Lesson4--栈和队列

【本节目标】

- 1.栈
- 2.队列
- 3.栈和队列面试题

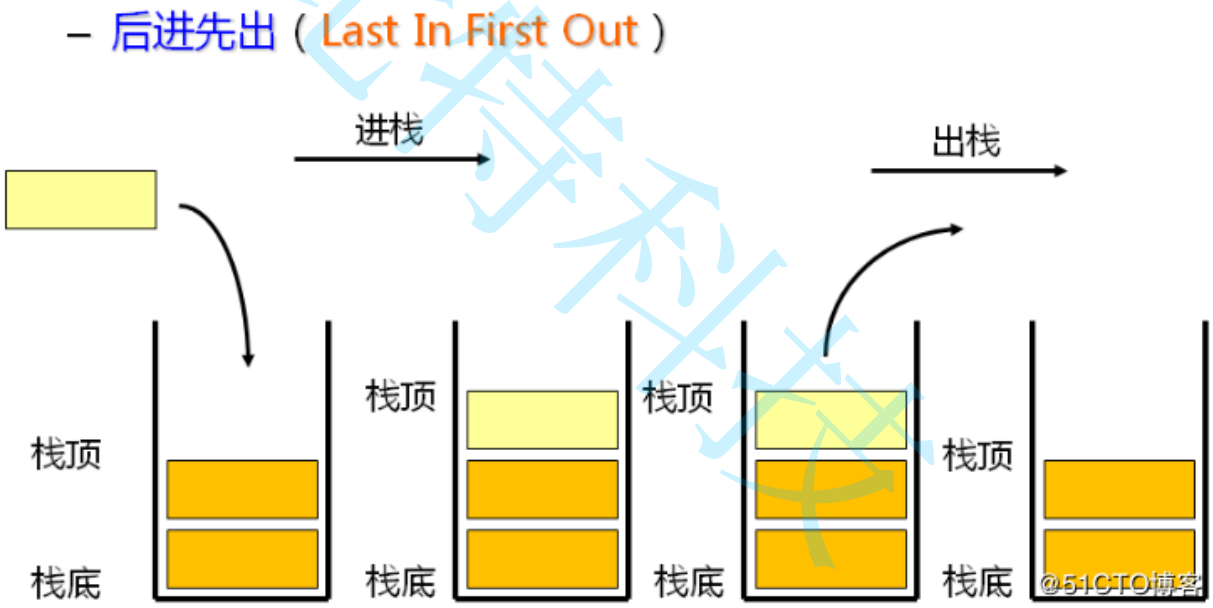
1.栈

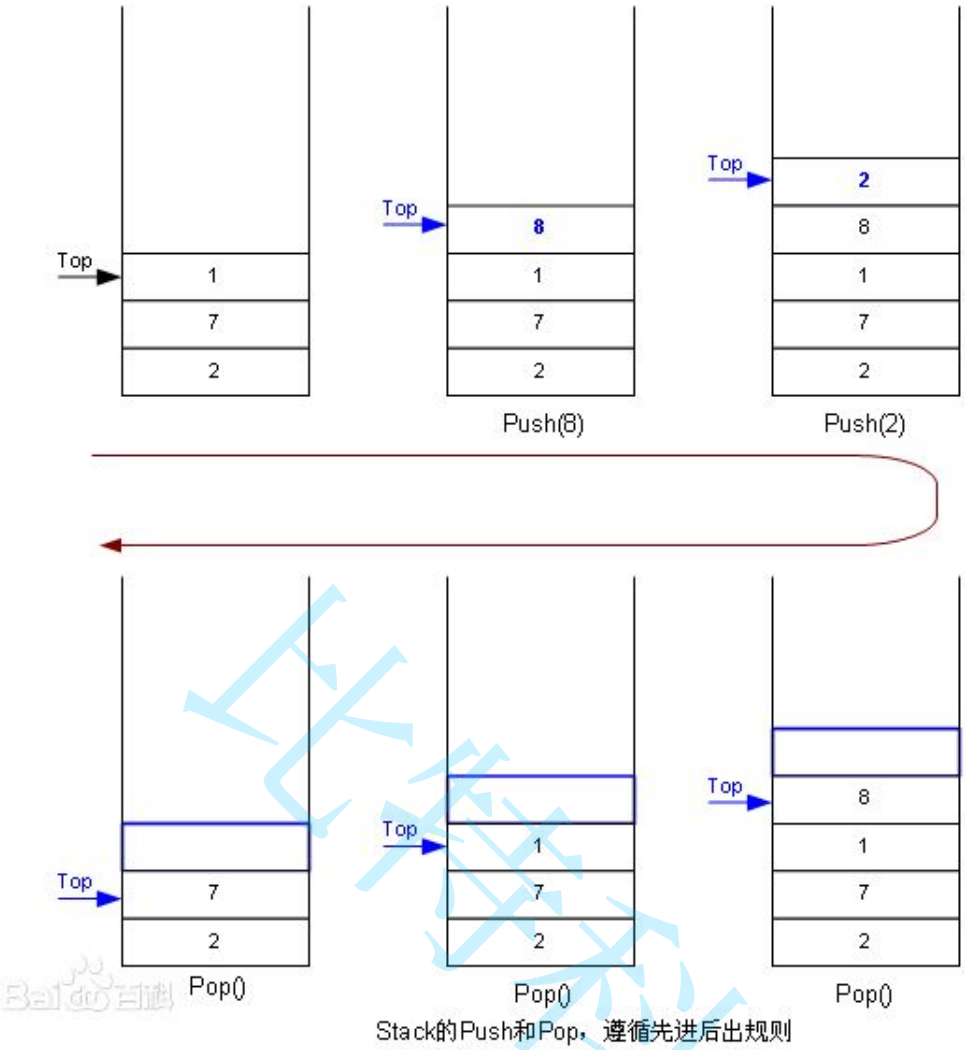
1.1栈的概念及结构

栈：一种特殊的线性表，其只允许在固定的一端进行插入和删除元素操作。**进行数据插入和删除操作的一端称为栈顶，另一端称为栈底。**栈中的数据元素遵守后进先出LIFO（Last In First Out）的原则。

