Praktische Informatik für Wirtschaftsmathematiker, Ingenieure und Naturwissenschaftler I (PIWIN I, 3 V + 1 Ü) WS 2002/03

Vorlesung:
Peter Kemper
Informatik, LS IV
GB V, R 403, Tel 3031
email: peter.kemper@udo.edu

Termine:
Di 10-12, GB V, R 420
Do10-11, GB IV, R 318

Übung:
Hans Decker
Informatik, Dekanat
GB V, R 301, Tel 2208
email: hans.decker@udo.edu

Termine:
14-tägig

Organisatorisches

- Materialien
  - Folien: im Netz (als PDF Dateien), begleitend zur Vorlesung
    http://ls4-www.cs.uni-dortmund.de/home/kemper
  - Skript: Vorlesung PIWIN I von Prof. Moraga WS 00/01, nur als Anhaltspunkt
    http://ls1-www.cs.uni-dortmund.de/Lehre/vorlesung_history.html)
  - Buch: Gumm/Sommer, Einführung in die Informatik, Oldenbourg,
    Es werden Kapitel 1-4 auszugsweise behandelt.
- Arbeitstechnik
  - Vorlesungsinhalte nachbereiten
  - Fragen stellen, Unklarheiten beseitigen
- Prüfung
  - nach der Vorlesung, Termine nach Vereinbarung
- Übung
  - Aufgabenzettel erhalten Sie jeweils in der Vorlesung
  - Lösungen erarbeiten Sie bitte zu Hause
  - Lösungen und Fragen werden in der Übung (14 tägig) diskutiert

  Übungsorganisation: Hans Decker (hans.decker@udo.edu)

Literatur

- im Handapparat der Bereichsbibliothek Informatik steht für Sie bereit:
  - Gumm,Sommer: Einführung in die Informatik, 3. Aufl. Oldenbourg (neuere 4. Auflage in der UniBib)
  - H.J.Appelrath, J.Ludewig: Skriptum Informatik, Teubner 1995, Sn19019, aktuell ist eigentlich die 5. Auflage von 2000, ...
  - A.V. Aho, J.D. Ullman: Informatik - Datenstrukturen und Konzepte der Abstraktion, Int. Thomson Publ. 1996, in der UniBib auch unter

    L Sr 373 (31 Exemplare!)

  Achtung: Sie finden in der BI unter 3000/... eine ganze Reihe von Büchern „Einführung in die Informatik“ o.ä., aus denen Sie sich ggfs, das für Sie besonders passende auswählen können. Themen sind häufig sehr ähnlich, Fokus der Programmiersprache variiert (imperativ, funktional).

  Empfehlung: wählen Sie ein Buch, das auf imperative/objektorientierte Programmierung fokussiert (nicht funktional, erkennbar z.B. an Sprachen wie Lisp, ML, Scheme, Haskell etc).

Lernziele

- allg. Grundbegriffe
  - Informatik, Spezifikation, Algorithmus, Programm, Prozeß
- Grundlagen imperativer Programmierung
  - Sequenz, Alternative, Wiederholung, Funktion, Rekursion
- Grundlagen objektorientierter Programmierung
  - Klasse, Objekt, Methode, Attribut, Vererbung, Schnittstelle
- Rechnerarithmetik
  - Zahlendarstellung und arithmetische Operationen
- Datenstrukturen
- Algorithmen
- Komplexität von Algorithmen
- Berechenbarkeit
- Programmentwurf

Heute: Was ist Informatik ?

- Themen heute:
  - Definition des Begriffs „Informatik“
  - Teilgebiete der Informatik
  - einige Begriffe zur Datendarstellung
- Anmerkung

  Informatik ist im Vergleich zu anderen Disziplinen eine relativ junge Wissenschaft, daher in ihren Inhalten, ihren Teilgebieten und ihrem Verhältnis zu anderen Disziplinen stetem Wandel unterzogen.

Informatik: eine Auswahl an Definitionen

- aus dem Informatik-Duden

  „Informatik (engl. Computer Science oder inzwischen auch Informatics) ist die Wissenschaft von der systematischen Verarbeitung von Informationen, besonders der automatischen Verarbeitung mit Hilfe von Digitalrechnern“

- der Akademie Francaise, angelehnt an „informatique“

  „Behandlung von Information mit rationalen Mitteln“

  wobei rationale Mittel nach Descartes auszeichnet:
  - „nur dasjenige gilt als wahr, was so klar ist, dass kein Zweifel bleibt“
  - „größere Probleme sind in kleinere aufzuspalten“
  - „es ist immer vom Einfachen zum Zusammengesetzten hin zu argumentieren“
  - „das Werk muss am Ende einer abschließenden Prüfung unterworfen werden“
- Informatik als Wort beinhaltet: Information und Automatik

Informatik: eine Auswahl an Definitionen

- im Studienführer Informatik

  „Informatik ist die (Ingenieur-)Wissenschaft von der theoretischen Analyse und Konzeption, der organisatorischen und technischen Gestaltung sowie der konkreten Realisierung von (komplexen) Systemen aus miteinander und mit ihrer Umwelt kommunizierenden (in gewissem Maß intelligenten und autonomen) Agenten oder Akteuren, die als Unterstützungssysteme für Menschen in unsere Zivilisation eingebettet werden müssen - mit Agenten/Akteuren sind Software-Module, Maschinen oder roboterartige Geräte gemeint.

Teilgebiete der Informatik

- Technische Informatik

  befasst sich mit dem Bau von Rechner und Rechnernetzen und allen damit verbundenen Fragegestellungen (Hardwareentwicklung)

- Praktische Informatik

  befasst sich mit der Programmierung von Rechnern und der Entwicklung von dafür benötigten Sprachen, Werkzeugen, Methoden (Softwareentwicklung)

- Theoretische Informatik

  befasst sich mit den (mathematischen) Grundlagen der Informatik, die von den anderen Teilgebieten genutzt werden (Grenzen des Berechenbaren, Komplexitätsbetrachtungen)

- Angewandte Informatik

  ergänzt die vorhergehenden Teilgebiete und befasst sich mit den vielfältigen Anwendungsmöglichkeiten für Rechensysteme

