5. TUTORIUM

Einführung in die Strukturierte Programmierung

Kurzer Ausflug nach SVG

- SVG = Scaleable Vector Graphic

- Auf XML basierendes Vektor-Grafik Format
- Unsere Befehle sind relativ einfach in SVG umzusetzen

- 2 Wichtige Punkte:
- SVG Dateien sind auch nur XML Dateien und sollten daher valid („syntaktisch korrekt“) sein. Daher: Ausgegebene Dateien per Validator (siehe URD) prüfen
- Reihenfolge der Überlappungen gibt sich aus Reihenfolge in der Datei. Daher:

Die Reihenfolge der eingegeben Daten muss mitgespeichert werden (siehe Datenstrukturen) !

Einlesen der Kommandos

- Gedanke im Hinterkopf: wie einfach kann ich ein neues Kommando hinzufügen …

- Zuständige Komponenten:
- Endlosschleife (wartet auf quit)
- Anzeigen des Prompts (Achtung, kommt aus Config-File)
- Funktion zum Einlesen der Zeile (warum kein scanf ??)
- Funktion zum klassifizieren des Inputs
- Aufruf der entsprechenden Funktion

- Warum der ganze Aufwand ?
- ... Erweiterbarkeit ...

Binär vs. Textdatei

Binär

- Lesbar über Hexeditor
- Einlesen / Schreiben
- über rb bzw. wb
- Achtung: Endianness bei nicht x86 Architektur

ASCII

- Lesbar über einen normalen Editor
- Einlesen / Schreiben über r bzw w
- Keine Endianness

Structs [1]

- Mehrere einzelne Variablen werden zu einem neuen Typ zusammengefasst, und können dann als ganzes verwendet werden [Prof. Kappe].

- Was bringt uns das ?
  - Für eine Gruppe von Variablen die logisch zusammen gehört brauchen wir nur noch ein struct.
  - Wir müssen nicht mehr jede einzelne Variable in eine Funktion übergeben, sondern nur ein struct.
  - Wir können structs ganz einfach in eine Binärdatei schreiben (Achtung Reihenfolge) und lesen.

Speichern der Objekte[1]

- Problem: Wir haben verschiedene Pointer die in der Reihenfolge der Eingabe gespeichert, gelöscht und (optionalerweise) geändert werden müssen.

- Dadurch ergeben sich einige Fragen:
  - Wie garantiere ich die Reihenfolge?
  - Was muss ich mir beim Löschen alles ansehen, bzw. umschreiben?
  - Wie übergebe ich meine Daten in die entsprechenden Funktionen?
  - Könnte das nicht schöner / einfacher gehen?

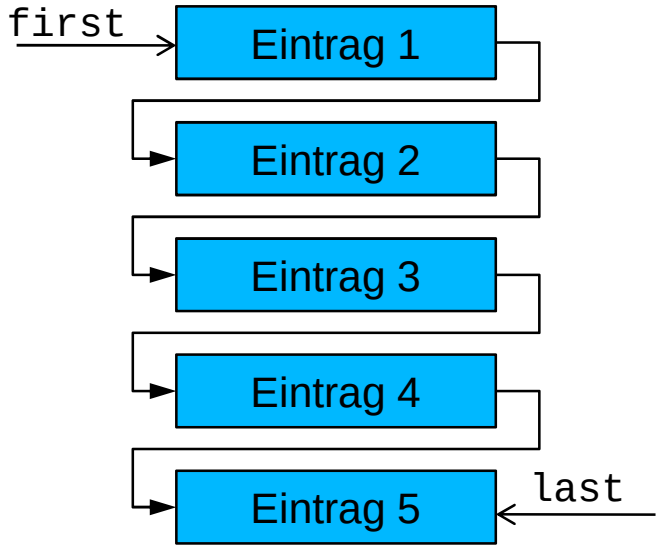
Speichern der Objekte[2]

- Zum Speicher gibt es jetzt 2 Ansätze:
  - Eigenes Array für jeden Elementtyp
    - einfach aber mit Nachteilen verbunden
  - Speichern eines “generelleren” Pointers und der Art des Pointers, casten in den entsprechenden Typ

Speichern der Objekte[4]

