FH Salzburg

Grundlagen der Informatik

Vorlesung 4: Datenstrukturen

Technik
Gesundheit
Medien

Datenstrukturen

- Beschreiben die Art der Organisation der Daten
  - im Speicher des Rechners während der Verarbeitung
  - auf Speichermedien

- Die Daten werden dabei in Elemente aufgeteilt.

- Die einzelnen Elemente werden in regelmäßige Relationen gesetzt,
  - z.B. eine Vorgänger- und Nachfolger-Relation.

Datenstrukturen

- Für viele Anwendungen ist die Wahl einer geeigneten Datenstruktur eine wesentliche Entscheidung

- Wie organisiert man die Daten im Speicher, damit sie günstig verarbeitet werden können?
  - Programmcode zur Verarbeitung ist einfach und kurz
  - wenig Anweisungen (schneller, weniger Prozessorbelastung)
  - inhaltlich verwandte Elemente stehen nah beieinander (schneller)
  - Geringer Speicherbedarf

Arrays (Felder)

- Datenstruktur zur Aneinanderreihung von Variablen gleichen Typs
- Können ein- oder mehrdimensional sein

Datentyp der Variable (hier Double)

Einzelne Elemente liegen im Speicher hintereinander: [][][][][][]

```
double tageseinnahmen[6];
```

Name des Arrays

Größe/Länge des Arrays (bei Deklaration!)

Arrays (Felder)

- Zugriff auf einzelne Elemente im Array über den Index

Index des Elements, auf das zugegriffen werden soll (Index 0 ist das erste Element)

```
tageseinnahmen[0] = 1730;
```

Wert, der dem ersten Element zugewiesen wird

Speicher nach der Zuweisung:
[1730] [ ] [ ] [ ] [ ] [ ]

Arrays (Felder)

- Warum Index 0 und nicht 1?

```
array[2] = …
```

Wert: [ ] [ ] [ ] [ ] [ ]
Index: [0] [1] [2] [3] [4]

Index ist eigentlich Offset (= Distanz zum ersten Element)

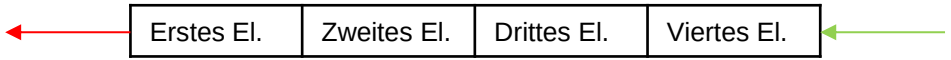
Stack

- Ein Stack ist ein LIFO (Last In, First Out) Speicher („Stapel“)
- Das letzte Element, das hinzugefügt wurde, ist das erste, das wieder entnommen wird → Reihenfolge bleibt erhalten
- Wichtigste Funktionen:
  - push: Fügt dem Stack ein Element hinzu
  - pop: Entfernt das zuletzt hinzugefügte Element vom Stack und gibt es zurück

Queue

- Eine Queue ist ein FIFO (First In, First Out) Speicher („Warteschlange“)

- Das erste Element, das hinzugefügt wurde, ist das erste, das wieder entnommen wird → Reihenfolge bleibt erhalten

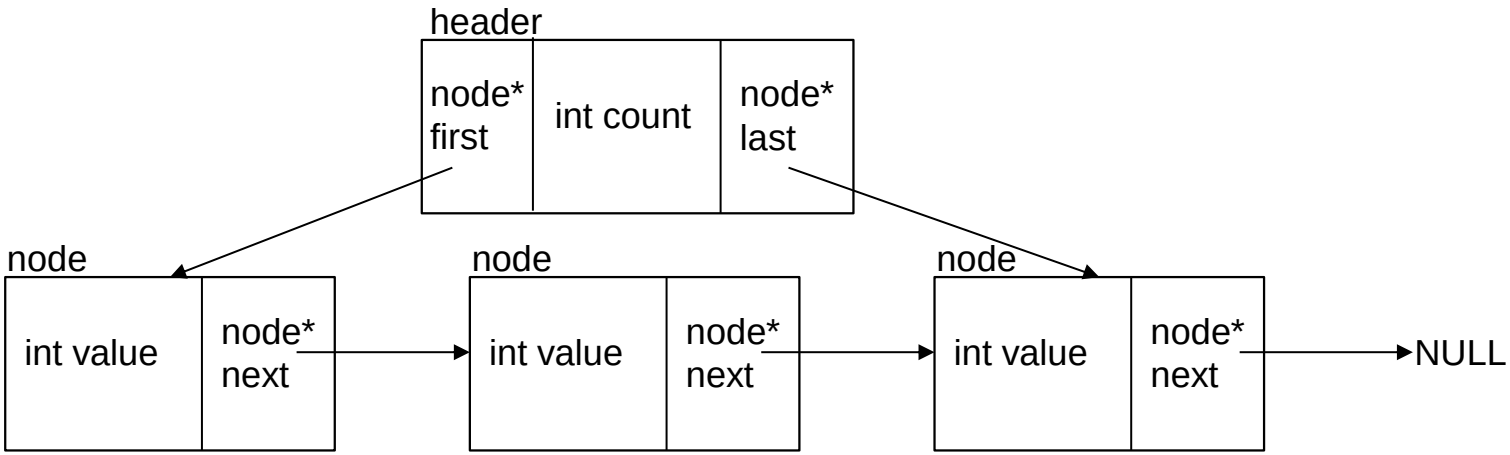


- Wichtigste Funktionen:
  - enqueue: Fügt der Queue ein Element hinzu
  - dequeue: Entfernt das als erstes hinzugefügte Element von der Queue und gibt es zurück

