Informatik I

1. Grundlagen

Jan-Georg Smaus

Albert-Ludwigs-Universität Freiburg

19. Oktober 2010

Informatik I

19. Oktober 2010 — 1. Grundlagen

1.1 Was ist Informatik?

1.2 Programmiersprachen und Programmieren

1.1 Was ist Informatik?

Keine Antwort!

Wir können hier keine umfassende endgültige Antwort geben.
Im Folgenden gibt es lediglich ein paar Puzzlestücke, Appetithäppchen.

Das Wort

- Manche betrachten „Informatik“ als abgeleitet von „Information“, also die Wissenschaft von der Information.
- Andere sehen in ihr ein Kofferwort von „Information“ und „Mathematik“.
- Auf Englisch heißt sie computer science oder computing science. Ist Informatik deshalb die Wissenschaft von den Computern? Edsger W. Dijkstra (1930-2002) sagte hierzu:

  > *Computer science is no more about computers than astronomy is about telescopes.*

Einordnung

- Informatik hat etwas von Mathematik und ist insofern eine Strukturwissenschaft (oder gar eine Geisteswissenschaft?).
- Informatik hat etwas von Elekrotechnik und ist insofern eine Ingenieurwissenschaft.
- Das Konstruieren von Software (Softwaretechnik, engl. software engineering) ist eine Teildisziplin der Informatik und spricht ebenfalls für eine Einordnung unter die Ingenieurwissenschaften.

Teilgebiete

Im ersten Semester hören Sie Vorlesungen zu folgenden Teilgebieten der Informatik:

Systeme I (Betriebssysteme)	Wie wird ein Computersystem verwaltet?
Technische Informatik	Aufbau und Funktionsweise von Rechnern
Informatik I	Grundlagen der Programmiersprachen und des Programmierens

1.2 Programmiersprachen und Programmieren

- I/O, Algorithmen, Programme
- Kochen mit Šmaus
- Zurück zu den Computern
- Programmiersprachen
- Programmierarbeit
- Schluss

Was tut ein Computer?

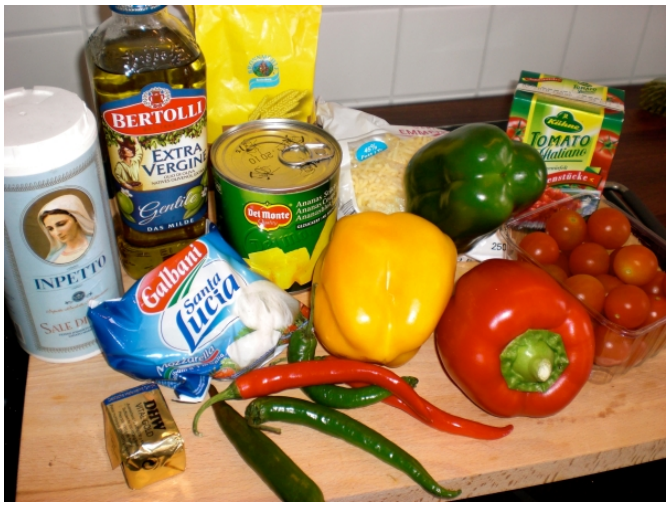
Um uns dieser Frage zu nähern, sollten wir vier Konzepte verstehen und unterscheiden:

- Input/Output,
- Algorithmus,
- Programm,
- (Berechnungs)prozess.

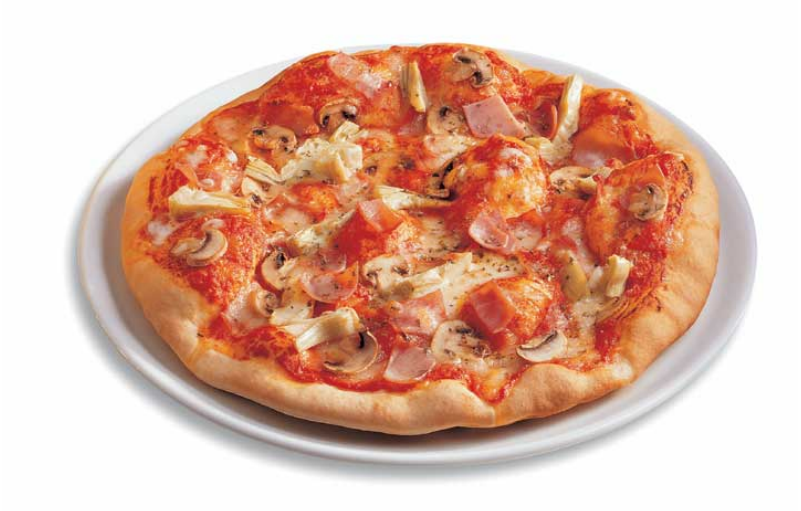
Eine hilfreiche Analogie ist das Kochen ...

