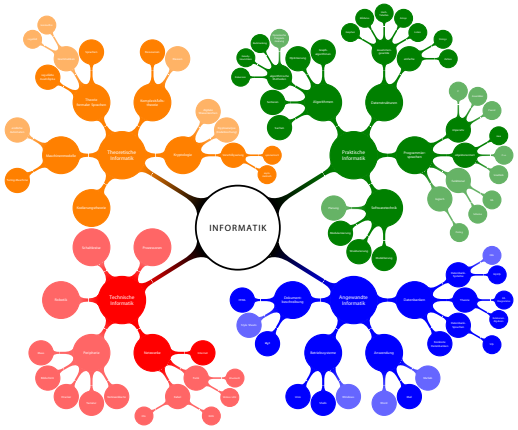
UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

Kapitel 8

Der Algorithmusbegriff

Von der vagen Idee zur Instruktionsfolge

Vorlesung Einführung in die Informatik 1 vom 19. November 2013 von Till Tantau



Lernziele von Kapitel 8

1. Den Begriff des Algorithmus verstehen
2. Einfache Lösungsideen als Algorithmen formulieren können
3. Probleme spezifizieren können
4. Programmiersprachen als Kommunikationsmittel für Algorithmen begreifen
5. Das Konzept des Übersetzers verstehen

Gliederung von Kapitel 8

Über das Wort »Algorithmus«.


كتاب الخوارزمي

Unkown author, public domain

- Das Wort stammt vermutlich vom Namen des Gelehrten *Muhammad ibn Musa, Abu Dscha'far al-Chwarizmi* (∗ ca. 783, † ca. 850)
- Entscheidend war sein Buch *Al-kitab al-muchtasar fi hisab al-dschabr wa-l-muqabala* (»Rechnen durch Ergänzung und Ausgleich«).

Das erste Programm stammt von einer Frau.

Unkown author, public domain

Aus de.wikipedia.org/wiki/Algorithmus

Der erste für einen Computer gedachte Algorithmus wurde 1842 von Ada Lovelace, in ihren Notizen zu Charles Babbages Analytical Engine, festgehalten. Sie gilt deshalb als die erste Programmiererin. Weil Charles Babbage seine Analytical Engine nicht vollenden konnte, wurde Ada Lovelaces Algorithmus nie darauf implementiert.

Duplieren nach Adam Riese (1574).

Unknown author, public domain

Lehret wie du ein zahl zweyfaltigen ſolt.
Thu ihm alſo: Schreib die zahl vor dich
mach ein Linien darunter
heb an zu forderſt
Duplir die erſte Figur. Kompt ein zahl die du mit
einer Figur ſchreiben magſt
ſo ſetz die unden. Wo mit zweyen
ſchreib die erſte
Die andere behalt im ſinn. Darnach duplir die ander
und gib darzu
das du behalten haſt
und ſchreib abermals die erſte Figur
wo zwo vorhanden
und duplir fort bis zur letzten
die ſchreibe ganz aus
als folgende Exempel auſweiſen.

Was sind Algorithmen?

- Ein *Algorithmus* ist eine abstrakte Folge von Handlungsanweisungen zur Lösung eines Problems.
- Eher schwaches Beispiel:
  Backrezept.
- Eher gutes Beispiel:
  Anleitung zum schriftlichen Multiplizieren.

Was ist der Unterschied zwischen Algorithmen und Programmen?

Eigenschaften von Algorithmen:

- Ein Algorithmus beschreibt *abstrakt*, wie ein Problem zu lösen ist.
- Ein Algorithmus ist *nicht* an einen bestimmten Computer oder eine *bestimmte Programmiersprache* gebunden.
- Derselbe Algorithmus kann oft *in vielen Situationen* benutzt werden.

Eigenschaften von Programmen:

1. Programme sind immer ganz *konkrete* Instruktionsfolgen.
2. Programme werden in einer ganz *bestimmten Programmiersprache* geschrieben.
3. Ein Programm löst *genau ein Problem* und nichts anderes.

