```cpp
#include <iostream>
#include <stack>
#include <string>
#include <cctype>

using namespace std;

bool isOperator(char c) {
    return (c == '+' || c == '-' || c == '*' || c == '/');
}

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

string infixToPostfix(string infix) {
    stack<char> s;
    string postfix = "";

    for (int i = 0; i < infix.length(); i++) {
        char c = infix[i];

        if (isspace(c)) {
            continue;
        }

        if (isalnum(c)) {
            postfix += c;
        }
        else if (c == '(') {
            s.push(c);
        }
        else if (c == ')') {
            while (!s.empty() && s.top() != '(') {
                postfix += s.top();
                s.pop();
            }
            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;
}

void generateThreeAddressCode(string postfix) {
    stack<string> s;
    int tempCount = 1;
    for (int i = 0; i < postfix.length(); i++) {
        char c = postfix[i];

        if (isalnum(c)) {
            string operand(1, c);
            s.push(operand);
        }
        else if (isOperator(c)) {
            string op2 = s.top();
            s.pop();
            string op1 = s.top();
            s.pop();
            string temp = "t" + to_string(tempCount++);

            cout << temp << " = " << op1 << " " << c << " " << op2 << endl;
            s.push(temp);
        }
    }
}

int main() {
    string infix;

    cout << "Enter an arithmetic expression (infix): ";
    getline(cin, infix);

    string postfix = infixToPostfix(infix);
    cout << "\nPostfix Expression: " << postfix << endl;

    cout << "\nThree Address Code: " << endl;
    generateThreeAddressCode(postfix);

    return 0;
}
```

```
OUTPUT:
Enter an arithmetic expression (infix): e=a+b/c-d
Postfix Expression: eabc/+d-
Three Address Code:
t1 = b / c
t2 = a + t1
t3 = t2 - d
```