**QURESHI FAISAL**

**S3-13**

**BATCH A**

**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:**

Postfix Expression: A postfix expression, also known as Reverse Polish Notation (RPN), is a mathematical expression in which every operator follows all of its operands. For example, the infix expression "3 + 5" would be written in postfix as "3 5 +". Postfix notation eliminates the need for parentheses to indicate the order of operations.

Abstract Data Type (ADT) Stack: A stack is a fundamental data structure that follows the Last-In-First-Out (LIFO) principle. In other words, the last element added to the stack is the first one to be removed. An ADT stack typically supports two main operations: push, which adds an element to the top of the stack, and pop, which removes and returns the top element of the stack.

## Example for Evaluation of Postfix Expression: 456\*+**postfix evolutions through stack in C**

**Algorithm:**

1) Add ) to postfix expression.

2) Read postfix expression Left to Right until ) encountered

3) If operand is encountered, push it onto Stack

[End If]

4) If operator is encountered, Pop two elements

i) A -> Top element

ii) B-> Next to Top element

iii) Evaluate B operator A

push B operator A onto Stack

5) Set result = pop

6) END

**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++;

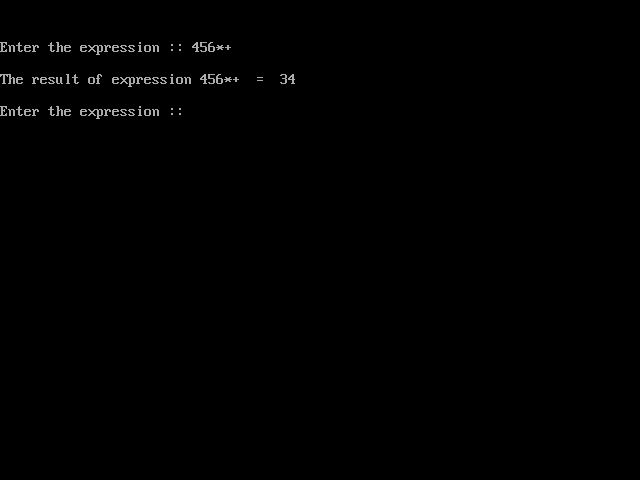
}

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

return 0;

}

**Output:**



**Conclusion :**

Evaluating a postfix expression using an ADT stack is a fundamental algorithm that showcases the practical application of stacks in computer science. This approach efficiently handles the evaluation of mathematical expressions and can be extended to support various operations and functions.