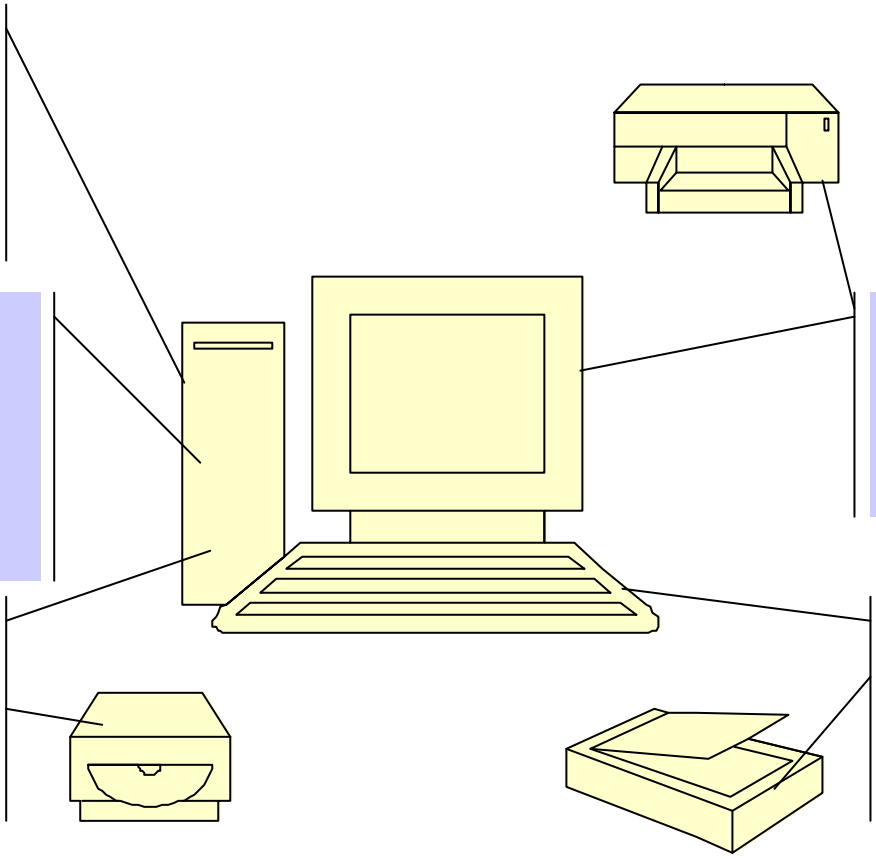
Was macht eigentlich ein Rechensystem aus ?

- Hardware: physikalische Komponenten, die spezielle Leistungen erbringen, Funktionen verfügbar machen

CPU, Prozessor führt Befehle aus Maschinensprache aus

Verbindungssystem (Bus) übermittelt Daten zwischen Komponenten

Speichermedien halten Daten (auch auf Dauer)



Ausgabemedien machen Daten sichtbar

Eingabemedien erlauben Dateneingabe

Was macht eigentlich ein Rechensystem aus ?

- Wofür wird es denn verwendet ?

  von Privatanwendern:

  – Textverarbeitung
  – Tabellenkalkulation
  – Elektronische Post (email)
  – Internet surfen, Informationsbeschaffung, ...

  von Firmen:

  – zusätzlich zu den vorherigen Punkten,
  – Verwaltung von Firmendaten und Arbeitsvorgängen, Produktionsplanung und -steuerung, Buchhaltung, ...
    - Datenbankapplikationen,
    - Enterprise Ressource Systeme wie SAP R/3
  – Steuerung automatisierter Fertigungsanlagen

- Warum geht das ?

  Rechensysteme sind flexibel einsetzbar, ihre Fähigkeiten lassen sich an die jeweiligen Anforderungen anpassen:

  Programmierung, Softwareentwicklung

Was macht eigentlich ein Rechensystem aus ?

- Hardware: -> Rechnerarchitektur in PIWIN II

eine sehr abstrakte Sichtweise:

Automat mit Gedächtnis / Speicher transformiert Eingaben in Ausgaben anhand gegebener Befehlsfolge

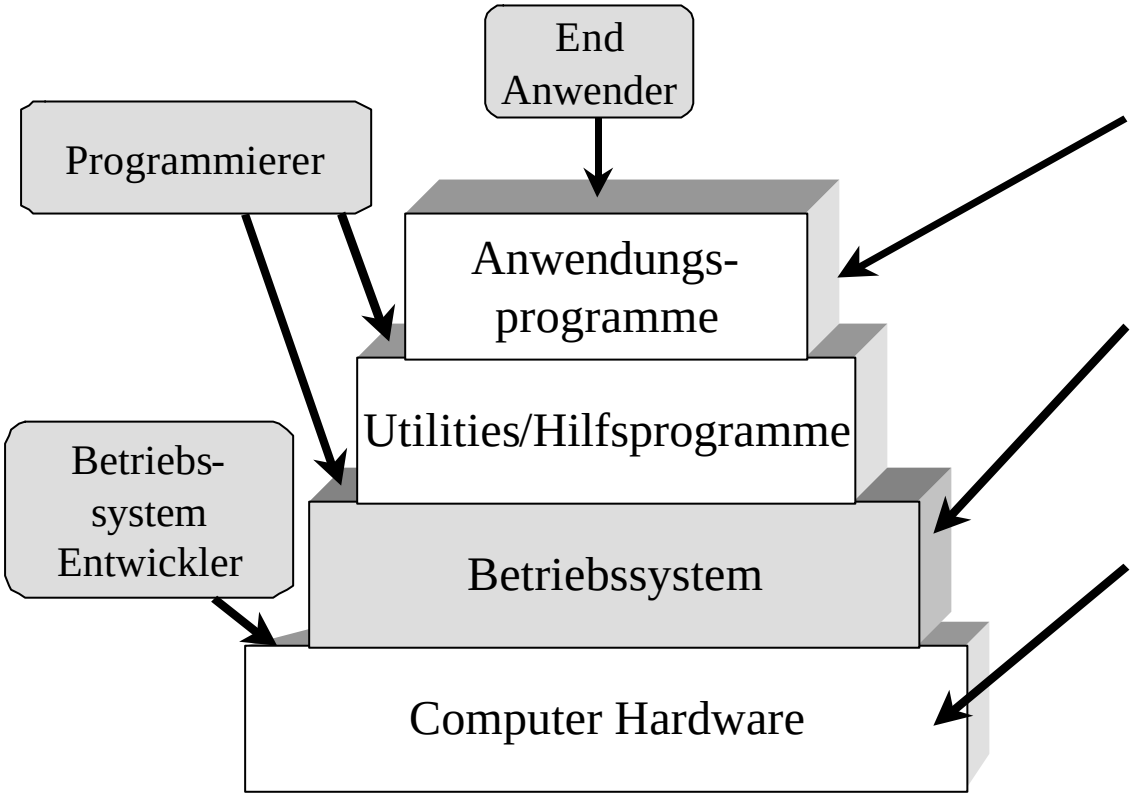
Besonderheit: Gestaltung von Berechnungsabläufen, Programmierbarkeit

- Software: Beschreibung, die bestehende Hardware / Softwarefunktionen nutzt, um eine komplexere Funktionalität zu erbringen

  Es existieren unterschiedliche Betrachtungsebenen

  - Anwender: Bedienung eines Softwaresystems
  - SW Entwickler: Programmstruktur des Softwaresystems
  - Betriebssystem: Darstellung des laufenden Programms (Prozesses) im Speicher des Rechners und die Abarbeitung der Befehlsfolge durch den Prozessor

Schichten (auch Sichten) eines Computer Systems



Beispiel:
Powerpointpräsentation
Powerpoint nutzt
- Dateisystem,
- logische E/A Geräte

Powerpointprozeß wird vom BS verwaltet, mittels CPU ausgeführt, Dateisystem, E/A Geräte werden unter Nutzung realer Hardware verfügbar gemacht.

