

# Data Structures in C++

CMPE226- Data Structures

Stacks (Cont.)

Infix, Prefix and Postfix Notation

# Infix, Prefix, and Postfix Notation

To Evaluate:  $2+3*5=?$  **\*** has precedence over **+**.

*Infix: Operator is between the operands. e.g. **A + B***

What about  $(2+3)*5=?$  **Parenthesis** has the precedence.

To avoid parenthesis, **prefix and postfix notation** can be used.

*Postfix: Operator comes after the operands. e.g. **A B +***

*Prefix: Operator comes before the operands. e.g. **+ A B***

Infix Expression	Prefix Expression	Postfix Expression
A + B	+ A B	A B +
A / B + C	?	?

Check the precedence first

# Conversion from Infix to Prefix, and Postfix Notation

Infix Expression	Prefix Expression	Postfix Expression
A + B	+ A B	A B +
A / B + C	?	?

Operations with the highest precedence are converted first. The portion that is converted is treated as a single operand. Evaluated left to right (except exponentiation)

Prefix Conversion:

A / B + C




X = /AB

+XC = **+ /ABC**

Postfix Conversion:

A / B + C



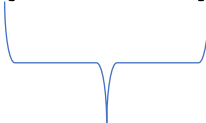
X = AB /

XC + = **AB / C +**

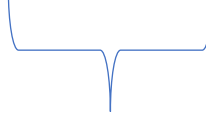
# Conversion from Infix to Prefix, and Postfix Notation

Infix Expression	Prefix Expression	Postfix Expression
$A / (B + C)$	?	?

Prefix Conversion:

$A / (B + C)$   
  
 $X = +BC$   
 $/AX = /A+BC$

Postfix Conversion:

$A / (B + C)$   
  
 $X = BC+$   
 $AX/= ABC+ /$

# Conversion from Infix to Prefix, and Postfix Notation

**Exp.** Convert the given Infix Expressions to their Prefix and Postfix equivalent.

Infix Expression	Prefix Expression	Postfix Expression
$A + B * C + D$		
$(A + B) * (C + D)$		
$A * B + C * D$		
$A + B + C + D$		

# Conversion from Infix to Prefix, and Postfix Notation

Convert the Infix Expressions to Prefix and Postfix Notation

Infix Expression	Prefix Expression	Postfix Expression
$A + B * C + D$	$++A*BCD$	$ABC*+D+$
$(A + B) * (C + D)$	$*+AB+CD$	$AB+CD+*$
$A * B + C * D$	$+*AB*CD$	$AB*CD*+$
$A + B + C + D$	$+++ABCD$	$AB+C+D+$

# Evaluate the Postfix Expression

- Evaluate from left to right. Find two operands and an operator.

Ex. 5 3 \* 6 8 2 / - +

15 6 4 - +

15 2 +

17

# Evaluate the Postfix Expression

- Use stacks to evaluate postfix expressions.

To evaluate a postfix expression process the expression left to right and do the following:

while more input

    get next item

    if the item is an operand

        push it in the operand stack

    if the item is an operator

        get the right operand from the operand stack (pop)

        get the left operand from the operand stack (pop)

        perform operation

        push the result onto the operand stack



# Evaluate the Postfix Expression

- Use stacks to evaluate postfix expressions.

## Push the operands into stack

If operator, pop 2 elements

Perform calculation and push the result into stack

Ex. Given the arithmetic expression in postfix notation, evaluate it and show the result.

5 3 \* 6 8 2 / - +

[illegible]

```

#include <iostream>
#include <string>
#include "LinkedStack.h"
using namespace std;

bool validDigit(char c){
    return (c>='0' && c<='9');
    //check if it is a digit
}

float eval(float a, float b, char c){
    switch(c){
        case '+': return a+b;
        case '-': return a-b;
        case '*': return a*b;
        case '/': return a/b;
    }
}

```

```

main(){
    float x,y;
    LinkedStack<float> st;
    char ch;
    string expr;
    cout<<"Enter an expression in Postfix:";
    cin>>expr;
    for(int i=0;i<expr.length();i++){
        ch=expr[i];
        if(validDigit(ch)){
            float val=float(ch-'0');
            st.push(val);
        }
        else{
            y=st.pop();
            x=st.pop();
            float res=eval(x,y,ch);
            st.push(res);
        }
    }

    cout << "Result=" << st.pop()<<endl;
}

```

# References

- CMPE226- Data Structures Lecture Notes by Cigdem Turhan
- Data Structures Using C++, D.S. Malik, Thomson Course Technology, 2nd Edition.
- Lecture Slides by Huamin Qu, The Hong Kong University of Science and Technology (2005)