Algorithmusbeschreibungen können unterschiedlich »genau« sein.

Die einzelnen Handlungsanweisungen eines Algorithmus können ganz unterschiedlich umfangreich sein:

- Eher elementar:
  »1. Tausche die ersten beiden Zahlen in der Liste.«
  »2. Wenn die erste Zahl größer ist als die zweite, mache bei Schritt 19. weiter.«
- Eher komplex:
  »1. Sortiere die Liste.«
  »2. Entferne das Maximum.«

5-Minuten-Aufgabe

Zur Übung

Ein möglicher Algorithmus zum Sortieren von drei Spielkarten funktioniert wie folgt:

1. Falls die linke Karte größer ist als die mittlere, tausche sie.
2. Falls die mittlere Karte größer ist als die rechte, tausche sie.
3. Falls die linke Karte größer ist als die mittlere, tausche sie.

Schreiben Sie einen Algorithmus zum Sortieren von vier Karten auf.
Zusatzfrage: Kommt Ihr Algorithmus mit der minimalen Anzahl an Schritten aus (oder geht es auch schneller)?

Es gibt immer wiederkehrende Strukturen in Algorithmen.

Die allermeisten Algorithmen benutzen Varianten der folgenden so genannten *Steuerungsanweisungen*:

1. Hintereinanderausführung von Anweisungen.
   Beispiel: Erst tue dies, dann jenes, dann dieses.
2. Bedingte Anweisungen.
   Beispiel: Falls dies gilt, tue jenes, sonst dieses.
3. Schleifen.
   Beispiel: Solange jenes gilt, tue folgendes:
   Beispiel: Tue folgendes 100 mal:
4. Sprünge (considered harmful!).
   Beispiel: Mache nun dort weiter.

Steuerungsanweisungen bei Adam Riese

1. Hintereinander-ausführung.
2. Bedingte Anweisungen.
3. Schleifen.
4. Sprünge.

Lehret wie du ein zahl zweyfaltigen ſolt.
Thu ihm alſo: Schreib die zahl vor dich
mach ein Linien darunter
heb an zu forderſt
Duplir die erſte Figur. Kompt ein zahl die du mit
einer Figur ſchreiben magſt
ſo ſetz die unden. Wo mit zweyen
ſchreib die erſte
Die andere behalt im ſinn. Darnach duplir die ander
und gib darzu
das du behalten haſt
und ſchreib abermals die erſte Figur
wo zwo vorhanden
und duplir fort bis zur letzten
die ſchreibe ganz aus
als folgende Exempel auſweiſen.

Steuerungsanweisungen bei Adam Riese

1. *Hintereinander-ausführung.*
2. Bedingte Anweisungen.
3. Schleifen.
4. Sprünge.

Lehret wie du ein zahl zweyfaltigen ſolt.
Thu ihm alſo: Schreib die zahl vor dich
mach ein Linien darunter
heb an zu forderſt
Duplir die erſte Figur. Kompt ein zahl die du mit einer Figur ſchreiben magſt
ſo ſetz die unden. Wo mit zweyen
ſchreib die erſte
Die andere behalt im ſinn. Darnach duplir die ander
und gib darzu
das du behalten haſt
und ſchreib abermals die erſte Figur
wo zwo vorhanden
und duplir fort bis zur letzten
die ſchreibe ganz aus
als folgende Exempel auſweiſen.

Steuerungsanweisungen bei Adam Riese

1. Hintereinander-ausführung.
2. *Bedingte Anweisungen.*
3. Schleifen.
4. Sprünge.

Lehret wie du ein zahl zweyfaltigen ſolt.
Thu ihm alſo: Schreib die zahl vor dich
mach ein Linien darunter
heb an zu forderſt
Duplir die erſte Figur. Kompt ein zahl die du mit
einer Figur ſchreiben magſt
ſo ſetz die unden. Wo mit zweyen
ſchreib die erſte
Die andere behalt im ſinn. Darnach duplir die ander
und gib darzu
das du behalten haſt
und ſchreib abermals die erſte Figur
wo zwo vorhanden
und duplir fort bis zur letzten
die ſchreibe ganz aus
als folgende Exempel auſweiſen.

