Prinzip und Verwendung der Datenstrukturen Stapel, Schlange und Dynamische Reihung[1]

Datenstrukturen allgemein

In einer Datenstruktur werden Daten auf eine bestimme Art und Weise verwaltet. Sie beinhaltet eine Menge von Werten und stellt Operationen zur Verfügung, mit denen auf die Werte zugegriffen werden kann. Dabei wird für eine Datenstruktur konkret nur die Wirkung der zugehörigen Operationen festgelegt. So wird für die Datenstruktur Stapel beispielsweise vorgegeben, dass die Operation `pop()` das oberste Element eines Stapels entnimmt und den Inhalt dieses Elementes zurückgibt. Wie die Operationen einer Datenstruktur intern implementiert sind, ist bei der Verwendung einer Datenstruktur dagegen irrelevant. Für die Verwendung von Datenstrukturen sind zwei Prinzipien wichtig:

Prinzip der Datenkapselung: auf die Werte und Operationen einer Datenstruktur kann nur über ihre definierte Schnittstelle zugegriffen werden.

Geheimnisprinzip: eine konkrete Implementierung der Operationen einer Datenstruktur ist verborgen und für die Verwendung der Datenstruktur irrelevant.

Bei der Lösung einer algorithmischen Problemstellung spielt die Wahl der verwendeten Datenstruktur eine wichtige Rolle. Sie legt das weitere algorithmische Vorgehen fest und beeinflusst das Laufzeitverhalten und den Speicherbedarf eines Verfahrens.

Im Folgenden werden die für das niedersächsische Abitur vorgegebenen Datenstrukturen Stapel, Schlange und Dynamische Reihung erläutert. Für sie ist zu beachten: je nach Datenstruktur müssen bei der Anwendung der zur Verfügung gestellten Operationen mögliche Laufzeitfehler bei der Verwendung (wie beispielsweise der Zugriff auf eine nicht existierende Position einer dynamischen Reihung oder das Entnehmen eines Elementes aus einem leeren Stapel) durch entsprechende Abfragen im Programm explizit ausgeschlossen werden.

Die Datenstruktur Stapel

Der Begriff „Stapel“ veranschaulicht (vgl. Abbildung 1), nach welchem Prinzip Daten in der Datenstruktur Stapel verwaltet werden. Ein neues Element wird immer oben auf den Stapel abgelegt. Möchte man ein Element entnehmen, so ist dies immer das oberste Element des Stapels, also das zuletzt abgelegte Element. Dies entspricht dem sogenannten LIFO-Prinzip – ein Begriff, der von der englischen Bezeichnung „last in – first out“ stammt.

Abbildung 1: ein Bücherstapel

[1] Die hier beschriebenen Datenstrukturen Stapel, Schlange und dynamische Reihung orientieren sich an den Vorgaben des Kerncurriculums für das Gymnasium – gymnasiale Oberstufe, die Gesamtschule – gymnasiale Oberstufe, das Kolleg für das Fach Informatik, Niedersächsisches Kultusministerium, 2017 sowie den Ergänzenden Hinweisen zum Kerncurriculum Informatik für die gymnasiale Oberstufe am Gymnasium und an der Gesamtschule sowie für das Kolleg, Niedersächsisches Kultusministerium, Stand Juni 2021.

Die Datenstruktur Stapel stellt zur Verwaltung ihrer Daten genau die Operationen aus Tabelle 1 bereit. Dabei kann der Inhaltstyp je nach Fragestellung variieren: Stapel vom Typ Zeichenkette oder vom Typ Ganzzahl sind genauso denkbar wie Stapel vom Typ einer vorher definierten Klasse. Anders als im Bücherstapel in Abbildung 1 kann bei der Datenstruktur Stapel dabei nicht auf Elemente innerhalb des Stapels zugegriffen werden. Auch stellt die Datenstruktur beispielsweise keine Operation zur Verfügung, um die Anzahl der Elemente eines Stapels zu bestimmen.

