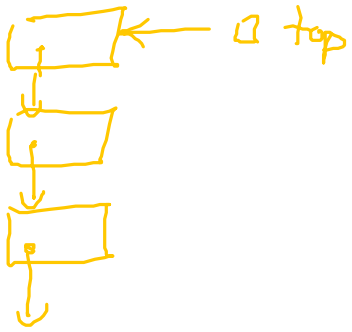
Stacks

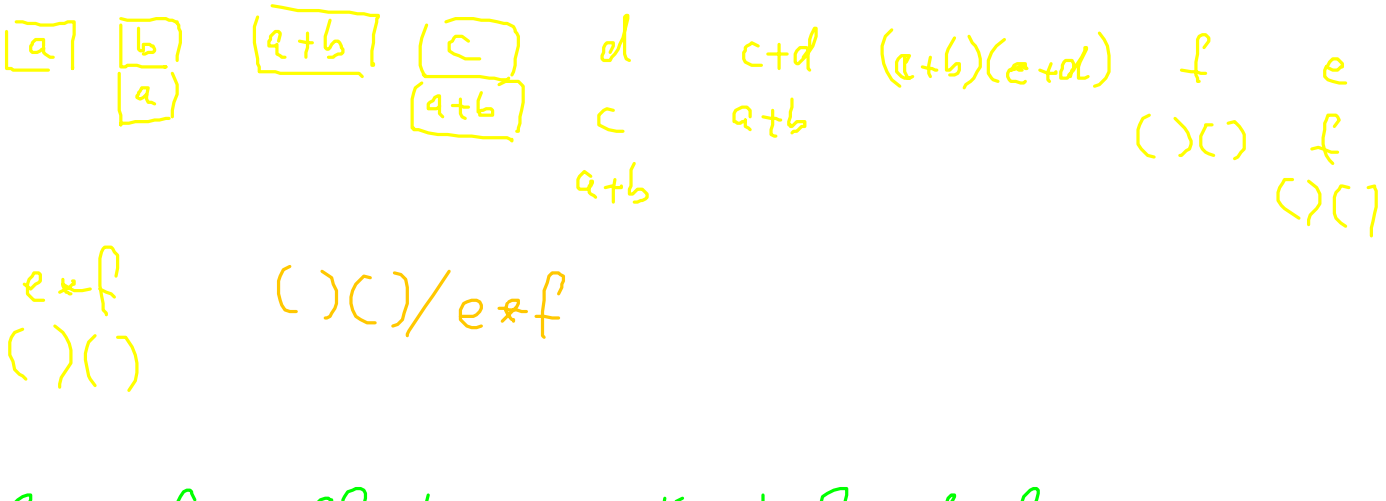
top
push
pop
isEmpty



RPN - Rechner (Reverse Polish Notation)

(a+b)(c+d)/(f*e) ← algebraische Notation

→ a b + c d + * f e * /



Klammerfreie Schreibweise arithmetischer Ausdrücke

Erkennung korrekter Klammerausdrücke

((( )( ))( ))

① Testen ob korrekt

② Finden der zugehörigen Paare

Algorithmus: Erkennung korrekter Klammerausdrücke

1. Lese die Folge der Klammern von links nach rechts
2. Falls (so pushe diese auf Stack
3. Falls) so nehme top Element vom Stack (ist ( )
   und erkläre diese zu Paar
4. Erkläre den Klammerausdruck als korrekt, wenn
   der Stack am Ende leer ist und wir zwischendrin
   vom leeren Stack gepoppt wird

→ "Stackbedingungen"

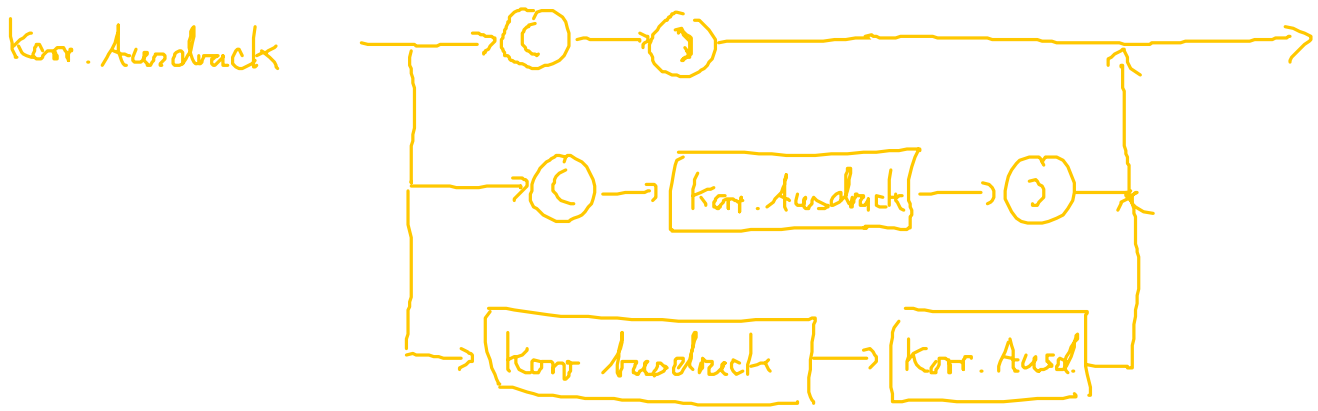
Bsp ((())()) ()
1 2 3 4 5 6 7 8

Stackfolge: $\emptyset$ C_1 C_2 C_1 C_4 C_1 $\emptyset$ C_7 $\emptyset$ } Stackbed. erfüllt
C_1 C_1

SATZ: Alg. erkennt genau die korrekten Klammerausdrücke als korrekt und er identifiziert zueinander gehörige Klammerausdrücke richtig

