



**Vidyavardhini's College of Engineering and Technology**  
**Department of Artificial Intelligence & Data Science**

<b>Experiment No.2</b>
Convert an Infix expression to Postfix expression using stack ADT.
Name:Dhruv Gharat
Roll No:11
Date of Performance:
Date of Submission:
Marks:
Sign:

**Experiment No. 2: Conversion of Infix to postfix expression using stack ADT Aim:**  
**To convert infix expression to postfix expression using stack ADT.**

**Objective:**

- 1) Understand the use of Stack.
- 2) Understand how to import 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:**

Postfix notation is a way of representing algebraic expressions without parentheses or operator precedence rules. In this notation, expressions are evaluated by scanning them from left to right



## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

and using a stack to perform the calculations. When an operand is encountered, it is pushed onto the stack, and when an operator is encountered, the last two operands from the stack are popped and used in the operation, with the result then pushed back onto the stack. This process continues until the entire postfix expression is parsed, and the result remains in the stack.

Conversion of infix to postfix expression

Expression	Stack	Output
2	Empty	2
*	*	2
3	*	23
/	/	23*
(	/(	23*
2	/(	23*2
-	/(-	23*2
1	/(-	23*21
)	/	23*21-
+	+	23*21-/
5	+	23*21-/5
*	+*	23*21-/53
3	+*	23*21-/53
	Empty	23*21-/53*+

#### Algorithm:

##### Conversion of infix to postfix

Step 1: Add ")" to the end of the infix expression

Step 2: Push "(" on to the stack

Step 3: Repeat until each character in the infix notation is scanned

IF a "(" is encountered, push it on the stack

IF an operand (whether a digit or a character) is encountered, add it to the postfix expression. IF a ")" is encountered, then

a. Repeatedly pop from stack and add it to the postfix expression until a "(" is encountered.



## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

b. Discard the "(" . That is, remove the "(" from stack and do not add it to the postfix expression IF an operator o is encountered, then

a. Repeatedly pop from stack and add each operator (popped from the stack) to the postfix expression which has the same precedence or a higher precedence than o b. Push the operator o to the stack

[END OF IF]

Step 4: Repeatedly pop from the stack and add it to the postfix expression until the stack is empty Step 5: EXIT

#### Code:

```
#include<stdio.h>

#include<ctype.h>

char stack[100]; int
top = -1; void
push(char x) {
stack[++top] = x; }
char pop() { if(top
== -1); return -1;
else return
stack[top--]; } int
priority(char x) {
if(x == '(') return 0;
if(x == '+' || x == '-')
return 1; if(x == '*' ||
x == '/') return 2;
return 0; } int main()
{
char exp[100]; char *e, x; printf("Enter the expression : ");
scanf("%s",exp); printf("\n"); e = exp; while(*e != '\0') { if(isalnum(*e))
```



## Vidyavardhini's College of Engineering and Technology

### Department of Artificial Intelligence & Data Science

```
printf("%c ",*e); else if(*e == '(') push(*e); else if(*e == ')') { while((x
= pop()) != '(') printf("%c ", x); } else { while(priority(stack[top]) >=
priority(*e)) printf("%c ",pop()); push(*e); } e++; } while(top != -1) {
printf("%c ",pop()); } return 0;
}
```

#### Output:

```
/tmp/Uux1dFb17F.o
Enter the expression : 123-+76*
1 2 3 - 7 6 * + |
```

#### Conclusion:

Convert the following infix expression to postfix  $(A+(C/D))*B$

scanned	stack	pe
(	(	empty
A	(	A
+	+	A
(	+(	A
C	+(	AC
/	+(/	AC
D	+(/	ACD
)	+	ACD/



ACD/+B\*

1. Calculator software.
2. Programming language compilers.
3. Expression evaluation.
4. Mathematical software.
5. Spreadsheet programs.
6. Computer algebra systems.



**Vidyavardhini's College of Engineering and Technology**  
**Department of Artificial Intelligence & Data Science**

7. Calculator hardware.
8. Query languages.
9. Expression parsing.
10. Scientific and engineering simulations.