Das Betriebssystem vermittelt zwischen Anwendungsprogrammen und realer Hardware. Z.B. Powerpoint hat eine „Save“ Funktion zum Speichern einer Präsentation in einer Datei. Das Betriebssystem stellt hierzu elementare Operationen zum Lesen / Schreiben von Dateien zur Verfügung und verwaltet Dateien. Es realisiert eine virtuelle Maschine.

Virtuelle Maschine

Eine virtuelle Maschine ist eine Menge von Funktionen /Diensten, die sich real nutzen lassen, jedoch durch Software hergestellt werden.

- häufig verwendetes Konzept der Informatik zur Beherrschung von Komplexität durch Zerlegen in beherrschbare Teilprobleme und Kopplung der Teillösungen mittels Schnittstellen (Teile&Herrsche), wobei eine Teillösung (VM) dann in unterschiedlichen Kontexten genutzt werden kann.
- z.B. Betriebssystem: erzeugt eine Schnittstelle festen Funktionsumfanges (=>VM), so dass Anwendungsprogramme auf dieser VM trotz unterschiedlicher HW Plattformen ablaufen können. Eine Leistung ist z.B. die real begrenzte Menge Hauptspeicher für das Anwendungsprogramm als quasi beliebig gross erscheinen zu lassen.
- z.B. Kommunikationsprotokolle: erzeugen eine Schnittstelle zur Übermittlung von Daten. Z.B. TCP/IP bewirken zuverlässige Verbindung zwischen Kommunikationspartnern, obwohl in Wirklichkeit einzelne Pakete über Zwischenstationen und mit unsicheren Verbindungen durch ein Netzwerk transferiert werden.

Arbeitsdefinition Rechensystem (im Skript: Prozessor)

„Ein Rechensystem zeichnet sich durch einen festen Vorrat an Operationen aus (Befehlsvorrat) und die Fähigkeit anhand einer Vorschrift (eines Programms) eine Folge von Operationen zu entwickeln und in Verbindung mit Eingabegrößen auszuführen. Ein Rechensystem verfügt über die Möglichkeit Daten und Befehle zu speichern. Die Durchführung von Operationen obliegt einer Zentraleinheit (CPU), Eingabedaten / Ausgabedaten werden mittels Peripheriegeräten eingelesen / ausgegeben."

- Diese Begriffsbestimmung eines Rechensystems ist abstrakt gehalten, ein Rechensystem kann physikalisch existieren, aber auch als eine virtuelle Maschine hergestellt werden oder lediglich ein formaler Mechanismus sein.
- Der folgende „endliche Automat" ist ein in der theoretischen Informatik häufig und in vielen Variationen verwendetes formales Konzept, um u.a. Berechnungsvorgänge formal zu betrachten.

Definition. Endlicher Automat

Ein endlicher Automat ist ein Tupel (A, S, s_0, F, d) mit

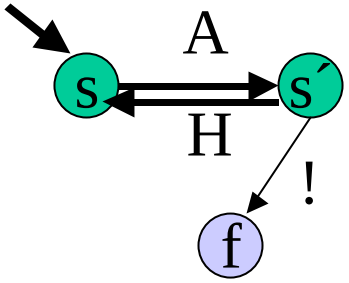
Alphabet *A*, einer endlichen Menge von Zeichen,

S einer endlichen Menge von Zuständen

$s_0 \in S$, einem Anfangszustand

$F \subset S$, einer Menge von Endzuständen

$d : A \times S \rightarrow S$, einer Übergangsfunktion



A als Eingabealphabet führt zu Betrachtung von Automaten, die bestimmte Zeichenfolgen erkennen und akzeptieren können, z.B. A!, AHA!, AHAHA!, ...

Variation: Unterscheidung Ein/Ausgabealphabet führt zu Automaten, die auf die Eingabe von Zeichenfolgen mit einer Ausgabefolge von Zeichen reagieren (Mealy-/Mooreautomat).

Besonderheit des Automaten: die Zustandsmenge wird benutzt, um durch den jeweils erreichten, aktuellen Zustand auch Informationen über die bisher gelesene Zeichenfolge zu speichern.

Mit Automaten lassen sich Berechnungsvorgänge formal fassen, der Zustand eines Rechensystems ist die Belegung aller Speicherzellen (und damit implizit auch der Stand der Berechnung, des Programmzählers), Zustandsübergänge erfolgen durch die Ausführung von Operationen aus dem Befehlsvorrat. Übergangsfunktion wird passend zum betrachteten Programm gewählt.

Grenzen der Leistungsfähigkeit eines endlichen Automaten

- Ein endlicher Automat „akzeptiert“ ein Wort einer Sprache, falls nach Einlesen der Zeichenfolge der aktuelle Zustand in F liegt.
- Eine Sprache ist eine Menge von Zeichenfolgen, z.B. A(HA)*! im vorherigen Beispiel. (* zeigt n Wiederholungen an, mit n aus {0,1,...})
- ein endlicher Automat als Akzeptor einer formalen Sprache ist wegen seines endlichen Speichers (=Zustandsmenge) in seiner Leistungsfähigkeit eingeschränkt
- z.B. folgende Sprache kann durch einen endlichen Automaten nicht erkannt werden:

