

Listen-Datentypen

Reiner Hüchting

2. März 2023

Themenüberblick

Listen-Datentypen

Listen in Java

Dynamische Arrays

Verkettete Listen

Themenüberblick

Listen-Datentypen

Listen in Java

Dynamische Arrays

Verkettete Listen

Listen-Datentypen

Gemeinsame Eigenschaften von Listen

- ▶ Elemente gleichen Typs
- ▶ Zugriff auf einzelne Elemente möglich
- ▶ Durchlaufen möglich
- ▶ Operationen: Hinzufügen und Löschen von Elementen

Abstrakter Datentyp „Liste“

- ▶ Wird durch abstrakte Eigenschaften wie oben definiert.
- ▶ Wird *nicht* durch konkrete Implementierungsdetails definiert.
 - ▶ *nicht* durch die konkrete Anordnung im Speicher
 - ▶ *nicht* durch Performance-Eigenschaften

Listen-Datentypen

Abstrakter Datentyp „Liste“

- ▶ Elemente gleichen Typs
- ▶ Zugriff/Durchlaufen möglich
- ▶ Hinzufügen und Löschen von Elementen

Konkrete Listen-Datentypen

- ▶ Arrays
- ▶ dynamische Arrays
- ▶ verkettete Listen

Listen-Datentypen

Arrays

- ▶ zusammenhängender Bereich im Speicher
- ▶ Zugriff und Durchlaufen mittels *Pointerarithmetik*

Vorteile

- ▶ Zugriff/Durchlauf sehr schnell
- ▶ *wahlfreier* Zugriff

Nachteile

- ▶ keine Größenänderung möglich
- ▶ Einfügen ggf. aufwendig oder unmöglich
- ▶ zusammenhängender Platz nötig

Listen-Datentypen

Dynamische Arrays

- ▶ verwendet intern Arrays
- ▶ *wahlfreier* Zugriff
- ▶ fügt die Möglichkeit zur Größenänderung hinzu

Vorteile

- ▶ Zugriff/Durchlauf sehr schnell
- ▶ Größenänderung möglich

Nachteile

- ▶ Einfügen ggf. aufwendig
- ▶ zusammenhängender Platz nötig

Listen-Datentypen

Verkettete Listen

- ▶ Elemente bestehen aus zwei Teilen:
 - ▶ Daten
 - ▶ Pointer/Referenz auf benachbarte Elemente

Vorteile

- ▶ Größenänderung möglich
- ▶ Einfügen bei bekannter Position schnell
- ▶ kein zusammenhängender Speicherbereich
 - ▶ dadurch bessere Speicher-Ausnutzung

Nachteile

- ▶ Durchlauf und Zugriff aufwendig
- ▶ kein *wahlfreier* Zugriff

Themenüberblick

Listen-Datentypen

Listen in Java

Dynamische Arrays

Verkettete Listen

Listen in Java

Wichtige Listen-Datentypen in Java

- ▶ klassische Arrays
- ▶ `ArrayList`
- ▶ `LinkedList`

Interfaces und abstrakte Typen

- ▶ `List`
 - ▶ *Interface*, definiert die Schnittstelle von Listen
- ▶ `AbstractList`
 - ▶ gemeinsame Basisklasse verschiedener Listen-Datentypen
 - ▶ stellt gemeinsame Funktionalität bereit
- ▶ `AbstractSequentialList`
 - ▶ Basisfunktionalität für Listen ohne wahlfreien Zugriff

Themenüberblick

Listen-Datentypen

Listen in Java

Dynamische Arrays

Verkettete Listen

Dynamische Arrays

Attribute eines dynamischen Arrays

- ▶ Array für die Daten
- ▶ tatsächliche und maximale Länge

Zentrale Operation: `realloc`

- ▶ neues Array für die Daten mit neuer Länge erzeugen
- ▶ alle Elemente an die neue Stelle kopieren
- ▶ maximale Länge aktualisieren

Anhängen von Elementen

- ▶ Element an erste freie Stelle schreiben und Größe aktualisieren
- ▶ ggf. vorher `realloc` durchführen

Dynamische Arrays

Anhängen von Elementen

- ▶ falls Array voll: `realloc` durchführen
- ▶ neues Element an erste freie Stelle schreiben
- ▶ Größe aktualisieren

Wie viel Speicher sollte bei `realloc` reserviert werden?

- ▶ Antwort: Z.B. immer verdoppeln.
- ▶ Ziel: Der Speicher muss exponentiell wachsen, damit `realloc` nicht zu oft notwendig ist.

Themenüberblick

Listen-Datentypen

Listen in Java

Dynamische Arrays

Verkettete Listen

Verkettete Listen

Attribute eines Listenelements

- ▶ Datensatz
- ▶ Zeiger/Referenzen auf die Nachbarelemente

Zwei typische Varianten

- ▶ einfach verkettete Liste
- ▶ doppelt verkettete Liste

Anhängen von Elementen

- ▶ Ende der Liste suchen
- ▶ neues Element anhängen

Verkettete Listen

Anhängen von Elementen

- ▶ Ende der Liste suchen
- ▶ neues Element anhängen

Markierung des Listen-Endes: Sentinel-Prinzip

- ▶ Verwendung eines *Dummy-Elements*
- ▶ wird nicht für Daten verwendet
- ▶ markiert das Ende der Liste

Vorteil des Dummy-Elements

- ▶ keine Sonderbehandlung der leeren Liste notwendig

Verkettete Listen

Implementierung der ganzen Liste

- ▶ Ein Listenelement ist gleichzeitig auch eine Liste.
- ▶ **Listen haben eine rekursive Struktur!**
- ▶ Container-Klasse für Liste ist dennoch oft nützlich

Attribute einer verketteten Liste

- ▶ Zeiger/Referenz auf Anfang der Liste oder Dummy

Vorteil der Container-Klasse

- ▶ kann Dummy vor Benutzer verstecken
- ▶ manche Operationen einfacher umsetzbar