栈操作

一个数组实现三个栈

```
class TripleInOne {
private:
    int *stack;
    int N = 3;
    int top[3]={0,0,0};
    int stackSize;
public:
    TripleInOne(int stackSize) {
        stack = new int[stackSize * N];
        this->stackSize = stackSize;
        for (int i = 0; i < N; i++) {
            top[i] = i * this->stackSize;
        }
    }
    void push(int stackNum, int value) {
        int idx = top[stackNum];
        if (idx < (stackNum + 1) * stackSize) {
            stack[idx] = value;
            top[stackNum]++;
        }
    }

    int pop(int stackNum) {
        if (this->isEmpty(stackNum)) {
            return -1;
        }
        int idx = top[stackNum];
        top[stackNum]--;
        return stack[idx - 1];
    }
    int peek(int stackNum) {
        if (this->isEmpty(stackNum)) {
            return -1;
        }
        int idx = top[stackNum];
        return stack[idx - 1];
    }
    bool isEmpty(int stackNum) {
        return top[stackNum] == stackNum * this->stackSize;

    }
};
```

栈的最小值

> 常数时间求出stack的最小值

```
class MinStack {
private:
    stack<int> x_stack;
    stack<int> minStack;
public:
    MinStack() {
        minStack.push(INT_MAX);

    }
    void push(int x) {
        x_stack.push(x);
        minStack.push(min(minStack.top(), x));
    }
    void pop() {
        x_stack.pop();
        minStack.pop();
    }
    int top() {
        return x_stack.top();
    }
    int getMin() {
        return minStack.top();
    }
};
```