Steuerungsanweisungen bei Adam Riese

1. Hintereinander-ausführung.
2. Bedingte Anweisungen.
3. *Schleifen.*
4. Sprünge.

Lehret wie du ein zahl zweyfaltigen ſolt.
Thu ihm alſo: Schreib die zahl vor dich
mach ein Linien darunter
heb an zu forderſt
Duplir die erſte Figur. Kompt ein zahl die du mit einer Figur ſchreiben magſt
ſo ſetz die unden. Wo mit zweyen
ſchreib die erſte
Die andere behalt im ſinn. Darnach duplir die ander
und gib darzu
das du behalten haſt
und ſchreib abermals die erſte Figur
wo zwo vorhanden
und duplir fort bis zur letzten
die ſchreibe ganz aus
als folgende Exempel auſweiſen.

Steuerungsanweisungen bei Adam Riese

1. Hintereinander-ausführung.
2. Bedingte Anweisungen.
3. Schleifen.
4. *Sprünge.*

Lehret wie du ein zahl zweyfaltigen ſolt.
Thu ihm alſo: Schreib die zahl vor dich
mach ein Linien darunter
heb an zu forderſt
Duplir die erſte Figur. Kompt ein zahl die du mit einer Figur ſchreiben magſt
ſo ſetz die unden. Wo mit zweyen
ſchreib die erſte
Die andere behalt im ſinn. Darnach duplir die ander
und gib darzu
das du behalten haſt
und ſchreib abermals die erſte Figur
wo zwo vorhanden
und duplir fort bis zur letzten
die ſchreibe ganz aus
als folgende Exempel auſweiſen.

Spezifikationen sind Problembeschreibungen.

- Probleme lassen sich nur dann sinnvoll lösen, wenn das Problem klar gestellt ist.
- Eine »klare Problemstellung« nennt man eine *Spezifikation*.
- Sie sollte die folgende Fragen umfassend beantworten:
  1. Was sind die relevanten Rahmenbedingungen?
  2. Welche Hilfsmittel und Basisoperationen sind zugelassen?
  3. Wann ist eine Lösung korrekt oder zumindest akzeptabel?

Gute Spezifikationen zu erstellen ist schwierig.

- Spezifikationen dienen oft dazu zu klären, was eine zu erstellende Software später leisten soll. Leider wissen Kunden aber nicht, was sie eigentlich wollen.
- Selbst Spezifikationen zu einfachsten Problemen lassen oft Fragen offen.
  Beispiel: Berechne die Summe der ersten *n* Zahlen. Als elementare Operationen stehen Additionen und Vergleiche zur Verfügung.
- Größe Spezifikationen sind in aller Regel in sich widersprüchlich.
  Beispiel: Auf Seite 50 der Spezifikation steht, dass beim Drücken des ersten Knopfes der Text rot wird. Auf Seite 150 derselben Spezifikation steht dann, dass beim Drücken dieses Knopfes der Text grün werden soll.

Von der Idee zur Instruktionsfolge.

Spezifikation

Gegeben ist eine natürliche Zahl. Sie soll verdoppelt werden.

Idee

Verdopple die Zahl stellenweise von rechts nach links und beachte den Übertrag.

Von der Idee zur Instruktionsfolge.

Spezifikation

Gegeben ist eine natürliche Zahl. Sie soll verdoppelt werden.

Algorithmus (Notation von Riese)

Thu ihm alſo: Schreib die zahl vor dich
mach ein Linien darunter
heb an zu forderſt
Duplir die erſte Figur. Kompt ein zahl die du mit einer Figur ſchreiben magſt
ſo ſetz die unden. Wo mit zweyen
ſchreib die erſte
Die andere behalt im ſinn. Darnach duplir die ander
und gib darzu
das du behalten haſt
und ſchreib abermals die erſte Figur
wo zwo vorhanden
und duplir fort bis zur letzten
die ſchreibe ganz aus

Von der Idee zur Instruktionsfolge.

