

Informatik I

Literatur

- Max Hailperin, Barbara Kaiser and Karl Knight. *Concrete Abstractions: An Introduction to Computer Science Using Scheme*. PWS Publishing, 1998.
- Harold Abelson, Gerald Jay Sussman with Julie Sussman. *Structure and Interpretation of Computer Programs*, 2. Auflage. MIT Press, 1996.
- Gerhard Goos. *Vorlesungen über Informatik*, Band 1. Springer-Verlag, 1997.

1 Grundlagen

1.1 Was ist Informatik?

Informatik = Information + Mathematik

computer science, computing science

Ingenieurwissenschaft, Systemwissenschaft

Definition der ACM

(Association of Computing Machinery)

Computer science is the systematic study of algorithms and data structures, specifically

1. their formal properties,
2. their mechanical and linguistic realizations, and
3. their applications.

Core Subjects

- Principles of Computer Organization
- Algorithms
- Theory of Computation
- Principles of Programming Languages

Teilgebiete

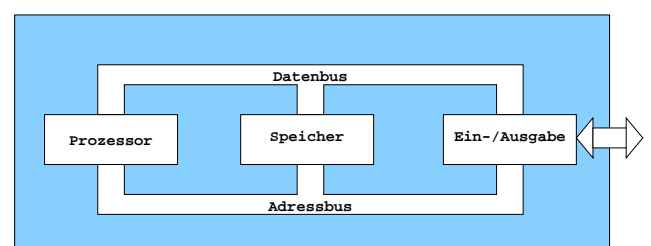
- Theoretische Informatik
- Technische Informatik
- Praktische Informatik
- Angewandte Informatik
- Informatik und Gesellschaft

1.2 Computer

1.2.1 Was ist ein Computer?

Eine Einheit, die Befehle ausführen kann.

1.2.2 Woraus besteht ein Computer?



Von Neumann-Rechner

Zentraleinheit

(central processing unit, *CPU*, eine oder mehrere)
Ausführung der Befehle

Speicher

Ablage für Daten und Befehle
Primärspeicher: random access memory (RAM, Hauptspeicher), read only memory (ROM)
Sekundärspeicher: Diskette, Festplatte, CD-ROM, DVD
Magnetbänder

Ein-/Ausgabe

Tastatur, Monitor, Maus, Drucker, Modem, Netzwerk

- Struktur des Rechners unabhängig vom Problem
→ Steuerung durch Programm
- Speicher enthält Programme und Daten
Organisation:
 - Hauptspeicher
identische Speicherzellen, Adressen
 - Sekundärspeicher
hierarchisches System (Dateien und Ordner)

John von Neumann

- geboren 28.12.1903 in Budapest
- gestorben 8.2.1957 in Washington, D.C.
- Studium der Chemie und Mathematik
- "Why would you want more than machine language?"

1.2.3 Womit rechnet ein Computer?

Zwei Zustände: aus / ein

Binärcode: Grundmenge $\mathbf{B} = \{\mathbf{0}, \mathbf{1}\}$ (Zeichenvorrat)

Bit: Element aus \mathbf{B}

Bitfolgen: 1 *Byte* $\hat{=}$ 8 Bit

1 *Wort* $\hat{=}$ 4 Byte $\hat{=}$ 32 Bit

1 *Langwort* $\hat{=}$ 2 Worte $\hat{=}$ 8 Byte $\hat{=}$ 64 Bit

Interpretation:

Zahlen, Zeichen, Speicheradressen, Befehle, ...

Wortlänge: Länge der Bitfolge, die die CPU verarbeiten kann

1.2.4 Welche Operationen kann ein Computer ausführen?

Logische Operationen: $\mathbf{0} \hat{=}$ falsch, $\mathbf{1} \hat{=}$ wahr

logisches Und			logisches Oder			logisches Nicht	
x	y	$x \wedge y$	x	y	$x \vee y$	x	$\neg x$
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1	0	1
1	1	1	1	1	1	1	0

Binäre Arithmetik: $\mathbf{0} \hat{=}$ 0, $\mathbf{1} \hat{=}$ 1

Addition			Multiplikation		
x	y	$x + y$	x	y	$x \cdot y$
0	0	0	0	0	0
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	1

Abgeleitete Operationen: $x + y = (\neg x \wedge y) \vee (x \wedge \neg y)$

$x \cdot y = x \wedge y$

Binäre Zahlendarstellung $\hat{=}$ Folge von Bits

Stellenwertsystem mit Basis 2

$$\begin{aligned}
 42_{10} &= 4 \cdot 10^1 + 2 \cdot 10^0 \\
 &= 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\
 &= 101010_2
 \end{aligned}$$

4	2	1	0	1	0	1	0
+	9	+	1	0	0	1	
<hr/>		1		<hr/>		1	
5	1	1	1	0	0	1	1

Weitere gebräuchliche Darstellungen

Oktalardarstellung (zur Basis 8)

$$\begin{aligned}
 42_{10} &= 5 \cdot 8^1 + 2 \cdot 8^0 \\
 &= 52_8
 \end{aligned}$$

Hexadezimaldarstellung (zur Basis 16)

$$\begin{aligned}
 42_{10} &= 2 \cdot 16^1 + 10 \cdot 16^0 \\
 &= 2A_{16}
 \end{aligned}$$

(benutze *A-F* als Ziffern für 10-15)

1.2.5 Was steuert einen Computer?

Ein Programm legt fest, **welche Daten** der Prozessor bearbeitet und **was** mit den Daten geschieht.

