线性表的特例：栈、队列

关振扬

September 14, 2025

本章主要内容

- 栈的逻辑结构
- 栈的实现
- 队列的逻辑结构
- 队列的实现
- RPN 和栈：简单的计算器

栈的逻辑结构

Definition

栈是一个特殊的线性表，其特殊之处为插入与删除的位置限制：限定只能在表的固定一端插入、删除。

允许插入、删除的一端称为栈顶、另一端称为栈底。

$$\text{栈底}(a_0, a_1, \ldots, a_{n-1})\ \text{栈顶}.$$

插入又称为：入栈、进栈、压栈
删除又称为：出栈、弹栈
这种结构是一种后进先出的结构。(Last In, First Out; LIFO)

栈：小问

- 假设有三个元素（记为 a, b, c）依次进栈，且每个元素只入栈一次（但入栈的时机不受限，可以是前面的元素出栈前或出栈后），那么可能的出栈序列有多少种？
- 例 1：三个都先入栈，然后都出栈，那么出栈序列是 c, b, a。
- 例 2：a, b 先入栈，然后 b 出栈，c 入栈，然后余下全部出栈，那么出栈序列为 b, c, a。
- 把 3 改成 n 呢？（这与 catalan 数字有关）

栈：ADT

ADT 类型名：`Stack`

数据的关系：
全部数据元素类型相同，相邻元素有前驱、后继关系

操作：构造、析构、长度（可无）、是否空栈、入栈、出栈、栈顶元素值

一般不具备查找功能，基本全部都是 $O(1)$ 时间复杂度（除了析构）

栈：ADT（续）

`initStack`（构造函数）

- 前置条件：没有
- 输入：没有
- 功能：初始一个空栈
- 输出：没有
- 后置条件：完成空栈的初始化

栈：ADT（续）

`destroyStack`（析构函数）

- 前置条件：栈已存在
- 输入：没有
- 功能：销毁栈
- 输出：没有
- 后置条件：完成析构

栈：ADT（续）

`getLength`

- 前置条件：栈已存在
- 输入：没有
- 功能：回传栈的长度
- 输出：如功能
- 后置条件：栈没改变

栈：ADT（续）

`isEmpty`

- 前置条件：栈已存在
- 输入：没有
- 功能：回传栈是否为空
- 输出：如功能
- 后置条件：栈没改变

栈：ADT（续）

`push`

- 前置条件：栈已存在
- 输入：数据元素 x
- 功能：对元素 x 进行入栈处理
- 输出：没有
- 后置条件：于栈顶插入新元素 x。

栈：ADT（续）

pop

- 前置条件：栈已存在
- 输入：没有
- 功能：进行出栈操作。若栈为空，则抛错。
- 输出：回传被出栈元素
- 后置条件：若栈非空，则进行了一次出栈；否则抛错。

