

第13题——用两个栈实现队列

用两个栈实现一个队列。队列的声明如下，请实现它的两个函数 `appendTail` 和 `deleteHead`，分别完成在队列尾部插入整数和在队列头部删除整数的功能。（若队列中没有元素，`deleteHead` 操作返回 -1）

示例 1：

输入：

```
["CQueue","appendTail","deleteHead","deleteHead"]
[[],[3],[],[]]
```

输出： `[null,null,3,-1]`

提示： $1 \leq \text{values} \leq 10000$ 最多会对 `appendTail`、`deleteHead` 进行 10000 次调用

C语言

```
typedef struct {
    int stackIn[500]; // 入栈
    int topIn; // 入栈的栈顶指针
    int stackOut[500]; // 出栈
    int topOut; // 出栈的栈顶指针
} CQueue;

// 初始化队列
CQueue* cQueueCreate() {
    CQueue *obj = malloc(sizeof(CQueue));
    obj->topIn = -1;
    obj->topOut = -1;
    return obj;
}

// 入队
void cQueueAppendTail(CQueue* obj, int value) {
    obj->stackIn[++obj->topIn] = value;
}

// 出队
int cQueueDeleteHead(CQueue* obj) {
    if(obj->topIn == -1 && obj->topOut == -1)
    {
        return -1; // 队列空返回-1
    }
    else if(obj->topOut == -1) // 出栈为空
    {
        while(obj->topIn > 0) // 将入栈元素逆序放进出栈，再返回入栈的栈顶元素
        {
```

```

        obj->stackOut[++obj->topOut]=obj->stackIn[obj->topIn--];
    }
    return obj->stackIn[obj->topIn--];
}
else//出栈不为空
{
    return obj->stackOut[obj->topOut--];
}
}

//释放内存空间
void cQueueFree(CQueue* obj) {
    free(obj);
}

/**
 * Your CQueue struct will be instantiated and called as such:
 * CQueue* obj = cQueueCreate();
 * cQueueAppendTail(obj, value);
 *
 * int param_2 = cQueueDeleteHead(obj);
 *
 * cQueueFree(obj);
 */

```

C++

```

class CQueue {
    stack<int> stack_1,stack_2;//定义两个栈
public:
    CQueue() {
        //如果栈不为空，那就把栈中所有元素弹出
        while (!stack_1.empty()) {
            stack_1.pop();
        }
        while (!stack_2.empty()) {
            stack_2.pop();
        }
    }

    void appendTail(int value) {
        stack_1.push(value);//将元素压入栈中
    }

    int deleteHead() {
        // 如果第二个栈为空
        if (stack_2.empty()) {
            while (!stack_1.empty()) {
                stack_2.push(stack_1.top());
                stack_1.pop();
            }
        }
    }
}

```

```
};  
  
}  
  
if (stack_2.empty()) {  
    return -1;  
} else {  
    int num = stack_2.top();//取出栈顶元素  
    stack_2.pop();//出栈  
    return num;//返回栈顶元素  
}  
};
```