压栈：栈的插入操作叫做进栈/压栈/入栈，**入数据在栈顶。**

出栈：栈的删除操作叫做出栈。**出数据也在栈顶。**

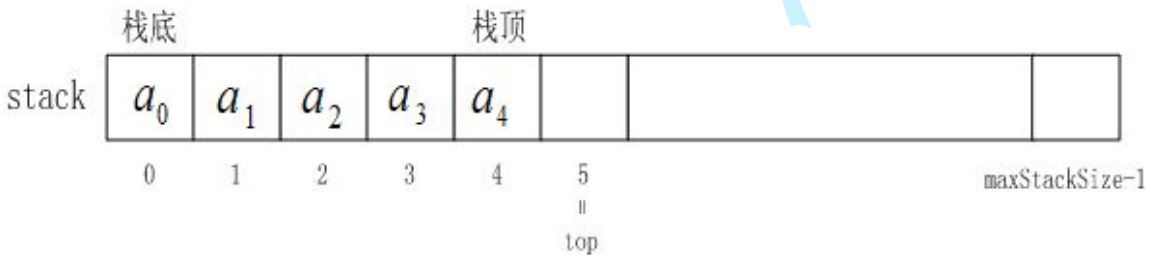




Stack的Push和Pop，遵循先进后出规则

1.2栈的实现

栈的实现一般可以使用**数组或者链表实现**，相对而言数组的结构实现更优一些。因为数组在尾上插入数据的代价比较小。



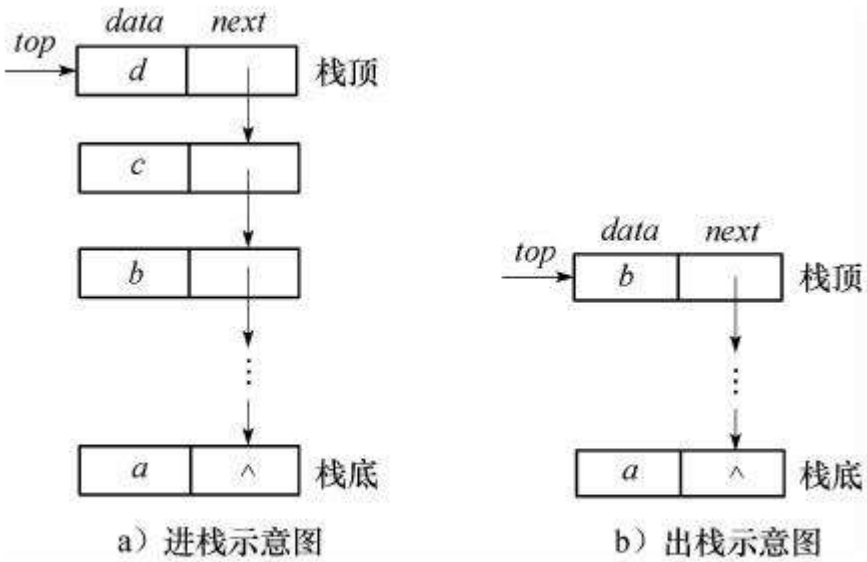


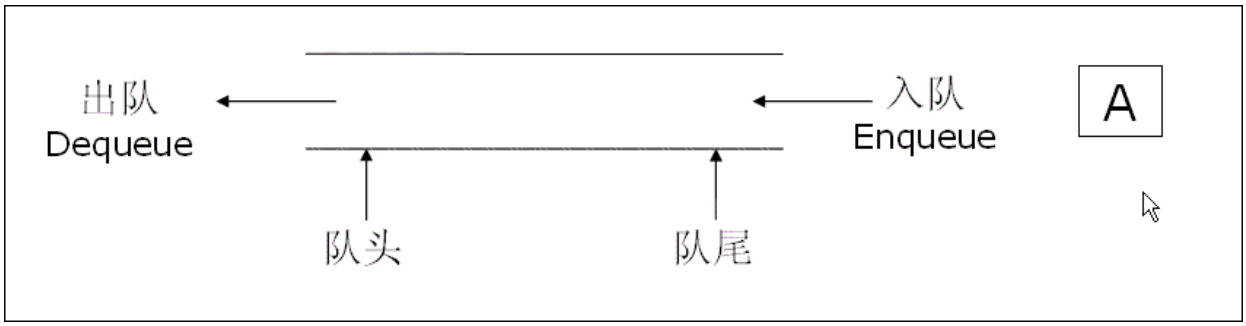
图 2-10 栈的进出栈示意图

```
// 下面是定长的静态栈的结构，实际中一般不实用，所以我们主要实现下面的支持动态增长的栈
typedef int STDataType;
#define N 10
typedef struct Stack
{
    STDataType _a[N];
    int _top; // 栈顶
}Stack;

// 支持动态增长的栈
typedef int STDataType;
typedef struct Stack
{
    STDataType* _a;
    int _top;       // 栈顶
    int _capacity;  // 容量
}Stack;

void StackInit(Stack* ps);
void StackDestory(Stack* ps);

void StackPush(Stack* ps, STDataType x);
void StackPop(Stack* ps);
STDataType StackTop(Stack* ps);
int StackEmpty(Stack* ps);
int StackSize(Stack* ps);

void TestStack();
```

