user: sakura

第46题——包含min函数的栈

定义栈的数据结构，请在该类型中实现一个能够得到栈的最小元素的 min 函数在该栈中，调用 min、push 及 pop 的时间复杂度都是 O(1)。

示例:

MinStack minStack = new MinStack(); minStack.push(-2); minStack.push(0); minStack.push(-3); minStack.min(); --> 返回 -3. minStack.pop(); minStack.top(); --> 返回 0. minStack.min(); --> 返回 -2.

辅助栈(1)

> 因为我们的时间复杂度是O(1),又需要获得最小元素,那么最好的办法就是在入栈的时候把最小值存下来,然后调用min的时候我妈嗯返回这个最小值即可.
>
> 具体怎么做呢,我们用到一个辅助栈,在初始化时存入一个int型元素的最大值
>
> 每一次入栈,我们也要给辅助栈入栈,当然,我们需要判断要入栈的值x和辅助栈的栈顶元素的大小,如果 x< min_stack.top()成立 ;那我们将这个最小值入栈即可.如果不成立,那我们入栈辅助栈的最小值即可,如此一来,辅助栈的长度始终为数据栈的长度+1,并且栈顶元素永远都是最小值.
>
> 出栈时我们让两个栈都出栈一个元素即可

```
class MinStack {
public:
    /** initialize your data structure here. */
    MinStack() {
        min_stack.push(INT_MAX);
    }
    void push(int x) {
        data_stack.push(x);
        if(x<min_stack.top())
        {
            min_stack.push(x);
        }
        else
        {
            min_stack.push(min_stack.top());
        }
    }

    void pop() {
     data_stack.pop();
     min_stack.pop();
    }
```

```
    int top() {
        return data_stack.top();
    }

    int min() {
        return min_stack.top();
    }
private:
    stack<int> data_stack;
    stack<int> min_stack;
};

/**
 * Your MinStack object will be instantiated and called as such:
 * MinStack* obj = new MinStack();
 * obj->push(x);
 * obj->pop();
 * int param_3 = obj->top();
 * int param_4 = obj->min();
 */
```