1.字符串

计算器问题

- 224. 基本计算器
- 227. 基本计算器 II
- 770. 基本计算器 IV
- 三叶通用题解

```
class Solution {
public:
    //添加一个INT_MIN的原因：运算符遇到(，应该先入栈，当前括号内有优先级更高的则计算，低的
    unordered_map<char,int> oper_pri = { {'+',1}, {'-',1}, {'*',2}, {'/',2}, {'%
    stack<char> opers;
    stack<long long> nums;
    int calculate(string s) {
        //处理所有空格
        string temp = "";
        for(char c : s){
            if(c != ' ') temp += c;
        }
        s = temp;

        //-除了减，还有一元运算符负的概念
        //所以：避免-5的处理，0-5更加通用
        nums.push(0);
        int n = s.size();
        for(int i = 0; i < n; i++){
            //处理左括号
            if(s[i] == '(') {
                opers.push('(');
                //-除了减，还有一元运算符负的概念
                //所以：避免-5的处理，0-5更加通用
                if(s[i + 1] == '-'){
                    i++;
                    nums.push(0);
                    opers.push('-');
                }
            }
            //处理右括号，要把当前括号中没计算完的计算完
            else if(s[i] == ')'){
                while(opers.top() != '('){
                    cal();
                }
                opers.pop();
            }
            //处理数字，要读就一次性把数字读完
            else if(s[i] >= '0' && s[i] <= '9'){
                int l = i;
                while(i + 1 < n && s[i + 1] >= '0' && s[i + 1] <= '9') i++;
                nums.push(stoll(s.substr(l,i - l + 1)));
            }
            //处理运算符：每次出现了优先级更低的运算符，才去算前一个优先级高的运算符，否则
```

```
            else{
                while(!opers.empty() && oper_pri[opers.top()] >= oper_pri[s[i]])
                    cal();
                }
                opers.push(s[i]);
            }
        }
        //计算所有剩余的运算法
        while(!opers.empty() && opers.top() != '('){
            cal();
        }
        return nums.top();
    }
    void cal(){
        long long b = nums.top(); nums.pop();
        long long a = nums.top(); nums.pop();
        char oper = opers.top(); opers.pop();
        long long res;
        switch (oper) {
            case '+':   res = a + b; break;
            case '-':   res = a - b; break;
            case '*':   res = a * b; break;
            case '/':   res = a / b; break;
            case '%':   res = a % b; break;
            case '^':   res = pow(a, b); break;
        }
        nums.push(res);
    }
};
```