Ziele des Informatik-Studiums

- **Was sollen Sie im Informatik-Studium lernen?**
  - **Abstraktionsfähigkeit**
    - Computer versteht Sprache der Anwender nicht
    - -> konkretes Problem abstrahieren
    - -> Modell meist **mathematische Konstrukte**
    - meist existieren schon Lösungen für Modellprobleme:ADS
  - **mit neuen Problemen schnell fertig zu werden**

- **Unterschied zu bisherigen Vorlesungen**
  - weniger algorithmisch -> gewöhnungsbedürftig
  - fummle viel mit abstrakten Sachen rum
  - "sage: **WAS**, nicht: **WIE**"
  - Denken auf höherer Ebene im Arbeitsleben unumgänglich

Beispiel für Abstraktion

- **in ADS: Sortieren von Zahlen durch Quicksort**

```
quicksort (int a[], int l, int r) {
int v, i, j, t;
if (r > l) {
  v = a[r]; i = l-1; j = r;
  for (;;) {
    while (a[++i] < v) ;
    while (a[--j] > v) ;
    if (i >= j) break;
    t = a[i]; a[i] = a[j]; a[j] = t;
  }
  t = a[i]; a[i] = a[r]; a[r] = t;
  quicksort(a, l, i-1);
  quicksort(a, i+1, r);
}
}
```

konkret -> abstrakt

- **Quicksort kann auch ganz andere Dinge sortieren:**
  - z.B. Wörter lexikografisch
  - Städte von Ost nach West
- **Quicksort muss lediglich wissen:**
  - Was sortiert werden soll
    - konkret: Zahlen, Wörter, Städte
    - abstrakt: Elemente einer **Menge M**
  - Wie sortiert werden soll
    - konkret: bei Zahlen der Größe nach
bei Wörtern lexikografisch
bei Städten von Ost nach West
    - abstrakt: gemäß einer **Ordnung**

abstraktes Quicksort

- **Ersetze int -> M < -> ≺**

```
quicksort (M a[], int l, int r) {
M v, t; int i, j;
if (r > 1) {
      v = a[r]; i = l-1; j = r;
      for (;;) {
            while (a[++i] ≺ v) ;
            while (a[--j] ≻ v) ;
            if (i >= j) break;
            t = a[i]; a[i] = a[j]; a[j] = t;
      }
  t = a[i]; a[i] = a[r]; a[r] = t;
  quicksort(a, l, i-1);
  quicksort(a, i+1, r);
}
}
```

abstrakt -> konkret

- **Gebe konkrete Instanz von M an:**
  - Zahlen: int
  - Wörtern: Menge der deutschen Wörter
  - Orten: Menge der betrachteten Orte
- **Gebe konkrete Instanz der Ordnung an:**
  - Zahlen: <
  - Wörtern: lexikografische Ordnung
  - Orten: liegt östlich von
- **erhalte durch Instanzierung aus einem abstrakten Algorithmus viele konkrete Algorithmen**
- **siehe OOS**

Vorteile & Probleme der Abstraktion

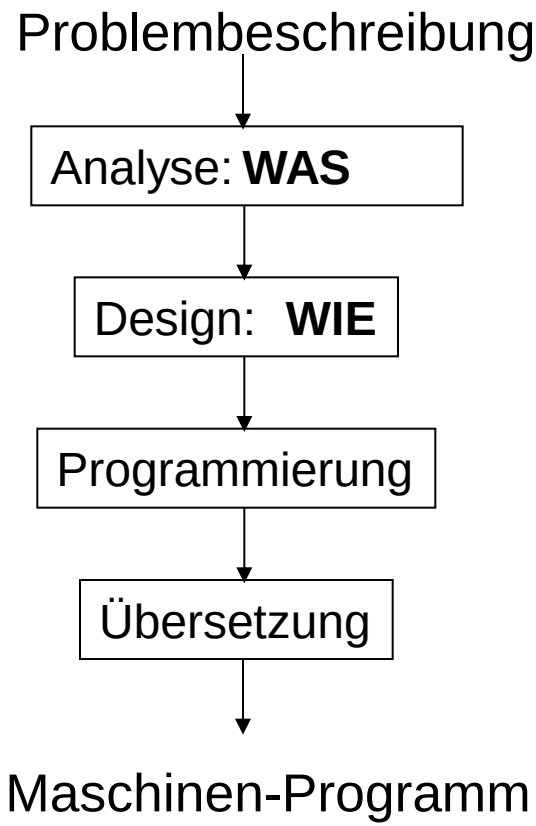
- **Vorteile**
  - Beschäftigung mit dem Wesentlichen
  - abstrakte Lösung für viele konkrete verwendbar
  - durch Abstraktion vom Anwendungsbereich Erleichterung der Kommunikation mit Entwicklern
- **Probleme**
  - weg vom Konkreten
  - schwierig vorzustellen
  - gewöhnungsbedürftig
  - Denken auf höherer Ebene

