Rajvaibhav Rahane

17u283 223045

SE-C Comp,Viit,Pune

***Title-Implement an infix to postfix expression conversion code.***

***CODE:***

*The Code is divided into 3 file-*

1. *Stack.cpp,*
2. *queue.cpp,*
3. *sd1\_ass6\_postfix.cpp-UI File*

*UI File-*

/\*

\*@Rajvaibhav Rahane

\*/

/\*

\*Program to convert Expression from Infix to Postfix

\*Application of classes Stack and Queue

\*/

#include<iostream>

#include"queue.cpp"

#include"stack.cpp"

using namespace std;

int priorityOfOperator(char op){

switch(op)

{

case '#': return 0;

case '(':

case ')': return 1;

case '+':

case '-': return 2;

case '\*':

case '%':

case '/': return 3;

case '^': return 4;

}

}

int main(){

string infixExpression;

char inFixI;

cout<<"Enter Infix Expression\t";cin>>infixExpression;

Stack<char> operatorsStack(infixExpression.length());

Queue<char> postfixExpression(infixExpression.length());

//operatorsStack.push('#');

for(int i=0;infixExpression[i]!='\0';i++){

inFixI=infixExpression[i];

if(isalnum(inFixI)){

postfixExpression.enQueue(inFixI);

}else if(inFixI=='('){

operatorsStack.push(inFixI);

}else if(inFixI==')'){

while(operatorsStack.peek()!='('){

postfixExpression.enQueue(operatorsStack.pop());

}

inFixI=operatorsStack.pop(); //pop '(' in inFixI

}else{

while(!operatorsStack.isEmpty() && priorityOfOperator(inFixI)<=priorityOfOperator(operatorsStack.peek())){

postfixExpression.enQueue(operatorsStack.pop());

}

operatorsStack.push(inFixI);

}

}

while(!operatorsStack.isEmpty()){

postfixExpression.enQueue(operatorsStack.pop());

}

string separator="";

cout<<"Postfix Expression : ";postfixExpression.printQueue(separator);

return 0;

}

***Output:***

