



- 2 Algorithmische Grundkonzepte
 - Thematik dieser Vorlesung
 - Algorithmische Grundkonzepte



- Thematik – Worum geht es in dieser Vorlesung?
- Historischer Überblick
- Grundbegriffe
- Algorithmus-Begriff
- Eigenschaften von Algorithmen



- 2 Algorithmische Grundkonzepte
 - Thematik dieser Vorlesung
 - Algorithmische Grundkonzepte



Informatik (nach Duden)

Informatik ist die Wissenschaft von der systematischen Verarbeitung von Informationen, besonders der automatischen Verarbeitung mit Hilfe von Digitalrechnern.

- Kunstwort aus den 60er Jahren
 - Informatik = Information + Technik **oder**
 - Informatik = Information + Mathematik
- Nicht auf Computer beschränkt (wie *computer science*)
- Zentrale Themen
 - Systematische Verarbeitung von **Information**
 - **Maschinen**, die diese Verarbeitung automatisch leisten




Informatik : Computer = Astronomie : Teleskope

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

(E.W. Dijkstra)



- Theoretische Informatik – z.B.
 - Mathematische Modelle
 - Theoretische Konzepte (Logik, Spezifikation, Komplexität)
- Praktische Informatik – z.B.
 - Techniken der Programmierung
 - Realisierung von Softwaresystemen
- Technische Informatik – z.B.
 - Struktur und Aufbau von Computern
- Angewandte Informatik
 - Anwendungen von Informationssystemen
 - z.B. Computergraphik, **Visual Computing** 
- *Interdisziplinär*: „Bindestrich-Informatiken“



- Informatik =
 - Systematische Verarbeitung von **Information**
 - **Maschinen**, die diese Verarbeitung automatisch leisten
- Benötigt **Abstraktion**
 - Theoretische Modellierung von realen Prozessen
 - Erkennen von (oft gleichen) Strukturen
 - Beschreibung von Vorgehensweisen
 - Beschreibung von deren Eigenschaften, z.B.
 - Korrektheit
 - Aufwand
- „Vorgehensweise“ → Algorithmus-Begriff



- Informatik =
 - Systematische Verarbeitung von **Information**
 - **Maschinen**, die diese Verarbeitung automatisch leisten
- Benötigt **Programmierung**
 - Realisierung eines abstrakten Modells (*Implementierung*)
 - Beschreibung in einer „Computersprache“
- „Vorgehensweise“ → Algorithmus-Begriff
- Programmierung als notwendiger Teil der Informatik
 - Informatiker : Programmierer = Architekt : Maurer ?
 - Informatiker : Programmierer = Architekt : Künstler ?



Algorithmus

Ein Algorithmus ist eine eindeutige Beschreibung eines Vorgangs.

- Vorgang = Bearbeitung von **Daten**
- Daten = **Information**
- Vorgang nicht trivial: es sind mehrere Schritte nötig
- *Wir präzisieren den Begriff im weiteren Verlauf der Vorlesung.*

- Wir betrachten Berechnungsvorgänge, die durch (abstrakte) Maschinen ausgeführt werden:

Prozessor

Ein Prozessor führt einen Arbeitsvorgang (Prozess) auf Basis einer eindeutig interpretierbaren Beschreibung – dem Algorithmus – aus.



- Kochrezepte
- Bauanleitungen, Bedienungsanleitungen, ...
 - Montageanleitung z.B. IKEA Möbel
 - Verwendung der Mikrowelle
 - Mensch-ärgere-dich-nicht Spiel
- Berechnungsvorschriften
 - Schriftliches Addieren
 - Berechnung des größten gemeinsamen Teilers zweier Zahlen



- 300 v. Chr. Euklids Algorithmus zur Bestimmung des ggT
- 800 n. Chr. Al-Chwarizmi
 - Abu Dscha'far Muhammad ibn Musa al-Chwarizmi
 - *Über das Rechnen mit indischen Ziffern*,
lateinisch *Algorismi de... = Al-Chwarizmi über...*
 - Abgeleiteter Begriff: **Algorithmus**
 - Auch *Algebra* hat ihren Ursprung bei Al-Chwarizmi!
- 1574: Adam Ries' Rechenbuch
- 1614: Logarithmentafeln (30 Jahre für Berechnung!)
- 1703: Duales Zahlensystem (Gottfried Wilhelm Leibniz)
- 1822: *Analytical Engine* (Charles Babbage; Ada Lovelace)
- 1931: Gödels Unvollständigkeitssatz
- 1936: Churchsche These



- Notation für Beschreibung
- Ausdrucksfähigkeit
- Berechenbarkeit
- Korrektheit
- Aufwand (Zeitbedarf, Geschwindigkeit)



- „Bilde die Summe aus zwei ganzen Zahlen x und y !“
- $f : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$ mit $f(x, y) = x + y$
- $z \leftarrow x + y$
- $z := x + y$
- $z = x + y$
- $(+ \ x \ y)$
- `leal (%rsi,%rdi), %eax`



- Verschiedene Notationen = gleiche Ausdrucksfähigkeit?
- Wahl der Sprache – universelle (Programmier-)sprache?
- Beispiel: Wegbeschreibung
 - in Bienensprache
 - zur Programmierung eines Roboters
- Beispiel: Dressierte Tiere, z.B. Hunde
 - Kommandos in menschlicher Sprache
 - Versteht das Tier tatsächlich Sprache?