  -> Fragen bei Unklarheiten

  Übungen machen

Abstraktion in Softwareentwicklung

allgemein:



Bsp:

gesucht: Programm, das Bewohner einer Straße nach Hausnr. sortiert liefert

Sortierung einer Adressliste mit gleicher Straße nach Hausnr. **abstrakter**

Verwendung von Quicksort für nat. Zahlen zur Sortierung der Adressliste nach Hausnr.

Implementierung von Quicksort für nat. Zahlen zur Sortierung einer lin. Liste von Adressen nach dem Feld Hausnr. **viel konkreter**

automatisch

1. Praktikumsvers.: Auftrag vom König

Zur Förderung der einheimischen Autoindustrie entschließt sich der König Einweg des Landes Einbahn in seinem ganzen Reich nur noch Einbahnstraßen zuzulassen, die so angeordnet sind, dass man nie wieder per Auto zum Ausgangspunkt zurück kommt.

Dadurch werden die Bürger des Landes Einbahn jedes Mal, wenn Sie von einem Ort zu einem anderen kommen möchten, vor die Entscheidung gestellt, ob sie den Zielort vom Ausgangsort aus ohne Ordnungswidrigkeit zu begehen, erreichen können, indem sie ein Auto benutzen, d.h. ob es sich lohnt ein neues Auto zu kaufen.

Um den potenziellen Autokäufern die Entscheidung zu erleichtern, gibt König Einweg der einzigen Softwarefirma FHAC TIWS GmvH des Landes den Auftrag ein Programm zu schreiben, das bei Eingabe des Einbahnstraßennetzes des Landes alle möglichen mit dem Auto zu erreichenden Verbindungen zwischen jeweils zwei beliebigen Orten des Landes liefert.

Vorgehen

- **Abstraktion des Problems finden, dass**
  - Problem mit bisher Erlerntem (Semester 1 & 2) lösbar
  - Problemlöser nicht unbedingt konkretes Problem kennen muss
- **Auswahl eines Problemlösenden Algorithmus**
- **Implementierung des Algorithmus in C++**
- **Deadline: jeweils Ende Praktikumsversuch!!! ->**
  - möglichst vorher Gedanken machen
  - bei Unklarheiten zur Abstraktion oder Algorithmus **fragen**
  - Diskussion:
    - jeweils nach Vorlesung

allgemeine Ziele der Vorlesung

- **Abstraktionsfähigkeit erlernen durch**
  - Umgang mit grafischen Modellen und mathem. Formalismen
  - Transformation von konkreten Problemen in abstrakte Modelle
- **Kommunikationsfähigkeit verbessern durch**
  - Vortragen von Übungsaufgaben
  - Vorstellen von Praktikumslösungen
  - Nachfragen bei Verständnisproblemen
- **Horizont erweitern durch**
  - Kennenlernen eines fremden Sprachparadigmas
  - Lösungsvereinfachung durch Einsatz von Nichtdeterminismus
  - Erlernen grundlegender Aussagen der Informatik

spezielle Ziele der Vorlesung

- **Erlernen der Grundlagen von WS & neues Sprachparadigma:**
  - WS & Logikprogrammierung an Programmiersprache Prolog
  - Programmieren durch Spezifizieren und Nichtdeterminismus
- **Umgang mit Nichtdeterminismus kennen lernen**
  - Auswahl unter mehreren Folgekonfigurationen offen lassen
    Bsp.: Suche nach Weg, Schachprogramm
- **Grundlagen und Implementierung von Programmiersprachen:**
  - formale Sprachen & Automatentheorie
  - Umwandlung: Nichtdeterminismus -> Determinismus
- **Lösbarkeit von Problemen: Gefühl & Beweistechniken**
- **Komplexität von lösbaren Problemen: gutartig oder bösartig**
- **Behandlung bösartiger Probleme: Approximationsalgorithmen**