

2. Daten und Algorithmen

- 2.1 Zeichenreihen
- 2.2 Datendarstellung durch Zeichenreihen
- 2.3 Syntaxdefinitionen
- 2.4 Algorithmen

2. Daten und Algorithmen

- 2.1 Zeichenreihen
- 2.2 Datendarstellung durch Zeichenreihen
- 2.3 Syntaxdefinitionen
- 2.4 Algorithmen

Wir betrachten zunächst die Daten (Objekte), die durch Algorithmen verarbeitet werden sollen.

Typische Daten sind Zahlen, z.B. die Zahl “drei”, die wie folgt dargestellt werden kann:

- 3
- DREI
- III
- drei ausgestreckte Finger einer Hand
- ...

- Wir unterscheiden bei einem Objekt
 - die Darstellung, (*Syntax*, “Bezeichnung”),
 - seine Bedeutung, (*Semantik*, “Information”).
- Einige Datendarstellungen sind für maschinelle Verarbeitung nicht geeignet.
- Alle geeigneten Datendarstellungen beruhen auf dem Grundprinzip der *Zeichenreihe*.

- Ein *Alphabet* ist eine endliche Menge, deren Elemente *Zeichen* genannt werden.
- Beispiele:
 - Menge der Großbuchstaben: $\{A, B, C, \dots, Z\}$
 - Menge der Dezimalziffern: $\{1, 2, 3, \dots, 9\}$
 - Menge der Vorzeichen: $\{+, -\}$
 - Menge der Richtungszeiger eines Lifts: $\{\uparrow, \downarrow\}$
- Alphabete, die genau zwei Zeichen enthalten heißen *binär*.
- Ein wichtiges binäres Alphabet besteht aus den *Binärziffern (Bits)* $\{0, 1\}$.

- Eine Zeichenreihe über einem Alphabet $\mathcal{A}$ ist eine (endliche) Folge von Zeichen aus $\mathcal{A}$.
- Formal ist auch eine leere Folge eine Zeichenreihe.
- Wir schreiben Zeichenreihen/Folgen $(x_1, x_2, \dots, x_n)$ auch als $x_1 x_2 \dots x_n$.
- Beispiele:
 - Sei $\mathcal{A}_1 = \{A, B, C, \dots, Z\}$
 - Die Folge INFORMATIK ist eine Zeichenreihe über $\mathcal{A}_1$.
 - Die Folge (Z,I,M,E,K) ist eine Zeichenreihe über $\mathcal{A}_1$.
 - Die Folgen Kröger und (H1234U) sind *keine* Zeichenreihen über $\mathcal{A}_1$.
Warum?
 - Sei $\mathcal{A}_2 = \{0, 1\}$
 - Die Folge 0 ist eine Zeichenreihe über $\mathcal{A}_2$.
 - Die Folge 1 ist eine Zeichenreihe über $\mathcal{A}_2$.
 - Die Folge 01 ist eine Zeichenreihe über $\mathcal{A}_2$.
 - Die Folge 10 ist eine Zeichenreihe über $\mathcal{A}_2$.
 - Die Folge 11 ist eine Zeichenreihe über $\mathcal{A}_2$.
 - Die Folge 00 ist eine Zeichenreihe über $\mathcal{A}_2$.
 - ...

2. Daten und Algorithmen

2.1 Zeichenreihen

2.2 Datendarstellung durch Zeichenreihen

2.3 Syntaxdefinitionen

2.4 Algorithmen

- Wir verwenden ausschließlich Zeichenreihen zur Bezeichnung von Daten.
- Im folgenden betrachten wir als Beispiel die Darstellung von natürlichen Zahlen, also Elementen der Menge $\mathbb{N}_0$.
- Die Zahl “dreizehn” lässt sich u.a. durch folgende Zeichenreihen bezeichnen:
 - 13 ($\mathcal{A} = \{0, 1, 2, \dots, 9\}$),
 - DREIZEHN ($\mathcal{A} = \{A, B, \dots, Z\}$),
 - ||||| ($\mathcal{A} = \{| \}$).
- Nicht alle diese Darstellungen sind für den praktischen Gebrauch (z.B. Rechnen) geeignet.
- Am besten geeignet ist die *Zifferndarstellung*, z.B. die allgemein gebräuchliche *Dezimaldarstellung* über dem Alphabet $\{0, 1, 2, \dots, 9\}$.

- Das allgemeine Prinzip der Zifferndarstellung ist wie folgt definiert:

Sei $p \in \mathbb{N}$, $p \geq 2$ und $\mathcal{A}_p = \{z_0, z_1, \dots, z_{p-1}\}$ ein Alphabet mit p Zeichen (Ziffern) $z_0, z_1, \dots, z_{p-1}$.