The limits of my language mean the limits of my world.
(L. Wittgenstein)



- Kann man „alles“ mit Algorithmen – also mit Computerprogrammen – berechnen?

Nein!

- Wo sind die Grenzen?
- Welche Probleme sind *nichtentscheidbar*? – z.B.
 - Halteproblem
 - semantische Eigenschaften von Algorithmen



- Algorithmen/Programme/Softwaresysteme sollen sich wie beabsichtigt – d.h., **korrekt** – verhalten!
- Folgen von Programmfehlern (*bugs*) s. z.B. [Wikipedia \(en\)](#):
 - 1962: Absturz der Venus-Sonde *Mariner 1* durch Übersehen eines Überstrichs in Spezifikation
 - 1968: HAL 9000 (*2001: A Space Odyssey*) entwickelt ein problematisches Eigenleben (**Fiktion**)
 - 1996: Verlust einer *Ariane 5*-Rakete nach dem Start (Fehler bei einer Typumwandlung)
 - 1999: Verlust der Mars-Sonde *Climate Oribiter* durch falsches Maßsystem
 - Jahr-2000-Problem (*Millenium bug*)
- Fehler können teuer sein!



- Welchen Aufwand benötigt ein Algorithmus für eine Eingabe?
 - Rechenzeit
 - Speicherbedarf
- Wie kann Aufwand (abstrakt) abgeschätzt werden?
 - in Abhängigkeit von Problemgröße
 - unabhängig von konkreter Rechenleistung
 - für Szenarien: am besten / im Mittel / am schlechtesten
- Welche Probleme können praktisch gelöst werden?
 - Mooresches Gesetz: „*Rechenleistung steigt exponentiell*“
(*Komplexität von Schaltkreisen verdoppelt sich etwa alle 18 Monate.*)
 - Reicht das?



- 2 Algorithmische Grundkonzepte
 - Thematik dieser Vorlesung
 - Algorithmische Grundkonzepte



Definition (Algorithmus)

Ein *Algorithmus* ist eine präzise¹, endliche Beschreibung eines allgemeinen Verfahrens unter Verwendung ausführbarer elementarer (Verarbeitungs-)Schritte.

¹ d.h. in einer festgelegten Sprache abgefasste



- Addition zweier positiver Dezimalzahlen mit Übertrag

$$\begin{array}{r} 3 \quad 3 \\ + \quad 4 \quad 8 \\ \quad 1 \\ \hline 8 \quad 1 \end{array}$$

- Test, ob eine gegebene Zahl eine Primzahl ist
- Sortieren einer Kartei
- Berechnung der Eulerschen Zahl $e = 2,7182818284590452\dots$
- Berechnung der Kreiszahl $\pi = 3,1415926535897932\dots$



- Terminierung
- Determinismus
- Semantik von Algorithmen



Definition (Terminierung)

Ein Algorithmus heißt *terminierend*, wenn er – für jede erlaubte Eingabe – nach endlich vielen Schritten abbricht.

- Terminieren die gezeigten Beispiele für Algorithmen?
- Berechnung der Eulerschen Zahl e

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \dots = \sum_{n=0}^{\infty} \frac{1}{n!}$$

- Ebenso wenig terminiert die Berechnung von π



- „Wahlfreiheit“ bei Ausführung
- Wir unterscheiden
 - **Deterministischer Ablauf:**
Eindeutige Vorgabe der Folge der auszuführenden Schritte
 - **Determiniertes Ergebnis:**
Eindeutiges Ergebnis bei vorgegebener Eingabe
- Ein *nichtdeterministischer* Algorithmus mit *determiniertem* Ergebnis heißt **determiniert**.
- Ein *deterministischer* Algorithmus ist immer *determiniert*.



- Ziehung der Lottozahlen
- Mensch-ärgere-dich-nicht Spiel
- Aufbau eines IKEA Möbels nach (nichttrivialer) Montageanleitung
- Berechnung der ersten n Stellen von π durch „Monte-Carlo-Integration“ (in etwa: Werfen von Dart-Pfeilen und Zählen der Treffer)



- **Deterministische** und **terminierende** Algorithmen definieren

$$f : \text{Eingabewerte} \rightarrow \text{Ausgabewerte}$$

- Die Ein-/Ausgabefunktion f ist eine formale Beschreibung der **Semantik** (Bedeutung) des Algorithmus: Was wird berechnet?