Input/Output

Eingabe (Input):



Ausgabe (Output):



Hier interessiert nur:

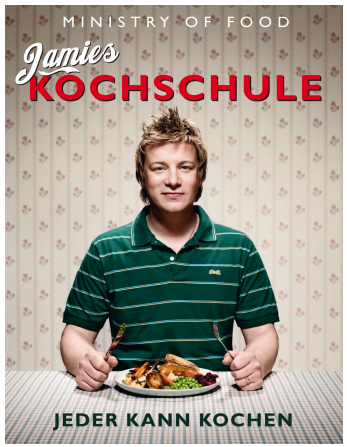
- Welche Zutaten stehen zur Verfügung?
- Wie schmeckt die fertige Pizza?

Algorithmus

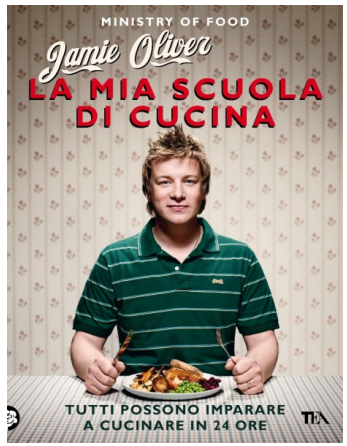
Wie wird die Pizza zubereitet?
Ich folge einem Algorithmus.
Wenn ich die Reihenfolge, in der die Paprika und die Pilze auf den Teig gelegt werden, ändere, ist das ein anderer Algorithmus, auch wenn das den Geschmack der Pizza vielleicht nicht beeinflusst.
Ein italienischer Beobachter könnte mir beim Pizzabacken zuschauen und meinen Algorithmus auf Italienisch aufschreiben, auch wenn er mein deutsches Kochbuch vielleicht nicht lesen kann.

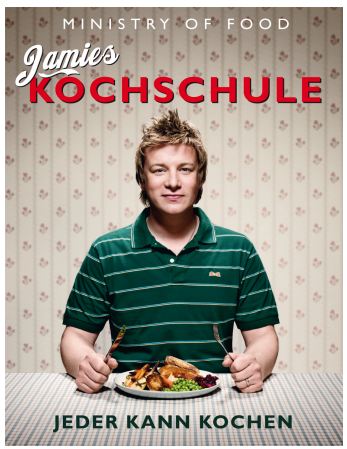
Programm

Ein Programm ist der Algorithmus notiert („aufgeschrieben“) in einer geeigneten Sprache, also ein Kochrezept.

