Imperative Programmierung

Übung 12: Warteschlangen und Stapel

Justin Kreikemeyer
Informatik, Uni Rostock

Plan

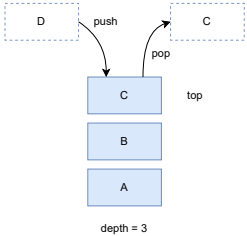
- Datenstruktur Stack
- Datenstruktur Queue
- Übungsaufgaben

Nächste Woche: Binärer Suchbaum und Heap.
Letzte Woche: Konsultation (**bringt Fragen mit!**)

Stack (aka Stapel, aka Keller)

LIFO-(Last in, first out)-Prinzip

Ops.: `push, pop, top, depth, empty`

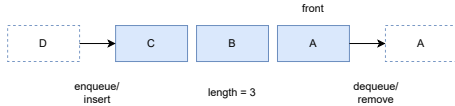


Why? z.B. Ablagestapel für Papier, Rekursion (call stack), ...

Queue (aka Warteschlange)

FIFO-(First in, first out)-Prinzip

Ops.: `enqueue, dequeue, front, empty`



Why? z.B. Musik-Playlist, Warteschlange (vor Webserver), ...

Spezifikation ADT Stack

Sorten: $[\mathbb{B}, \mathbb{N}, \mathit{Element}]$
Definition Sorte: $[\mathit{Stack}]$:

$\mathit{init} : \mathit{Stack}$
$\mathit{push} : \mathit{Element} \times \mathit{Stack} \to \mathit{Stack}$
$\mathit{empty} : \mathit{Stack} \to \mathbb{B}$
$\mathit{pop} : \mathit{Stack} \to \mathit{Stack}$ (entferne oberstes Element)
depth : **??** (Höhe des Stapels)
top : **??** (oberstes Element des Stapels)

$\mathit{empty}(\mathit{init}) = \mathit{True}$
$\forall\, s \in \mathit{Stack};\ \forall\, e \in \mathit{Element} \bullet \mathit{empty}(\mathit{push}(e, s)) = \mathit{False}$
pop? top? depth?

Aufgabe: Füllt die mit **??** markierten Lücken.
Hinweis: Es geht um das Prinzip und nicht darum die VL auswendig zu lernen.

Implementierung Stack und Queue

Implementierung des ADTs Stack

- auf Basis verketteter Liste
- auf Basis eines Arrays (Achtung: Fehlerbehandlung, da Höche begrenzt)
- ...

Implementierung des ADTs Queue

- auf Array-Basis als Ringspeicher (Achtung: Fehlerbehandlung, da Höche begrenzt)
- auf Basis verketteter Liste mit Kapselung
- ...

Mehr Details: Tafel/Vorlesung.

Fragen?

Aufgaben

Lösen Sie die folgenden Aufgaben! Nutzen Sie dazu die Konzepte aus dieser Übung!
Weitere Aufgaben können jederzeit beim Übungsleiter erfragt werden.

Aufgaben: Stack

Gegeben Sei die Implementierung eines Stacks auf Basis einer verketteten Liste (s. Stud.IP) und die Spezifikation eines Stacks auf der folgenden Folie.

- Vereinfachen Sie folgende Terme zur Normalform:
  a) $depth(push(A, push(B, push(C, pop(push(D, init))))))$
  b) $top(pop(push(A, push(B, push(C, push(D, init))))))$
  c) $depth(init)$
  d) $pop(init)$
- Stellen Sie die obigen Ausdrücke in der Implementierung dar und lassen sich die Ergebnisse auf dem Bildschirm ausgeben. Ist die Spezifikation korrekt implementiert? Beheben Sie ggf. den/die Fehler!
- Schreiben Sie auf Basis der Stack-Implementierung ein Programm, welches für einen gegebenen String prüft, ob alle öffnenden Klammern eine entsprechende Schließende Klammer besitzen, e.g., `"(()(()))"` $\rightarrow 1$, `"(()())"` $\rightarrow 0$.
- Erweitern Sie das Programm auf alle drei Klammern, e.g., `"(){}[()]"` $\rightarrow 1$; `"({}}"` $\rightarrow 0$.

Vollständige Spezifikation Stack

$Element \in \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, *\}$
$errorelement = *$
$Element_v = Element \setminus \{errorelement\}$
Sorten: $[\mathbb{B}, \mathbb{N}, Element]$
$[Stack]$

$$
\begin{array}{|l}
init : Stack \\
push : Element \times Stack \to Stack \\
empty : Stack \to \mathbb{B} \\
pop : Stack \to Stack \\
depth : Stack \to \mathbb{N} \\
top : Stack \to Element \\
\hline
\forall\, s \in Stack;\ \forall\, e \in Element_v \bullet \\
\quad push(errorelement, s) = s \\
\ldots
\end{array}
$$

Vollständige Spezifikation Stack

$$
\begin{array}{|l}
\hline
\ldots \\
\hline
\forall\, s \in \mathit{Stack};\ \forall\, e \in \mathit{Element}_v \bullet \\
\quad \mathit{empty}(\mathit{init}) = \mathit{True} \\
\quad \mathit{empty}(\mathit{push}(e, s)) = \mathit{False} \\
\hline
\quad \mathit{top}(\mathit{init}) = \mathit{errorelement} \\
\quad \mathit{top}(\mathit{push}(e, s)) = e \\
\hline
\quad \mathit{pop}(\mathit{init}) = \mathit{init} \\
\quad \mathit{pop}(\mathit{push}(e, s)) = s \\
\hline
\quad \mathit{depth}(\mathit{init}) = 0 \\
\quad \mathit{depth}(\mathit{push}(e, s)) = 1 + \mathit{depth}(s) \\
\hline
\end{array}
$$