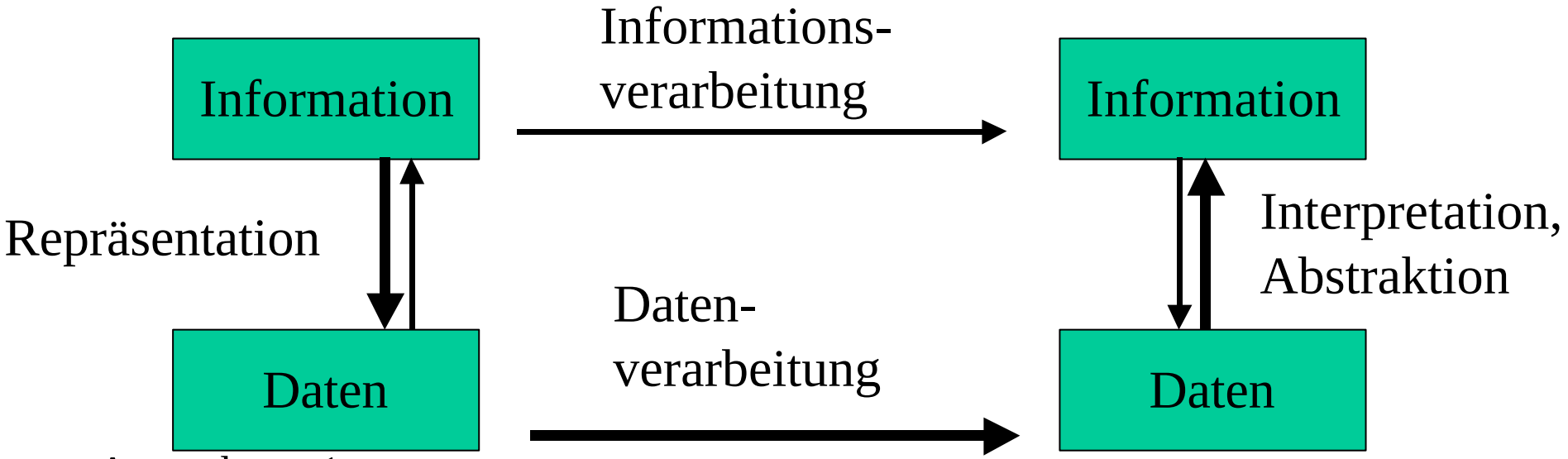
$$L = \{w \mid a^n b^n, n \in N\}$$

- daher wird auf dem Gebiet „Automatentheorie“ eine Vielzahl von verschiedenen Automaten betrachtet, der Bezug zwischen Sprachen und Automaten ist ein klassisches Gebiet der theoretischen Informatik (Stichwort: Chomsky Hierarchie), das Auswirkungen auf die praktische Informatik hatte:

  LALR1 Grammatik **----** Algorithmus **--->** endlicher Automat, der Worte dieser Sprache erkennt, wird zur Generierung von Parsern, insbesondere für Compiler genutzt. => Programmiersprachen haben LALR1 Grammatik.

Wozu wird ein Rechensystem genutzt ?

Wunsch: Informationsverarbeitung Wirklichkeit: Datenverarbeitung



Anmerkung 1:

Die grundsätzliche Crux der Informatik besteht darin, dass ein System ohne eigenes Verstehen, ohne eigene Erkenntnis geschaffen wird, dass dennoch ein sinnvolles Verhalten zeigen soll.

Anmerkung 2:

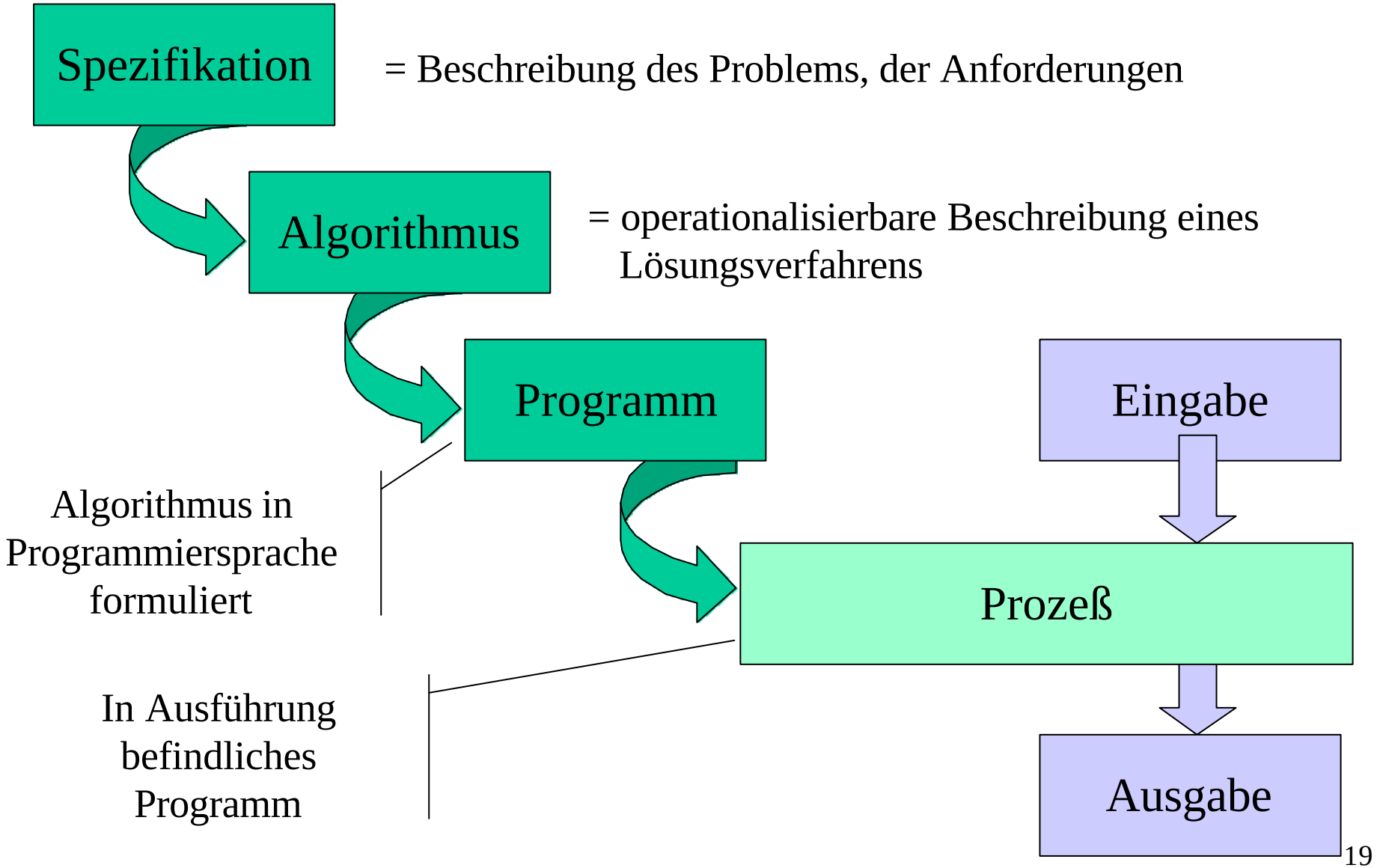
Repräsentation von Informationen durch Daten kann auf unterschiedlichen Ebenen erfolgen, wir werden uns heute noch mit der elementarsten Ebene, nämlich durch Werte 0 und 1 befassen.

Wo stehen wir jetzt ?

- Was ist Informatik ?
- Was macht ein Rechensystem aus ?
- Daten vs Informationen führt zur Frage:

  Wie werden Daten in einem Rechner dargestellt ?
  - Buchstaben, Zeichenketten, Texte, ...
  - Grafiken
  - Algebren
    - Boolesche Algebra mit Operationen AND, OR, NOT
    - Natürliche Zahlen, ganze Zahlen, reellwertige Zahlen mit Operationen Addition, Subtraktion, Multiplikation, Division, Modulo, ...
    - Achtung: Genauigkeit der Darstellung und damit auch von Berechnungen ist begrenzt !!! Wertebereiche für Zahlen sind beschränkt !!!
- Wir wollen uns letztlich jedoch mit dem Entwurf von Algorithmen und Programmen, der Programmierung von Rechensystemen und zugehörigen Programmiersprachen befassen.

Stationen im Entwurf von Algorithmen und Programmen



Wie geht es weiter ?