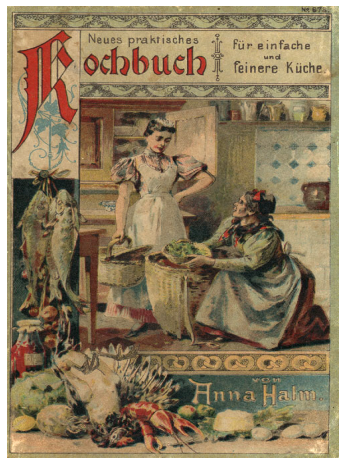


$\neq$

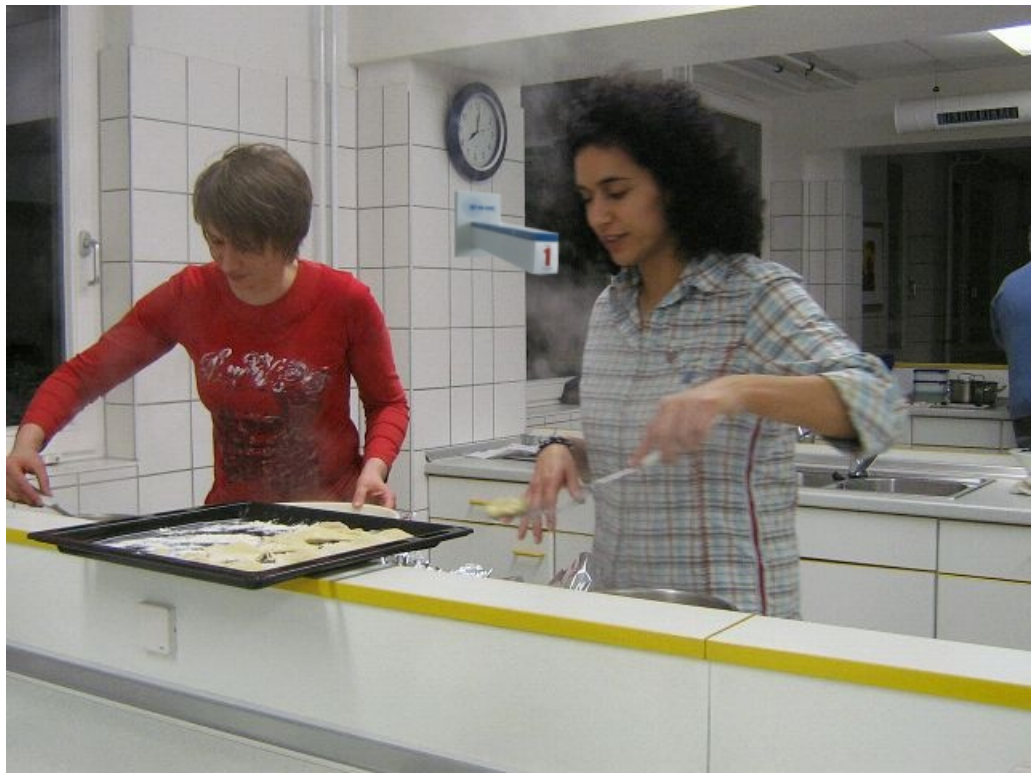




$\neq$



Prozess



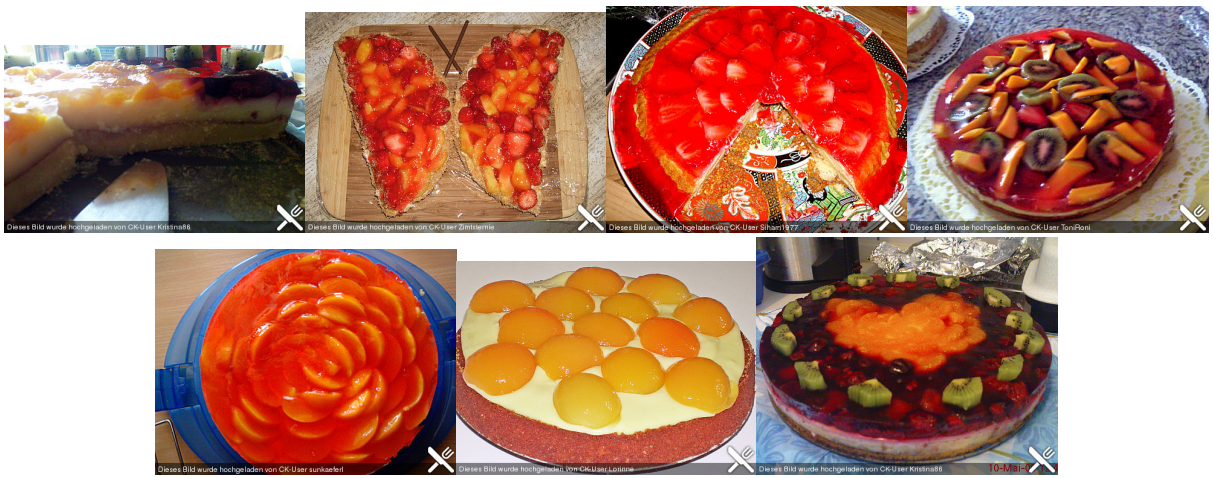
Der Vorgang des Kochens, also das Ausführen eines Programms, an einem bestimmten Ort zu einer bestimmten Zeit.

Unsere Vorstellung vom Kochen

Die Analogie hinkt vielleicht ein wenig:

- Zwar hat jedes Kochbuch seine eigene Sprache (Sprachstil), aber diese ist nicht sehr präzise festgelegt.
- Kochrezepte sind meistens nicht „idiotensicher“ genug. Sie lassen Freiheiten, und sie setzen manches Wissen voraus.
- Die meisten Rezepte sind für festgelegte Mengen von festgelegten Zutaten geschrieben.
  Parametrisierte Rezepte sind näher an Computerprogrammen ...

Parametrisierte Rezepte: Art der Zutaten



Obstkuchen

Zutaten

4 mittelgroße Eier, 1 Tasse Zucker, [...] Obst nach Saison, z.B. Beeren oder Mandarinenfilets

Zubereitung

[...] Den Pudding auf dem Kuchen verteilen und mit Obst nach Saison belegen. [...]

