NAME: GAURAV KISHOR PATIL

DIV: 2 ROLL NO:54

BATCH: C

Experiment No.3
Evaluate Postfix Expression using Stack ADT.
Date of Performance:
Date of Submission:

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 functions of an ADT are accessed in an application program

Theory:

An arithmetic expression consists of operands and operators. For a given expression in an postfix form, stack can be used to evaluate the expression. The rule is when ever an operands comes into the string push it on to the stack and when an operator is found then last two elements from the stack are poped and computed and the result is pushed back on to the stack. One by one whole string of postfix expression is parsed and final result is obtained at an end of computation that remains in the stack.

Algorithm

Algorithm : EVAL_POSTFIX

Input : E is an expression in an postfix form.

Output : Result after computing the expression.

Data Structure : An array representation of stack is used with top as a pointer to the top most element.

```
1.    Append # as an delimiter at an end of expression.
2.    item = READ_SYMBOL()
3.    while item != '#' do
      if item =operand then
            PUSH(item)
      else
            op=item
            y=POP()
            x=POP()
            t=x op y
            PUSH(t)
      end if
      item = READ_SYMBOL()
      end while
4.    value = POP()
5.    stop
```

Code:

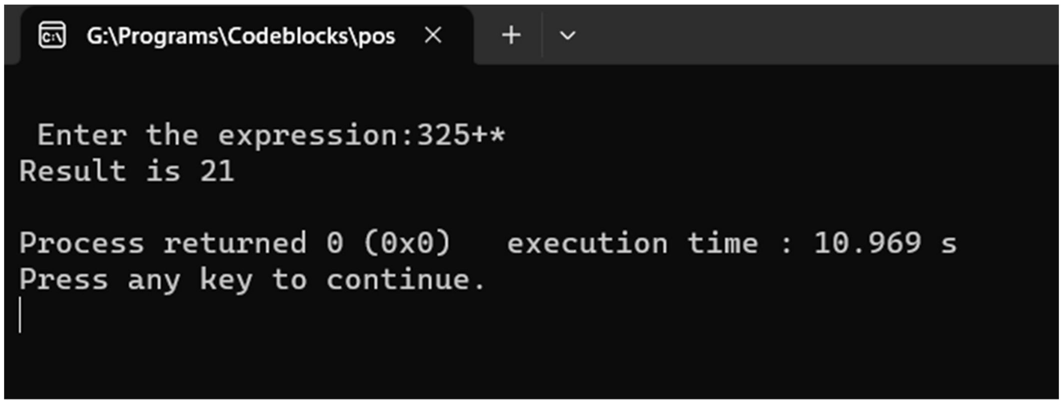
```
#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>


#define MAX 10


int stack[MAX];

char s[20];
```

```
int top, m;


void push(char);

char pop();

void evaluate();


int main()

{

    top = -1;

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

    gets(s);

    evaluate();

    return 0; // Return from main

}


char pop()

{

    if (top == -1)

    {

        printf("Stack is empty\n");

        return -1;

    }

    else

    {

        char ch = stack[top];

        top--;
        return ch;
    }
}

void push(char ch)
{
    if (top == MAX - 1)
    {
        printf("Stack is overflow\n");
    }
    else
    {
        top++;
        stack[top] = ch;
    }
}

void evaluate()
{
    int n1, n2, n3;
    int i = 0;
    n3 = 0; // Initialize n3

    while (s[i])
    {
        if (isdigit(s[i]))
```

```
{
   m = s[i] - '0';
   push(m);
}
else
{
   n1 = pop();
   n2 = pop();
   switch (s[i])
   {
   case '+':
      n3 = n2 + n1;
      break;
   case '-':
      n3 = n2 - n1;
      break;
   case '/':
      n3 = n2 / n1;
      break;
   case '*':
      n3 = n2 * n1;
      break;
   case '%':
      n3 = n2 % n1;
      break;
   default:
```

```
            printf("Unknown operator\n");
            return; // Exit the function instead of exiting the program
        }
        push(n3);
    }
    i++;
  }

  printf("Result is %d\n", n3);
}
```

Output:



```
 Enter the expression:325+*
Result is 21

Process returned 0 (0x0)   execution time : 10.969 s
Press any key to continue.
```