Übung 3: Paradigmen

http://people.f4.htw-berlin.de/~hebold/htw/pka/exercises/konzepte-Paradigmen.pdf

Das von-Neumann-Rechnerkonzept (auch von-Neumann-Architektur) zählt zur archetypischen Realisierung des imperativen Programmierparadigmas. Warum?

Imperative Konzept $\widehat{=}$ Befehlsorientiert
Fetch, Execute-Zyklus

Die Turing-Maschine realisiert ebenfalls das imperativen Programmierparadigma. Warum?

Jeder Zustand verknüpft über Befehle, vgl. Überführungsfunktion

Wiese wird vom von-Neumann-Rechner**konzept** aber von der Turing-**Maschine** gesprochen?

Konzept: Abstraktion
Maschine: Konkrete Idee (auch wenn so nicht realisierbar)

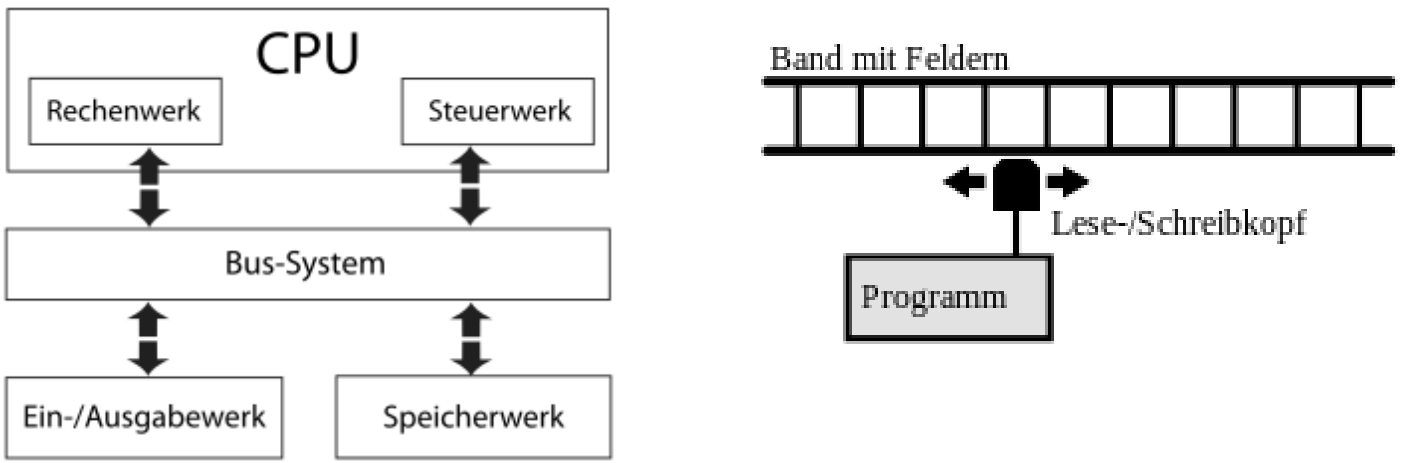

Abbildung : Von-Neumann, Turing-Maschine

Im Zusammenhang mit dem Neumann-Rechnerkonzept ist die Rede vom von-Neumann-Flaschenhals, wenn Nachteile des Konzepts genannte werden.

---

a) Was ist darunter zu verstehen?
Alle Befehle müssen durch den Bus

---

b) Gibt es eine vergleichbare Problematik für die Turing-Maschine?
Schreib/Lesekopf kann nur entweder schreiben oder lesen

Nennen Sie wenigstens einen konzeptionellen Unterschied zwischen von-Neumann-Rechnerkonzept und Turing-Maschine.

Von TM ausgehend:

1. Daten und Programme liegen **nicht** im selben Speicher
2. keine Nummerierung auf dem Band
3. keine Sprungadressen
4. kann nur 1 Feld gehen pro Befehl

Setzt die Turing-Maschine das von-Neumann-Rechnerkonzept um?

Nein, weil

1. Bei TM: Daten $\neq$ Programme
2. TM hat keine Sprungadresse

oder **Ja** mit Einschränkungen (s.o)

Wie könnte das Paradigma der strukturierten Programmierung in das von-Neumann-Rechnerkonzept integriert werden?

Überwachen, bzw. Regeln der Sprunganweisungen.
D.h. Begrenzter Bereich z.B. bei if-Anweisungen

Wieso verletzt das Konzept der lokalen static-Variablen in C das Paradigma der funktionalen Programmierung? (Beispiel!)

Pardigma der f. Programmierung: Funktionsausgabe nur abhängig von Eingabe. D.h. bei gleicher Eingabe gleiche Ausgabe.

```
int f(int i) {
  static int x = 0;
  // Ausfuehrung bei Objekt-Init, nicht bei Methodenaufruf
  x++;
  return x+i;
}
```

Wieso verletzen Pointer in C das Paradigma der funktionalen Programmierung? (Beispiel!)

Pardigma der f. Programmierung: Funktionsausgabe nur abhängig von Eingabe. D.h. bei gleicher Eingabe gleiche Ausgabe.

```
int f(int *i) {
  *i = 1234 // Veraendern der Speicheradresse und somit der Eingabe
   ...
}
```