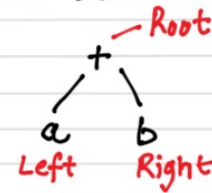


3.4.3

REVERSE POLISH NOTATION (RPN):

A mathematical equation is always **INORDER**; means, all the operators are between operands. E.g: $n + e \div m$

operands operators



- A computer doesn't recognise inorder equations.

- It converts every inorder equation to its equivalent postorder equation.

WRITE

infix

NOTATION $a + b$

prefix

 $+ab$

postfix

 $ab+$

↓ Reverse Polish Notation (RPN)

READ

inorder

Left, Root, Right

preorder

Root, Left, Right

postorder

Left, Right, Root

- This conversion eliminates the complication in formation of solution order. E.g: in postorder there is no need of BODMAS implementation.

- Conversion from infix to other forms of notations is only possible through **BINARY TREE**, i.e. firstly BT is made and then read in a different required order...



Zak
ZAFAR ALI KHAN

3.4.3 Translation software.pdf - Adobe Acrobat Pro DC

File Edit View Window Help

Home Tools 9608-syllabus-201... 4.1.2 Algorithms.pd... 3.4.3 Translation so... x

DL: 16.0 kbps UL: 227.3 kbps Sign in

17 / 19

an advantage of being easy to implement with quick access time.

Practical implications:

- Reverse Polish expression can be processed directly from left to right
- RPN is free from ambiguities.
- Does not require brackets: the user simply performs calculations in the order that is required, letting the automatic stack store intermediate results on the fly for later use.
- There is no requirement for the precedent rules required in infix notation.
- Calculations occur as soon as an operator is specified. Thus, expressions are not entered wholesale from left to right but calculated one piece at a time.
- The automatic stack permits the automatic storage of intermediate results for later use. **This key feature is what permits RPN calculators to easily evaluate expression of arbitrary complexity also; they do not have limits on the complexity of expressions they can evaluate.**
- In RPN calculators, no equals key is required to force computation to occur.
- Users must know the size of the stack, because practical implementations of RPN use different sizes for the stack.

9:43 PM 01/05/2017

Walk through chart for infix-postfix conversion.

1, INFIX

2, BINARY TREE

3, POSTFIX (L, Ri, Root)

$(a+b)/c$

Follow BODMAS

$ab+c\div$ (RPN)

→ INFIX

↓

BINARY TREE

↓

POSTFIX (RPN)

↓

STACK

↓

Are there any variables in STACK data structure?

Yes →

No (means only constants) → Solve and find absolute answer.

Find RPN of following infix notation:

1, $a+b/g*h*j-k$

2, $((a+b)/((g*h)*(j-k)))$

3, $((a-b)*(c+d))/e$

4, $abc-d$

5, $a+b+c/d$

—BODMAS—

1, $a+b/g*h*j-k$

$abg\div h*j*+k-$

2, $((a+b)/((g*h)*(j-k)))$

$ab+gh*jk-*\div$

③

$ab-cd+*e\div$ (RPN)



Find postfix variables
Replace with their values
using stack
randomly where;

Classwork
 $A = 2$
 $B = 3$
 $C = 4$

$A - B * C$

$((((15 - 10) \div (15 \div 3)) + 9) * 5)$

15 10 - 15 3 ÷ ÷ 9 + 5 *

Stack operations:

- 15 - 10 → 5
- 15 ÷ 3 → 5
- 5 ÷ 5 → 1
- 1 + 9 → 10
- 10 * 5 → 50