Beweis: vollständige Ind. nach Anzahl der Klammern

Erinnerung: Korrekte Klammerausdrücke genügen folgender Grammatik (Syntaxdiagramm)



1. A korrekt $\Rightarrow$ Algo erkennt A als korrekt und ermittelt die richtige Paarung

Ind. Anf.: Länge = 2 $\Rightarrow$ A = ( )

$\Rightarrow$ Algo erkennt A als korrekt und findet richtige Paarung

Ind. Vor.: Beh 1. sei korrekt für alle Ausdrücke der Länge $< n$

Schluss auf Länge $n > 2$

Sei A korrekt mit n Klammern

$\Rightarrow$ (Grammatik) $A = (_1\ B\ )_2$ oder $A = B\,C$, (B, C korrekt)

(a) (b)

Fall a) Stackfolge

$\emptyset\ C_1 \mid\mid\mid\mid\mid\mid\ C_1$ $\quad (_1()()_2$

$C_1 C_1 C_1 C_1 C_1$

nur Klammern aus B

auf B trifft Ind. Vor. zu $\Rightarrow$ Algo erkennt richtige Paarung und B als korrekt

liest nun Schluss $)_2$, dann ist nun $(_1$ im Stack und Algo erklärt sie als zugehörig

Fall b) $A = BC$ $\quad (())()$ (B C)

Stackfolge von A $\stackrel{!}{=}$ Stackfolge von B gefolgt von Stackfolge von C

$(|||||_1') \; \emptyset \; (|||||) \; \emptyset$

Stackfolge von B $\uparrow$ Stackfolge von C, ebenfalls alles richtig erkannt nach Ind. Vor.

$\Rightarrow$ Ind Vor. Stack dann leer, Klammern richtig erkannt

2. Wenn Stackkeel erfüllt $\Rightarrow$ A korrekt und Algo hat richtige Klammerung erkannt

Ind. Anf: Länge $= 2 \Rightarrow$ $A = ()$

einzige Ausdruck mit ≤ 2 Klammern bei dem Stackkeel

erfüllt

A korrekt

Ind. Annahme: Sei 2. richtig für Ausdrücke der Länge $< n$

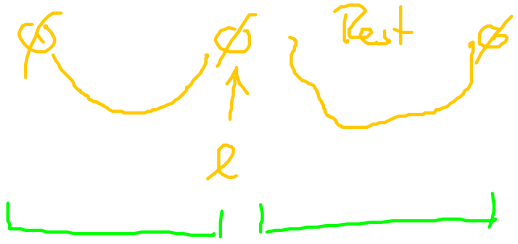
Ind Schluss auf n

A habe Länge n und Stackbed seien erfüllt

Betrachte erste Klammer, bei deren Lesen der Stack leer wird
Sei diese Klammer an Position l (spätestens bei $l = n$ der Fall)

a) $l < n$ b) $l = n$

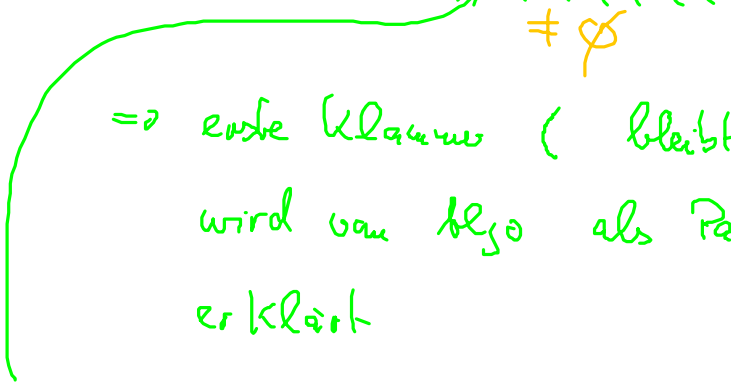
a) Stackfolge



Kürzere Stackfolgen für die die Stackbed gelten
$\Rightarrow$ (Ind. Vor.) die gehören zu korrekten Ausdrücken B, C
$\Rightarrow$ A = BC $\Rightarrow$ A korrekt nach Grammatik

b) Stackfolge $\emptyset$ (|||||||) $\emptyset$

$\neq \emptyset$



⇒ erste Klammer (bleibt bis Schluss auf Stack und wird von dort als Partner zur letzten Klammer) erkannt

⇒ für innere Stackfolge ohne die erste Klammer (erfüllt Stackbed

⇒ gehört nach Ind. Vor. zu korr. Ausdruck B

⇒ A = (B) ⇒ A korrekt nach Grammatik □

Applet dazu (verwendet Klasse Stack)

verwenden Hilfsarray: partner

$$partner[i] := \begin{cases} k & \text{Zeichen an Stelle } k \text{ und } i \text{ bilden ein Paar von Klammern} \\ -1 & \text{Zeichen ist keine Klammer} \end{cases}$$

Bsp: $(a+b)(c+d)$

partner

4	-1	-1	-1	0	9	-1	-1	-1	5
0	1	2	3	4	5	6	7	8	9

Struktogramm

Java Applet

5.8 Warteschlangen (Queues)

Liste: Einfügen am Ende
Löschen am Anfang

5.9 Abschließende Bemerkungen

2 Arten um strukturierte Datentypen zu definieren

Hierarchisch / Konstruktiv

double — Zahlen
↓
double [] — Vektoren
↓
double [] [] — Matrizen

Axiomatisch

Methoden und Komponenten, Verhalten wird über die Methoden festgelegt

LinkedList, Stack ...

Kapitel 6 Algorithmen auf Arrays

3 Anwendungen:

- Suche eines Elements
- lineare Gleichungssysteme lösen

- kürzeste Wege ermitteln