```cpp
#include <iostream>

#include <stack>

#include <string>

#include <cctype>


using namespace std;


// Function to return precedence of operators

int precedence(char op) {

    switch (op) {

        case '+':

        case '-':

            return 1;

        case '*':

        case '/':

            return 2;

        default:

            return 0;

    }

}


// Function to check if the character is an operator

bool isOperator(char c) {

    return c == '+' || c == '-' || c == '*' || c == '/';

}


// Function to convert infix expression to postfix expression

string infixToPostfix(const string& infix) {

    stack<char> s;

    string postfix = "";


    for (char c : infix) {
        if (isalnum(c)) {
            postfix += c;
        } else if (c == '(') {
            s.push(c);
        } else if (c == ')') {
            while (!s.empty() && s.top() != '(') {
                postfix += s.top();
                s.pop();
            }
            if (!s.empty()) {
                s.pop();
            }
        } else if (isOperator(c)) {
            while (!s.empty() && precedence(s.top()) >= precedence(c)) {
                postfix += s.top();
                s.pop();
            }
            s.push(c);
        }
    }

    while (!s.empty()) {
        postfix += s.top();
        s.pop();
    }

    return postfix;
}

// Function to evaluate postfix expression
int evaluatePostfix(const string& postfix) {
    stack<int> s;
    for (char c : postfix) {
        if (isdigit(c)) {
            s.push(c - '0');
        } else if (isOperator(c)) {
            int val2 = s.top();
            s.pop();
            int val1 = s.top();
            s.pop();

            switch (c) {
                case '+':
                    s.push(val1 + val2);
                    break;
                case '-':
                    s.push(val1 - val2);
                    break;
                case '*':
                    s.push(val1 * val2);
                    break;
                case '/':
                    s.push(val1 / val2);
                    break;
            }
        }
    }

    return s.top();
}

int main() {
    string infix;
    cout << "Enter an infix expression: ";
    cin >> infix;

    string postfix = infixToPostfix(infix);
    cout << "Postfix expression: " << postfix << endl;

    int result = evaluatePostfix(postfix);
    cout << "Result of evaluation: " << result << endl;

    return 0;
}
```