TITLE 37

Write a C program to convert the given in-fix expression to pre-fix expression using STACK

OBJECTIVE:

To convert the given in-fix expression to pre-fix expression using STACK

PROBLEM STATEMENT:

In this program we use STACK to convert a in-fix expression to pre-fix expression.

ALGORITHM:

START

INPUT: Read from the users

COMPUTATION: Computing how infix expression is converted to prefix expression using STACK

DISPLAY: Displaying the converted prefix expression

STOP

PROGRAM:

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<string.h>

#define SIZE 100

char stack[SIZE];

int top = -1;

void push(char c);

char pop();

int isoperator(char symbol);

int precedence(char symbol);

void InfixToPrefix(char infix\_exp[], char prefix\_exp[]);

void main()

{

char infix[SIZE], prefix[SIZE];

printf("\n\n Enter Infix expression : ");

gets(infix);

InfixToPrefix(infix,prefix);

printf("\n Prefix Expression: ");

puts(prefix);

}

void InfixToPrefix(char infix\_exp[], char prefix\_exp[])

{

int i, j, k, pos, len;

char item, x, rev[SIZE];

pos=0;

len=strlen(infix\_exp);

for(k=len-1;k>=0;k--)

{

rev[pos]=infix\_exp[k];

pos++;

}

rev[pos]='\0';

strcpy(infix\_exp,rev);

for(i=0; infix\_exp[i]!='\0'; i++)

{

if(infix\_exp[i] == ')')

infix\_exp[i] = '(';

else if(infix\_exp[i] == '(')

infix\_exp[i] = ')';

}

push('(');

strcat(infix\_exp,")");

i=0;

j=0;

item=infix\_exp[i];

while(item != '\0')

{

if(item == '(')

{

push(item);

}

else if( isdigit(item) || isalpha(item))

{

prefix\_exp[j] = item;

j++;

}

else if(isoperator(item) == 1)

{

x=pop();

while(isoperator(x) == 1 && precedence(x)>= precedence(item))

{

prefix\_exp[j] = x;

j++;

x = pop();

}

push(x);

push(item);

}

else if(item == ')')

{

x = pop();

while(x != '(')

{

prefix\_exp[j] = x;

j++;

x = pop();

}

}

else

{

printf("\nInvalid infix Expression.\n");

break;

}

i++;

item = infix\_exp[i];

}

if(top > 0)

printf("\n Invalid infix Expression.");

prefix\_exp[j] = '\0';

pos=0;

len=strlen(prefix\_exp);

for(k=len-1;k>=0;k--)

{

rev[pos]=prefix\_exp[k];

pos++;

}

rev[pos]='\0';

strcpy(prefix\_exp,rev);

}

void push(char c)

{

if(top >= SIZE-1)

printf("\n Stack Overflow.");

else

{

top++;

stack[top] = c;

}

}

// Define pop operation

char pop()

{

char c;

c='\0';

if(top < 0)

printf("\n Stack Underflow.");

else

{

c = stack[top];

top--;

}

return c;

}

int isoperator(char symbol)

{

if(symbol == '^' || symbol == '\*' || symbol == '/' || symbol == '+' || symbol == '-')

return 1;

else

return 0;

}

int precedence(char symbol)

{

if(symbol == '^')

return(5);

else if(symbol == '/')

return(4);

else if(symbol == '\*')

return(3);

else if(symbol == '+')

return(2);

else if(symbol == '-')

return(1);

else

return(0);

}

CONCLUSION:

This program helps us to understand the conversion of infix to pre-fix expression using STACK.

OUTPUT:

Enter Infix expression : (A+B\*C)

Prefix Expression: +A\*BC