CC4

Chapter III - IV: Stacks, Queues, and Evaluation of Expressions

College of Information Technology and Computer Science

by: RBMB,FAP

1

Stacks and Queues

- **Stacks:** are data structures that has :
  a) n number of elements **ε**
  b) Container (open on one side)

  Stacks are commonly described as Last-In First-Out (LIFO) data structures.

- **Queues:** are data structures that has:
  a) n number of elements **ε**
  b) Container (open on both sides)

  Queues are commonly described as First-In First-Out (FIFO) data structures.

College of Information Technology and Computer Science

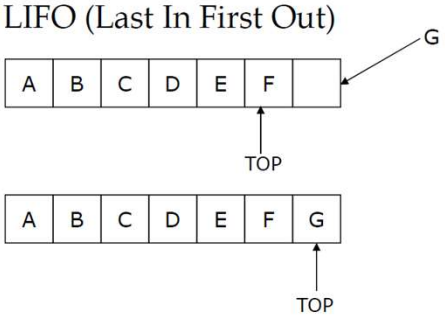
by: RBMB,FAP

2

Stacks

An ordered list in which all insertions and deletions are made at one end called the TOP.

LIFO (Last In First Out)



College of Information Technology and Computer Science

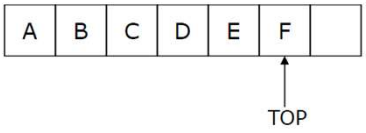
by: RBMB,FAP

3

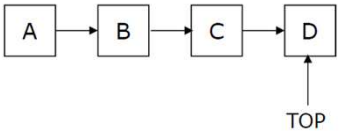
Stacks

Representation:

- One-dimensional array



- Singly linked-list



College of Information Technology and Computer Science

by: RBMB,FAP

4

Stacks

A stack is **full** when: Top = n-1

A stack is **empty** when: Top = -1

Insert and element in the stack: **Push**

Push condition: ?

Delete an element in the stack: **Pop**

Pop condition: ?

5

Stacks

Real-Life examples?

Technical Examples:

- **ALUs** in CPUs usually **compute arithmetic** using stacks. Most calculators still do ("stack machines").
- Most **compilers** also **parse syntax and evaluate expressions** using stacks before translating into low-level code.

6

Evaluation of Expressions using Stacks

Expression is made up of operands, operators and delimiters.

Operands can be any legal variable names or constants in programming languages.

Operations are described by operators:

Basic arithmetic operators: + - * /

Unary operators: - +

Relational operators: ==, >, <, >=, <=

7

Evaluation of Expressions using Stacks

Our concern: how the expressions are evaluated

The compiler accepts expressions and produce correct result by reworking the expressions into postfix form. Other forms include infix and prefix.

Prefix : <Operator> <Operand1> <Operand2>

Postfix : <Operand1> <Operand2> <Operator>

Infix : <Operand1> <Operator> <Operand2>

8

Evaluation of Expressions using Stacks

Expression: A + B
Prefix : +AB
Postfix: AB+

Expression: (A+B*C)/D
Prefix : /+A*BCD
Postfix: ABC*+D/

9

Infix to Postfix using Stacks

IN-Stack Priority (ISP) – The priority of the operator as an element of the stack.

IN-Coming Priority (ICP) – The priority of the operator as current token.

SYMBOL	ISP	ICP
)	--	--
^	3	4
*, /, %	2	2
+, -	1	1
(	0	4

Note: ISP and ICP of # sign = -1

10

Infix to Postfix using Stacks

Algorithm:

If *token(x)* is an operand, print in the output

Else if *token(x)* is a '**)**', print and pop all operators until '**(**', discard '**(**'

Else

while the ISP of *stack(y)* >= the ICP of *token(x)*, print and pop all operators. Push *token(x).*

Push *token(x)*

If *token(x)* is a **'#'** and stack is not empty, pop and print all stack contents until **'#'**

College of Information Technology and Computer Science

by: RBMB,FAP

11

Infix to Postfix using Stacks

Exercise: Convert the expression from Infix to Postfix using stacks

E = 5+3*7-9%4^2+6/3#

College of Information Technology and Computer Science

12

Infix to Postfix using Stacks

token(x)	*stack(y)*	*output*
5	#	5
+	#+	5
3	#+	53
*	#+*	53
7	#+*	537
-	#-	537*+
9	#-	537*+9
%	#-%	537*+9
4	#-%	537*+94
^	#-%^	537*+94

College of Information Technology and Computer Science

by: RBMB,FAP

13

Infix to Postfix using Stacks

token(x)	*stack(y)*	*output*
2	#-%^	537*+942
+	#+	537*+942^%-
6	#+	537*+942^%-6
/	#+/	537*+942^%-6
3	#+/	537*+942^%-63
#	#	**537*+942^%-63/+**

College of Information Technology and Computer Science

by: RBMB,FAP

14

Infix to Postfix using Stacks

Exercise: Convert the expression from Infix to Postfix using stacks

E = ((5+3)*7)-9%4^(2+6/3)#

College of Information Technology and Computer Science

15

Queues

- An ordered list which all insertions take place at one end, called the REAR, while all deletions take place at the other end, called the FRONT.
- FIFO (First-In-First-Out)
  Elements are processed in the same order as they were received. The first element inserted in the queue will be the first one to be removed.

College of Information Technology and Computer Science

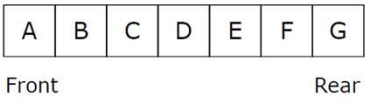
by: RBMB,FAP

16

Queues

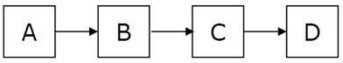
Representation

One-dimensional Array



Front Rear

Singly linked-list



Rear Front

College of Information Technology and Computer Science

by: RBMB,FAP

17

Queues

A queue is **full** when: Rear = n-1

A queue is **empty** when: Rear = -1

Insert condition: ?

Delete condition: ?

College of Information Technology and Computer Science

by: RBMB,FAP

18