`Stack()`	Ein leerer Stapel wird angelegt
`isEmpty(): Wahrheitswert`	Wenn der Stapel kein Element enthält, wird der Wert `wahr` zurückgegeben, sonst der Wert `falsch`
`top(): Inhaltstyp`	Der Inhalt des obersten Elements des Stapels wird zurückgegeben, das Element aber nicht entnommen.
`push(inhalt: Inhaltstyp)`	Ein neues Element mit dem übergebenen Inhalt wird oben auf den Stapel gelegt.
`pop(): Inhaltstyp`	Das oberste Element des Stapels wird entnommen. Der Inhalt dieses Elements wird zurückgegeben.

Tabelle 1

Beispiel: die Verwaltung von neu eingetroffenen Büchern einer Bücherei

Eine Bücherei verwaltet die neu mit der Post eingetroffenen Bücher in der Datenstruktur Stapel namens `B_Stapel` vom Typ Zeichenkette. Darin wird zunächst jeweils nur der Titel eines Buches erfasst. Dieser wird in den Stapel eingefügt und das tatsächliche Buch auf einen Bücherstapel gelegt. Am Ende eines Tages können die Bücher dann nacheinander vom Stapel (sowohl die zugehörigen Daten der Datenstruktur als auch die real existierenden Bücher vom Bücherstapel) entnommen, vollständig katalogisiert und im Sortiment aufgenommen werden. In Abbildung 2 wird eine beispielhafte Belegung der zwei Stapel nach Hinzufügen von drei Büchern dargestellt.

`"Computer-Netzwerke"`
`"Einführung in die Kryptographie"`
`"Ideen der Informatik"`

Visualisierung der Belegung des Stapels `B_Stapel`

tatsächlicher Bücherstapel

Abbildung 2

Aufgaben

1) Zeichnen Sie die Belegung des Stapels `B_Stapel` nach zweimaligem Aufruf der Methode `B_Stapel.pop()`.

2) In der Bücherei wird angefragt, ob das Buch mit dem Titel „Ideen der Informatik“ vorrätig ist. Beschreiben Sie Gemeinsamkeiten und Unterschiede zwischen der algorithmischen Suche nach dem Buch in der Datenstruktur `B_Stapel` und der menschlichen Suche im tatsächlichen Bücherstapel.

3) Eine Methode `zaehlen` soll für den als globale Variable vorliegenden Stapel `B_Stapel` bestimmen, wie viele Elemente darin enthalten sind. Dabei soll sich der Stapel `B_Stapel` nach Ausführung der Methode `zaehlen` im gleichen Zustand befinden wie vor der Ausführung der Methode. Implementieren Sie die Methode `zaehlen` in einer im Unterricht verwendeten Programmiersprache.

4) Vergleichen Sie die Verwaltung der Titel der ankommenden Bücher in einem Stapel mit der in einer statischen Reihung.

Die Datenstruktur Schlange

Auch der Begriff „Schlange“ veranschaulicht, nach welchem Prinzip Daten in der Datenstruktur Schlange verwaltet werden. In Analogie zu einer Warteschlange wird ein neues Element immer am Ende der Schlange angehängt. Möchte man ein Element entnehmen, so ist dies immer das vorderste Element der Schlange, also das zuerst eingefügte Element. Dies entspricht dem sogenannten FIFO-Prinzip – ein Begriff, der von der englischen Bezeichnung „first in – first out“ stammt.

Die Datenstruktur Schlange stellt zur Verwaltung ihrer Daten genau die Operationen aus Tabelle 2 bereit. Dabei kann der Inhaltstyp wie beim Stapel je nach Fragestellung variieren: eine Schlange vom Typ Zeichenkette oder vom Typ Ganzzahl ist genauso denkbar wie eine Schlange vom Typ einer vorher definierten Klasse.

