

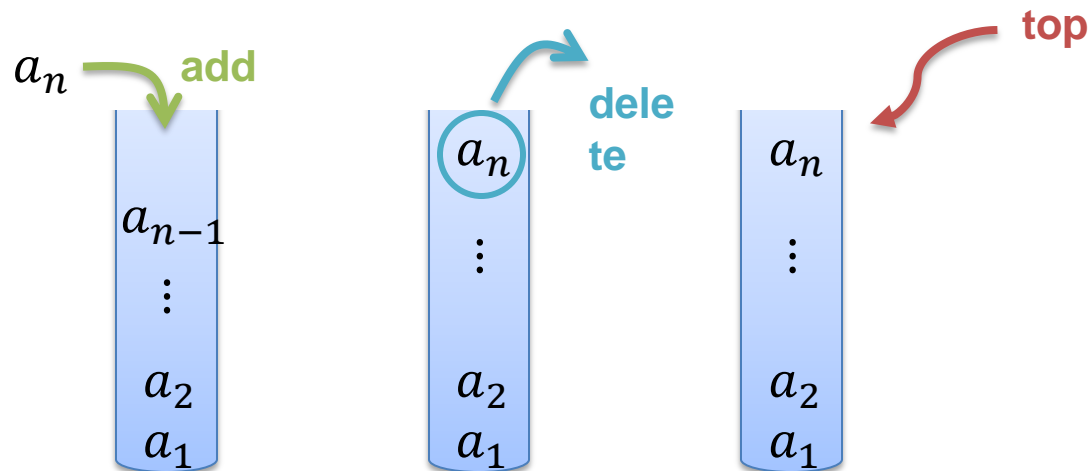
Infix, Prefix, and Postfix

Kuan-Yu Chen (陳冠宇)

2020/10/05 @ TR-313, NTUST

Review

- Stack
 - A **stack** is an **ordered** list in which insertions and deletions are made at one end called the **top**
 - Stack Permutation



Expressions

- When pioneering computer scientists conceived the idea of higher-level programming languages, they were faced with many hurdles
 - How to generate machine-language instructions to evaluate an arithmetic expression

$$A \div B - C + D \times E - A \times C$$

Operator (運算子)

Operand (運算元)

- The first problem with understanding the meaning of an expression is to decide in what order the operations are carried out
 - Specify the order by using parentheses

$$A \div (B - C) + D \times (E - A) \times C$$

$$(A \div B) - C + (D \times E) - (A \times C)$$

priority	operator
1	unary minus
2	not
3	*, /, div, mod, and
4	+, -, or, xor
5	<, <=, =, <>, >=, >, in

Infix & Postfix Notations

- If e is an expression with operators (運算子) and operands (運算元), the conventional way of writing e is called **infix**
 - The operators come **in-between** the operands

$$A \div B - C + D \times E - A \times C$$

- The **postfix** form of an expression calls for each **operator to appear after its operands**

$$AB \div C - DE \times +AC \times -$$

operation	postfix
$T_1 := A/B$	$T_1 C - DE * + AC * -$
$T_2 := T_1 - C$	$T_2 DE * + AC * -$
$T_3 := D * E$	$T_2 T_3 + AC * -$
$T_4 := T_2 + T_3$	$T_4 AC * -$
$T_5 := A * C$	$T_4 T_5 -$
$T_6 := T_4 - T_5$	T_6

Infix to Postfix

- It is simple to describe an algorithm for producing postfix from infix

- (1) Fully parenthesize the expression.
- (2) Move all operators so that they replace their corresponding right parentheses.
- (3) Delete all parentheses.

– Take $A \div B - C + D \times E - A \times C$ for example

- $((((A \div B) - C) + (D \times E)) - (A \times C))$

- $((((A \div B) - C) + (D \times E)) - (A \times C))$

- $AB \div C - DE \times +AC \times -$

Infix & Prefix Notations

- If e is an expression with operators and operands, the conventional way of writing e is called **infix**
 - The operators come **in-between** the operands

$$A \div B - C + D \times E - A \times C$$

- In the **prefix** form of an expression, the **operators precede their operands**

- Take $A \div B - C + D \times E - A \times C$ for example

- $((((A \div B) - C) + (D \times E)) - (A \times C))$

- $((((A \div B) - C) + (D \times E)) - (A \times C))$

- $- + - \div ABC \times DE \times AC$

infix	prefix
$A * B / C$	$/*ABC$
$A / B - C + D * E - A * C$	$- + - / ABC * DE * AC$
$A * (B + C) / D - G$	$- /* A + BCDG$


Examples.

