**EXPERIMENT NO 3**

~ ARYA RAUL

COMPS 20

**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 evaluation, also known as Reverse Polish Notation (RPN) evaluation, is a method of evaluating mathematical expressions without the use of parentheses. It involves scanning the expression from left to right and using a stack to keep track of operands and operators

No need for parentheses, reducing ambiguity.

Easily implementable using a stack data structure.

No need for operator precedence rules, as the order of operations is determined by the position of operators in the expression.

**Algorithm:**

1.#

2. item = READ\_Symbol()

3. while item != ‘#’ do

if item=operand then

push(item)

else

operand = item

y = pop()

x = pop()

t = xpopy

PUSH(t)

end if

item = READ Symbol()

end while

4. value = pop()

5. stop

**Code:**

#include<stdio.h>

#include<conio.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++;

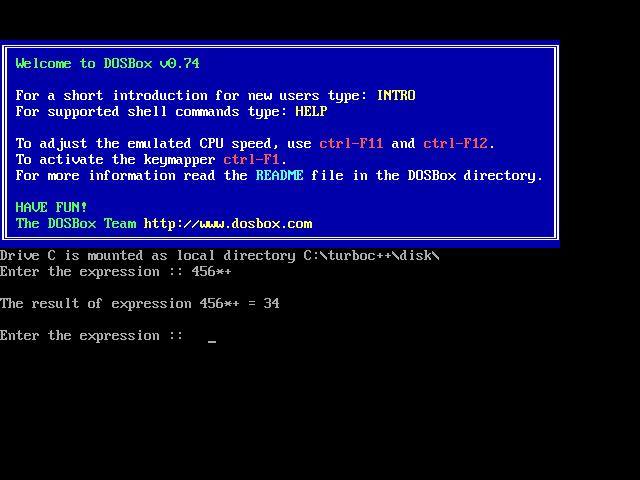
}

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

return 0;

}

**Output:**



**Conclusion:**

In conclusion, postfix evaluation is an efficient method for evaluating mathematical expressions by utilizing a stack-based approach. It eliminates the need for parentheses and follows a straightforward algorithm to process operators and operands sequentially. This approach ensures optimal time complexity and is widely used in various programming applications, offering a concise and effective solution for expression evaluation.