0.前言

在上节课的内容中，我们详细学习了线性存储结构的内容，也就是数组和链表的内容

数组和链表作为一种底层的数据结构，通过对他们的进一步封装，实际上是可以得到跟多其他的数据结构的

这一节我们就即将讨论一下两种基于数组和链表实现的高级封装的数据结构：栈和队列

首先需要声明的是：栈和队列的学习重点在于对其结构特性的掌握，至于栈和队列的底层是通过数组实现还是链表实现，这个事情大家可以不必关心

因为同属于线性存储结构，我们既可以通过数组实现一个栈结构，也可以通过链表实现一个栈结构，队列结构也是亦然。

数组和链表与栈和队列的排列组合实现关系：

	栈	队列
数组	数组栈	数组队列
链表	链表栈	链表队列

1.栈结构

①栈结构的特点：元素先进后出

栈的结构特性是：元素先进后出（First In Last Out，简称FILO）

也就是说，在我们向栈结构中添加元素后，如果将元素从栈结构中取出，那么元素出来的顺序和元素放入栈结构的顺序正好是相反的

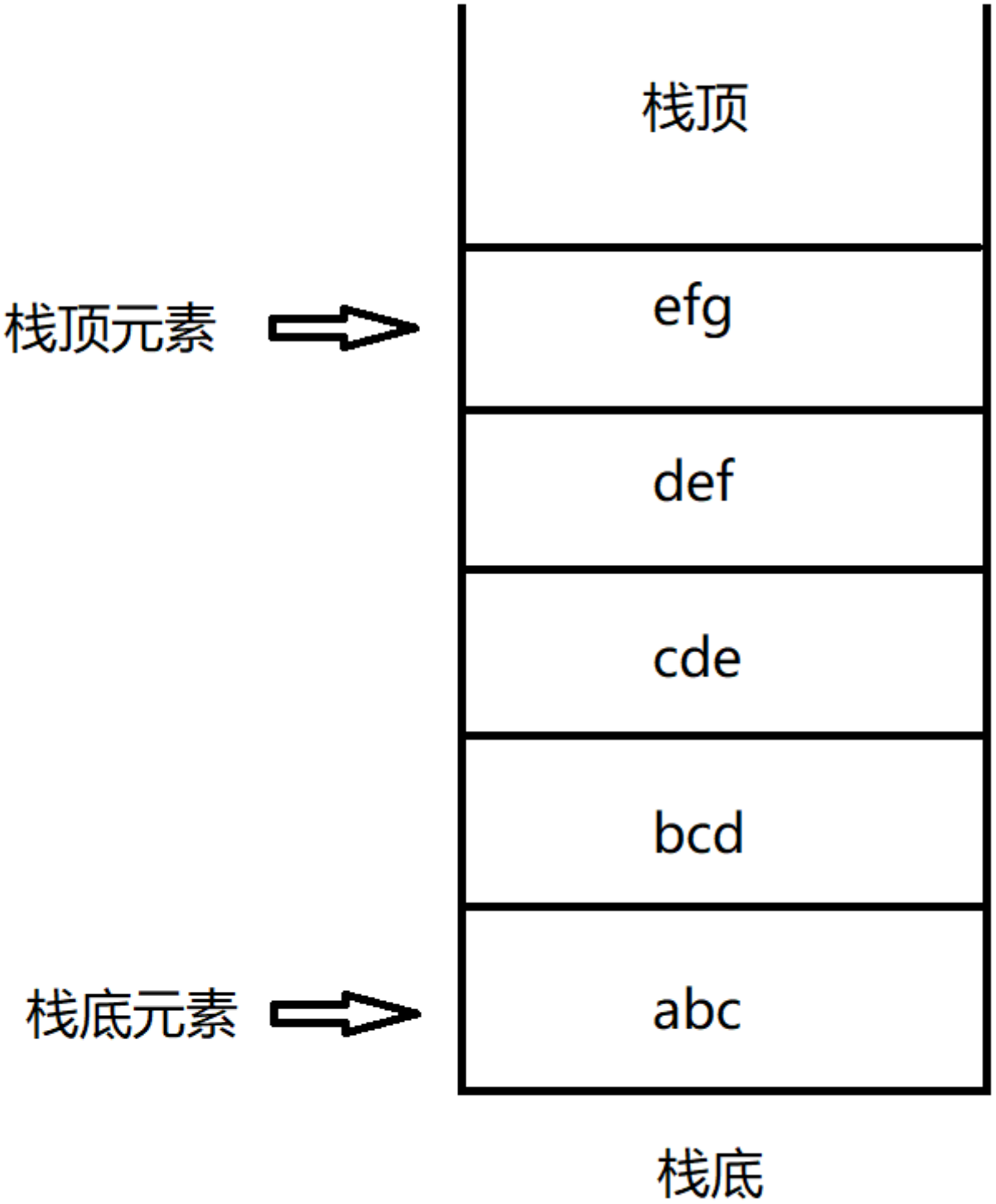
我们将元素放入栈结构中的操作称之为元素入栈

将元素从栈结构中取出的操作称之为元素出栈

将最先进入栈结构中的元素称之为栈底元素，这个方向同时称之为栈底

将最后进入栈结构中的元素称之为栈顶元素，这个方向同时称之为栈顶

栈结构图如下：



打个比方，大家都从影视剧中见过手枪的弹夹，通常我们第一颗压入弹夹的子弹将会被最后一个被打出去；反之最后一颗被压入弹夹的子弹会被第一个打出去

这也是为什么很多程序员将元素入栈也称之为压栈，不得不说这个说法还是很形象的

②Java中的栈结构：Stack

在Java中存在一个栈结构的实现类，这个类是Stack类

Stack类是Vector类的子类，从集成的角度来说，Stack类是List接口的实现类，同样属于Collection接口的实现类之一

从这个角度来说，Stack类同样具有List接口下其他实现类类似的功能，能够有序可重复的保存元素

但是除了这些特性之外，Stack类还提供了一些额外的方法，能够从数据结构的角度体现出栈结构的特性

方法名称	方法作用	方法说明
push(E e)	元素入栈	将制定元素e加入栈结构中
E pop()	元素出栈	返回栈顶元素，并且将这个元素从栈结构中删除；如果再次调用这个方法，返回的将是下一个栈顶元素
E peek()	看栈顶	返回栈顶元素，但是这个元素并不会从栈结构中删除；如果再次调用这个方法，那么两次返回的将是相同的栈顶元素

③栈结构的典型题

1.数组逆序

题目说明：

将一个数组中的所有元素以倒置的顺序重新存放在这个数组中

题目案例：

给定数组：[1,4,2,7,5,9,6]

倒序存储：[6,9,5,7,2,4,1]

