```
/*
* 输入: 前缀表达式
* 输出: 后缀表达式和计算结果
============================
* 输入样例:  + 1 2 =
* 样例输出:  1 2 +
*            3
*/

#include<iostream>
#include <string>
#include <stack>
#include <cstdlib>

using namespace std;

/* Calculator 类 */
class Calculator
{
private:
    string m_pfixExp;        // 后缀表达式
    void inToPost();         // 前缀表达式转化为后缀表达式
public:
    Calculator():m_pfixExp("") {}    // 构造对象的时候将后缀表达式的值设为空串
    void calPfixExp();       // 后缀表达式的计算
};

// 前缀转后缀
void Calculator::inToPost()
{
    char c;
```

```
stack<char> opStack;
cout << "Please input the infix expression, end with '=': " << endl;
while (cin.get(c))
{
    if (c != '=')
    {
        if ((c >= '0' && c <= '9') || c == ' ')
        {
            m_pfixExp.insert(m_pfixExp.end(), c);
        }
        else if (c == '(')
        {
            m_pfixExp.insert(m_pfixExp.end(), ' ');
            opStack.push(c);
        }
        else if (c == ')')
        {
            m_pfixExp.insert(m_pfixExp.end(), ' ');
            while (!opStack.empty() && opStack.top() != '(')
            {
                m_pfixExp.insert(m_pfixExp.end(), opStack.top());
                opStack.pop();
            }
            if (opStack.empty())
            {
                cout << "error! ')' is not matched!" << endl;
                exit(1);
            }
            else
            {
                opStack.pop();
```

```
            }
        }
        else if (c == '+' || c == '-' || c == '*' || c == '/')
        {
            m_pfixExp.insert(m_pfixExp.end(), ' ');
            while (!opStack.empty() &&
                   opStack.top() != '(' &&
                   ((opStack.top() == '*' || opStack.top() == '/') || c == '+' || c == '-
'))
            {
                m_pfixExp.insert(m_pfixExp.end(), opStack.top());
                opStack.pop();
            }
            opStack.push(c);
        }
        else
        {
            cout << "invalid character, ignore it!" << endl;
        }
    }
    // 遇等号结束
    else
    {
        while (!opStack.empty())
        {
            if (opStack.top() == '(')
            {
                cout << "error! '(' is not matched!" << endl;
                exit(1);
            }
            m_pfixExp.insert(m_pfixExp.end(), opStack.top());
```

```
                opStack.pop();
            }
            break;
        }
    }
}

// 后缀的计算
void Calculator::calPfixExp()
{
    inToPost();     //  先转换
    cout << "PostfixExpression: " << m_pfixExp << endl;
    stack<double> digitStack;
    double iTmp, opNum1, opNum2;
    string sTmp("");
    for (string::iterator itor = m_pfixExp.begin();
         itor != m_pfixExp.end(); itor++)
    {
        sTmp.clear();
        while (itor != m_pfixExp.end() && *itor >= '0' && *itor <= '9')
        {
            sTmp.insert(sTmp.end(), *itor);
            itor++;
        }
        if (!sTmp.empty())
        {
            iTmp = atof(sTmp.c_str());
            digitStack.push(iTmp);
        }
        if (itor != m_pfixExp.end() && *itor != ' ')
        {
```

```
if (digitStack.size() >= 2)
{
    opNum2 = digitStack.top();
    digitStack.pop();
    opNum1 = digitStack.top();
    digitStack.pop();
    switch (*itor)
    {
    case '*':
        digitStack.push(opNum1*opNum2);
        break;
    case '/':
        digitStack.push(opNum1/opNum2);
        if (opNum2 <= 1e-15 && opNum2 >= -1e-15)
        {
            cout << "error! 0 can't be divisor" << endl;
            exit(1);
        }
        break;
    case '+':
        digitStack.push(opNum1+opNum2);
        break;
    case '-':
        digitStack.push(opNum1-opNum2);
        break;
    default:
        break;
    }
}
else
{
```

```
                cout << "error! too much operators" << endl;
                exit(1);
            }
        }
        if (itor == m_pfixExp.end())
        {
            break;
        }
    }

    if (digitStack.size() != 1)
    {
        cout << "error! you need more operators" << endl;
        exit(1);
    }
    else
    {
        cout << "The answer: " << digitStack.top() <<endl;
        digitStack.pop();
    }
}

int main(int argc, char *argv[])
{
    Calculator test;    // 实例化 Calculator 对象
    test.calPfixExp();

    return 0;
}
```