`Schlange()`	Eine leere Schlange wird angelegt.
`isEmpty(): Wahrheitswert`	Wenn die Schlange kein Element enthält, wird der Wert `wahr` zurückgegeben, sonst der Wert `falsch`
`head(): Inhaltstyp`	Der Inhalt des vordersten Elements der Schlange wird zurückgegeben, das Element aber nicht entnommen.
`enqueue(inhalt: Inhaltstyp)`	Ein neues Element mit dem angegebenen Inhalt wird am Ende an die Schlange angehängt.
`dequeue(): Inhaltstyp`	Das vorderste Element wird entnommen. Der Inhalt dieses Elements wird zurückgegeben.

Tabelle 2

Beispielhafte Darstellung der Belegung eines Stapels `stapel1` vom Typ Ganzzahl und einer Schlange `schlange1` vom Typ Ganzzahl:

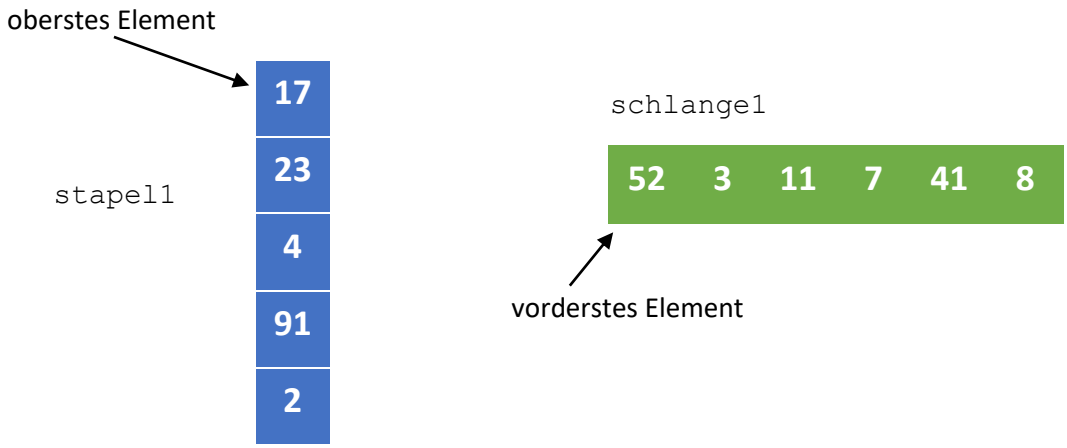

Abbildung 3

Aufgaben

5) In Abbildung 3 sind beispielhafte Belegungen eines Stapels `stapel1` sowie einer Schlange `schlange1` jeweils vom Typ Ganzzahl dargestellt. Zeichnen Sie den Zustand von `stapel1` und `schlange1` nach den folgenden Methodenaufrufen und gehen Sie bei jeder Teilaufgabe wieder vom Anfangszustand wie in Abbildung 3 aus:

```
a) stapel1.push(schlange1.dequeue());
b) stapel1.pop();
   schlange1.enqueue(stapel1.pop());
c) schlange1.enqueue(schlange1.dequeue());
```

6) Implementieren Sie in einer im Unterricht verwendeten Programmiersprache eine Operation `hinzufuegen(s: Schlange vom Typ Ganzzahl, z: Ganzzahl): Wahrheitswert`, die nur dann die Zahl `z` am Ende an die Schlange `s` anhängt, wenn sie noch nicht in der Schlange `s` enthalten ist. In dem Fall gibt sie `true` zurück, andernfalls `false`.

7) Diskutieren Sie die Bedeutung und Notwendigkeit der Operation `isEmpty()` für die beiden Datenstrukturen Stapel und Schlange.

8) Entscheiden Sie für die beiden folgenden Anwendungen jeweils, welche Datenstruktur (Stapel oder Schlange) sich besser zur Verwaltung der Daten eignet und begründen Sie Ihre Entscheidung:
   a) Die Druckaufträge für einen Netzwerkdrucker sollen verwaltet werden.
   b) Ein einfaches Grafikprogramm soll um eine „Undo“-Methode erweitert werden, mit der alle ausgeführten Aktionen schrittweise rückgängig gemacht werden können.
   c) Ein Pizzalieferdienst verwaltet alle eingehenden Bestellungen.