2.队列

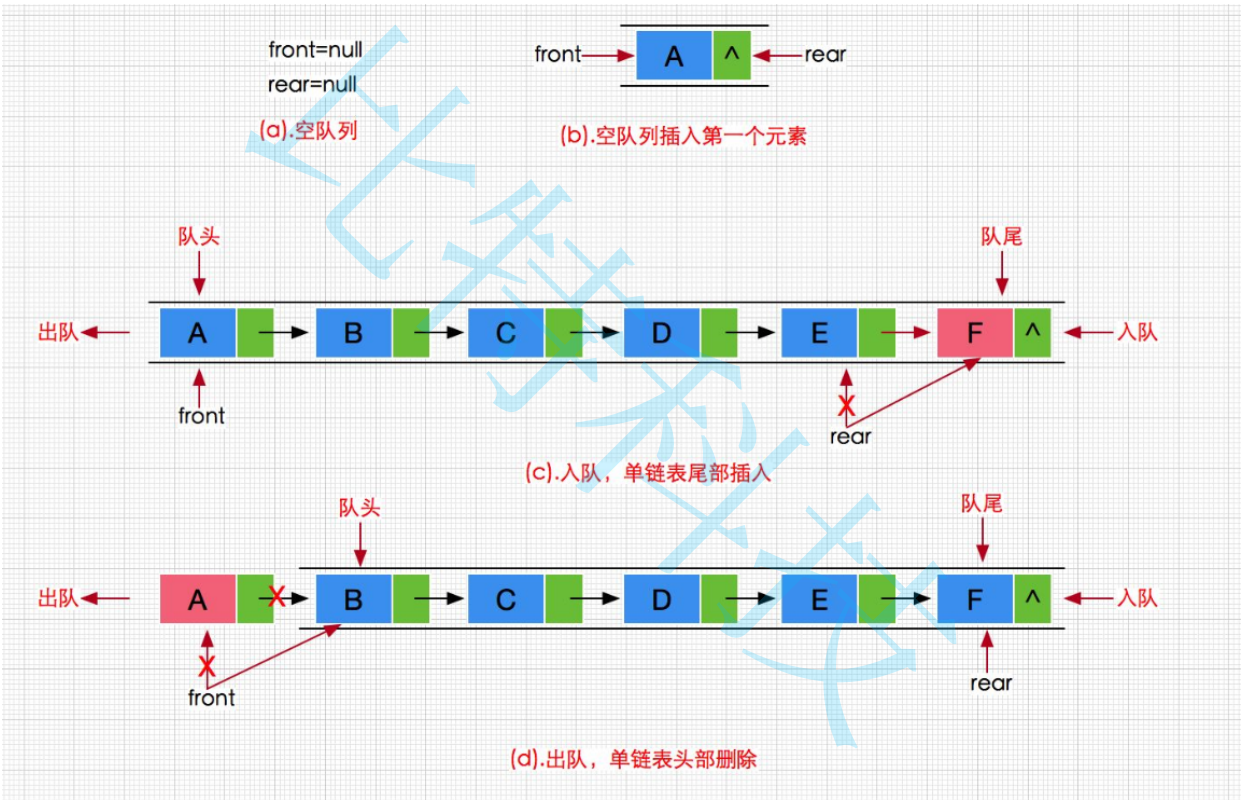
2.1队列的概念及结构

队列：只允许在一端进行插入数据操作，在另一端进行删除数据操作的特殊线性表，队列具有先进先出FIFO(First In First Out) 入队列：进行插入操作的一端称为**队尾** 