- Darstellung von elementaren Daten
- Grundlagen der Programmierung
  - Spezifikationen, Algorithmen, Programme
  - Daten und Datenstrukturen
  - imperative Programmierung:
    - Verfeinerung, elementare Operationen, Sequenz, Selektion, Iteration, funktionale Algorithmen und Rekursion, Variablen und Wertzuweisungen, Prozeduren, Funktionen und Modularität, Zuweisung, Sequenz
  - objektorientierte Programmierung
- Programmiersprachenkonzepte
  - Syntax und Semantik
  - imperative, objektorientierte, funktionale und logische Programmierung
- Berechenbarkeit und Entscheidbarkeit von Problemen
- Effizienz und Komplexität von Algorithmen
- Programmentwurf, Softwareentwurf

Grundbegriffe

- Bits
  - kleinstmögliche Einheit der Information
  - erlaubt Antwort auf eine Frage mit nur zwei Antwortmöglichkeiten, z.b. {ja,nein},{wahr,falsch},{schwarz,weiß},{links,rechts}, meist durch {0,1} codiert, technisch umgesetzt durch
    - Ladungen: 0 = ungeladen, 1 = geladen
    - Spannungen: 0 = 0 Volt, 1=5 Volt
    - Magnetisierung: 0 = unmagnetisiert, 1 = magnetisiert

  Wir gehen im folgenden von {0,1} als verfügbar aus.
- Bitfolgen
  - basieren auf Sequenzen $\{0,1\}^n$, n ist natürliche Zahl
  - erlauben Codierung von Mengen, z.B.

    000 = Süd 001 = West 010 = Nord 011 = Ost

    100 = Südost 101 = Nordwest 110 = Nordost 111 = Südwest

  Es gibt genau 2^N mögliche Bitfolgen der Länge N.
- Hexadezimalzahlen
  - Bitfolgen werden schnell unübersichtlich, daher Blöcke aus 4 Bits als „Ziffer“

    0000=0, 0001=1, 0010=2, 0011=3, 0100=4, 0101=5, 0110=6, 0111=7

    1000=8, 1001=9, 1010=A, 1011=B, 1100=C, 1101=D, 1110=E, 1111=F

  entspricht Zahlendarstellung zur Basis 16.

Grundbegriffe

- Byte
  - Rechner behandeln nicht einzelne Bits, kleinste betrachtete Bitfolge ist das Byte = 8 Bits,
  - grobere Granularität kommt nur als Vielfaches von 8 Bit vor,

    z.B. 16 Bit, 32 Bit, 64 Bit Rechner
  - 1 Byte erlaubt 2^8=256 Werte zu unterscheiden, z.B. zur Codierung von Buchstaben
- übliche Abkürzungen: B = Byte, b = bit

$1k = 1024 = 2^{10}$ (k = kilo)

$1M = 1024 \cdot 1024 = 2^{20}$ (M = mega)

$1G = 1024 \cdot 1024 \cdot 1024 = 2^{30}$ (G = giga)

$1T = 1024 \cdot 1024 \cdot 1024 \cdot 1024 = 2^{40}$ (T = tera)

$1P = 1024 \cdot 1024 \cdot 1024 \cdot 1024 \cdot 1024 = 2^{50}$ (P = Peta)

$1E = 1024 \cdot 1024 \cdot 1024 \cdot 1024 \cdot 1024 \cdot 1024 = 2^{60}$ (E = exa)

für Längen,Zeiten:

$1m = 10^{-3}$ (m = milli)

$1\mu = 10^{-6}$ (μ = mikro)

$1n = 10^{-9}$ (n = nano)

$1p = 10^{-12}$ (p = pico)

$1f = 10^{-15}$ (f = femto)

Datendarstellung

- Texte
- Programme
- Bilder
- Zahlen, Algebren
  - boolesche Algebra, Wahrheitswerte
  - natürliche Zahlen
  - ganze Zahlen
  - reellwertige Zahlen
- Bei der Darstellung von Zahlen werden wir erkennen, dass nicht alle angenehmen, aus der Mathematik vertrauten Eigenschaften von Zahlen auf einem Rechner erhalten bleiben.

Datendarstellung, Textdarstellung

- Texte: Zeichenfolgen aus Buchstaben und Satzzeichen
- Darstellung mittels Bitfolgen

=> Codierung jedes Buchstabens / Zeichens durch Bitfolge

Üblich:

- ASCII = American Standard Code for Information Interchange
  - 7 bit (= max 128 Zeichen), Tabelle mit Nummerierung aller Zeichen
  - z.B. „a“ hat Nummer 97, „A“ hat Nummer 65, „?“ hat Nummer 63
  - Klein- und Großbuchstaben nach Alphabet durchnumeriert
  - übliche Erweiterung auf PCs: 8bit, weitere Sonderzeichen, z.B. Umlaute
  - Erweiterung in Europa: Latin-1, (nach Norm ISO 8859-1)
- aber auch Unicode, z.B. von Java verwendet
  - 16 bit (= max 65526 Zeichen)
  - siehe http://www.unicode.org
  - als Obermenge weltweit geläufiger Zeichensätze

Datendarstellung, Programme, Grafiken

Programme

- Ein Programm wird zunächst meist als Text (Quelltext) erzeugt und wie normaler Text repräsentiert.
- Übersetzungsprogramme (Compiler) erzeugen daraus Programmcode in Maschinensprache.
- In jeder Form müssen alle Anteile eines Programms durch Bitfolgen codiert werden.

Grafiken, Bilder

- Rastergrafik
  - Grafik wird aus einer Folge einzelner Rasterpunkte dargestellt
  - einzelner Rasterpunkt durch 1 Bit, 1 oder mehrere Bytes (wg Farbe) codiert
- Vektorgrafik
  - Grafik wird aus Linien zusammengesetzt
  - für die Anfangs- / Endpunkte etc codiert werden müssen

Datendarstellung: Logische Werte, Boolesche Werte

- Boolesche Algebra: {false,true} (oft auch als {0,1}) mit Operationen
  - Und-Verknüpfung: AND,
  - Oder-Verknüpfung: OR,
  - Negation: NOT,
  - Exklusives Oder: XOR
- Darstellung in Rechnern erfordert meist 1 Byte (mindestens) als kleinster behandelter Dateneinheit
  - 1 Bit wäre im Prinzip ausreichend, jedoch im Rechner ist ein einzelnes Bit nur als Element innerhalb eines Bytes und über das zugehörige Byte adressierbar
  - Bitfelder dagegen lassen sich mit Platzverbrauch 1 Bit je boolescher Variable verwalten, wobei das Bitfeld insgesamt jedoch eine Größe in ganzen Bytes haben muss.

Datendarstellung: Darstellung von Zahlen

Satz

Jede natürliche Zahl n besitzt zur Basis p ≥ 2 (b aus N) eine eindeutige m-stellige p-adische Darstellung der Form

$$n = \sum_{i=0}^{m-1} \alpha_i \cdot p^i \quad \text{mit } 0 \leq \alpha_i < p \text{ und } m \geq \log_p n$$

Bemerkung: Ziffern dürfen Basiswert p nicht erreichen!

