Mathematical Expressions

- We work with mathematical expressions on a regular basis.
  - Easy to determine the order of evaluation.
  - Easy to calculate.
- But the task is more difficult in computer programs.
  - A program can not visualize the expression to determine the order of evaluation.
  - Must examine one token at a time.

Types of Expressions

- Three different notations can be used:
  - infix: `A + B * C`
  - prefix: `+ A * B C`
  - postfix: `A B C * +`

Infix to Postfix

- Infix expressions can be easily converted by hand to postfix notation.

```
A * B + C / D
```

1. Fully parenthesize the expression.

```
((A * B) + (C / D))
```

2. For each set of (), move operator to the end of the closing parenthesis.

```
((A B *) (C D /) +)
```

Infix to Postfix (cont)

- The expression at the end of step 2:

```
((A B *) (C D /) +)
```

3. Remove all of the parentheses.

```
A B * C D / +
```

- Which results in the postfix version.

Evaluating Postfix Expressions

- We can evaluate a valid postfix expression using a stack structure.
- For each token:
  - If the current token is an operand, push its value onto the stack.
  - If the current token is an operator:
    - pop the top two operands off the stack.
    - perform the operation (top value is RHS operand).
    - push the result of the operation back on the stack.
- The final result will be the last value on the stack.

Postfix Evaluation Examples

- To illustrate the use of the algorithm, assume
  - the existence of an empty stack, and
  - the following variable assignments

```
A = 8        C = 3
B = 2        D = 4
```

- Evaluate the valid expression:

```
A B C + * D /
```

Postfix Example #1

Token	Alg Step	Stack	Description
ABC+*D/	1	8	push value of A
A**B**C+*D/	1	8 2	push value of B
AB**C**+*D/	1	8 2 3	push value of C
ABC**+***D/	2(a)	8	pop top two values: y = 3, x = 2
	2(b)	8	compute z = x + y or z = 2 + 3
	2(c)	8 5	push result (5) of the addition
ABC+*D/	2(a)		pop top two values: y = 5, x = 8
	2(b)		compute z = x * y or z = 8 * 5
	2(c)	40	push result (40) of the multiplication
ABC+***D**/	1	40 4	push value of D
ABC+*D**/**	2(a)		pop top two values: y = 4, x = 40
	2(b)		compute z = x / y or z = 40 / 4
	2(c)	10	push result (10) of the division

Postfix Example #2

- What happens if the expression is invalid?

A B * C D +

Token	Alg Step	Stack	Description
AB*CD+	1	8	push value of A
A**B***CD+	1	8 2	push value of B
AB*CD+	2(a)		pop top two values: y = 2, x = 8
	2(b)		compute z = x * y or z = 8 * 2
	2(c)	16	push result (16) of the multiplication
AB***C**D+	1	16 3	push value of C
AB*C**D**+	1	16 3 4	push value of D
AB*CD**+**	2(a)	16	pop top two values: y = 4, x = 3
	2(b)	16	compute z = x + y or z = 3 + 4
	2(c)	16 7	push result (7) of the addition
Error	***xxxxxx***	***xxxxxx***	***Too many values left on the stack.***

Postfix Example #3

- What happens if there are too many operators for the given number of operands?

A B * + C /

Token	Alg Step	Stack	Description
AB*+C/	1	8	push value of A
A**B***+C/	1	8 2	push value of B
AB*+C/	2(a)		pop top two values: y = 2, x = 8
	2(b)		compute z = x * y or z = 8 * 2
	2(c)	16	push result (16) of the multiplication
AB***+**C/	2(a)		pop top two values: y = 16, x = ?
Error	***xxxxxx***	***xxxxxx***	***Only one value on stack, two needed.***