Parametrisierte Rezepte: Menge der Zutaten

1. Sie braten Pfannkuchen: die Zubereitung des Teiges ist unabhängig von der Menge der Zutaten exakt dieselbe.
2. Dann folgt etwas wie: „Solange die Teigschüssel nicht leer ist, tauche die Kelle in die Schüssel, schütte den Inhalt in die heiße Pfanne, . . . “
3. Wenn Sie sehr große Zutatenmengen haben, stimmt auch 1 nicht mehr: Sie werden eine größere Schüssel brauchen oder den Teig in mehreren Fuhren machen müssen.
4. Das klingt doch schon nach einem ganz spannenden Algorithmus!
5. Allerdings dürfte ein typisches Pfannkuchenrezept nicht explizit genug sein, um als Programm für diesen Algorithmus gelten zu können.

Algorithmus

Vorschrift zur Durchführung einer Berechnung (Folge von Einzelschritten) mit folgenden Eigenschaften:

Effektivität	Jeder Einzelschritt ist ausführbar.
Determiniertheit	Der nächste Einzelschritt ist stets festgelegt.
Finitheit	Die Vorschrift ist endlich.
Terminierung	Die Berechnung endet nach endlich vielen Einzelschritten.
Generalität	Die Vorschrift kann eine Klasse von Problemen lösen.
Präzision	Die Bedeutung jedes Einzelschritts ist eindeutig festgelegt.

Vorschrift, Einzelschritt

- Es geht hier um den Inhalt der Vorschrift, wie auch immer sie notiert ist.
- Wir verstehen noch nicht genau, was ein „Einzelschritt“ sein könnte. Das macht nichts!
- Erst recht wissen wir nicht, wie ein Einzelschritt notiert werden könnte. Auch das macht nichts!
- Entscheidend ist hier nur: Irgendwie wird ein Einzelschritt notiert, und wenn der Computer (oder unser Koch) dies liest, weiß er genau, was er zu tun hat.

Weitere Beispiele für Algorithmen

(aus Mathematik und Alltag)

- Schriftliche Addition, Multiplikation, Division
- Lösen von linearen Gleichungen
- Wegbeschreibungen
- Bedienungsanleitungen, Spielregeln

Und nochmals: Irgendwie muss man Algorithmen notieren, wenn man sie einem Computer oder Menschen kommunizieren will, aber die Notation ist nicht der Algorithmus.

Programm

Ein Programm ist ein Algorithmus notiert (implementiert) in einer Programmiersprache.
Es gibt verschiedene Programmiersprachen, aber sie alle sind formale Sprachen, d.h., sie sind exakt, durch strikte Regeln, definiert. Das unterscheidet sie von natürlichen Sprachen wie Deutsch oder Italienisch.

Berechnungsprozess

- Der Ablauf eines Programms auf einem bestimmten Rechner zu einer bestimmten Zeit.
- In dieser Vorlesung spielt der Begriff des Prozesses keine große Rolle, obwohl wir natürlich unsere Programme auch gelegentlich mal laufen lassen wollen.
- In Betriebssystemen dreht sich alles um Prozesse. Z. B.: Wieviel Rechenzeit auf welchem Prozessor bekommt welcher Prozess wann spendiert?

Input/Output, Algorithmus, Programm, (Berechnungs)prozess

- Ein Input/Output-Verhalten kann evtl. durch verschiedene Algorithmen erreicht werden.
- Ein Algorithmus kann in verschiedenen Programmiersprachen, also durch verschiedene Programme, implementiert werden.
- Ein Programm kann mehrmals (sehr, sehr oft) auf verschiedenen Computern auf der ganzen Welt laufen, gehört also zu vielen Prozessen.

Programmiersprachen

- Systemprogrammiersprachen
  - Nah an der Maschine
  - Abbildung auf Maschine offensichtlich
- Höhere Programmiersprachen
  - Idealisiertes Berechnungsmodell
  - Abbildung auf Maschine einfach
- Deklarative Programmiersprachen
  - Einfacheres Berechnungsmodell
  - Abbildung auf Maschine schwierig

Elemente von Programmiersprachen

So wie Sätze in natürlicher Sprache aus Wörtern und Satzzeichen gemäß einer bestimmten Grammatik zusammengefügt werden, so werden Programme in einer Programmiersprache aus Grundbausteinen unter Verwendung von Kombinationsmitteln zusammengefügt.
Es kommt noch ein Konzept hinzu: Abstraktionsmittel, um Programmstücke zu benennen.

