21143 Tanmay Karmarkar

Infix to Postfix

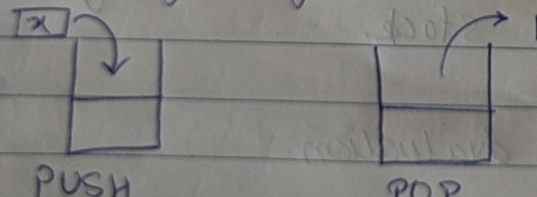
Problem statement:
Implement C++ program for expression conversion from infix to postfix, evaluation based on given conditions.

Objective: To
1. Implement a C++ program ~~from~~ for infix to postfix conversion.
2. A function to evaluate postfix expression

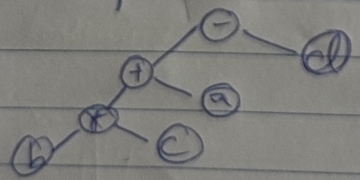
Outcome: Student will be able to
write & execute C++ program to convert infix to postfix & evaluate postfix expression.

S/W & H/W requirements:
Windows 10, VS CODE, mingw, 8GB RAM, 512GB SSD.

Theory: Stack follows LIFO rule. The element that enters last goes out first. It has 2 main operations push & pop.



Infix expression: a+b*c-d
Postfix expression: abc*+d-



Algorithm:

int pref (char c)

1. if return 1 if operator + or -.
2. Return 2 if operator * or /.
3. Return 3 if operator ^.
4. Return -1 if else.

Algorithm for infix to postfix.

String itf (string c)

1. Create a char stack St.
2. Create a string ret; which is empty.
3. Traverse through input string from start to end.
4. If s[i] is between "a" to "z" concatenate it to ret.
5. If s[i] = '(' push it to the stack.
6. If s[i] = ')' concatenate top element to ret until '(' is read
7. Else while stack is not empty prec (st.top()) >= prec (s[i]). concatanate top to string ret. pop after concatenation. else push it to stack.
8. After for loop, while stack is not empty concatanate the string ret & pop the stack.

Algorithm for postfix evaluation.

1. Create an int stack st.
2. Travel through the string. s.
3. If s[i] >= '0' && s[i] <= '9' push it in the stack. Else. int op2 = st.top();
   st.pop();
   int op1 = st.top();
   st.pop();

Create a switch statement.

4. For case '+' push (op1+op2) in stack
5. For '-' push (op1-op2) in stack.
6. For '*' push (op1*op2) in stack.
7. For '/' push (op1/op2) in stack.
8. return (st.top()).

Pseudocode

int prec(char c)
same as algorithm.

```
string itf (string c)
{ stack<char> st
string ret;
If (s[i] >='a' && s[i] <= 'z' )
{  ret += s[i]
}
If ( s[i] = '(')
  { st.push(s[i])
else if  (s[i] ==')')
{ while (!st.empty () && prec (st.top () >=prec (s[i]))
   ret += st.top();
    st.pop () }
} st.push (s[i])
while (!st.empty())
  ret += st.top()
  st.pop()
```

```
int peval(string c)
stack <int> st;
fo (int i=0; i<c.length(); i++)
 if( c[i]>='0' && c[i]<='9')
{  st.push(c[i]-'0')
}
else: int op2=st.top(),st.pop(); int op1=st.top;st.pop();
    switch(c[i])
     case '+': st.push(op1+op2)
   similar for other operators.

} return(st.top())
```