232-用栈实现队列

题述

232. 用栈实现队列

难度 简单 589

请你仅使用两个栈实现先入先出队列。队列应当支持一般队列支持的所有操作（`push`、`pop`、`peek`、`empty`）：

实现 `MyQueue` 类：

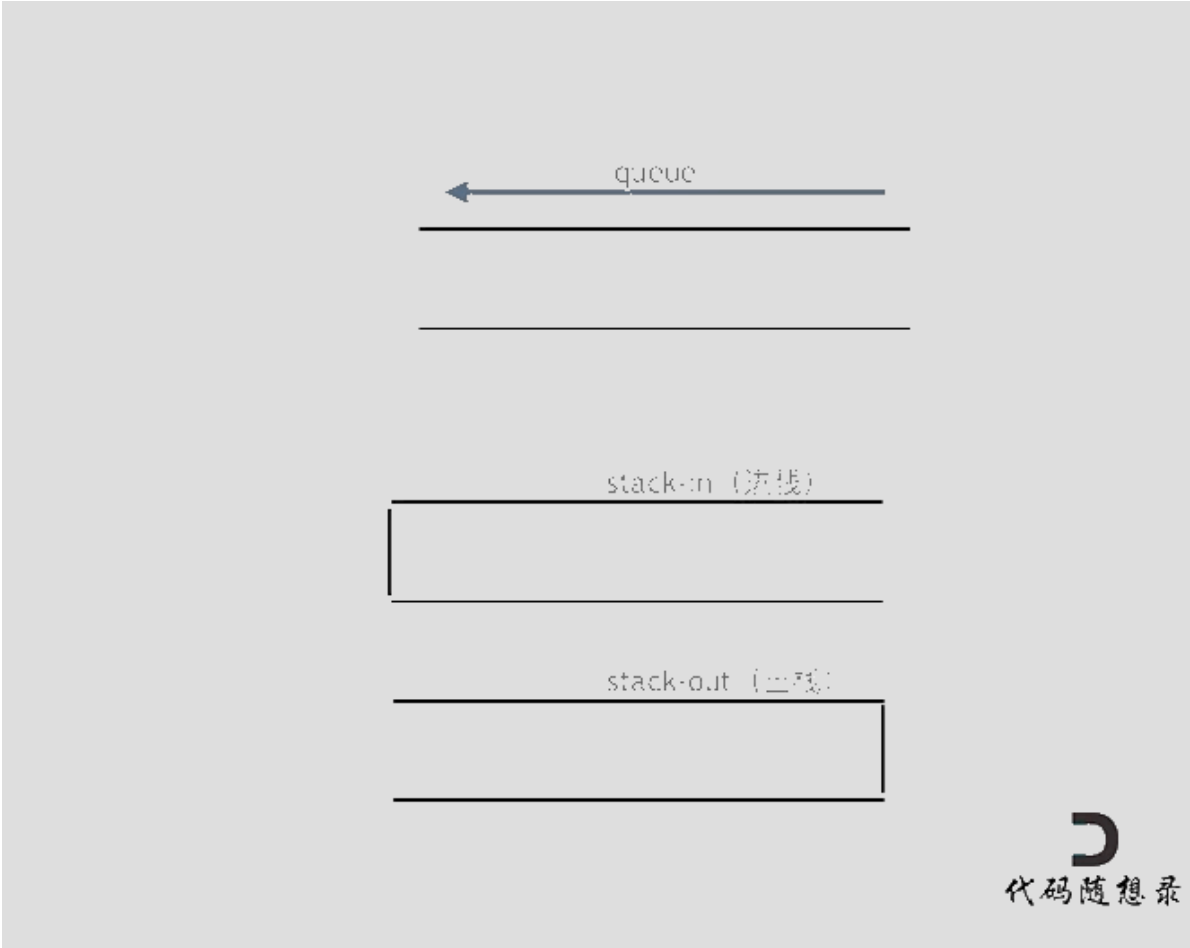
- `void push(int x)` 将元素 x 推到队列的末尾
- `int pop()` 从队列的开头移除并返回元素
- `int peek()` 返回队列开头的元素
- `boolean empty()` 如果队列为空，返回 `true` ；否则，返回 `false`

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。

思路

我们使用了两个栈，一个是输入栈，一个是输出栈



在push数据的时候，只要数据放进输入栈就好，**但在pop的时候，操作就复杂一些，输出栈如果为空，就把进栈数据全部导入进来（注意是全部导入）**，再从出栈弹出数据，如果输出栈不为空，则直接从出栈弹出数据就可以了。

题解

C++

```
class MyQueue {
public:
    stack<int> stIn;
    stack<int> stOut;

    //构造函数
    MyQueue()
    {

    }

    //入队
    void push(int x)
    {
        stIn.push(x);
    }

    int pop()
    {
        //stOut为空时，才从stIn中导入数据
        if(stOut.empty())
        {
```

```
            while(!stIn.empty())
            {
                 // 从stIn导入数据直到stIn为空
                stOut.push(stIn.top());
                stIn.pop();
            }
        }
        int result = stOut.top();
        stOut.pop();
        return result;
    }

    int peek()
    {
        int result=this->pop();
        stOut.push(result);    //peek操作只取值不弹出
        return result;
    }

    bool empty()
    {
        if(stIn.empty()&&stOut.empty())
        {
            return true;
        }
        else
        {
            return false;
        }
    }
};

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

Python

```python
class MyQueue:

    def __init__(self):
        """
        使用两个栈 stIn和stOut
        stIn负责push操作
        stOut负责pop操作
        """
        self.stIn = []
        self.stOut = []

    def push(self, x: int) -> None:
        self.stIn.append(x)


    def pop(self) -> int:
        if self.empty():
            return None
        if self.stOut:
            return self.stOut.pop()
        else:
            for i in range(len(self.stIn)):
                self.stOut.append(self.stIn.pop())
            return self.stOut.pop()


    def peek(self) -> int:
        res = self.pop()
        self.stOut.append(res)
        return res
```

```
    def empty(self) -> bool:
        return not (self.stIn or self.stOut)


# Your MyQueue object will be instantiated and called as such:
# obj = MyQueue()
# obj.push(x)
# param_2 = obj.pop()
# param_3 = obj.peek()
# param_4 = obj.empty()
```

思考

学会复用代码