- Programm: Daten mit spezieller Bedeutung
- Programm beschreibt Berechnungsprozess
- Abstrakte Beschreibung: Algorithmus

1.3 Algorithmen beschreiben Prozesse

Beispiel: Umwandlung in Binärdarstellung

Sei $n \geq 0$ eine ganze Zahl.

1. Falls $n = 0$, schreibe "0". Fertig.
2. Teile n durch 2 mit Ergebnis q und Rest r .
3. Schreibe r als Ziffer.
4. Falls $q = 0$, fertig.
5. Weiter bei 2. mit $n \leftarrow q$.

Beispiel: Grosseinkauf

- Erstelle einen Einkaufszettel.
- Fahre zum Supermarkt.
- Fülle einen Einkaufswagen gemäss Einkaufszettel.
- Bezahle.
- Lade die gekauften Waren ein.
- Fahre nach Hause.

Verfeinerung: Männlicher Einkäufer

„Fülle einen Einkaufswagen gemäss Einkaufszettel“

1. Wähle eine Ware vom Einkaufszettel aus.
2. Fahre mit dem Einkaufswagen zum Standort der Ware.
3. Lade die gewünschte Menge in den Einkaufswagen.
4. Streiche die Ware vom Einkaufszettel.
5. Falls noch Waren auf dem Einkaufszettel, fahre mit Punkt 1 fort.

Weitere Beispiele

- Kochrezepte
- Wegbeschreibungen
- Spielregeln

1.3.1 Definition: Algorithmus

Vorschrift zur Durchführung eines Prozesses mit folgenden Eigenschaften:

Effektivität

Jeder Teilschritt ist ausführbar.

Determiniertheit

Der nächstfolgende Teilschritt ist immer festgelegt.

Fintheit

Die Vorschrift ist endlich.

Terminierung

Die Prozess endet nach endlich vielen Teilschritten.

1.4 Programmierung

1.4.1 Was ist eine Programmiersprache?

Sprache zum Aufschreiben von Algorithmen

- ausführbar auf einem (realen) Computer
- künstliche (formale) Sprache, da sie vom Computer „verstanden“ werden muss

1.4.2 Arten von Programmiersprachen

Maschinensprachen

Bitmuster im Speicher, das die Ausführung der Befehle steuert

Abhängig vom Prozessor

Beispiel: FF54F330

Assemblersprachen

Mnemonische Abkürzungen für Bitmuster

Verwendung von symbolischen Namen

Abhängig von Prozessorfamilie

Beispiel: `call 48(%ebx,%esi,8)`

Höhere Programmiersprachen

Virtuelle Maschine

- imperative Programmiersprachen:
FORTRAN, Pascal, C, Ada
- objekt-orientierte Programmiersprachen:
Smalltalk, Eiffel, Cecil, Self, [Java](#)

Abstraktes Modell

funktionale und logische Programmiersprachen:
[Scheme](#), Haskell, ML, Prolog

1.4.3 Elemente von Programmiersprachen

Grundbausteine

- Schreibweisen für Konstanten (Literele)
vgl. Wörter mit fester Bedeutung (aus dem Wörterbuch)
- Namen (Bezeichner, Identifier)
Wörter mit frei wählbarer Bedeutung

Kombinationsmittel zur Konstruktion von Programmstücken aus vorhandenen Programmstücken
vgl. Grammatik

Abstraktionsmittel zur Benennung von Programmstücken

1.4.4 Der Programmierprozess

1. Konstruieren von Programmstücken mithilfe von Grundbausteinen und Kombinationsmitteln.
2. Benennen von Programmstücken mithilfe von Abstraktionsmitteln, die Programmstücke vergessen, und nur die Namen weiter verwenden.
3. Weiter bei Punkt 1.

1.4.5 Was jedes Programm braucht

- Beschreibung der Eingabe- und Ausgabe-Daten
- Beschreibung der Wirkung des Programms (mit Beispielen)
- Definitionen (Programmstücke)
- Testumgebung

1.4.6 Beispiel: Parkplatzproblem

Auf einem Parkplatz stehen PKWs und Motorräder. Es handelt sich um n Fahrzeuge mit r Rädern.

Bestimme die Anzahl P der PKWs.

Lösungsansatz: Lineares Gleichungssystem

Sei M die Anzahl der Motorräder.

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

Lösung des Gleichungssystems

Multipliziere erste Gleichung mit -2 und addiere beide Gleichungen.

$$\begin{array}{r} -2M + -2P = -2n \\ 2M + 4P = r \\ \hline 2P = r - 2n \end{array}$$

Also: $P = r/2 - n$

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. Es handelt sich um n Fahrzeuge mit r Rädern.

Bestimme die Anzahl P der PKWs.

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

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

Ausgabe: $P \in \mathbf{N}$, falls die Nachbedingung erfüllbar, sonst „Keine Lösung“

Nachbedingung: Für gewisse $M, P \in \mathbf{N}$ gilt

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