Die Funktion $Z : \mathcal{A}_p \rightarrow \mathbb{N}_0$ bildet jedes Zeichen aus $\mathcal{A}_p$ auf eine natürliche Zahl wie folgt ab:

$$Z(z_i) = i \quad \text{für } i = 0, \dots, p-1.$$

Eine Zeichenreihe $x = x_n x_{n-1} \dots x_1 x_0$ über $\mathcal{A}_p$ (d.h. $x_i \in \mathcal{A}_p$ für $0 \leq i \leq n$) bezeichnet die Zahl

$$\mathcal{Z}(x) = p^n \cdot Z(x_n) + p^{n-1} \cdot Z(x_{n-1}) + \dots + p \cdot Z(x_1) + Z(x_0).$$

- Zur Verdeutlichung schreiben wir auch x_p . x_p heißt p -adische Zahlendarstellung der Zahl $\mathcal{Z}(x_p) \in \mathbb{N}_0$.

- Nochmal die Formel

$$\mathcal{Z}(x) = p^n \cdot Z(x_n) + p^{n-1} \cdot Z(x_{n-1}) + \dots + p \cdot Z(x_1) + Z(x_0).$$

- Beispiele:

- $p = 10$ und $\mathcal{A}_{10} = \{z_0, z_1, \dots, z_9\}$ erhält man die Dezimaldarstellung wenn man statt z_i gleich $Z(z_i)$ schreibt (also z.B. statt z_3 schreibe $Z(z_3) = 3$):

$$\mathcal{Z}(983_{10}) = 10^2 \cdot 9 + 10^1 \cdot 8 + 10^0 \cdot 3 = \text{“neunhundertdreiundachtzig”}.$$

(Wir schreiben direkt $\mathcal{A}_{10} = \{0, 1, \dots, 9\}$)

- $p = 2$ und $\mathcal{A}_2 = \{0, 1\}$ (Binärdarstellung):

$$\begin{aligned} \mathcal{Z}(1111010111_2) &= \\ 2^9 \cdot 1 + 2^8 \cdot 1 + 2^7 \cdot 1 + 2^6 \cdot 1 + 2^5 \cdot 0 + 2^4 \cdot 1 + 2^3 \cdot 0 + 2^2 \cdot 1 + 2 \cdot 1 + 1 &= \\ \text{“neunhundertdreiundachtzig”}. \end{aligned}$$

- $p = 8$ und $\mathcal{A}_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$ (Oktaldarstellung):

$$\mathcal{Z}(1727_8) = 8^3 \cdot 1 + 8^2 \cdot 7 + 8 \cdot 2 + 7 = \text{“neunhundertdreiundachtzig”}.$$

- $p = 16$ und $\mathcal{A}_{16} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

(Hexadezimaldarstellung):

$$\mathcal{Z}(3D7_{16}) = 16^2 \cdot 3 + 16 \cdot 13 + 7 = \text{“neunhundertdreiundachtzig”}.$$

- Führende Nullen sind in der Definition der p -adischen Zahldarstellung zugelassen, z.B. ist die Zeichenreihe 000983 eine zulässige Dezimaldarstellung und bezeichnet die gleiche Zahl wie die Zeichenreihe 983.
- Offensichtlich können führende Nullen (d.h. führende Ziffern 0, also z_0) immer weggelassen werden, außer bei der Bezeichnung “0” für die Zahl “null”.
- Für jede Zahl aus $\mathbb{N}_0$ gibt es für beliebiges $p \geq 2$ eine p -adische Darstellung.
- Betrachtet man nur Darstellungen ohne führende Nullen, so ist (zu festem $p \geq 2$) die p -adische Zahldarstellung eindeutig.

- Zur Entwicklung von Algorithmen werden wir typischerweise die Dezimaldarstellung der natürlichen Zahlen verwenden.
- Allgemein gibt es für die meisten Daten eine *Standarddarstellung* bei denen die Lesbarkeit der Darstellung für den menschlichen Benutzer im Vordergrund steht.
- Wir verwenden hier folgende Standardbezeichnungen wie sie in den üblichen höheren Programmiersprachen gebräuchlich sind:
 - Natürliche Zahlen $\mathbb{N}_0$: Dezimaldarstellung (ohne führende Nullen).
 - Ganze Zahlen $\mathbb{Z}$: wie natürliche Zahlen, ggf. mit Vorzeichen “-”, z.B. 3,-3.
 - Reelle Zahlen $\mathbb{R}$: *Gleitpunktdarstellung*, siehe später.
 - Wahrheitswerte $\mathbb{B}$: *TRUE* und *FALSE* für “wahr” bzw. “falsch”.

