DATA STRUCTURE

" STACK"

IT SHINTA P

STMIK MDP

APRIL **2008**

Characteristics of a Stack

A stack is a collection of elements, which can be stored and retrieved one at a time.

Elements are retrieved in reverse order of their time of storage, i.e. the latest element stored is the next element to be retrieved.

A stack is sometimes referred to as a Last-In-First-Out (LIFO) or First-In-Last-Out (FILO) structure. Elements previously stored cannot be retrieved until the latest element (usually referred to as the 'top' element) has been retrieved.

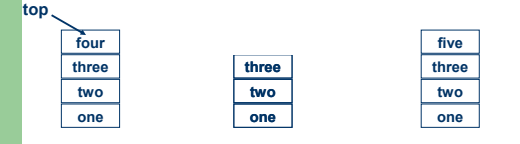
Characteristics of a Stack (Cont.)

- New nodes can only be added to the top of the stack
- Nodes may only be removed from the top of the stack
- The depth of a stack is the number of elements it contains
- It is therefore a last-in, first-out structure (LIFO)

Typical Operations on Stack :

OPERATION	PRE-CONDITION	POST-CONDITION
push (Object item)	stack not full	stack +1, item on top of stack
pop()	stack not empty	stack –1, top item removed
empty()	none	stack same
full()	none	stack same

Manipulation of a Stack



Stack with depth of 4 | Stack with depth of 3 | Stack with depth of 4

push(one)
push(two)
push(three)
push(four)

pop( )

push(five)

Stack Application

- Reverse the line order of a text file
- Check to see if brackets match
- Evaluation of complex expressions (intermediate values stored)
- Activation stack (method calls)
- Recursion

Stack Applications (Cont.)

- When analyzing arithmetic expressions, it is important to determine whether an expression is balanced with respect to parentheses
  - (a+b*(c/(d-e)))+(d/e) => Infix
- Problem is further complicated if braces or brackets are used in conjunction with parenthesis
- Solution is to use stacks! => Postfix

Additional Stack Applications

- Consider two case studies that relate to evaluating arithmetic expressions
  - Postfix and infix notation
- Expressions normally written in infix form
  - Binary operations inserted between their operands
- A computer normally scans an expression string in the order that it is input; easier to evaluate an expression in postfix form

Additional Stack Applications (continued)

TABLE 5.4
Postfix Expressions

Postfix Expression	Infix Expression	Value
5 6 *	5 * 6	30
5 6 1 + *	5 * (6 + 1)	35
5 6 * 10 -	(5 * 6) - 10	20
4 5 6 * 3 / +	4 + ((5 * 6) / 3)	14

Additional Stack Applications (continued)

- Advantage of postfix form is that there is no need to group subexpressions in parentheses
- No need to consider operator precedence

Aplikasi **Stack** (Cont.)

Notasi *POSTFIX*: Ekspresi penulisan matematis dimana operator berada disamping nilai data/elemen/variabel penampung data.

Contoh : A + B * C = A B C * +

notasi *infix* notasi *postfix*

Aplikasi **Stack** (Cont.)

Algoritma Konversi *INFIX* ke *POSTFIX*

Level Operator :
Level tertinggi : pangkat (^)
Level menengah : kali (*) dan bagi (/)
Level terendah : tambah (+) dan kurang (-)

1. Jika simbol adalah '(', maka simbol tersebut kita *push* ke dalam *stack*.

Aplikasi Stack (Cont.)

2. Jika simbol adalah ‘)’, maka *pop* semua elemen *stack* sampai terakhir kali simbol ‘(‘ di *push*. Semua elemen tersebut menjadi output kecuali tanda kurung.
3. Jika simbol adalah operand, maka simbol tersebut langsung menjadi output.

Aplikasi Stack (Cont.)

4. Jika simbol adalah sebuah operator, maka :
   1). Jika level operator TOP *stack* >= level operator simbol, maka elemen TOP *stack* kita *pop*.
   2). Jika level operator TOP *stack* < operator simbol, maka operator simbol di *push* ke dalam *stack*.

Aplikasi Stack (Cont.)

Simbol yang diproses	(	(	A	+	B	)	*	C	/	D	+	E	^	F	)	/	G	;
TOP STACK ------>	(	(	(	+	+	(	*	*	/	/	+	+	^	^		/	/	
		(	(	(	(		(	(	(	(	(	(	+	+				
					(	(								(	(			
OUTPUT			A		B	+		C	*	D	/	E		F	^+		G	/

Aplikasi Stack (Cont.)

Latihan :

Ubah notasi *infix* berikut menjadi notasi *postfix*.

1. ((P+Q^(R-S))/(T*(U+V)/W))
2. (P-Q+(R/S^T)*(U-V+W)/X)*Y)

NEXT LECTURE CLASS

“QUEUE” ➔➔➔

Shinta