**NAME: SUVIDHI V. PAREEK**

**SE (COMPS) / DIV: 3 / ROLL NO. 10**

**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:**

* **"Stack-based postfix evaluation" Scan the string from left to right.**
* **When you come across an operand, place it on the stack; when you come across an operator, remove the corresponding operands from the stack, perform the operation, and place the result back on the stack.**
* ***ALGORITHM:***

***Step 1 − scan the expression from left to right***

***Step 2 − if it is an operand push it to stack***

***Step 3 − if it is an operator pull operand from stack and perform operation***

***Step 4 − store the output of step 3, back to stack***

***Step 5 − scan the expression until all operands are consumed***

***Step 6 − pop the stack and perform operation.***

***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++;***

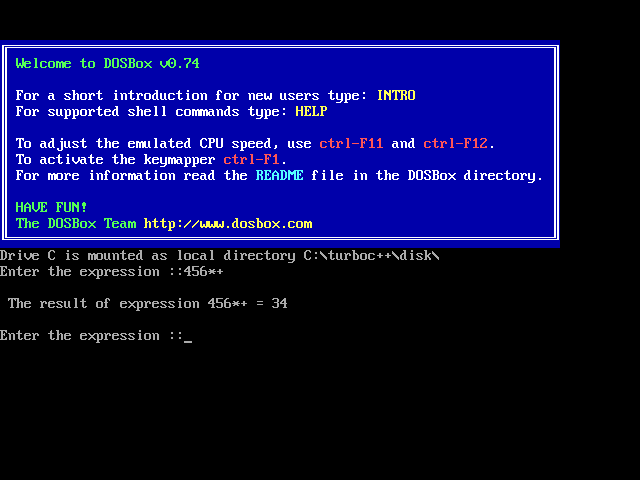
***}***

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

***return 0;***

***}***

***OUTPUT:***

******

***CONCLUSION:  
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.***