- Eine Dezimaldarstellung (z.B. 983) stellt eine natürliche Zahl dar, die verarbeitet werden kann.
- Die Zeichenreihe selbst (und nicht die dargestellte Zahl) könnte aber auch Gegenstand der Verarbeitung sein.
- Zeichenreihen können also nicht nur Darstellungen von Objekten sein, sondern auch selbst Objekte, die dargestellt werden müssen.
- Zeichenreihen heißen in diesem Zusammenhang *Texte*.
- In der Praxis wird zur Bildung von Texten häufig das sog. *ASCII-Alphabet* benutzt.
- Das ASCII-Alphabet repräsentiert eine Menge von Zeichen, die wir im folgenden als *CHAR* bezeichnen.
- Die Elemente von *CHAR* finden Sie in allen gängigen Lehrbüchern.

- Auf Rechenanlagen wird meist eine andere Darstellung der Daten gewählt (typischerweise Zeichenreihen über den oben benannten Alphabeten $\mathcal{A}_2$, $\mathcal{A}_8$ und $\mathcal{A}_{16}$).
- Aus technischen Gründen kann die kleinste Speichereinheit eines Computers (ein sog. *Bit*) nur zwei Zustände speichern:
 - Zustand 1: es liegt (elektr.) Spannung an.
 - Zustand 0: es liegt keine Spannung an.
- Daher werden Werte (Daten/Objekte) als Bitmuster (Zeichenreihe über dem Alphabet $\mathcal{A}_2 = \{0, 1\}$) codiert gespeichert.
- Intern kann der Computer also z.B. die natürlichen Zahlen in Binärcodierung repräsentieren.
- Ganze Zahlen können intern ebenfalls leicht als Zeichenkette über dem Alphabet $\mathcal{A}_2 = \{0, 1\}$ codiert werden (Genaueres darüber werden Sie in der Vorlesung “Rechnerarchitektur” lernen).

- Die Repräsentation der reellen Zahlen in Binärdarstellung ist etwas komplizierter.
- Üblicherweise wird sowohl im Rechner als auch in den höheren Programmiersprachen die sogenannte *Gleitpunktdarstellung* verwendet. Eine Zahl z wird z.B. im Rechner dargestellt durch

$$z = m \cdot 2^e,$$

wobei sowohl m (*Mantisse*) als auch e (*Exponent*) wiederum binär repräsentiert werden (können).

- In den meisten höheren Programmiersprachen (z.B. Java) wird eine Zahl dargestellt durch $z = m \cdot 10^e$, z.B. 3.14, -7.45, $1.33E - 2$ (für $1.33 \cdot 10^{-2}$).
- Eine genaue Spezifikation lernen wir im nächsten Abschnitt kennen.
- **Wichtig:** Für viele reelle Zahlen gibt es gar keine derartige Darstellung (z.B. für $\sqrt{2}$). Die darstellbaren Zahlen heißen auch *Gleitpunktzahlen*.
- Dieser Aspekt der maschinengerechten Darstellung von Daten ist bei der Entwicklung von Algorithmen möglicherweise wichtig! **Warum?**

- Ein weiterer Aspekt der maschinengerechten Darstellung ist, dass die Ressourcen (Speicherzellen) einer Rechenanlagen begrenzt sind.
- Es stehen daher für die Darstellung von Daten immer nur endlich viele Bits zur Verfügung.
- Typischerweise definiert jede Programmiersprache elementare (*primitive* oder *atomare*) *Datentypen*, die Teilmengen der obengenannten Mengen $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{B}$, *CHAR* sind.
- Dabei stehen zur Darstellung der Werte für jeden Datentyp eine fixe Anzahl von Bits zur Verfügung, d.h. die Zeichenketten der Werte eines Typs haben eine fixe *Länge*.
- Werte, deren Darstellung mehr als die für den Typ zur Verfügung stehenden Bits benötigt, können nicht dargestellt werden.
- Die Länge eines Datentyps hat damit offenbar Einfluss auf den Wertebereich des Typs.
- Auch dieser Aspekt der maschinengerechten Darstellung von Daten ist bei der Entwicklung von Algorithmen möglicherweise wichtig!

Warum?