出队列：进行删除操作的一端称为**队头**



2.2队列的实现

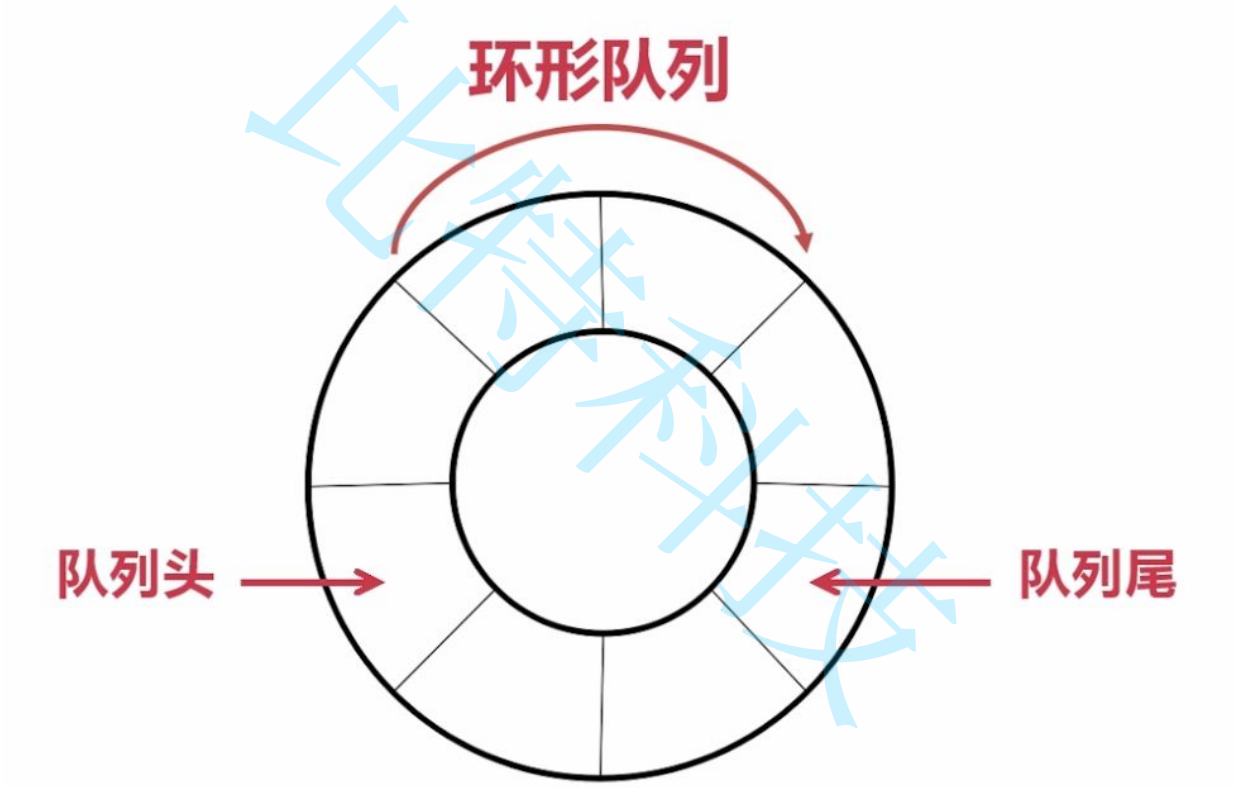
队列也可以数组和链表的结构实现，使用链表的结构实现更优一些，因为如果使用数组的结构，出队列在数组头上出数据，效率会比较低。

