# 题目

实现一个MyQueue类，该类用两个栈来实现一个队列。

**示例：**

MyQueue queue = new MyQueue();

queue.push(1);

queue.push(2);

queue.peek(); // 返回 1

queue.pop(); // 返回 1

queue.empty(); // 返回 false

**说明：**

你只能使用标准的栈操作--也就是只有push to top, peek/pop from top, size 和is empty操作是合法的。

你所使用的语言也许不支持栈。你可以使用list或者deque（双端队列）来模拟一个栈，只要是标准的栈操作即可。

假设所有操作都是有效的（例如，一个空的队列不会调用pop或者peek操作）。

# 分析

**思路：**

我们用stack1、stack2来表示两个栈：

入队的话就先入栈 stack1；

出队的话先判断stack2是否为空，如果为空就把stack1中的元素全部压入 stack2，因为stack1顶部的元素是后进入的，压入stack2之后stack2顶部的元素就是先进入的，所以可以放心出队了，如果stack2不为空，那么就出栈stack2就好了。

判定是否为空就直接看看stack1和stack2是不是都为空就好了。

peek操作和pop操作类似，只不过只需要返回队列头部元素就好了。所以可以把pop和peek合并一下。

**代码：**

class MyQueue {

stack<int> s1, s2;

public:

/\*\* Initialize your data structure here. \*/

MyQueue() {

}

/\*\* Push element x to the back of queue. \*/

void push(int x) {

s1.push(x);

}

/\*\* Removes the element from in front of queue and returns that element. \*/

int pop() {

int val = peek();

s2.pop();

return val;

}

/\*\* Get the front element. \*/

int peek() {

if(s2.empty()){

while(!s1.empty()){

int val = s1.top(); s1.pop();

s2.push(val);

}

}

return s2.top();

}

/\*\* Returns whether the queue is empty. \*/

bool empty() {

return s1.empty() && s2.empty();

}

};

/\*\*

\* Your MyQueue object will be instantiated and called as such:

\* MyQueue\* obj = new MyQueue();

\* obj->push(x);

\* int param\_2 = obj->pop();

\* int param\_3 = obj->peek();

\* bool param\_4 = obj->empty();

\*/