- Beispiele

- Addition zweier ganzer Zahlen

$$f : \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z} \text{ mit } f(x, y) = x + y$$

- Test, ob eine Zahl Primzahl ist

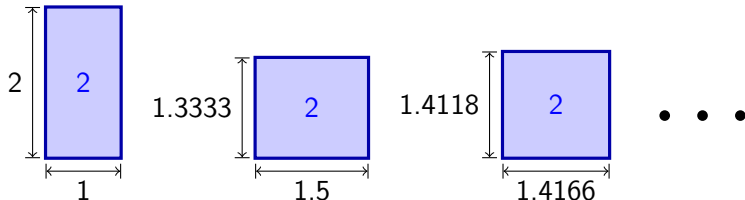
$$f : \mathbb{N} \rightarrow \{\text{wahr}, \text{falsch}\} \text{ mit } f(n) = \begin{cases} \text{wahr} & \text{falls } n \text{ prim} \\ \text{falsch} & \text{sonst} \end{cases}$$



- Bestimme Quadratwurzel \sqrt{A}
 - Betrachte Rechteck mit **Flächeninhalt** $A = a \cdot b$,
z.B. $a = 1, b = A$
 - Konstruiere neues Rechteck mit $a' \cdot b' = A$

$$a' = \frac{a+b}{2} \quad \text{und} \quad b' = \frac{A}{a'}$$

- Folge von immer „quadratischeren“ Rechtecken mit Fläche A
und Seitenlängen $a \approx b \rightarrow \sqrt{A}$
- Beispiel: $A = 2$ ($\sqrt{2} \approx 1.414213562373095\dots$)





- Elementare Operationen
 - Sequentielle Ausführung (von Bausteinen)
 - Parallele Ausführung (z.B. durch mehrere Prozessoren)
 - Bedingte Ausführung (Fallunterscheidung, Auswahl)
 - Bedingte Wiederholung (Schleife)
 - Unter-„Programm“
 - Rekursion
 - ...
-
- *Welche Bausteine sind essentiell, welche optional?*



- Code = Notation in einer Programmiersprache
- **Pseudocode** = Notation in Anlehnung an Programmiersprachen
 - Konzentration auf das Wesentliche (oft Programmfragmente)
 - Vereinfachung bedeutet einfachere Lesbarkeit
 - Oft teils mathematische Notation oder natürliche Sprache
 - *Es gibt keine Spezifikation*



- Sequenz = Hintereinanderausführung von Anweisungen
- Beispiel: (einfaches) Kaffeekochen

```
(1) Koche Wasser  
(2) Gib Kaffeepulver in Tasse  
(3) Fülle Wasser in Tasse
```

- Eine Anweisung pro Zeile **oder**
- Trennung durch Semikolon `Koche Wasser; ...`
- *Nummerierung nicht nötig*



- Entwurfsprinzip der Schrittweisen Verfeinerung
 - *Strukturierter* Entwurf von Algorithmen
 - Idee von Unterprogrammen

(2) Gib Kaffeepulver in Tasse

verfeinert zu

- (2.1) Öffne Kaffeepackung
- (2.2) Entnimm Löffel Kaffeepulver
- (2.3) Kippe Löffel in Tasse
- (2.4) Schließe Kaffeepackung



- Bedingte Ausführung heißt auch **Auswahl** oder **Selektion**
- „**Wenn Bedingung erfüllt dann mache ...**“
- Entspricht einer *Fallunterscheidung*
- Notation als

```
if Bedingung then Schritt fi
```

```
if Bedingung then  
    Schritt a  
else  
    Schritt b  
fi
```

- „Klammerung“ durch **if-fi** erleichtert Lesbarkeit!



- auch **Iteration**, in Programmiersprachen: **Schleife** (*loop*)
- „Solange *Bedingung* erfüllt ist, mache ...“
- Notation als

```
while Bedingung do  
    ...  
od
```

```
do  
    ...  
while Bedingung
```

- **do-od** Klammerung
- Variante **repeat...until** Bedingung



- Berechne \sqrt{A}
- Idee: $A = a \cdot b$ ist Flächeninhalt eines Rechtecks

```
a ← 1
b ←  $\frac{A}{a}$ 

while |a - b| > ε do
    a ←  $\frac{1}{2}(a + b)$ 
    b ←  $\frac{A}{a}$ 
od
```

- Das ist ein möglicher Pseudocode
- Fehlerschranke ε
- z.B. alternative Fehlerabschätzung: while $|A - a^2| > \varepsilon$ do ...



