# SSN College of Engineering, Kalavakkam

# Department of Computer Science and Engineering

# III Semester - CSE 'A ',’B’ & ‘C’

# UCS 1312 Data Structures Lab Laboratory

# Academic Year: 2019-2020 Batch: 2018-2022

# Exercise 4:Application of Stack

**Name:Rahul Ram M**

**Class:CSE - B**

**Register Number:185001121**

**• Create stack ADT as a header file “stack.h” .**

**• Convert the given infix expression into postfix expression by including “stack.h” in**

**the driver program to perform the conversion.**

**• Throw error for unbalanced symbols.**

**• Evaluate the postfix expression using “stack.h”.**

**• Test for the following expressions**

**a. (2+5) \* (3-6) / (7\*8)**

**b. 7 - (((3+2) \* (6+1)) / (5+6)**

**c. (((3+2) \* (2+5)**

**//stack.h**

typedef struct mystack

{

char data;

struct mystack \*next;

}stack;

typedef struct mystack2

{

float data;

struct mystack2 \*next;

}stack2;

int testexp(char exp[]);

stack\* push(char ele,stack \*top);

char pop(stack\*\* top);

int ismatching(char c1,char c2);

char \*postfix(char exp[]);

float evaluate(char exp[]);

float pop2(stack2\*\* top);

stack2\* push2(float ele,stack2 \*top);

int testexp(char exp[])

{

stack \*top;

top=NULL;

int i=0,s=1;

while(exp[i])

{

if(exp[i]=='('||exp[i]=='{'||exp[i]=='[')

{

top=push(exp[i],top);

}

else if(exp[i]=='}'||exp[i]==')'||exp[i]==']')

{

if(top!=NULL)

{

s=ismatching(pop(&top),exp[i]);

}

else

{

return 0;

}

if(s==0)

{

return 0;

}

}

i++;

}

if(top==NULL)

{

return 1;

}

else

{

return 0;

}

}

char \*postfix(char exp[])

{

stack \*top1;

top1=NULL;

char c;

char \*nexp=malloc(sizeof(char)\*50);

int i=0,n=0;

while(exp[i])

{

if(exp[i]=='('||exp[i]=='{'||exp[i]=='[')

{

top1=push(exp[i],top1);

}

else if(exp[i]=='+'||exp[i]=='-')

{

if(top1!=NULL)

{

if((top1->data)=='+'|| (top1->data)=='-' ||(top1->data)=='\*' ||(top1->data)=='/')

{

c=pop(&top1);

nexp[n]=c;

n++;

nexp[n]='\0';

top1=push(exp[i],top1);

}

else

{

top1=push(exp[i],top1);

}

}

else

{

top1=push(exp[i],top1);

}

}

else if(exp[i]=='\*' || exp[i]== '/')

{

if(top1!=NULL)

{

if(top1->data=='\*' || top1->data=='/')

{

c=pop(&top1);

nexp[n]=c;

n++;

nexp[n]='\0';

top1=push(exp[i],top1);

}

else

{

top1=push(exp[i],top1);

}

}

else

{

top1=push(exp[i],top1);

}

}

else if(exp[i]=='}'||exp[i]==')'||exp[i]==']')

{

c=pop(&top1);

nexp[n]=c;

n++;

nexp[n]='\0';

c=pop(&top1);

}

else

{

nexp[n]=exp[i];

n++;

nexp[n]='\0';

}

i++;

}

while(top1!=NULL)

{

c=pop(&top1);

nexp[n]=c;

n++;

nexp[n]='\0';

}

return nexp;

}

float evaluate(char exp[])

{

stack2 \*top2;

top2=NULL;

float temp;

float c,x;

int i=0;

while(exp[i])

{

if(exp[i]=='+'||exp[i]=='-'||exp[i]=='\*' || exp[i]== '/')

{

temp=pop2(&top2);

x=pop2(&top2);

if(exp[i]=='+')

temp=x+temp;

else if(exp[i]=='-')

temp=x-temp;

else if(exp[i]=='\*')

temp=x\*temp;

else

temp=x/temp;

top2=push2(temp,top2);

}

else

{

c=exp[i]-'0';

top2=push2(c,top2);

}

i++;

}

return temp;

}

stack\* push(char ele,stack \*top)

{

stack \*new;

new=(stack \*)malloc(sizeof(stack));

new->data=ele;

new->next=top;

top=new;

return top;

}

stack2\* push2(float ele,stack2 \*top)

{

stack2 \*new;

new=(stack2 \*)malloc(sizeof(stack2));

new->data=ele;

new->next=top;

top=new;

return top;

}

char pop(stack\*\* top)

{

char temp;

stack \*new;

new=(stack \*)malloc(sizeof(stack));

new=(\*top);

temp=new->data;

(\*top)=(\*top)->next;

free(new);

return temp;

}

float pop2(stack2\*\* top)

{

float temp;

stack2 \*new;

new=(stack2 \*)malloc(sizeof(stack2));

new=(\*top);

temp=new->data;

(\*top)=(\*top)->next;

free(new);

return temp;

}

int ismatching(char c1,char c2)

{

if(c1=='{'&&c2=='}')

return 1;

else if(c1=='('&&c2==')')

return 1;

else if(c1=='['&&c2==']')

return 1;

else

return 0;

}

**//main**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include"stack.h"

int main()

{

int ch=1,t;

char exp[50],postexp[50];

while(ch==1)

{

t=1;

printf("Enter the infix expression:");

scanf("%s",exp);

t=testexp(exp);

if(t==0)

{

printf("Unbalanced expression!\n");

}

else

{

printf("Balanced expression!\n");

strcpy(postexp,postfix(exp));

int j=0;

printf("Required postfix expression!\n");

while(postexp[j]!='\0')

{

printf("%c" ,postexp[j]);

j++;

}

printf("\n");

printf("Result:%.2f\n",evaluate(postexp));

}

printf("Continue(1.Yes 2.No):");

scanf("%d",&ch);

}

return 0;

}

**Sample I/P O/P:**

Enter the infix expression:(2+5)\*(3-6)/(7\*8)

Balanced expression!

Required postfix expression!

25+36-\*78\*/

Result:-0.38

Continue(1.Yes 2.No):1

Enter the infix expression:7-(((3+2)\*(6+1))/(5+6)

Unbalanced expression!

Continue(1.Yes 2.No):1

Enter the infix expression:7-((3+2)\*(6+1))/(5+6)

Balanced expression!

Required postfix expression!

732+61+\*56+/-

Result:3.82

Continue(1.Yes 2.No):1

Enter the infix expression:(((3+2)\*(2+5)

Unbalanced expression!

Continue(1.Yes 2.No):2