- Für uns üblich: Dezimalzahlen p=10 m nach Bedarf

$$1996_{10} = 1 \cdot 10^3 + 9 \cdot 10^2 + 9 \cdot 10^1 + 6 \cdot 10^0$$

- In Rechner üblich: Binärzahlen p=2 m=16,32 oder 64

$$1110_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 8_{10} + 4_{10} + 2_{10} = 14_{10}$$

- gelegentlich zur Dokumentation von Zahlenwerten / Adressen:
  Hexadezimal p=16 oder Oktal p=8

Umrechnung

Umrechung von Dezimalzahlen in Binärzahlen durch ganzahlige Division und Modulo Operation, d.h.

$$n = \sum_{i=0}^{m-1} \alpha_i \cdot p^i = \mathrm{p} \cdot \left(\sum_{i=1}^{m-1} \alpha_i \cdot p^{i-1}\right) + \alpha_0$$

$$\alpha_0 = n \bmod p \qquad \sum_{i=1}^{m-1} \alpha_i \cdot p^{i-1} = n \div p$$

$$\alpha_i = (n \div p^i) \bmod p$$

also fortgesetztes Dividieren,

der Rest r (mathematisch formal: modulo) liefert die Ziffernfolge,

z.B. $4711_{10} = 1001001000111_2$

4711 / 2 = 2355 mit Rest 1 -> „rechteste“ 1 in der Binärdarstellung

2355 / 2 = 1177 mit Rest 1 -> „vorletzte“ 1 in der Binärdarstellung

Anmerkung: geläufige Rechenoperationen sind für p-adische Zahlendarstellung unabhängig von p gültig

Aber Achtung: bisher nur natürliche Zahlen !!! Ganze Zahlen machen Ärger!

Datendarstellung: Darstellung von ganzen Zahlen

ein 1. Versuch:

- Extra Bit für das Vorzeichen, 0: positive Zahl, 1: negative Zahl
- 4 Bit (16 Werte) reichen also für -7,...,+7 wobei +0, -0 unterschieden werden

Nachteil:

- Addition (Subtraktion) ist nicht mehr so einfach wie üblich

-3	1011
+ +5	+ 0101
-----	---------
= +2	= ????

- üblicher Ausweg: Zweierkomplementdarstellung

Bemerkung: Der Freiheitsgrad, eine Codierung von Zahlen frei wählen zu können, wird gezielt genutzt, um Rechenoperationen einfach zu halten.

Datendarstellung: Darstellung von ganzen Zahlen

Zweierkomplement für feste Anzahl Bits m, hier m = 4

also 16 Zahlen können codiert werden, wir wählen {-8,-7, ..., 6,7}

- Idee 1: wähle 0000 als 0
- Idee 2: wähle für positive Zahlen herkömmliche binäre Codierung, Vorteil: Addition für positive Zahlen nach Schulmethode
- Idee 3: verwende Vorzeichenbit, aber
  - codiere kleinste negative Zahl -8 = 1000,
  - -7 = 1001, -6 = 1010, -5 = 1011, -4 = 1100, -3 = 1101, -2 = 1110
  - codiere größte negative Zahl -1 = 1111,

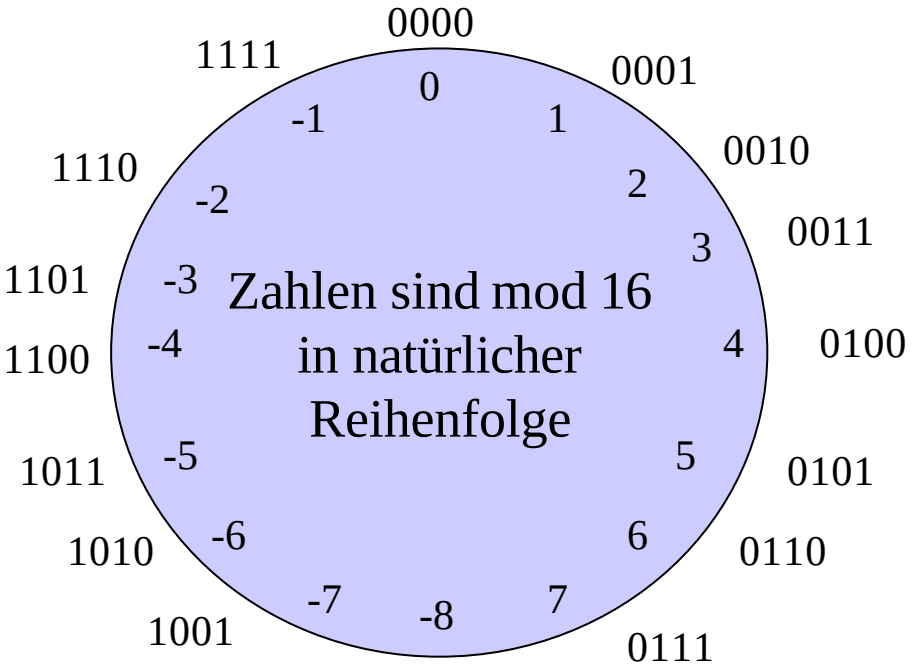
Vorteile:

- Addition für ganze Zahlen nach Schulmethode ist ok
- Reihenfolge der Werte modulo 16 ist ok
- Vorzeichen ist unmittelbar erkennbar

Allgemein, für m Bits

$$z = \alpha_{m-1} \cdot (-2^{m-1}) + \sum_{i=0}^{m-2} \alpha_i \cdot 2^i$$

Zweierkomplementdarstellung



-4 mod 16 = 12 mod 16 = 1100
-1 mod 16 = 15 mod 16 = 1111
-8 mod 16 = 8 mod 16 = 1000
d.h. alle Zahlen mit Differenz als Vielfachem von 16 habe die selbe Darstellung

- Besonderheit: 1000
  - eine Zahl (Bitfolge) und ihr bitweises Komplement haben als Summe gerade 1111 = -1, d.h. x + komplement(x) = -1
  - in umgekehrter Richtung genutzt, macht dies die Negation einer Zahl einfach: x + (-x) = 0, also -x = komplement(x)+1
  - damit wird die Subtraktion von Zahlen durch Negation und Addition einfach
- Anmerkung: wegen der richtigen Anordnung funktioniert die Addition für Binärzahlen und Zweierkomplementzahlen gleichermassen
- aber Achtung: begrenzter Wertebereich!!! Zulässigkeit prüfen !!!