Die Datenstruktur Dynamische Reihung

Anders als in einer statischen Reihung können in einer dynamischen Reihung theoretisch beliebig viele Elemente verwaltet werden. Außerdem bietet sie zusätzlich zum direkten Zugriff auf alle enthaltenen Elemente verschiedene Operationen zum Einfügen oder Löschen von Elementen an, so dass alle übrigen Elemente geeignet verschoben werden. Die Datenstruktur Dynamische Reihung stellt zur Verwaltung ihrer Daten die Operationen aus Tabelle 3 bereit[2]. Dabei kann der Inhaltstyp auch hier je nach Fragestellung variieren.

Die Nummerierung der Elemente einer dynamischen Reihung beginnt hier[2] mit dem Index 0.

`DynArray()`	Eine leere dynamische Reihung wird angelegt.
`isEmpty(): Wahrheitswert`	Wenn die dynamische Reihung kein Element enthält, wird der Wert `wahr` zurückgegeben, sonst der Wert `falsch`.
`getItem(index:Ganzzahl):Inhaltstyp`	Der Inhalt des Elements an der Position `index` wird zurückgegeben.
`append(inhalt: Inhaltstyp)`	Ein neues Element mit dem übergebenen Inhalt wird am Ende der dynamischen Reihung angefügt.
`insertAt(index: Ganzzahl, inhalt: Inhaltstyp)`	Ein neues Element mit dem übergebenen Inhalt wird an der Position `index` in die dynamische Reihung eingefügt. Das Element, das sich vorher an dieser Position befunden hat, und alle nachfolgenden werden nach hinten verschoben. Entspricht der Wert von `index` der Länge der dynamischen Reihung, so wird ein neues Element am Ende der dynamischen Reihung angefügt.
`setItem(index: Ganzzahl, inhalt: Inhaltstyp)`	Der Inhalt des Elementes an der Position `index` wird durch den übergebenen Inhalt ersetzt.
`delete(index: Ganzzahl)`	Das Element an der Position `index` wird entfernt. Alle folgenden Elemente werden um eine Position nach vorne geschoben.
`getLength(): Ganzzahl`	Die Anzahl der Elemente der dynamischen Reihung wird zurückgegeben.

Tabelle 3

[2] gemäß den Ergänzenden Hinweisen zum Kerncurriculum Informatik für die gymnasiale Oberstufe am Gymnasium und an der Gesamtschule sowie für das Kolleg, Niedersächsisches Kultusministerium, Stand Juni 2021.

Aufgabe Wetterstation

Die Daten einer Wetterstation werden in einer global definierten dynamischen Reihung `wetterdaten` vom Typ `Messwert` (vgl. Klassenkarte in Abbildung 4) verwaltet.

Messwert
- temperatur: Fließkommazahl - luftfeuchtigkeit: Fließkommazahl - zeitstempel: Zeichenkette
c Messwert(t: Fließkommazahl, l: Fließkommazahl, z: Zeichenkette) + getTemperatur(): Fließkommazahl + getLuftfeuchte(): Fließkommazahl + getZeitstempel(): Zeichenkette

Abbildung 4

a) Begründen Sie, warum eine dynamische Reihung im Vergleich zu einer statischen Reihung, einem Stapel oder einer Schlange für die Verwaltung der Wetterdaten besonders geeignet ist.

b) Eine Operation `bestimmeMax` soll den Zeitstempel eines Messwertes in der dynamischen Reihung `wetterdaten` mit der höchsten vorkommenden Temperatur bestimmen. Implementieren Sie die Operation `bestimmeMax` in einer im Unterricht verwendeten Programmiersprache.

c) Aufgrund von Übertragungsfehlern sind in der dynamischen Reihung `wetterdaten` auch fehlerhafte Messwerte enthalten. Eine Operation `bereinigeLuftfeuchte` soll alle Objekte vom Typ `Messwert` aus der Reihung `wetterdaten` entfernen, deren Wert der Luftfeuchtigkeit größer als 100 ist. Implementieren Sie die Operation `bereinigeLuftfeuchte` in einer im Unterricht verwendeten Programmiersprache.