



PIMPRI CHINCHWAD EDUCATION TRUST'S.
PIMPRI CHINCHWAD COLLEGE OF ENGINEERING
(An Autonomous Institute)

S.Y. B. TECH

Year: 2024 – 25

Semester: I

Name: Abhishek Joshi

PRN: 123B1B150

Department: Computer Engineering

Division: C (C1)

Course: Data Structures Laboratory

Course Code: BCE23PC02

Date:

Assignment No -8

Aim: Write a program to convert infix expression to postfix, infix expression to prefix and evaluation of postfix and prefix expression.

a. Implement a restaurant waitlist system using the queue data structure. Restaurant waitlist provide following facility:

- Add Party to Waitlist
- Seat Party
- Display Waitlist

- **Source Code :**

```
#include <iostream>
using namespace std;

#define MAX 100

int precedence(char op) {
    if (op == '+' || op == '-') return 1;
    if (op == '*' || op == '/') return 2;
    return 0;
}

string infixToPostfix(string infix) {
    char stack[MAX];
    int top = -1;
    string postfix;

    for (char token : infix) {
        if (isalnum(token)) {
            postfix += token;
```

```

    } else if (token == '(') {
        stack[++top] = token;
    } else if (token == ')') {
        while (top != -1 && stack[top] != '(') {
            postfix += stack[top--];
        }
        top--;
    } else {
        while (top != -1 && precedence(token) <= precedence(stack[top])) {
            postfix += stack[top--];
        }
        stack[++top] = token;
    }
}

while (top != -1) {
    postfix += stack[top--];
}

return postfix;
}

```

```

string infixToPrefix(string infix) {
    string reversedInfix;
    for (int i = infix.length() - 1; i >= 0; i--) {
        if (infix[i] == '(') {
            reversedInfix += ')';
        } else if (infix[i] == ')') {
            reversedInfix += '(';
        } else {
            reversedInfix += infix[i];
        }
    }

    string postfix = infixToPostfix(reversedInfix);
    string prefix;

    for (int i = postfix.length() - 1; i >= 0; i--) {
        prefix += postfix[i];
    }

    return prefix;
}

```

```

int evaluatePostfix(string postfix) {
    int stack[MAX];
    int top = -1;

```

```

for (char token : postfix) {
    if (token >= '0' && token <= '9') {
        stack[++top] = token - '0';
    } else {
        int operand2 = stack[top--];
        int operand1 = stack[top--];
        switch (token) {
            case '+': stack[++top] = operand1 + operand2; break;
            case '-': stack[++top] = operand1 - operand2; break;
            case '*': stack[++top] = operand1 * operand2; break;
            case '/': stack[++top] = operand1 / operand2; break;
        }
    }
}

return stack[top];
}

int evaluatePrefix(string prefix) {
    int stack[MAX];
    int top = -1;

    for (int i = prefix.length() - 1; i >= 0; i--) {
        char token = prefix[i];
        if (token >= '0' && token <= '9') {
            stack[++top] = token - '0';
        } else {
            int operand1 = stack[top--];
            int operand2 = stack[top--];
            switch (token) {
                case '+': stack[++top] = operand1 + operand2; break;
                case '-': stack[++top] = operand1 - operand2; break;
                case '*': stack[++top] = operand1 * operand2; break;
                case '/': stack[++top] = operand1 / operand2; break;
            }
        }
    }

    return stack[top];
}

int main() {
    string infix, postfix, prefix;

    cout << "Enter infix expression: ";
    cin >> infix;

    postfix = infixToPostfix(infix);

```

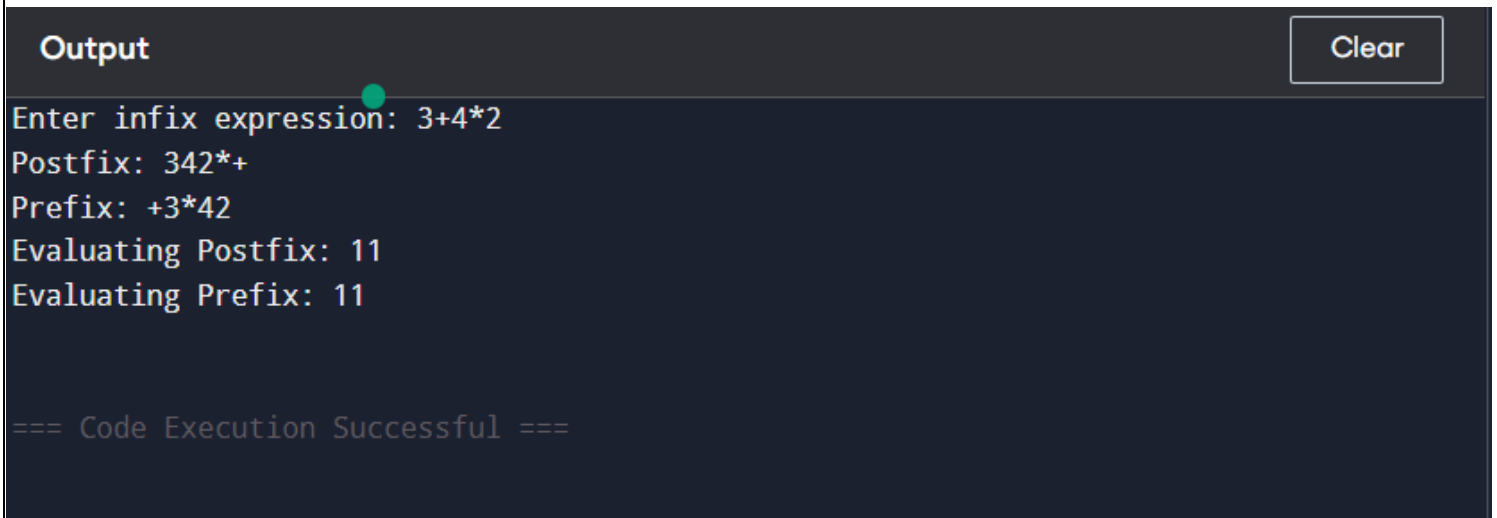
```
prefix = infixToPrefix(infix);

cout << "Postfix: " << postfix << endl;
cout << "Prefix: " << prefix << endl;

cout << "Evaluating Postfix: " << evaluatePostfix(postfix) << endl;
cout << "Evaluating Prefix: " << evaluatePrefix(prefix) << endl;

return 0;
```

Screen Shot of Output :



The screenshot shows a terminal window with a dark background. At the top, there is a header bar with the word "Output" on the left and a "Clear" button on the right. Below the header, the program's output is displayed in a monospaced font. The output shows the user entering the infix expression "3+4*2", followed by the calculated postfix expression "342*+", the prefix expression "+3*42", and the evaluation of both postfix and prefix expressions resulting in the value "11". At the bottom, a green message indicates that the code execution was successful.

```
Output Clear
Enter infix expression: 3+4*2
Postfix: 342*+
Prefix: +3*42
Evaluating Postfix: 11
Evaluating Prefix: 11

=== Code Execution Successful ===
```

Conclusion: we studied stack applications to convert infix expressions to postfix and prefix, and to evaluate them using a stack.

