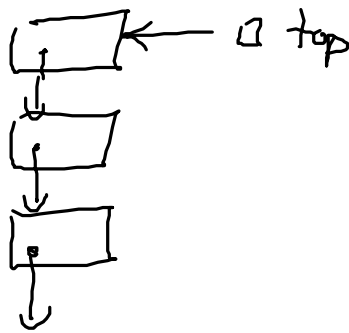


Stacks

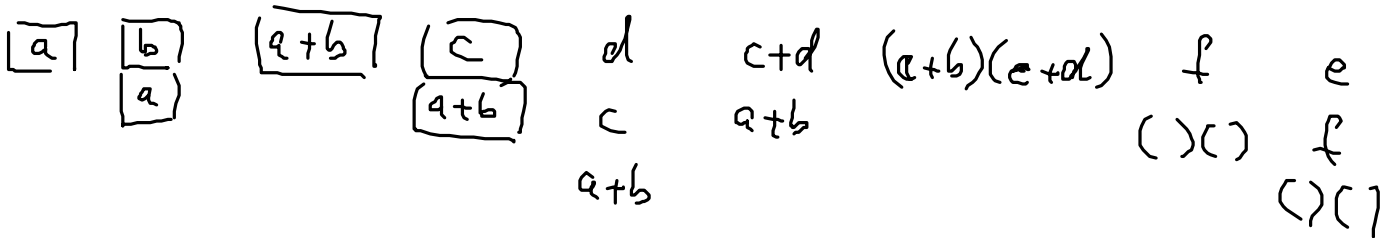
top
push
pop
isEmpty



RPN - Redner (Reverse Polish Notation)

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

→ $ab + cd + * fe * /$

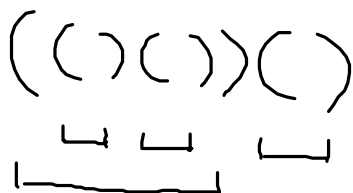


$e * f$
 $()()$

$()() / e * f$

klammerfreie Schreibweise arithmetischer Ausdrücke

Erkennung korrekter Klammersausdrücke



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 wie zwischen dem von leeren Stack gepoppt wird

→ "Stackbedingungen"

Bsp

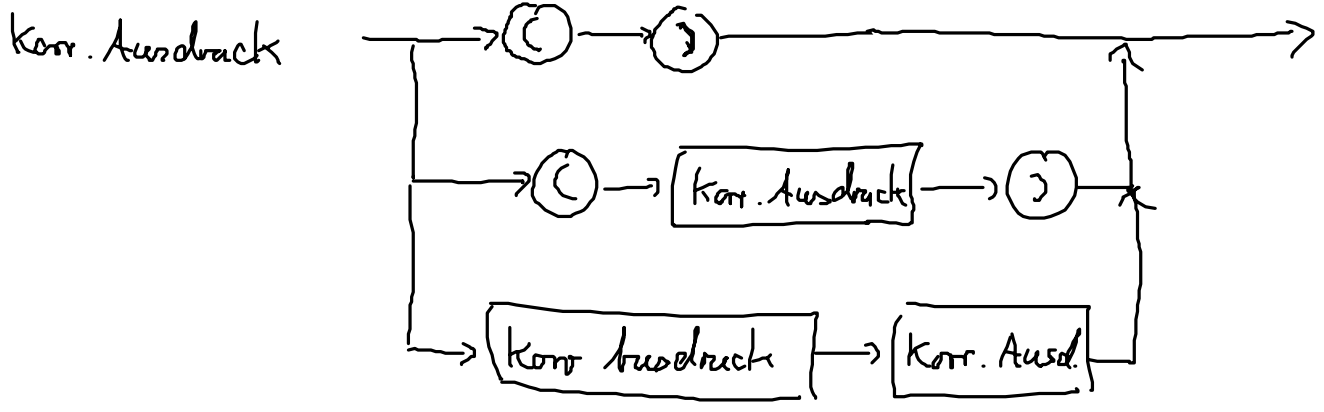
$$\begin{array}{cccccccc} (&) & (&) &) & (&) \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline & & \underbrace{\quad} & \underbrace{\quad} & & \underbrace{\quad} & & \underbrace{\quad} \end{array}$$

