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.
```

解题思路：特殊栈的建立 需要额外一个栈去维护，保持特殊栈的属性

```
class MinStack {
public:
    /** initialize your data structure here. */
    stack<int> normal,minval;
    MinStack() {

    }

    void push(int x) {
        normal.push(x);
        if(minval.empty())
        {
            minval.push(x);//如果 minval 中为空 则直接放入
        }
        else
        {
            if(x<=minval.top())//如果 minval 中不为空，此时需要比较新放入元素和原有栈顶元素大小
            {
                minval.push(x);//如果放入栈的值比 minval 栈顶小，则更新栈顶元素
            }
            else
```

```
            {
                minval.push(minval.top());//如果放入栈的值比 minval 栈顶元素
大，则重复放入栈顶元素
            }
        }
    }

    void pop() {
        normal.pop();
        minval.pop();
    }

    int top() {
        return normal.top();
    }

    int min() {
        return minval.top();
    }
};

/**
 * 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();
 */
```