- Given an infix expression $A + B \times C - D \div E$, please write down the prefix and postfix expressions

$$A + B \times C - D \div E$$


$$((A + (B \times C)) - (D \div E))$$

– Prefix

$$((A + (B \times C)) - (D \div E))$$


$$- +A \times BC \div DE$$

– Postfix

$$((A + (B \times C)) - (D \div E))$$


$$ABC \times +DE \div -$$

Examples..

- Given an infix expression $(A + B) \times C \div (D - E \div F)$, please write down the prefix and postfix expressions

$$(((A + B) \times C) \div (D - (E \div F)))$$

– Prefix

$$\div \times + ABC - D \div EF$$

– Postfix

$$AB + C \times DEF \div - \div$$

Examples...

- Given an infix expression $A \wedge \neg(B > C) \vee (D \vee \neg E)$, please write down the prefix and postfix expressions

$$((A \wedge (\neg(B > C))) \vee (D \vee (\neg E)))$$

– Prefix

$$((A \wedge (\neg(B > C))) \vee (D \vee (\neg E)))$$



$$\vee \wedge A \neg > B C \vee D \neg E$$

– Postfix

$$((A \wedge (\neg(B > C))) \vee (D \vee (\neg E)))$$



$$A B C > \neg \wedge D E \neg \vee \vee$$

Examples....

- Given a postfix expression $ABCD + \times E \div -$, please write down the infix expression
 - The pattern for postfix is $\langle operand_1, operand_2, operator \rangle = operand_1 operator operand_2$

$$ABCD + \times E \div -$$

$$ABC\boxed{D} + \times E \div -$$

$$C + D$$

$$A\boxed{BCD} + \times E \div -$$

$$B \times (C + D)$$

$$A\boxed{BCD + \times E \div} -$$

$$B \times (C + D) \div E$$

$$\boxed{ABC D + \times E \div -}$$

$$A - B \times (C + D) \div E$$

Examples.....

- Given a prefix expression $-A \div \times B + CDE$, please write down the infix expression
 - The pattern for prefix is $\langle operator, operand_1, operand_2 \rangle = operand_1 operator operand_2$

$$-A \div \times B + CDE$$

$$-A \div \times B + CDE$$

$$C + D$$

$$-A \div \times B + CDE$$

$$B \times (C + D)$$

$$-A \div \times B + CDE$$

$$B \times (C + D) \div E$$

$$-A \div \times B + CDE$$

$$A - B \times (C + D) \div E$$

Algorithm to Convert Infix to Postfix.

Step 1: Add ")" to the end of the infix expression

Step 2: Push "(" on to the stack

Step 3: Repeat until each character in the infix notation is scanned

IF a "(" is encountered, push it on the stack

IF an operand (whether a digit or a character) is encountered, add it to the postfix expression.

IF a ")" is encountered, then

a. Repeatedly pop from stack and add it to the postfix expression until a "(" is encountered.

b. Discard the "(" . That is, remove the "(" from stack and do not add it to the postfix expression

IF an operator 0 is encountered, then

a. Repeatedly pop from stack and add each operator (popped from the stack) to the postfix expression which has the same precedence or a higher precedence than 0

b. Push the operator 0 to the stack

[END OF IF]

Step 4: Repeatedly pop from the stack and add it to the postfix expression until the stack is empty

Step 5: EXIT

Algorithm to Convert Infix to Postfix..

- Take $A - (B \div C + (D \% E \times F) \div G) \times H$ for example

Infix Scanned	Stack	Postfix Expression
	(
A	(A
-	(-	A
((- (A
B	(- (A B
/	(- (/	A B
C	(- (/	A B C
+	(- (+	A B C /
((- (+ (A B C /
D	(- (+ (A B C / D
%	(- (+ (%	A B C / D
E	(- (+ (%	A B C / D E

Step 1: Add ")" to the end of the infix expression
 Step 2: Push "(" on to the stack
 Step 3: Repeat until each character in the infix notation is scanned
 IF a "(" is encountered, push it on the stack
 IF an operand (whether a digit or a character) is encountered, add it to the postfix expression.
 IF a ")" is encountered, then
 a. Repeatedly pop from stack and add it to the postfix expression until a "(" is encountered.
 b. Discard the "(" . That is, remove the "(" from stack and do not add it to the postfix expression
 IF an operator 0 is encountered, then
 a. Repeatedly pop from stack and add each operator (popped from the stack) to the postfix expression which has the same precedence or a higher precedence than 0
 b. Push the operator 0 to the stack
 [END OF IF]
 Step 4: Repeatedly pop from the stack and add it to the postfix expression until the stack is empty
 Step 5: EXIT

Algorithm to Convert Infix to Postfix...

