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:

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.

Algorithm:

Step 1: If a character is an operand push it to Stack.
Step 2: If the character is an operator. Pop two elements from the Stack. ...
Step 3: Step 1 and 2 will be repeated until the end has reached.
Step 4: The Result is stored at the top of the Stack, return it.
Step 5: End.

Code :

```
#include<stdio.h>
int stack[20];
int top = -1;
```

```c
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:

```
Output

/tmp/yDe09ILnSy.o
Enter the expression :: 456*+
The result of expression 456*+  =  34
```

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.