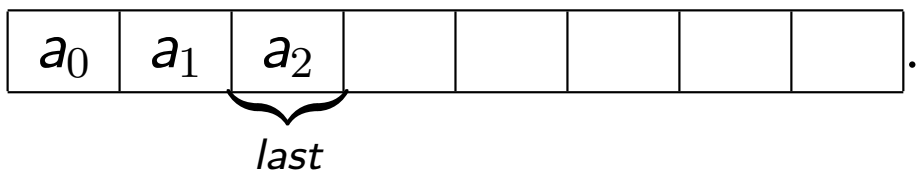
栈：ADT（续）

`getTop`

- 前置条件：栈已存在
- 输入：没有
- 功能：回传栈顶的值；若栈为空，则抛错。
- 输出：回传栈顶元素值
- 后置条件：栈没改变

顺序结构

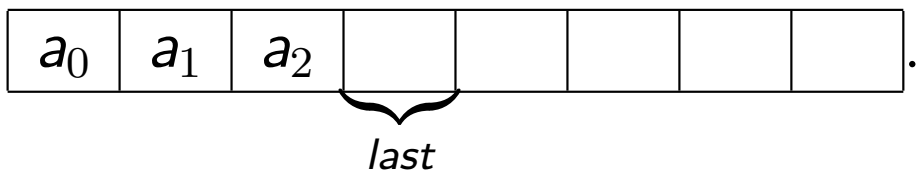
这是利用数组为基础实现，大致跟向量表很像。



关键是有一个指标 `last`，标示最后一个数据的存放位置。我们用这个来代表栈顶。

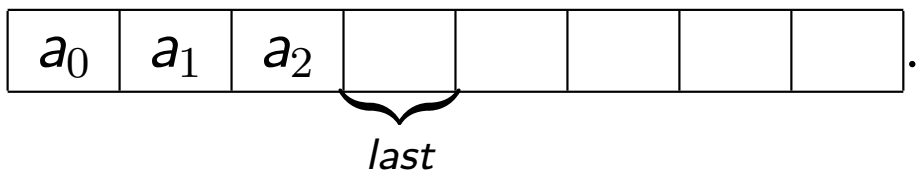
顺序结构：进栈

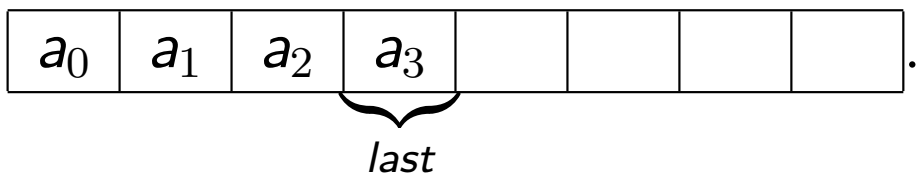
进栈很简单，先把 `last` 往右移，把数据放入。



顺序结构：进栈

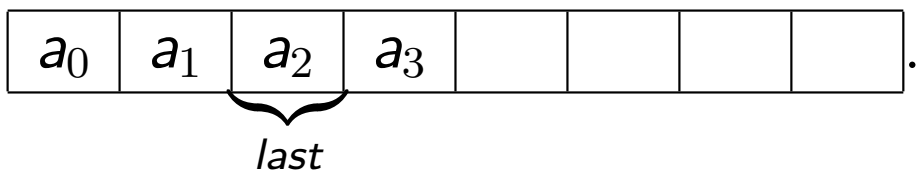
进栈很简单，先把 `last` 往右移，把数据放入。





顺序结构：出栈

出栈也很简单，把 `last` 往左移，再把数据回传。

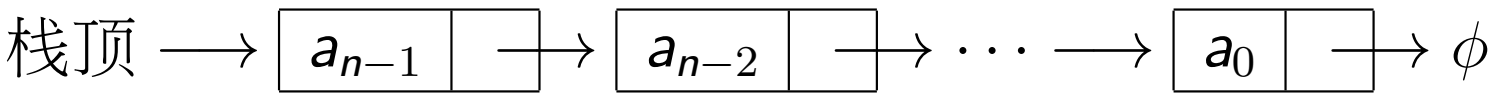


那我们在这里应该要同时注意到空栈时，`last` 要等于 -1，不然我们又要处理特殊情况……

顺序结构：其他注意点

- 我们注意到数组是由长度限制。那当然可以适当时候扩容，但那样就没了每次入栈都是 $O(1)$ 的时间复杂度……
- 如果是有限长度数组，则其实可以有两个栈共享空间……
- 链接结构呢？

链接结构



- 这次我们不用空结点。
- 所有插入都在开始（栈顶）的位置，删除也是。
- 操作基本几跟单链表的于第 0 个位置插入、删除一样。
- 空表为栈顶指针指向空白。

队列的逻辑结构

Definition

队列是一个特殊的线性表，其特殊之处为插入与删除的位置限制：限定只能在表的固定一端插入、另外一端删除。

允许插入的一端称为队尾、删除的一端称为队头。

$$\text{队头}(a_0, a_1, \ldots, a_{n-1})\ \text{队尾}.$$

插入又称为：入队、进队
删除又称为：出队
这种结构是一种先进先出的结构。(First In, First Out; FIFO)

