

Lösungsvorschlag

Thema: Information, Repräsentation/Abstraktion, Bits/Bytes, Textkodierung

Präsenzaufgaben

Aufgabe 1: Information vs. Daten.

Kernidee: *Daten* sind rohe Zeichen/Zahlen (Bits, Bytes). *Information* entsteht erst durch *Interpretation im Kontext*.

Beispiele:

- 42: als Temperatur \Rightarrow 42°C (heiß), als Alter \Rightarrow 42 Jahre, als Hausnummer \Rightarrow Adresse.
- 2025-08-23: als Datum (23. 8. 2025) \Rightarrow Zeitpunkt; als Teil einer Artikelnummer \Rightarrow ID-Fragment ohne Zeitbedeutung.

Merke: Dieselben Daten liefern je nach Kontext unterschiedliche Information.

Aufgabe 2: Repräsentation oder Abstraktion? (mit Begründung)

Aufgabe 2:a) *Sensor misst Temperatur \rightarrow Zahl in $^{\circ}\text{C}$* : **Repräsentation**. Ein physikalischer Zustand wird als Datenzahl dargestellt.

Aufgabe 2:b) *Bildbetrachter zeigt PNG als Foto an*: **Abstraktion** (Interpretation). Aus Dateidaten werden Pixel und für den Menschen „Bildinhalt“.

Aufgabe 2:c) *MP3-Encoder aus WAV*: **Repräsentation** (Formatwechsel & Kompression). Es bleibt dieselbe Information (Audioinhalt), nur in anderer Datenrepräsentation; formal Daten \rightarrow Daten, aber auf Repräsentationsebene.

Aufgabe 2:d) *Statistiktool erkennt Trend*: **Abstraktion**. Aus Messdaten wird eine inhaltliche Aussage (Trend/Modell) abgeleitet.

Aufgabe 3: Bits, Bytes, Wortbreite.

Aufgabe 3:a) **Gruppenweise I/O:** (i) *Bus-/Cache-Breiten*: Speicher und Caches arbeiten in Blöcken (Cacheline, z. B. 64 B). (ii) *ALU-/Registerbreite*: CPU verarbeitet 32/64 Bit am effizientesten. (iii) *Ausrichtung/ECC*: geringere Overheads, Fehlerkorrektur auf Wortbasis.

Aufgabe 3:b) **Wortbreite:** Anzahl Bits pro Register/ALU-Operation (typ. 32/64 Bit). 64 Bit \Rightarrow größerer Adressraum, größere Ganzzahlen, breitere Pointer; oft mehr Durchsatz.

Aufgabe 3:c) **Byteadressierung vs. 64-Bit-Register:** Kein Widerspruch. Adressen verweisen auf Bytes (kleinste adressierbare Einheit), *laden/speichern* kann aber in 8/16/32/64 Bit-Paketen erfolgen.

Aufgabe 4: Pipeline „Foto mit Smartphone“.

Messung: Photonen \rightarrow Sensor (Bayer-Filter).

Repräsentation: Analog \rightarrow Digital (A/D), Demosaicing, \rightarrow Rohdaten, dann JPEG/HEIC (Kompression, Metadaten/EXIF).

Verarbeitung: Weißabgleich, Rauschminderung, HDR, Schärfung.

Abstraktion: Anzeige fürs Auge; ggf. Objekterkennung (z. B. „Gesicht“, „Text“), also inhaltliche Information aus Pixeln.

Aufgabe 5: Textkodierung — ASCII vs. Unicode.

Aufgabe 5:a) **ASCII-Lücken:** z. B. „ä“, „€“, „Ω“. Viele 8-Bit-Codepages entstanden (ISO-8859-1, Windows-1252 ...); gleiches Byte \Rightarrow anderes Zeichen \Rightarrow *Mojibake*.

Aufgabe 5:b) Codepunkt vs. Kodierung: *Codepunkt* (z.B. U+00E4 „ä“) ist die abstrakte Nummer; *Kodierung* ist die Bytefolge (UTF-8: C3 A4; UTF-16LE: E4 00; UTF-32LE: E4 00 00 00).

Aufgabe 5:c) Vorteil UTF-8: ASCII bleibt 1 Byte; weltweit alle Zeichen möglich (1–4 Byte); robust und verbreitet im Web/Dateien/APIs.

Hausaufgaben

Aufgabe 1: Recherche: Mojibake.

Beispiel 1: „ä“ wurde als UTF-8 C3 A4 gespeichert, aber als ISO-8859-1 gelesen \Rightarrow Anzeige „Ã¤“. **Ursache:** Leser interpretiert C3 als „Ã“ und A4 als „¤“. **Beispiel 2:** „€“ (U+20AC) in Windows-1252 ist 80. Wird als ISO-8859-1 gelesen (wo 0x80 ein Steuerzeichen ist) \Rightarrow Platzhalter „ “ oder gar nichts. **Gegenmittel:** Encoding konsequent auf UTF-8 festlegen und deklarieren (HTTP/HTML/Datei-Header/DB-Kollation).

Aufgabe 2: UTF-8 zum Anfassen.

Zeichen	Codepunkt	UTF-8-Bytes (hex)
A	U+0041	41
ä	U+00E4	C3 A4
€	U+20AC	E2 82 AC

Begründung: ASCII (U+0000–U+007F) \Rightarrow 1 Byte. U+0080–U+07FF \Rightarrow 2 Byte (z. B. „ä“). U+0800–U+FFFF \Rightarrow 3 Byte (z. B. „€“). U+10000– \Rightarrow 4 Byte (z. B. viele Emojis).

Aufgabe 3: Datenmenge einschätzen.

Bild: 800×600 Pixel = 480 000 Pixel.

Aufgabe 3:a) Graustufen 8 bpp \Rightarrow 480 000 Byte \approx 468,75 KiB (da 1 KiB = 1024 B).

Aufgabe 3:b) RGB 24 bpp \Rightarrow $480\,000 \times 3 = 1\,440\,000$ B \approx 1,37 MiB.

Aufgabe 3:c) PNG nutzt verlustfreie Kompression (u. a. Filter + Deflate) und Redundanzen (gleichförmige Flächen, Muster) \Rightarrow Datei oft deutlich kleiner als Rohdaten.

Aufgabe 4: Transferaufgabe: Schritte-App.

Rohdaten (Beispiele): Beschleunigung (x/y/z), Gyro, GPS-Schritte, Zeitstempel. **Modell (Features):** Schritt-Erkennung per Schwellenwerten/Frequenzanalyse; Aggregation zu Tageszähler; Aktivitätslevel (z. B. „niedrig/normal/hoch“) per Grenzwerten. **Risiken:** Falsche Abstraktion (z. B. Fahrt im Bus als „Schritte“), Bias (Handhaltung), Datenschutz (Ortungsdaten). Gegenmaßnahmen: Glättung, Sensorfusion, Kalibrierung, lokale Verarbeitung, klare Datenschutzeinstellungen.

Hinweis: Dies ist ein ausführlicher Lösungsvorschlag; alternative korrekte Begründungen/Lösungswege sind möglich.