**NAME: ASHWINI SHIVAJI RATHOD**

**CLASS:SE**

**DIV:3**

**Experiment no 3:Evaluation of postfix Expression using stack ADT**

**Aim:** Implementation of Evaluation of Postfix Expression using stack ADT

**Objective:**

1) Understand the use of stack

2) Understand importing an ADT in an application program

3) Understand the instantiation of stack ADT in an application Program

4) Understand how the member function of an ADT are accessed in an application program

**Theory:**

The code you provided implements a basic calculator using a stack-based approach to evaluate arithmetic expressions involving integers and the operators +, -, \*, and /. It utilizes the concept of a stack data structure to keep track of operands and perform the calculations in the correct order.

**Algorithm:**

1.Initialize an integer array ‘stack’ and an integer variable ‘top’ to keep track of the top of the stack. Initialize ‘top’ to -1.

2.Implement the ‘push()’ function to add an element to the stack.

3.Implement the ‘pop()’ function to retrieve and remove the top element from the stack.

4.In the ‘main()’ function, declare a character array 'exp' to hold the input expression and a character pointer 'e' to traverse the expression.

5.Prompt the user to input an arithmetic expression.

6.Traverse each character of the expression using the pointer ‘e’.

7.If the character is a digit, convert it to an integer and push it onto the stack.

8.If the character is an operator (+, -, \*, /), pop the top two elements from the stack, perform the respective operation, and push the result back onto the stack.

9.Continue the traversal until the end of the expression.

10.After the traversal, the final result will be left on the stack. Pop and print this result.

**Code :**

#include<stdio.h>

#include<ctype.h>

int stack[20];

int top = -1;

void push(int x)

{

stack[++top] = x;

}

int pop()

{

return stack[top--];

}

int main()

{

char exp[20];

char \*e;

int n1,n2,n3,num;

printf("Enter the expression :: ");

scanf("%s",exp);

e = exp;

while(\*e != '\0')

{

if(isdigit(\*e))

{

num = \*e - 48;

push(num);

}

else

{

n1 = pop();

n2 = pop();

switch(\*e)

{

case'+':

{

n3 = n1 + n2;

break;

}

case '-':

{

n3 = n2 - n1;

break;

}

case '\*':

{

n3 = n1 \* n2;

break;

}

case '/':

{

n3 = n2 / n1;

break;

}

}

push(n3);

}

e++;

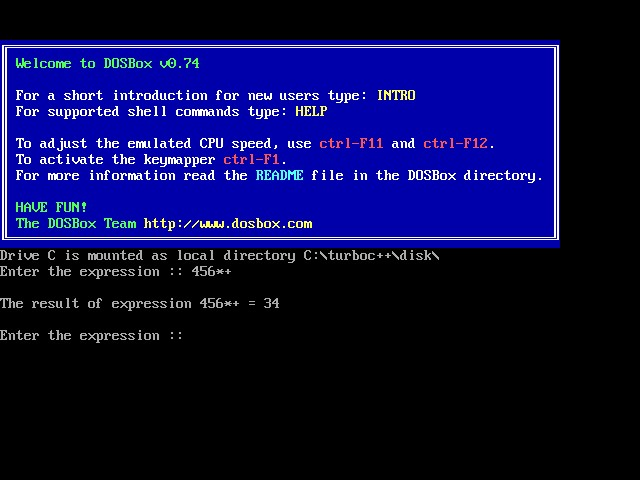
}

printf("\nThe result of expression %s = %d\n\n",exp,pop());

return 0;

}

**Output:**

****

**Conclusion :**

To evaluate a postfix expression we can use a stack. Iterate the expression from left to right and keep on storing the operands into a stack. Once an operator is received, pop the two topmost elements and evaluate them and push the result in the stack again.