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

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 heißen Folgen der Grammatik (Syntaxdiagramm)



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

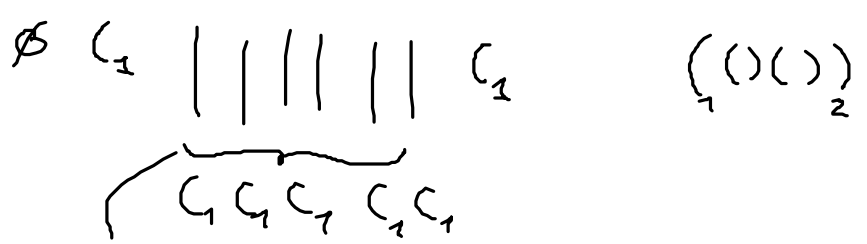
Ind. hyp.: Länge = 2 \Rightarrow A = ()
 \Rightarrow also erkennt A als korrekt und findet richtige Paarung

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

Schluss auf Länge n > 2

Sei A korrekt mit n Klammern
 \Rightarrow (Grammatisch) $A = \underbrace{(\ B)}_a$ oder $A = \underbrace{B C}_b$, $\overbrace{B, C \text{ korrekt}}$

Fall a) Stackfolge



nur Klammern aus B

auf B trifft Ind. Ver. zu \Rightarrow also erkennt richtige Paarung
und B als korrekt

liest zum Schluss $)_2$, dann ist nur $(_1$ im Stack
und also erklärt sie als zugehörig

Fall b) $A = BC$ $\underbrace{(())}_{B} \underbrace{())}_{C}$

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

$(| | | | |)' \emptyset$ $(+ + + +) \emptyset$
 $\underbrace{\hspace{10em}}_{\text{Stackfolge von } B} \uparrow \underbrace{\hspace{10em}}_{\text{Stackfolge von } C}$, ebenfalls alles richtig erkannt nach Ind. Ver.
 \Rightarrow Ind. Ver.
Stack dann leer, Klammern richtig erkannt

2. Wenn Stack leer erfüllt $\Rightarrow A$ korrekt und also hat richtige Klammerung erkannt

Ind. Anf: Länge = 2 $\Rightarrow A = \underline{()}$
einzige Ausdruck mit ≤ 2
Klammern bei dem Stack leer

erfüllt

A korrekt

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

Ind Schluss auf n

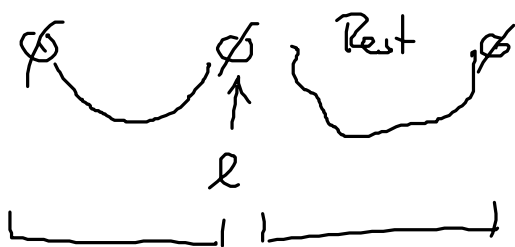
A habe Länge n und Stackbed seien erfüllt

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) $l = n$

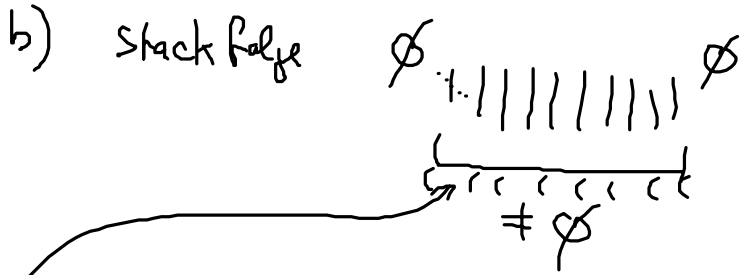
a) Stackfolge



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



\Rightarrow erste Klammer (bleibt bis Schluss auf Stack und wird von \emptyset als Partner zur letzten Klammer) erklärt

\Rightarrow für innere Stackfolge ohne die erste Klammer (erfüllb Stackbed

\Rightarrow gehört nach Ind. Ver. zu korrekter Ausdruck \mathcal{B}

$\Rightarrow A = \{ \mathcal{B} \} \Rightarrow A$ korrekt nach Grammatik \square

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