**OPERATOR PRECEDENCE PARSING**

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<ctype.h>

#include<stdlib.h>

char str[20],stk[20],pstk[20],dstr[20];

int tos=-1,flag=0,ptr=0,rm=-1,i,j,n;

char q[9][9]={

{'>','>','<','<','<','<','>','<','>'},

{'>','>','<','<','<','<','>','<','>'},

{'>','>','>','>','<','<','>','<','>'},

{'>','>','>','>','<','<','>','<','>'},

{'>','>','<','<','<','<','>','<','>'},

{'<','<','<','<','<','<','=','<','E'},

{'>','>','>','>','>','E','>','E','>'},

{'>','>','>','>','>','E','>','E','>'},

{'<','<','<','<','<','<','E','<','A'},

};

char c[9]={'+','-','\*','/','^','a','(',')','$'};

//push operation

void pushin(char a)

{

tos++;

stk[tos]=a;

}

//pop operation

char popout()

{

char a;

a=stk[tos];

tos--;

return(a);

}

//finding which column and row

int find(char a)

{

switch(a)

{

case'+':return 0;

case'-':return 1;

case'\*':return 2;

case'/':return 3;

case'^':return 4;

case'(':return 5;

case')':return 6;

case'a':return 7;

case'$':return 8;

}

return-1;

}

// displaying shift operation

void display(char a)

{

printf("\n");

for(i=ptr;i<=strlen(str);i++)

printf("%c",str[i]);

printf("\t\t");

printf(" SHIFT %c\t\t",a);

for(i=0;i<=tos;i++)

printf("%c",stk[i]);

}

void display1(char a)

{

if(a!='(')

{

if(isalpha(a))

{

printf("\n");

for(i=ptr;i<=strlen(str);i++)

printf("%c",str[i]);

printf("\t\t");

printf(" REDUCE E--> %c\t\t",a);

for(i=0;i<=tos;i++)

printf("%c",stk[i]);

}

else if(a==')')

printf("\n REDUCE E-->(E)\t\t");

else

{

printf("\n");

for(i=ptr;i<=strlen(str);i++)

printf("%c",str[i]);

printf("\t\t");

printf(" REDUCE E-->E %c E\t",a);

for(i=0;i<=tos;i++)

printf("%c",stk[i]);

}

}//if

}

//relating stack contain with the table

int rel(char a,char b,char d)

{

if(isalpha (a))

a='a';

if(isalpha(b))

b='a';

if(q[find(a)][find(b)]==d)

return 1;

else

return 0;

}

void main()

{

clrscr();

printf("\n\n\t The productions used are:\n\t");

printf("E-->E\*E/E+E/E^E/E\*E/E-E\n\tE-->E/E \n\tE-->a/b/c/d/e.../z");

printf("\n\t Enter an expression that terminals with $:");

fflush(stdin);

i=-1;

//getting the string

while(str[i]!='$')

{

i++;

scanf("%c",&str[i]);

}

strcpy(dstr,str);

// if you encounter ( or ) perform nothing

for(j=0;j<i;j++)

{

if((str[j]=='(')||(str[j]==')')||(str[j+1]=='(')||(str[j+1]==')'))

{}

//if u don't enter any thing then print error

else

if(((isalpha(str[j])==0)&&(isalpha(str[j+1])==0))||((isalpha(str[j])!=0)&&(isalpha(str[j+1])!=0)))

{

printf("ERROR");

getch();

exit(0);

}

}

// if you enter an expression then proceed

if((((isalpha(str[0]))!=0)||(str[0]=='('))&&(((isalpha(str[i-1]))!=0)||(str[i-1]==')')))

{

pushin('$');

// printing the table

printf("\n\n\n\t+\t-\t\*\t/\t^\ta\t(\t)\t$\n\n");

for(i=0;i<9;i++)

{

printf("%c",c[i]);

for(j=0;j<9;j++)

printf("\t%c",q[i][j]);

printf("\n");

}

getch();

for(i=0;i<=35;i++)

printf("--");

printf("\nInputBuffer\tAction\t\t\tStack\n");

for(i=0;i<=35;i++)

printf("--");

//executing forever

while(1)

{

// if top of the stack and current input string are $ then accept and exit

if(str[ptr]=='$' && stk[tos]=='$')

{

for(i=0;i<=tos;i++)

printf("%c",stk[i]);

printf("\t\t");

for(i=ptr;i<=strlen(str);i++)

printf("%c",str[i]);

printf("\n\n\t Mr Tenzin Ur String is:ACCEPT!");

break;

}

//if stack top and input string is related by < or = then pushing input string into stack

else if(rel(stk[tos],str[ptr],'<')||rel(stk[tos],str[ptr],'='))

{

pushin(str[ptr]);

display(str[ptr]);

ptr++;

}

//if related by < then pop stack till the relation is > or = and reduce it

else if(rel(stk[tos],str[ptr],'>'))

{

do

{

rm++;

pstk[rm]=popout();

display1(pstk[rm]);

}

while(!rel(stk[tos],pstk[rm],'<'));

}

// other wise the string is not accepted

else

{

printf("\n\n\t Mr Sutheendran Ur String is:NOT ACCEPTED!!!!!!!");

getch();

exit(1);

}

}//end while(1)

getch();

}//end if

else

{

printf("ERROR");

getch();

}//end else

}//end main

Result: The program of implementation of operator precedenceparsing has been executed successfully.

**Output 1**

The productions used are:

E-->E\*E/E+E/E^E/E\*E/E-E

E-->E/E

E-->a/b/c/d/e.../z

Enter an expression that terminals with $:a+b\*c$

+ - \* / ^ a ( ) $

+ >><<<<><>

- >><<<<><>

\* >>>><<><>

/ >>>><<><>

^ >><<<<><>

a <<<<<< = < E

( >>>>> E > E >

) >>>>> E > E >

$ <<<<<< E < A

------------------------------------------------------------------------

InputBuffer Action Stack

------------------------------------------------------------------------

a+b\*c$ SHIFT a $a

+b\*c$ REDUCE E--> a $

+b\*c$ SHIFT + $+

b\*c$ SHIFT b $+b

\*c$ REDUCE E--> b $+

\*c$ SHIFT \* $+\*

c$ SHIFT c $+\*c

$ REDUCE E--> c $+\*

$ REDUCE E-->E \* E $+

$ REDUCE E-->E + E $$ $

Mr SutheendranUr String is:ACCEPT