思路解析：

上节课中我们使用一个临时变量和两个下标的方式实现了这个功能

这节课我们可以利用栈结构的特性，将一个数组中的全部元素入栈

然后再逐个出栈，并重新保存在这个数组中的方式，实现数组逆序功能

2.10进制正整数转2进制

题目说明：

将一个给定的10进制正整数，转换为一个2进制字符串进行输出

题目案例：

给定正整数：12

返回对应的2进制字符串输出：0b1100

思路解析：

将10进制转换为2进制，可以通过整除法倒排余数的方式实现

将正整数不断与2进行整除操作，并使用一个栈结构，将整除产生的余数不断入栈，并且使用整除之后的商值重复上述过程

直至这个正整数取值为0之后，将栈结构中的余数依次出栈，出栈过程就是倒排余数的过程

最终形成的结果就是这个正整数对应的2进制取值

2.队列结构

①队列结构的定义：元素先进先出

队列的结构特性是：元素先进先出（First In First Out，简称FIFO）

队列结构的特征就和现实生活中的排队买东西一样，先来排队的先买，一切先来后到，元素按照进入队列的顺序出队列

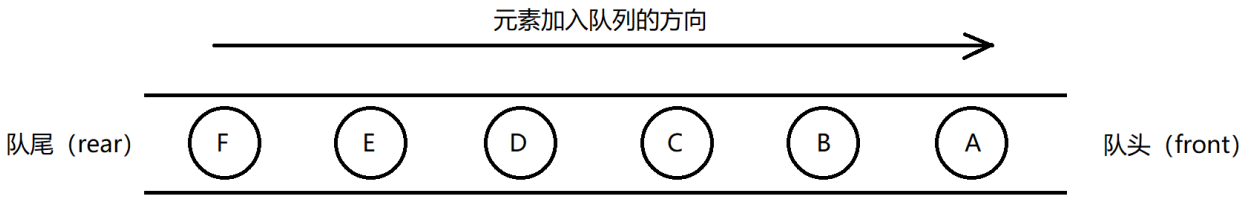
我们将元素加入队列的操作，称之为元素入队列

将元素从队列中取出的操作，称之为元素出队列

将元素出队列的一端称之为队头（front）

将元素入队列的一端称之为队尾（rear）

队列结构图如下：



做个形象一些的比喻，队列就好比一根管子，然后我们挡住其中一端，并且从另一端向管子中放入玻璃球，而管子的粗细正好能够容纳一个玻璃球

当我们放开被挡住的一端的时候，最先放入管子的玻璃球将会最先滚动出来，最后放入管子中的玻璃球，将会最后滚动出来

②Java中的队列结构：LinkedList

在Java中同样存在对队列结构的实现类，这个实现类使我们比较熟悉的LinkedList类型

没错，就是昨天我们详细分析过的链表在Java中的实现类

实际上，Collection接口下，不仅仅有List和Set两大子接口，实际上还存在着一个名为Queue的接口

这个接口从名字上来看，本身就是队列的含义，并且这个接口还有一个子接口名为Deque（双端队列）

而LinkedList类，一方面实现了List接口，一方面也实现了Deque接口，所以我们可以认为LinkedList本身就是一个通过链表实现的双端队列结构

说明：双端队列是一种两端可以同时元素入队列、出队列的结构，也就是说，两端同为队头和队尾

同样的，在LinkedList类型中，也提供了一些方法，用来支持队列操作：

方法名称	方法作用	方法说明
add(E e)	元素入队列	将元素e加入队列，在使用指定长度的队列结构时，如果队列已满，再次调用这个方法将导致IllegalStateException异常
offer(E e)		将元素e加入队列，在使用指定长度的队列结构时，如果队列已满，再次调用这个方法仅仅导致元素无法加入队列，并且方法返回false，并不会抛出异常
E remove()	队头元素出队列	返回队头元素，并将这个元素从队头中删除，如果此时队列是空队列，那么调用这个方法将导致抛出NoSuchElementException异常
E poll()		返回队头元素，并将这个元素从队头中删除，如果此时队列是空队列，那么调用这个方法将返回null，但是并不会抛出异常
E element()	看队列头元素	返回队列头元素，但是并不会导致元素出队列，如果此时队列是空队列，那么将会导致抛出NoSuchElementException异常
E peek()		返回队列头元素，但是并不会导致元素出队列，如果此时队列是空队列，那么调用这个方法将返回null，但是并不会抛出异常
E removeLast()	删除队尾元素	返回队尾元素，并将这个元素从队列中删除，如果此时队列是空队列，那么调用这个方法会导致抛出NoSuchElementException异常