- Als Verfeinerung für Ansatz 2 kann nun statt einem dynamischen Array eine eigene Datenstruktur verwendet werden:

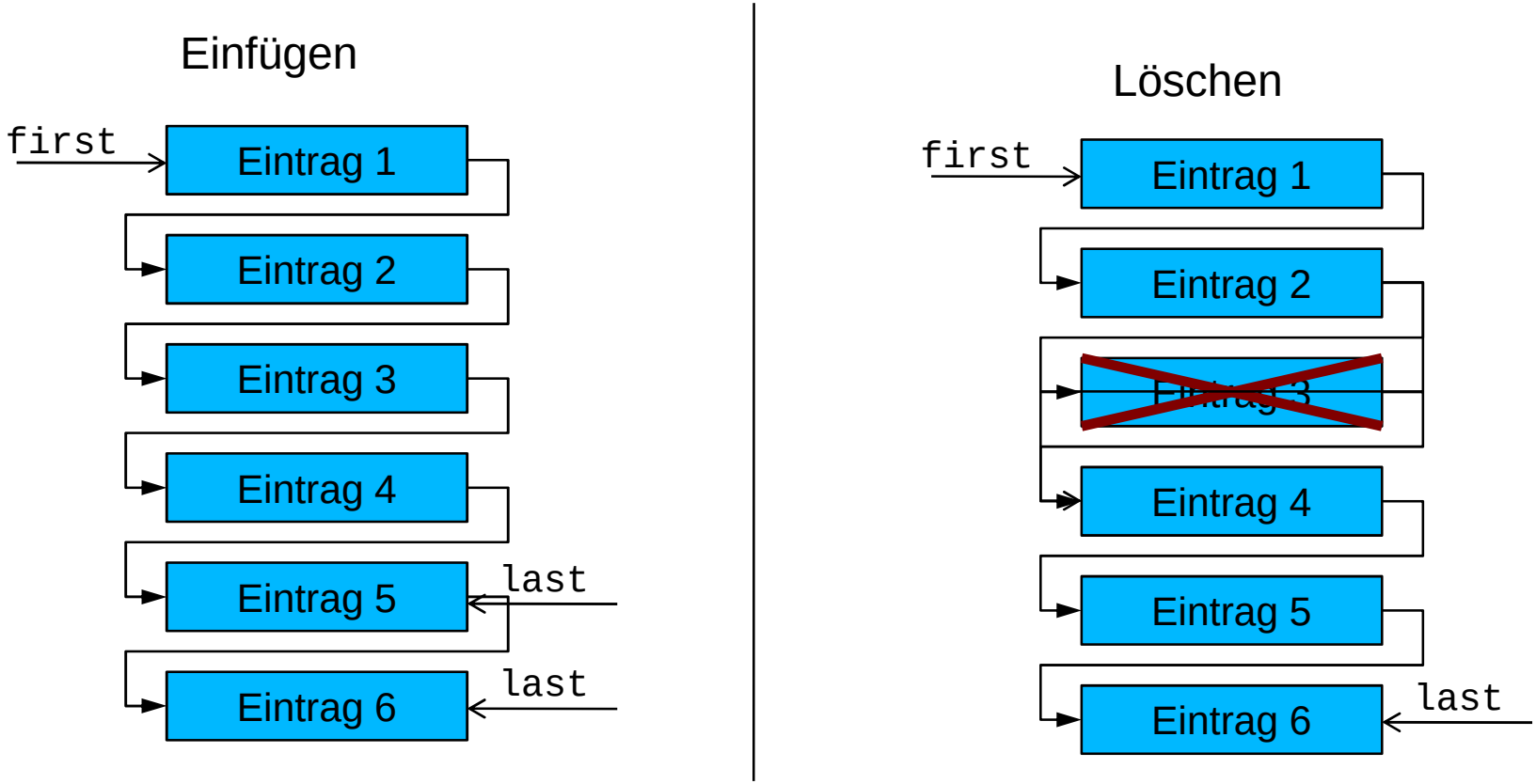
- Einfach verkette Liste:



Zu jedem Element wird ein Pointer auf das nächste Element gehalten. Zusätzlich brauchen wir noch Pointer `first` (und ev. `last`).

Speichern der Objekte[4]

- Zugriff auf die Liste:



Vielen Dank