- Schreibweise ähnlich wie in „richtigen“ Programmiersprachen
- Pseudocode spiegelt Bedeutung (Semantik) wider
- Es fehlt jedoch eine strenge Syntax
- Kompromiss zwischen formaler Schreibweise und Lesbarkeit



- von lateinisch *recurrere* = *zurücklaufen*
- *Rekursion, die*: siehe *Rekursion*
- Selbstbezug – hier in Defintion von Algorithmen

- Rekursion wird ein zentrales Thema sein,
v.a. auch im Sommersemester!
- Viele Algorithmen lassen sich damit sehr elegant ausdrücken.



- **Syntax** = formale Regeln, wie Sätze gebildet werden
- **Semantik** = Bedeutung von Sätzen
 - semantisch korrekt = Sinn ergebend
 - semantisch „falsch“ = „Unsinn“
- Beispiel:
 - *Der Elefant aß die Erdnuss.* – syntaktisch korrekt, sinnhaft
 - *Der Elefant aß Erdnuss die.* – syntaktisch falsch!
 - *Die Erdnuss aß den Elefanten.* – syntaktisch korrekt, sinnlos!

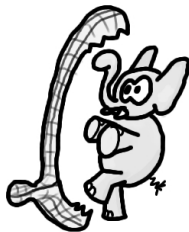
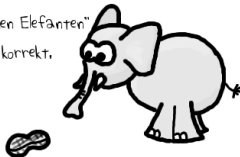


Der Satz

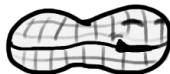
"Die Erdnuss frisst den Elefanten"

ist zwar syntaktisch korrekt,

aber semantisch



rölper



Semantik?!

Hell, Yeah!



- **Grammatik** = Regelwerk zur Beschreibung der *Syntax*
- Beispiel: *Produktionsregel* für einfache Sätze der Form
Satz \mapsto Subjekt Prädikat Objekt
- Generierte Sprache =
alle durch Anwendung der Regeln erzeugbare Sätze



- Formale Beschreibung von Grammatiken zur
- Festlegung der Syntax von Kunstsprachen
- Produktionsregeln/Ersetzungsregeln der Form

`LinkeSeite ::= RechteSeite`

- `LinkeSeite` = Name des zu definierenden Konzepts
 - z.B. `<Satz>`
- `RechteSeite` = Definition in Form einer Liste
 - Element = Konstanten (Terminale) oder andere Konzepte (in `<>` Klammern)
 - z.B. `<Subjekt> <Prädikat> <Objekt>`
 - Trennzeichen `|` bedeutet „oder“ und trennt Alternativen
z.B. `Elefant | Maus`



```
<Satz>      ::= <Subjekt> <Prädikat> <Objekt>  
<Subjekt>   ::= Elefant | Erdnuss  
<Prädikat>  ::= aß | sah  
<Objekt>    ::= Erdnuss | Maus
```

- Welche Sprache generiert diese Grammatik?



```
<atom>      ::= ...  
<bedingung> ::= ...  
  
<sequenz>   ::= <block>; <block>  
<auswahl>   ::= if <bedingung> then <block> fi |  
                if <bedingung> then <block>  
                    else <block> fi  
<schleife>   ::= while <bedingung> do <block> od  
<block>      ::= <atom> | <sequenz> |  
                <auswahl> | <schleife>
```

- Hier fehlt Definition von Termen <atom> und <bedingung>!



- Daten = zu verarbeitende Informationseinheiten
- **Datentyp** =
 - Zusammenfassung gleichartiger Daten und
 - Operationen, die auf diesen Daten erlaubt sind
- Beispiele
 - natürliche Zahlen \mathbb{N} , ganze Zahlen \mathbb{Z}
 - Wahrheitswerte $\{\text{true}, \text{false}\}$ mit Operationen
Negation ($\neg p$), logisches Und ($p \wedge q$), logisches Oder ($p \vee q$)
- Datentypen als Algebren
 - Algebra = Wertemenge (Sorte) + Operationen
 - Oft *mehrsortige Algebren* z.B. $\leq: \mathbb{Z} \times \mathbb{Z} \rightarrow \{\text{true}, \text{false}\}$
- *Syntax* definiert Regeln zum Bilden von Termen
z.B. $(x + y) \leq z \vee \neg p$
- *Semantik* kann sich je nach Datentyp unterscheiden



- **Informatik** =
Systematische und automatische Verarbeitung von Information
- **Algorithmus**
 - **Terminierung**
 - **Determinismus** (Ablauf/Ergebnis)
 - Algorithmus als Funktion: **Semantik**
- Notation von Algorithmen in **Pseudocode**
- Sprachen und Grammatiken (**Syntax**)
- Datentypen
- *Als nächstes*: Grundkonzepte in Java