Spezifikation

Gegeben ist eine natürliche Zahl. Sie soll verdoppelt werden.

Algorithmus (modernere Notation)

Für jede Ziffer, beginnend mit der letzten, tue folgendes:

1. Verdopple die Ziffer.
2. Addiere einen eventuell vorhandenen Übertrag hinzu.
3. Schreibe die letzte Ziffer der Summe unter die gerade behandelte Ziffer.
4. Merke die erste Ziffer des Übertrags.

Falls nun noch ein Übertrag vorhanden ist, schreibe diesen links von allem.

Von der Idee zur Instruktionsfolge.

Spezifikation

Gegeben ist eine natürliche Zahl. Sie soll verdoppelt werden.

Algorithmus (Pseudocode)

```
1  input Ziffern z_1, ..., z_n
2  c ← 0
3  for i ← n, ..., 1 do
4      d ← 2z_i + c
5      z'_i ← d mod 10
6      c ← ⌊d/10⌋
7  z'_0 ← c
8  for i ← 0, ..., n do
9      output z'_i
```

Von der Idee zur Instruktionsfolge.

Spezifikation

Gegeben ist eine natürliche Zahl. Sie soll verdoppelt werden.

Algorithmus (Java)

```
void verdopple(int[] ziffern, int n)
{
  int[] ziffern_verdoppelt = new int[n+1];
  int carry = 0;
  for (int i = n; i >= 1; i = i-1)
  {
    int d = 2*ziffern[i] + carry;
    ziffern_verdoppelt[i] = d % 10;
    carry = d / 10;
  }
  ziffern_verdoppelt[0] = carry;
  for (int i = 0; i <= n; i = i+1)
    System.out.print(i);
}
```

Programmiersprachen dienen zum Aufschreiben von Algorithmen.

- In einer Programmiersprache lässt sich ein Algorithmus so formulieren, dass ein Computer ihn versteht.
- Programmiersprachen sind unterschiedlich »maschinennah«: das Spektrum reicht von »Maschinensprache« bis zu Hochsprachen und es gibt hunderte von ihnen.
- Genau wie natürliche Sprachen müssen Programmiersprachen von Menschen mehr oder weniger mühselig erlernt werden.

Übersetzer sind Programme, die Sprachen ineinander übersetzen.

- Die CPU »versteht« lediglich Maschinensprache.
- Wir wollen aber in einer Hochsprache wie Java programmieren,
- weshalb wir einen *Übersetzer* brauchen.
- Programme werden heutzutage von einer ganzen Kette von Übersetzern in immer maschinennähere Sprachen übersetzt.

Zusammenfassung

Algorithmus

Ein *Algorithmus* ist eine abstrakte Folge von Handlungsanweisungen zur Lösung eines Problems.

Programmiersprache

Ein *Programmiersprache* definiert eine Art, Algorithmen aufzuschreiben.

Programm

Ein *Programm* ist ein in einer Programmiersprache verfasster Text, der einen Algorithmus *implementiert.*

Spezifikation

Eine *Spezifikation* legt fest, was ein Programm leisten soll.

Übersetzer

Ein *Übersetzer* ist ein Programm, das Programmtexte aus einer Programmiersprache in eine andere übersetzt.

Zusammenfassung

Der Arbeitsfluss beim Lösen eines Problems.

1. Ausgangspunkt ist ein *Problem*.
2. Um sich klar zu machen, was man will, erstellt man ein *Spezifikation*.
3. Dann entwickelt man eine *Lösungsidee*.
4. Diese verfeinert man zu einem *Lösungsalgorithmus*.
5. Diesen implementiert man als ein *Programm* in einer Hochsprache.
6. Auf dieses wendet man einen *Übersetzer* an, um eine Instruktionsfolge zu erhalten.