ACM ICPC Team Reference

Jeferson

August 7, 2015

Mathematical Expression Solver

```
//Solver for mathematical expressions
void doOp(stack<double> &num, stack<char> &op){
        double A = num.top(); num.pop();
        double B = num.top(); num.pop();
        char oper = op.top(); op.pop();
        double ans;
        if(oper == '+'){
                ans = A+B;
        }else if(oper == '-'){
                ans = B-A;
        }else if(oper == '*'){
                ans = A*B;
        }else{
                if(A != 0){
                        ans = B/A;
                }else{
                        //division by 0
                        ans = -1;
                }
        }
        num.push(ans);
}

double parse(string s){
    stack<char> op;
    stack<double> num;
    map<char,int> pr;

    //setting the priorities, greater values with higher pr
    pr['+'] = 0;
    pr['-'] = 0;
    pr['*'] = 1;
    pr['/'] = 1;

    for (int i = 0; i < s.size(); i++){
        if (s[i] == ')'){
            while(!op.empty() && op.top() != '('){
                doOp(num,op);
            }
            op.pop();
        } else if(s[i] == '('){
                        op.push('(');
        } else if(!(s[i] >= '0' && s[i] <= '9')){
            while(!op.empty() && pr[s[i]] <= pr[op.top()] && op.top() != '('){
                doOp(num,op);
            }
            op.push(s[i]);
        } else {
            double ans = 0;
            while(i < s.size() && s[i] >= '0' && s[i] <= '9'){
                ans = ans * 10 + (s[i] - '0');
                i++;
            }
            i--;
            num.push(ans);
        }
    }
    while (op.size()) {
        doOp(num,op);
    }
    return num.top();
}
```