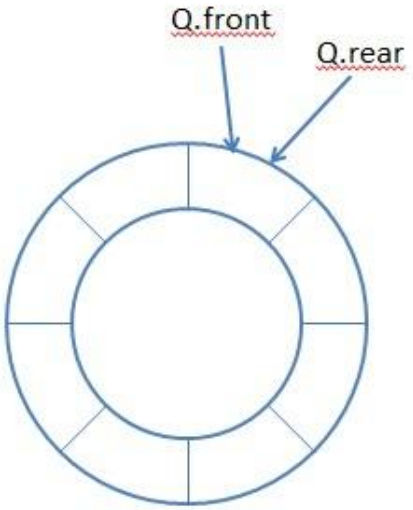



```
typedef int QUDataType;

typedef struct QueueNode
{
    struct QueueNode* _next;
    QUDataType _data;
}QueueNode;

typedef struct Queue
{
    QueueNode* _front; // 队头
    QueueNode* _rear;  // 队尾

}Queue;
```

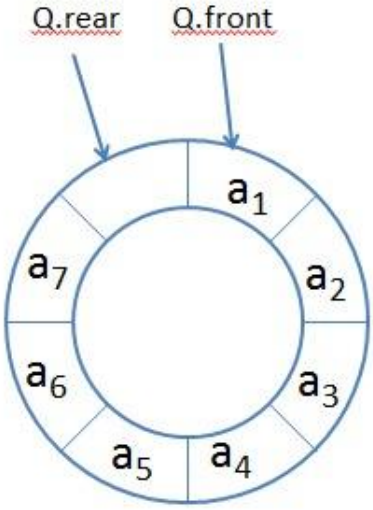
```
void QueueInit(Queue* pq);
void QueueDestory(Queue* pq);

QueueNode* BuyQueueNode(QUDataType x);
void QueuePush(Queue* pq, QUDataType x);
void QueuePop(Queue* pq);
QUDataType QueueFront(Queue* pq);
QUDataType QueueBack(Queue* pq);
int QueueEmpty(Queue* pq);
int QueueSize(Queue* pq);

void TestQueue();
```

另外扩展了解一下，实际中我们有时还会使用一种队列叫循环队列。如操作系统课程讲解生产者消费者模型时可以就会使用循环队列。环形队列可以使用数组实现，也可以使用循环链表实现。





（a）空的循环队列



为了能使用Q.rear=Q.front来区别是队空还是队满，我们常常认为出现左图时的情况即为队满的情况，此时：rear+1=front

（b）满的循环队列

3.栈和队列面试题

1. 括号匹配问题。OJ链接
2. 用队列实现栈。OJ链接
3. 用栈实现队列。OJ链接
4. 实现一个最小栈。OJ链接
5. 设计循环队列。OJ链接