表达式求值——栈的运用

```
#include<iostream>
#include<string>
#include<stack>
#include<cmath>
#include<iomanip>
using namespace std;
int getprior(char op)
{
    switch (op)
    {
    case '+':
    case '-':
        return 1;
    case '*':
    case '/':
    case '%':
        return 2;
    case '^':
        return 3;
    case '(':
        return 0;
    case '#':
        return -1;
    }
}
double calculate(double a,char op,double b)
{
    double c;
    switch (op)
    {
    case '+':
    {c = b + a; break; }
    case '-':
    {c = b - a; break; }
    case '*':
    {c = b * a; break; }
    case '/':
    {c = b / a; break;}
    case '%':
    {c = (int)b % (int)a; break; }
    case '^':
    {c = pow(b, a); break; }
```

```
    }
    return c;
}
int main()
{
    string str;
    cin >> str;
    stack<double> stack1_num;
    stack<char> stack2_op;
    stack2_op.push('#');
    for (int i = 0; i < str.length(); )
    {
        if ((int)str[i] >= 0 && (int)str[i] <= 9)
        {
            stack1_num.push((double)str[i]);
        }
        else if (str[i] == '(')
        {
            stack2_op.push('(');
            i++;
        }
        else if (str[i] == ')')
        {
            i++;
            while (stack2_op.top() != '(' && stack2_op.size() > 1)
            {
                double a = stack1_num.top();
                stack1_num.pop();
                double b = stack1_num.top();
                stack1_num.pop();
                char op = stack2_op.top();
                stack2_op.pop();
                double result = calculate(a, op, b);
                stack1_num.push(result);
            }
            stack2_op.pop();
            if (stack2_op.size() < 1)
            {
                cout << "ERROR IN INFIX NOTATION" << endl;
                return 0;
            }
        }
        else if (str[i] == '+' || str[i] == '-' || str[i] == '*' || str[i] == '/' || str[i]
== '^' || str[i] == '%')
```

```
        {
            while (getprior(str[i]) <= getprior(stack2_op.top()) && stack1_num.size() >=
2 && str[i] != '^')
            {
                double a = stack1_num.top();
                stack1_num.pop();
                double b = stack1_num.top();
                stack1_num.pop();
                char op = stack2_op.top();
                stack2_op.pop();
                double result = calculate(a, op, b);
                stack1_num.push(result);
            }
            stack2_op.push(str[i]);
            i++;
        }
        else
        {
            cout << "ERROR IN INFIX NOTATION" << endl;
            return 0;
        }
    }
    while (stack1_num.size() >= 2)
    {
        double a = stack1_num.top();
        stack1_num.pop();
        double b = stack1_num.top();
        stack1_num.pop();
        char op = stack2_op.top();
        stack2_op.pop();
        double result = calculate(a, op, b);
        stack1_num.push(result);
        stack2_op.pop();
    }
    stack2_op.pop();
    if (!stack2_op.empty())
    {
        cout << "ERROR IN INFIX NOTATION" << endl;
        return 0;
    }
    else if (stack1_num.size() != 1)
    {
        cout << "ERROR IN INFIX NOTATION" << endl;
        return 0;
```

```
        }
        else
        {
            cout << fixed << setprecision(2) <<stack1_num.top() << endl;
            return 0;
        }

        return 0;
}
```