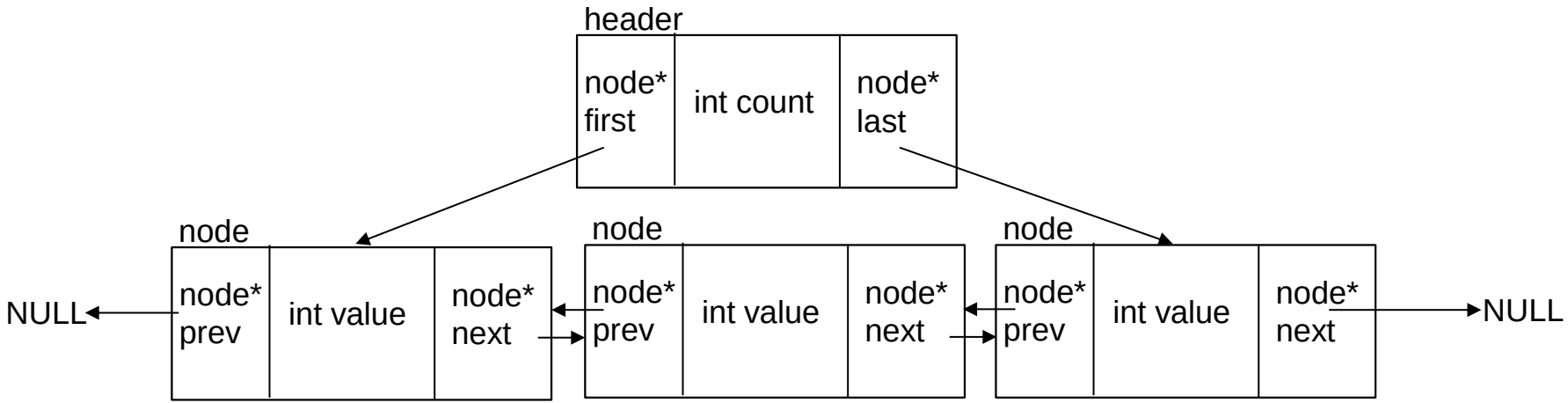
Listen

- Aneinanderreihung von Daten
- Dynamische Länge (keine Initialgröße nötig)
- Besteht aus Einzelelementen (Nodes)
- Reihenfolge bleibt erhalten

Listen (einfach verkettet)



Listen (doppelt verkettet)



Binäre Bäume

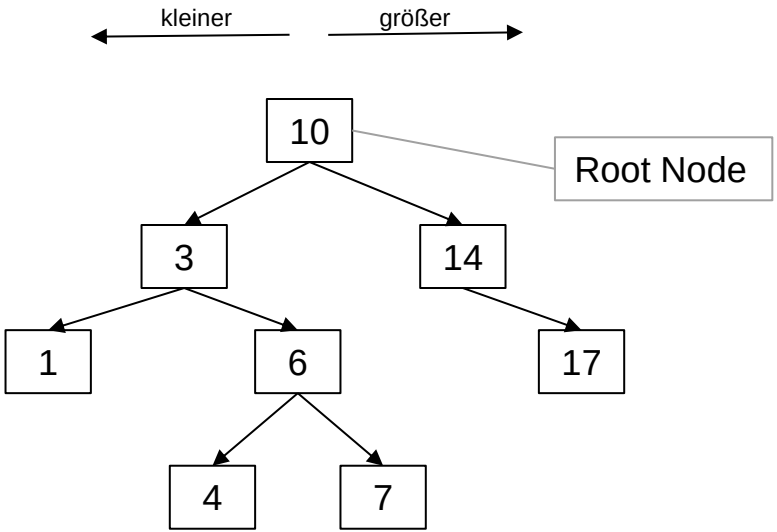
- Datenstruktur zum sortierten Speichern → Reihenfolge bleibt nicht erhalten

- Besteht aus Nodes, die jeweils kein, einen oder zwei Kind-Nodes haben können

Binäre Bäume - Aufbau

- Erstes Element im Baum wird der Root Node
- Neue Elemente werden immer ausgehen vom Root Node hinzugefügt
- Ist das neue Element kleiner oder gleich groß, kommt es nach links im Baum, ist es größer nach rechts

Binäre Bäume - Aufbau



Reihenfolge der Elemente:
10, 14, 3, 17, 6, 1, 7, 4

Binäre Bäume - Traversieren

- In-Order (aufsteigend sortiert):

  links – ausgeben – rechts

- Pre-Order (topologisch sortiert):

  ausgeben – links – rechts

- Post-Order (Eltern-Nodes immer nach Kind-Nodes):

  links – rechts – ausgeben

Andere Datenstrukturen

- Graph
- Hashtable
- ...