**SSN COLLEGE OF ENGINEERING (Autonomous)**

**Affiliated to Anna University**

**DEPARTMENT OF CSE**

**UCS 1312 Data Structures Lab Laboratory**

**EX5: APPLICATIONS OF STACK**

**=====================================================================================REGISTRATION NO: 185001112**

**NAME: PRATHYUSH S**

**CLASS: CSE-B (SEMESTER-3)**

**=====================================================================================Aim:**

To 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. And 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)

**Source Code:**

1. **stack.h**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

struct sNode

{

char data;

struct sNode \*next;

};

void pushi(struct sNode\*\* top\_ref, int new\_data)

{

struct sNode\* new\_node = (struct sNode\*) malloc(sizeof(struct sNode));

new\_node->data = new\_data;

new\_node->next = (\*top\_ref);

(\*top\_ref) = new\_node;

}

int popi(struct sNode\*\* top\_ref)

{

char res;

struct sNode \*top;

top = \*top\_ref;

res = top->data;

\*top\_ref = top->next;

free(top);

return res;

}

int match(char character1, char character2)

{

if (character1 == '(' && character2 == ')')

return 1;

else if (character1 == '{' && character2 == '}')

return 1;

else if (character1 == '[' && character2 == ']')

return 1;

else

return 0;

}

int balance(char exp[])

{

int i = 0;

struct sNode \*stack = NULL;

while (exp[i])

{

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

pushi(&stack, exp[i]);

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

{

if (stack == NULL)

return 0;

else if ( !match(popi(&stack), exp[i]) )

return 0;

}

i++;

}

if (stack == NULL)

return 1;

else

return 0;

}

struct Stack

{

int top;

unsigned capacity;

int\* array;

};

struct Stack\* createStack( unsigned capacity )

{

struct Stack\* stack = (struct Stack\*) malloc(sizeof(struct Stack));

if (!stack)

return NULL;

stack->top = -1;

stack->capacity = capacity;

stack->array = (int\*) malloc(stack->capacity \* sizeof(int));

if (!stack->array)

return NULL;

return stack;

}

int isEmpty(struct Stack\* stack)

{

return stack->top == -1 ;

}

char peek(struct Stack\* stack)

{

return stack->array[stack->top];

}

char pop(struct Stack\* stack)

{

if (!isEmpty(stack))

return stack->array[stack->top--] ;

return '$';

}

void push(struct Stack\* stack, char op)

{

stack->array[++stack->top] = op;

}

int isOperand(char ch)

{

return (ch >= '1' && ch <= '9');

}

int Prec(char ch)

{

switch (ch)

{

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '^':

return 3;

}

return -1;

}

int infixToPostfix(char\* exp)

{

int i, k;

struct Stack\* stack = createStack(strlen(exp));

if(!stack)

return -1 ;

for (i = 0, k = -1; exp[i]; ++i)

{

if (isOperand(exp[i]))

exp[++k] = exp[i];

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

push(stack, exp[i]);

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

{

while (!isEmpty(stack) && peek(stack) != '(')

exp[++k] = pop(stack);

if (!isEmpty(stack) && peek(stack) != '(')

return -1;

else

pop(stack);

}

else

{

while (!isEmpty(stack) && Prec(exp[i]) <= Prec(peek(stack)))

exp[++k] = pop(stack);

push(stack, exp[i]);

}

}

while (!isEmpty(stack))

exp[++k] = pop(stack );

exp[++k] = '\0';

printf( "%s", exp );

return 0;

}

int evaluatePostfix(char\* exp)

{

struct Stack\* stack = createStack(strlen(exp));

int i;

if (!stack) return -1;

for (i = 0; exp[i]; ++i)

{

if (isdigit(exp[i]))

{

push(stack, exp[i] - '0');

}

else

{

int val1 = pop(stack);

int val2 = pop(stack);

switch (exp[i])

{

case '+': push(stack, val2 + val1); break;

case '-': push(stack, val2 - val1); break;

case '\*': push(stack, val2 \* val1); break;

case '/': push(stack, val2/val1); break;

}

}

}

return pop(stack);

}

1. **postfix.c**

#include<stdio.h>

#include"stack.h"

int main()

{

char exp[100];

int ch;

do

{

printf("\nEnter the Expression:");

scanf(" %s",exp);

if (balance(exp))

printf("\nThe Equation is Balanced \n");

else

{

printf("The Equation is not Balanced \nError\n");

goto abcd;

}

printf("The postfix Expression is:");

infixToPostfix(exp);

printf("\n");

int ab=evaluatePostfix(exp);

printf("\nEvaluation of Postfix Expression:%d\n",ab);

abcd:

printf("\nEnter the Choice \n1.to continue\n0.Exit\t");

scanf("%d",&ch);

}while(ch==1);

return 0;

}

**OUTPUT**

gml36:Desktop cseb64$ gcc postfix.c -o postfix

gml36:Desktop cseb64$ ./postfix

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

The Equation is not Balanced

Error

Enter the Choice

1.to continue

0.Exit 1

Enter the Expression:(((3+2)\*(2+5)

The Equation is not Balanced

Error

Enter the Choice

1.to continue

0.Exit 1

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

The Equation is Balanced

The postfix Expression is:25+36-\*78\*/

Evaluation of Postfix Expression:0

Enter the Choice

1.to continue

0.Exit 1

Enter the Expression:(3+2)\*(2+5)

The Equation is Balanced

The postfix Expression is:32+25+\*

Evaluation of Postfix Expression:35

Enter the Choice

1.to continue

0.Exit 0