Überprüfung der Zulässigkeit von Resultaten

- Bei fester Bitzahl m wird nur ein fester, endlicher Wertebereich von Zahlen repräsentiert.
- Dieser Wertebereich kann durch arithmethische Operationen verlassen werden
  - nach oben: Overflow, z.B.

    bei m=4 und Binärzahlen:
    - rechne 10 + 7, dh 1010 + 0111 liefert 1 0001, also Übertrag an der 1. Stelle

    bei m=4 und Zweierkomplement:
    - rechne 1 + 7, dh 0001 + 0111 liefert 0 1001, also Übertrag in das Vorzeichenbit hinein
  - nach unten: Underflow, z.B. bei m=4 und Zweierkomplement:
    - rechne -4 - 6, d.h. 1100 + 1011 liefert 1 0111, also Übertrag an der 1. Stelle
- Allgemein: sei a und b die Einträge in der 1. Stelle und c Übertrag, der in die 1 Stelle gelangt, dann ist das Resultat unzulässig, wenn
  - bei Binärzahlen: $a+b+c > 1$
  - bei Zweierkomplementzahlen: $a+b-c = -1$ oder $a+b-c = 2$

Datendarstellung: Darstellung von Zahlen

- Standardformate
  - wie bereits angedeutet, Rechner realisieren Zahlendarstellungen nur für bestimmte Wertebereiche (festes m),
  - diese Wertebereiche dienen als Datentyp für Variable, analog zu

    $x \in N$ wird vereinbart : int x
  - z.B. in einer Programmiersprache wie Delphi (Turbo-Pascal) oder Java werden folgende Bereiche angeboten

Bereich	Größe	Delphi	Java
-128,...,127	8 bit	shortint	byte
-32768,...,32767	16 bit	integer	short
$-2^{31},...,2^{31}-1$	32 bit	longint	int
$-2^{63},...,2^{63}-1$	64 bit		long
0,...,255	8 bit	byte	
0,...,65535	16	word	

Datendarstellung: Gleitpunktzahlen

- bisher natürliche und ganze Zahlen
- bei gebrochenen Zahlen zeigen sich Schwierigkeiten bei der Genauigkeit der Darstellung

$$x = \sum_{i=0}^{m-1} \alpha_i \cdot 2^i + \sum_{i=1}^{n-1} \alpha_i \cdot 2^{-i}$$

  - Hinweise auf Schwierigkeiten gibt es bereits im Dezimalsystem
    - π, keine endliche Darstellung im Dezimalsystem
    - Periodische, gebrochene Dezimalzahl bei 1/3
  - bei Binärzahlen tritt Problem auch bei Zahlen mit endlicher Dezimaldarstellung auf:
    - dezimal 0.1 wird binär zu 0.0001100110011001...
  - wiederum stehen natürlich nur eine endliche und feste Anzahl Bits zur Verfügung ...
- Abhilfe: Gleitpunktzahlen, Real-Zahlen

Datendarstellung: Gleitpunktzahlen

- Gleitpunktzahlen bestehen aus 3 Teilen
  - Vorzeichenbit: V, gibt an, ob die Zahl positiv oder negativ ist
  - Exponent: E, gibt für eine Basis (typisch: p=2), einen Exponenten als Binärzahl an, mit der die Mantisse zu multiplizieren ist
  - Mantisse: M, ist eine Folge von Binärziffern $m_1,...,m_n$, die interpretiert wird als

$$m_1 \cdot 2^{-1} + m_2 \cdot 2^{-2} + ... + m_n \cdot 2^{-n}$$

- Gleitpunktzahlen werden typischerweise normiert, d.h. der Exponent wird so gewählt, dass vor dem Komma eine 1 steht. Dadurch werden keine Bits der Mantisse durch führende Nullen verschwendet: $1, m_1 m_2 ... m_n \cdot 2^E$

- Beispiel, 32 bit, bias für E ist +127, um negatives E zu vermeiden

  0 01110010 01001010110101101111111

  V:+ E: 114-127=-13 M: 23 bit Zahlenwert: 0,0001577567163622

- Umrechung zwischen Dezimal- und Binären Gleitpunktzahlen führen zu Rundungsfehlern aufgrund endlicher, fester Anzahl Bits in der Darstellung!

Datendarstellung: Gleitpunktzahlen

- Umrechung und arithmethische Operationen führen zu Rundungsfehlern mit Gleitpunktzahlen wegen der festen Anzahl Bits für die Zahlendarstellung

V(1bit)	E(8bit)	Mantisse(23bit)	Zahlenwert
0	01111011	10011001100110011001100	0,0999999940...
0	01111011	10011001100110011001101	0,1000000015...

Resultate sind also nicht exakt im mathematischen Sinne !!!

Für praktische Anwendungen sind Gleitpunktzahlen als Darstellung von reellwertigen Zahlen jedoch durchaus brauchbar, dennoch sollte man sich des grundsätzlichen Problems bewusst sein.

Eine konkrete praktische Auswirkung für die Programmierung ist, dass Vergleiche von Gleitpunktzahlen x,y auf Gleichheit nur bzgl eines Abweichungsepsilons sinnvoll sind:

$$|x - y| < \varepsilon$$

Datendarstellung: Gleitpunktzahlen

- Gleitpunktzahlen in Programmiersprachen

Bereich	Bytes	Delphi	Java
+- 2,9 E -39 .. 1,7 E 38	6	real	
+- 1,5 E -45 .. 3,4 E 38	4	single	float
+- 5,0 E -324 .. 1,7 E 308	8	double	double
+- 3,4 E -4932 .. 1,1 E 4932	10	extended	

Die Notation mit „E“ wie Exponent bedeutet:

$$3{,}1415\ \mathrm{E}\ 2 = 3{,}1415 \cdot 10^2 = 314{,}15$$

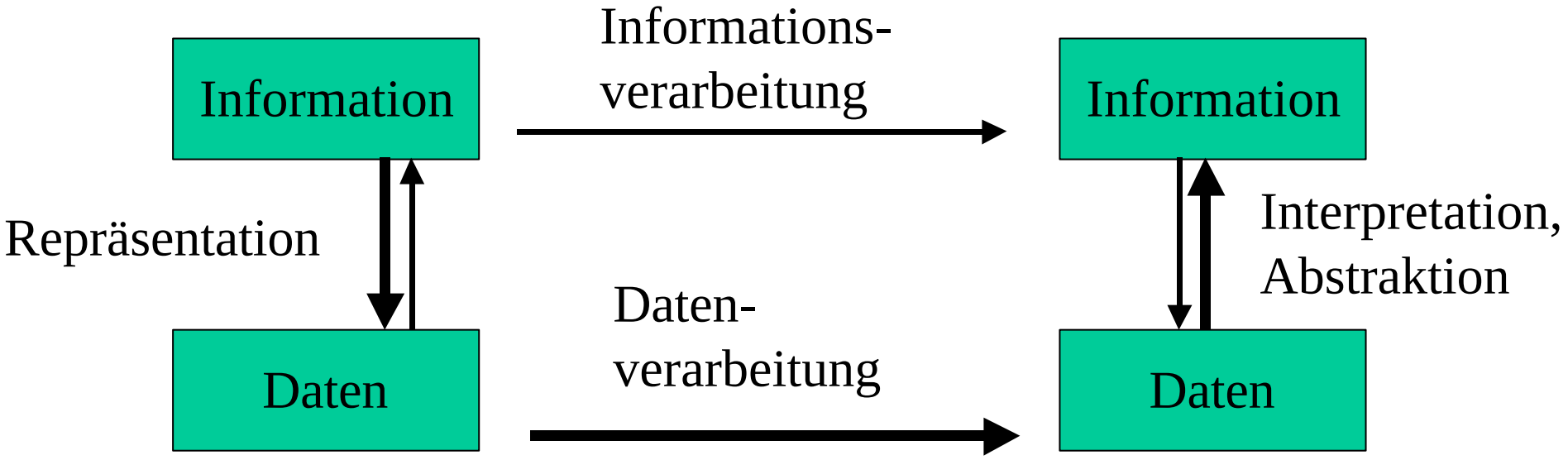
und entstammt Norm IEEE 754, ist in Programmiersprachen üblich.