E pollLast()		返回队尾元素，并将这个元素从队列中删除，如果此时队列是空队列，那么调用这个方法将返回null，但是并不会抛出异常

③队列结构的典型题

1.用两个栈实现一个队列

题目说明：

利用栈结构元素先进后出的特点，使用两个栈结构模拟一个队列结构，实现先进先出的功能

题目案例：

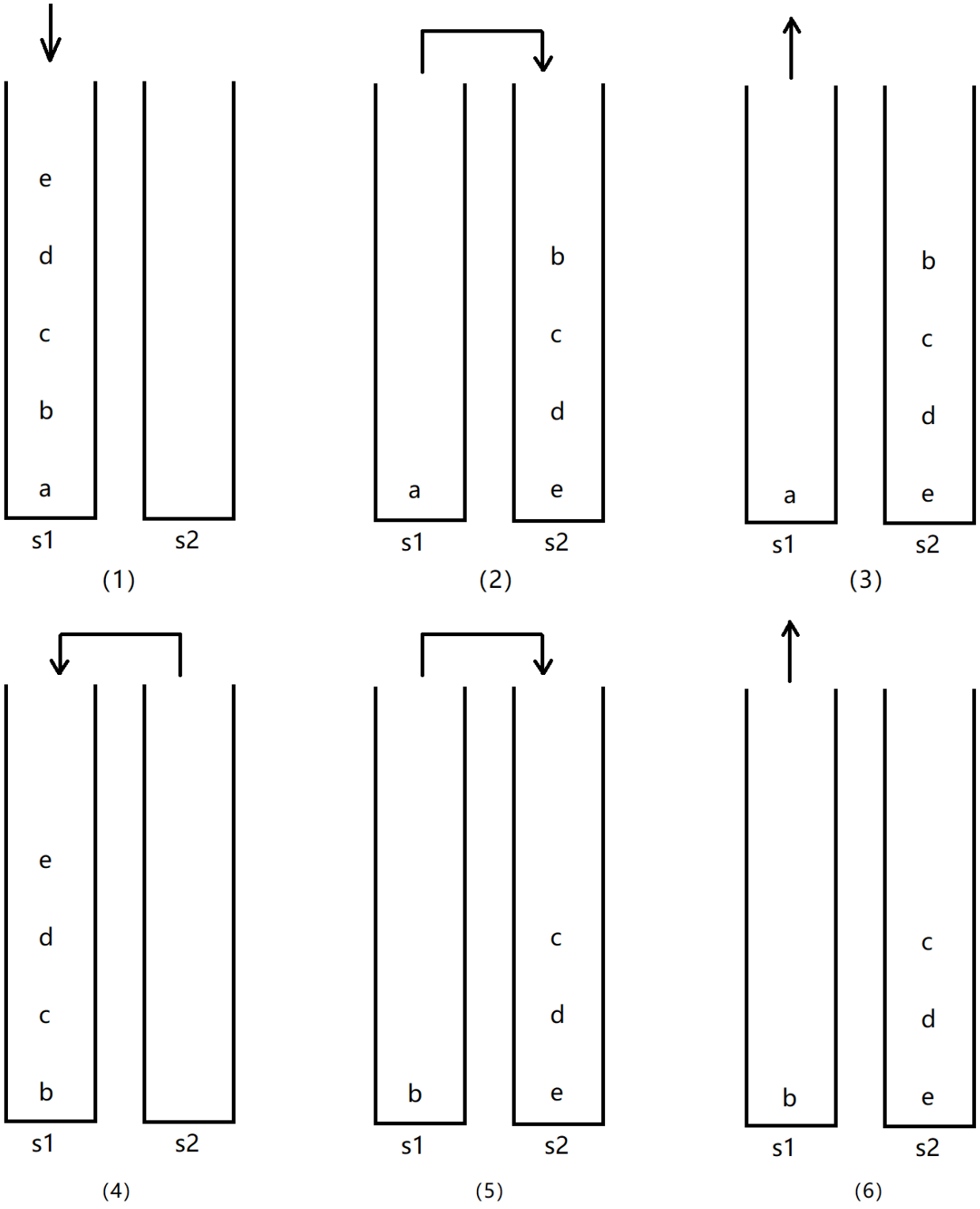
给定元素加入结构顺序：a b c d e

元素出结构顺序：a b c d e

这个结构功能等同于一个队列结构，但是内部要求使用两个栈结构实现

思路解析：

步骤图示如下：



步骤(1)：将加入队列结构的全部元素加入栈s1当中

步骤(2)：在元素出队列的时候，将栈s1中除了栈底元素之外的其他元素全部出栈并加入栈s2当中

步骤(3)：将栈s1当中暴露出来的栈底元素出栈，即为出队列方法的返回值