Arbeitsablauf beim Programmieren

1. Konstruiere Programmstücke mit Hilfe von Grundbausteinen und Kombinationsmitteln, und ggf. bereits vorhandenen Programmstücken.
2. Benenne und beschreibe Programmstücke mit Hilfe von Abstraktionsmitteln. Vergiss die so abstrahierten Programmstücke und verwende nur die Namen weiter.
3. Mach weiter bei 1 bis Problem gelöst.

Man nennt dies Bottom-up Design.
Bei größeren Problemen empfiehlt sich Top-down Design:

1. Betrachte ein Teilproblem als gelöst.
2. Benenne und beschreibe es.
3. Verschiebe seine Lösung auf später.

Was jedes Programm haben sollte

Data and test-driven development

- Datenanalyse
- Signatur
  - Beschreibung der Eingabe- und Ausgabedaten
  - Beschreibung der Wirkung des Programms
- Testumgebung (Beispiele)
- Definitionen (Programmstücke)

Beispiel: Parkplatzproblem

Auf einem Parkplatz stehen PKWs und Motorräder.
Seien die Anzahl der Fahrzeuge n und die Anzahl der Räder r gegeben.
Bestimme die Anzahl P der PKWs.

Lösungsansatz: Lineares Gleichungssystem

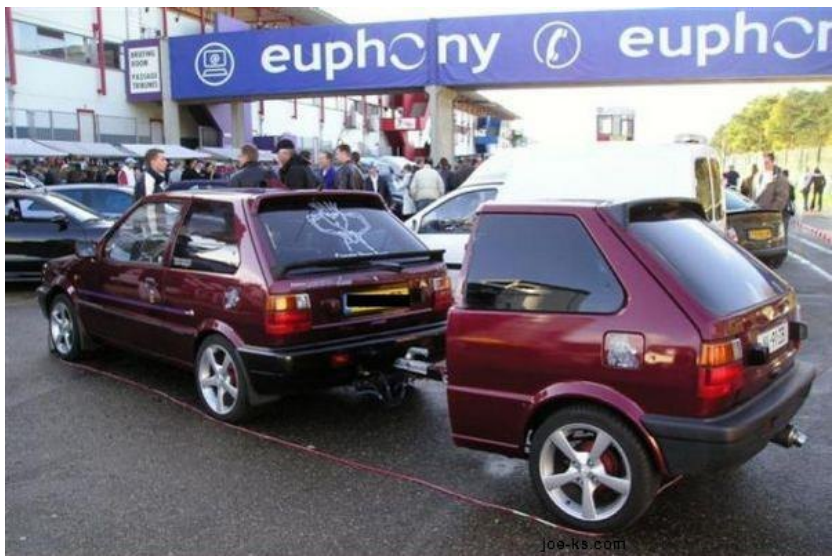
Bezeichne mit M die Anzahl der Motorräder.

$$
\begin{array}{rcl}
M + P & = & n \\
2M + 4P & = & r \\
\hline
M & = & n - P \\
2M + 4P & = & r \\
\hline
M & = & n - P \\
2(n - P) + 4P & = & r \\
\hline
2n - 2P + 4P & = & r \\
\hline
2n + 2P & = & r \\
\hline
2P & = & r - 2n \\
\hline
P & = & r/2 - n
\end{array}
$$

Probleme mit der Lösung

$P(n, r) = r/2 - n$

- $P(3, 9) = 1.5$



- $P(5, 2) = -4$
- $P(2, 10) = 3$

Vollständige Spezifikation (Parkplatzproblem)

Auf einem Parkplatz stehen PKWs und Motorräder. Seien die Anzahl der Fahrzeuge n und die Anzahl der Räder r gegeben.
Bestimme die Anzahl P der PKWs.

Eingabe: $n, r \in \mathbb{N}$

Vorbedingung: r ist gerade, $2n \leq r \leq 4n$

Ausgabe: Ein (das!) $P \in \mathbb{N}$, falls die Nachbedingung erfüllbar, sonst „Keine Lösung“

Nachbedingung: Es existieren $M, P \in \mathbb{N}$ so dass gilt:

$$\begin{aligned} M + P &= n \\ 2M + 4P &= r \end{aligned}$$

Fazit

- Ein Programm beschreibt einen Berechnungsprozess zur Lösung eines Problems.
- Eine Spezifikation/Signatur ist ein wichtiger Schritt zur Lösung eines Problems.
- Eingaben außerhalb der Signatur erzeugen unsinnige Ausgaben.
- Programme brauchen Signaturen!

Literatur I