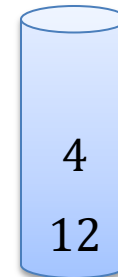
- Take $A - (B \div C + (D \% E \times F) \div G) \times H$ for example

Infix Scanned	Stack	Postfix Expression
E	(- (+ (%	A B C / D E
*	(- (+ (*	A B C / D E %
F	(- (+ (*	A B C / D E % F
)	(- (+	A B C / D E % F *
/	(- (+ /	A B C / D E % F *
G	(- (+ /	A B C / D E % F * G
)	(-	A B C / D E % F * G / +
*	(- *	A B C / D E % F * G / +
H	(- *	A B C / D E % F * G / + H
)		A B C / D E % F * G / + H * -

Evaluation of Postfix Expression

- Step 1: Add a ")" at the end of the postfix expression
- Step 2: Scan every character of the postfix expression and repeat Steps 3 and 4 until ")" is encountered
- Step 3: IF an operand is encountered, push it on the stack
IF an operator \circ is encountered, then
- Pop the top two elements from the stack as A and B as A and B
 - Evaluate $B \circ A$, where A is the topmost element and B is the element below A.
 - Push the result of evaluation on the stack
- [END OF IF]
- Step 4: SET RESULT equal to the topmost element of the stack
- Step 5: EXIT

Infix: $12 \div 4$
Postfix: $12\ 4\ \div$



$A = 4$
 $B = 12$

Example

- Consider the postfix expression given as $9\ 3\ 4\ *\ 8\ +\ 4\ /\ -$, please evaluate the expression

Step 1: Add a ")" at the end of the postfix expression

Step 2: Scan every character of the postfix expression and repeat Steps 3 and 4 until ")" is encountered

Step 3: IF an operand is encountered, push it on the stack
IF an operator θ is encountered, then

- Pop the top two elements from the stack as A and B as A and B
- Evaluate $B\ \theta\ A$, where A is the topmost element and B is the element below A.
- Push the result of evaluation on the stack

[END OF IF]

Step 4: SET RESULT equal to the topmost element of the stack

Step 5: EXIT

Character Scanned	Stack
9	9
3	9, 3
4	9, 3, 4
*	9, 12
8	9, 12, 8
+	9, 20
4	9, 20, 4
/	9, 5
-	4

Algorithm to Convert Infix to Prefix

Step 1: Reverse the infix string. Note that while reversing the string you must interchange left and right parentheses.
Step 2: Obtain the postfix expression of the infix expression obtained in Step 1.
Step 3: Reverse the postfix expression to get the prefix expression

- Take $(A - B \div C) \times (A \div K - L)$ for example
 - Step1: $(L - K \div A) \times (C \div B - A)$
 - Step2: $LKA \div -CB \div A - \times$
 - Step3: $\times -A \div BC - \div AKL$

Evaluation of Prefix Expression

Step 1: Accept the prefix expression

Step 2: Repeat until all the characters in the prefix expression have been scanned

- (a) Scan the prefix expression from right, one character at a time.
- (b) If the scanned character is an operand, push it on the operand stack.
- (c) If the scanned character is an operator, then
 - (i) Pop two values from the operand stack
 - (ii) Apply the operator on the popped operands
 - (iii) Push the result on the operand stack

Step 3: END

Example

- Consider the prefix expression $+ - 2 7 * 8 / 4 12$, please apply the algorithm to evaluate this expression

Step 1: Accept the prefix expression
Step 2: Repeat until all the characters in the prefix expression have been scanned

- (a) Scan the prefix expression from right, one character at a time.
- (b) If the scanned character is an operand, push it on the operand stack.
- (c) If the scanned character is an operator, then
 - (i) Pop two values from the operand stack
 - (ii) Apply the operator on the popped operands
 - (iii) Push the result on the operand stack

Step 3: END

Character scanned	Operand stack
12	12
4	12, 4
/	3
8	3, 8
*	24
7	24, 7
2	24, 7, 2
-	24, 5
+	29

Appendix – Convert Infix to Prefix

```
Step 1: Scan each character in the infix
        expression. For this, repeat Steps
        2-8 until the end of infix expression
Step 2: Push the operator into the operator stack,
        operand into the operand stack, and
        ignore all the left parentheses until
        a right parenthesis is encountered
Step 3: Pop operand 2 from operand stack
Step 4: Pop operand 1 from operand stack
Step 5: Pop operator from operator stack
Step 6: Concatenate operator and operand 1
Step 7: Concatenate result with operand 2
Step 8: Push result into the operand stack
Step 9: END
```

Questions?



kychen@mail.ntust.edu.tw