Single / double wird auch als Rechnen mit einfacher oder doppelter Genauigkeit (precision) bezeichnet.

Mit 2^{64} Elementen lassen sich viele numerische Probleme hinreichend genau behandeln.

Daten vs Informationen

Wunsch: Informationsverarbeitung Wirklichkeit: Datenverarbeitung



bisher betrachtet:

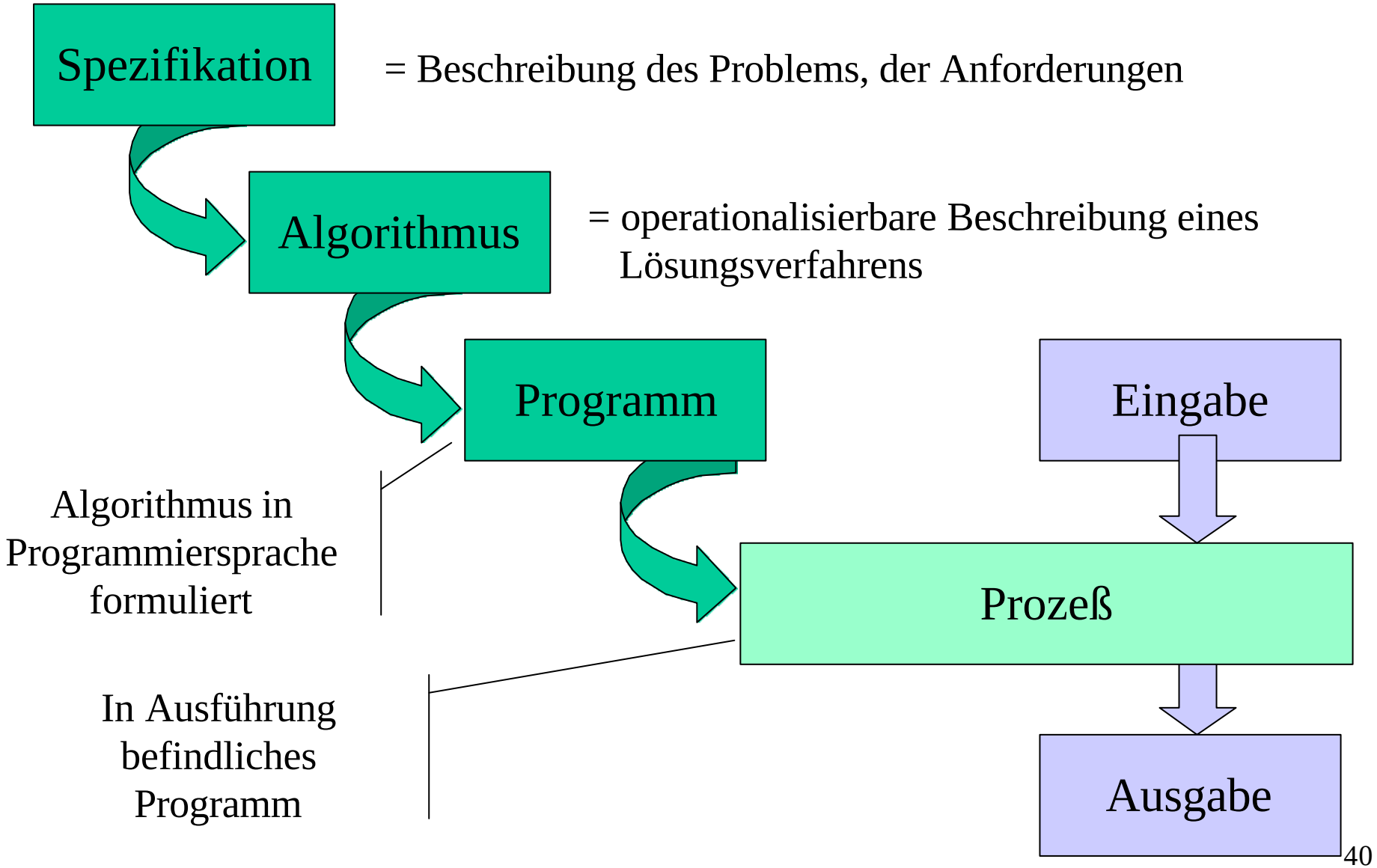
Behandlung von einfachen mathematischen Objekten, nämlich Zahlen (natürliche, ganze Zahlen, reellwertige Zahlen)

=> Repräsentation und Interpretation wesentlich um Rechensystem mit seinen Fähigkeiten zur Datenverarbeitung für die Informationsverarbeitung sinnvoll nutzen zu können

Wo stehen wir jetzt ?

- Was ist Informatik ?
- Was macht ein Rechensystem aus ?
- Wie werden Daten in einem Rechner dargestellt ?
  - Buchstaben, Zeichenketten, Texte, ...
  - Grafiken
  - Algebren
    - Boolesche Algebra mit Operationen AND, OR, NOT
    - Natürliche Zahlen, ganze Zahlen, reellwertige Zahlen mit Operationen Addition, Subtraktion, Multiplikation, Division, Modulo, ...
    - Achtung: Genauigkeit der Darstellung und damit auch von Berechnungen ist begrenzt !!! Wertebereiche für Zahlen sind beschränkt !!!
- Wir wollen uns im folgenden mit dem Entwurf von Algorithmen und Programmen, der Programmierung von Rechensystemen und zugehörigen Programmiersprachen befassen.

Stationen im Entwurf von Algorithmen und Programmen



Wie geht es weiter ?

- Grundlagen der Programmierung
  - Spezifikationen, Algorithmen, Programme
  - Daten und Datenstrukturen
  - imperative Programmierung:
    - Verfeinerung, elementare Operationen, Sequenz, Selektion, Iteration, funktionale Algorithmen und Rekursion, Variablen und Wertzuweisungen, Prozeduren, Funktionen und Modularität, Zuweisung, Sequenz
  - objektorientierte Programmierung
- Programmiersprachenkonzepte
  - Syntax und Semantik
  - imperative, objektorientierte, funktionale und logische Programmierung
- Berechenbarkeit und Entscheidbarkeit von Problemen
- Effizienz und Komplexität von Algorithmen
- Programmentwurf, Softwareentwurf