步骤(4)：将栈s2当中的元素再次倒回到栈s1当中，保证后续入队列元素的顺序是正确的

2.用两个队列实现一个栈

题目说明：

利用队列结构元素先进先出的特点，使用两个队列结构模拟一个栈结构，实现先进后出的功能

题目案例：

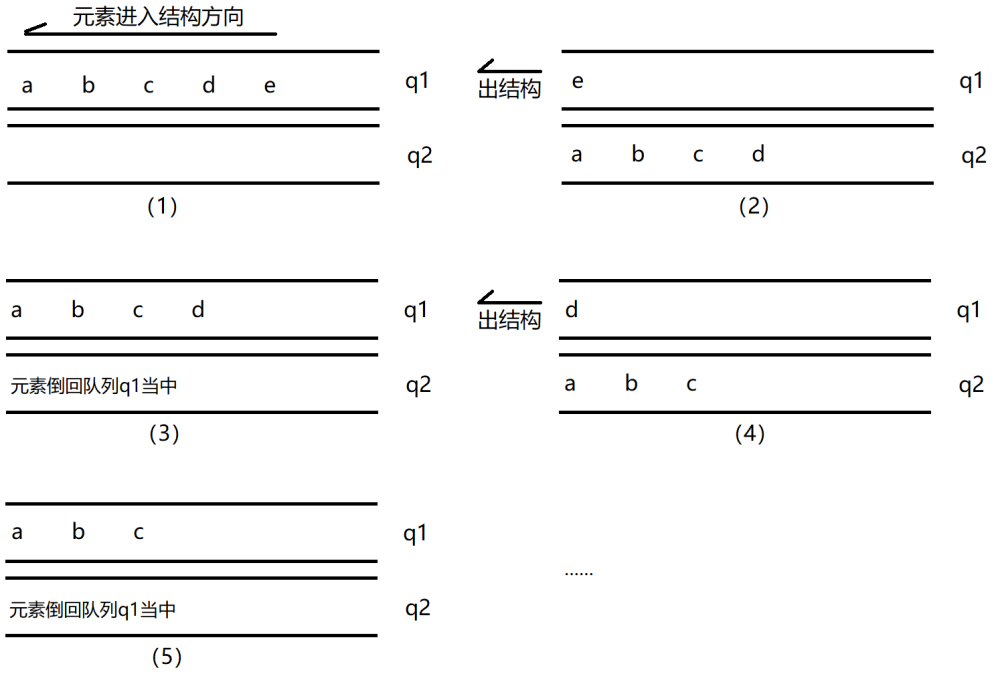
给定元素加入结构顺序：a b c d e

元素出结构顺序：e d c b a

这个结构功能等同于一个栈结构，但是内部要求使用两个队列结构实现

思路解析：

步骤图示如下：



步骤(1)：将全部加入结构的元素加入队列q1

步骤(2)：当元素e要退出结构的时候，将q1中除了元素e之外的其他元素全部出队列q1，并同时加入队列q2

步骤(3)：将队列q1中的元素e出队列，此时队列q1空

步骤(4)：将队列q2中的全部元素出队列，送回到空队列q1当中

……

以此类推，就能够实现元素的倒序输出