- Die Programmiersprache Java bietet folgende primitive Datentypen (1 Byte = 8 Bit):

Typname	Länge	Wertebereich
<code>boolean</code>	1 Byte	Wahrheitswerte <code>{true, false}</code>
<code>char</code>	2 Byte	Alle Unicode-Zeichen
<code>byte</code>	1 Byte	Ganze Zahlen von -2^7 bis $2^7 - 1$
<code>short</code>	2 Byte	Ganze Zahlen von -2^{15} bis $2^{15} - 1$
<code>int</code>	4 Byte	Ganze Zahlen von -2^{31} bis $2^{31} - 1$
<code>long</code>	8 Byte	Ganze Zahlen von -2^{63} bis $2^{63} - 1$
<code>float</code>	4 Byte	Gleitkommazahlen (einfache Genauigkeit)
<code>double</code>	8 Byte	Gleitkommazahlen (doppelte Genauigkeit)

- Es gibt in Java also Grunddatentypen für $\mathbb{B}$, `CHAR`, verschiedene Teilmengen von $\mathbb{Z}$ und verschiedene Teilmengen von $\mathbb{R}$.

- Um Objekte eines bestimmten Typs zu verarbeiten stellen höhere Programmiersprachen auch Operationen auf Objekten des entsprechenden Typs zur Verfügung.
- Beispiel:
Für das “Rechnen” mit natürlichen Zahlen stehen die Grundrechenarten wie $+$, $·$, usw. als Basisoperationen zur Verfügung.
- Die Operationen, die Java für seine Grunddatentypen bereitstellt, lernen wir später kennen.
- Mathematisch formal handelt es sich bei diesen Grundoperationen typischerweise um *Funktionen*.
- Beispiel:
Zur Addition zweier natürlicher Zahlen verwenden wir die Funktion

$$+ : \mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$$

die zwei natürliche Zahlen auf eine natürliche Zahl abbildet.

- Die Funktion $+$ ist also eine zweistellige Funktion. Grundsätzlich sind natürlich n -stellige Funktionen ($n \geq 0$) erlaubt. **Ist die Bedingung $n \geq 0$ im Kontext von Programmiersprachen sinnvoll?**

- Üblicherweise schreiben wir statt “ $+(x, y)$ ” “ $x + y$ ” um die beiden Zahlen $x \in \mathbb{N}_0$ und $y \in \mathbb{N}_0$ zu addieren.
- Diese Schreibweise wird *Infixschreibweise* genannt.
- Grundsätzlich gibt es
 - *Präfixschreibweise*: im Beispiel $+(x, y)$;
 - *Infixschreibweise*: im Beispiel $(x + y)$;
 - *Postfixschreibweise*: im Beispiel $(x, y) +$;
- Operationen, die als Ergebnis Objekte vom Typ $\mathbb{B}$ ergeben, heißen auch *Prädikate*, z.B. die Operation $<$ zum Vergleich zweier natürlicher Zahlen:

$$<: \mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{B}$$

($1 < 2$ ergibt den Wert $TRUE \in \mathbb{B}$, $3 < 1$ ergibt den Wert $FALSE \in \mathbb{B}$).

- Boolesche (Wahrheits-)Werte $\mathbb{B} = \{TRUE, FALSE\}$:

$$\wedge : \mathbb{B} \times \mathbb{B} \rightarrow \mathbb{B}$$

$$\vee : \mathbb{B} \times \mathbb{B} \rightarrow \mathbb{B}$$

$$\neg : \mathbb{B} \rightarrow \mathbb{B}$$

- Natürliche Zahlen $\mathbb{N}_0 = \{0, 1, 2, \dots\}$:

$$+ : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$

$$- : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$

$$\cdot : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$

$$: : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$

$$= : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{B}$$

$$\neq : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{B}$$

$$> : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{B}$$

$$< : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{B}$$

- Ganze Zahlen $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$:
 $+, -, \cdot, :, =, \neq, <, >, \dots$
- Reelle Zahlen $\mathbb{R}$ (die reellen Zahlen):
 $+, -, \cdot, :, =, \neq, <, >, \dots$
- Zeichen (Charakter) $CHAR = \{"A", "B", \dots, "a", "b", \dots, "1", "2", \dots, "!", \dots\}$
(z.B. alle druckbaren ASCII-Zeichen)
 $=, \neq, <, >, \dots$
- **Achtung:** obwohl z.B. $+: \mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{N}_0$ und $+: \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ gleich "benannt" sind, sind es zwei unterschiedliche Operationen!
- Wenn zwei unterschiedliche Operationen gleich benannt sind, spricht man von *Überladen*.
- **Achtung:** Obwohl Operationen auf den Grunddatentypen meist als gegeben vorausgesetzt werden, verbirgt sich hinter jeder Operation typischerweise wieder ein Algorithmus zu deren Berechnung.