队：ADT

ADT 类型名：`Queue`

数据的关系：
全部数据元素类型相同，相邻元素有前驱、后继关系

操作：构造、析构、长度（可无）、是否空队、入队、出队、队头元素值

一般不具备查找功能，基本全部都是 $O(1)$ 时间复杂度（除了析构）

队：ADT（续）

`initQueue`（构造函数）

- 前置条件：没有
- 输入：没有
- 功能：初始一个空队
- 输出：没有
- 后置条件：完成空队的初始化

队：ADT（续）

`destroyQueue`（析构函数）

- 前置条件：队已存在
- 输入：没有
- 功能：销毁队
- 输出：没有
- 后置条件：完成析构

队：ADT（续）

`getLength`

- 前置条件：队已存在
- 输入：没有
- 功能：回传队的长度
- 输出：如功能
- 后置条件：队没改变

队：ADT（续）

`isEmpty`

- 前置条件：队已存在
- 输入：没有
- 功能：回传队是否为空
- 输出：如功能
- 后置条件：队没改变

队：ADT（续）

`enqueue`

- 前置条件：队已存在
- 输入：数据元素 x
- 功能：对元素 x 进行入队处理
- 输出：没有
- 后置条件：于队尾插入元素为 x。

队：ADT（续）

`dequeue`

- 前置条件：队已存在
- 输入：没有
- 功能：进行出队操作。若队为空，则抛错。
- 输出：回传被出队元素
- 后置条件：若队非空，则进行了一次出队；否则抛错。

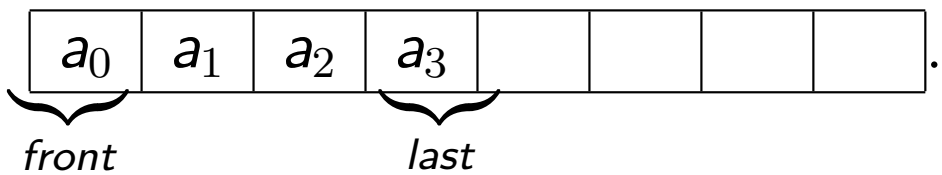
队：ADT（续）

`peekQueue`

- 前置条件：队已存在
- 输入：没有
- 功能：回传队头的值；若队为空，则抛错。
- 输出：回传队头元素值
- 后置条件：队没改变

顺序结构

这是利用数组为基础实现，大致跟向量表很像。

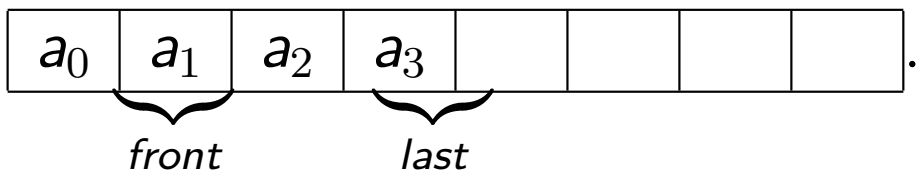


关键是有两个指标 `front` 和 `last`，分别指向队头、队尾。

顺序结构：进队、出队及问题点

进队与进栈的处理一样……

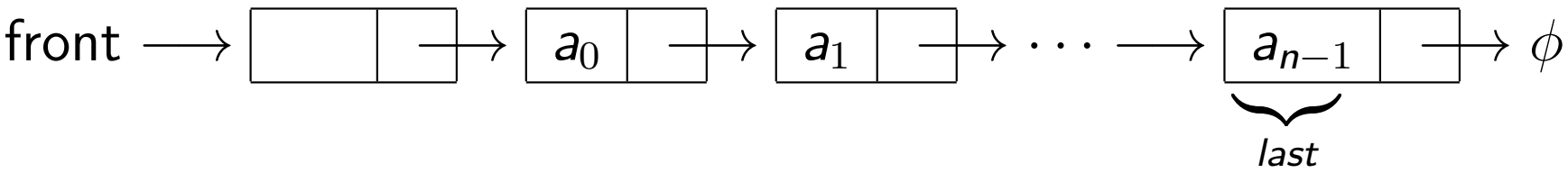
出队的话就是 front 往后移，再回传数值。



队空就是 `front==last+1`。

如果是固定数组长度，那么有空间用完的问题。这个如果用循环数组的位置，那么还是有满队的问题。向量表的话就是最后前面有一大堆碎片。

链接结构



- 这边我们还是用空结点。（你可以想一下不用空结点会怎样）
- 出队在队头方向，跟单链表删首个元素一样，只是要特别注意如果删的唯一存在的数据，那么尾指针位置要调整。
- 入队就是利用我们多写了的尾指针来帮助了。
- 空队就是队头、队尾指针都指向同一个结点（也就是空结点）。

栈的一个简单应用

- 我们可以用栈做一个简单的计算器，这边大概介绍一下概念。
- 为了方便介绍，假设计算的表达式就只有四则运算。
- 表达式是用 RPN (Reverse Polish Notation) 来表达的。可以看一下下列对应：

$$
\begin{aligned}
3+5 &\Longrightarrow 35+ \\
5-2 &\Longrightarrow 52- \\
3+5*4-2 &\Longrightarrow 354*+2- \\
(3+5)*4-2 &\Longrightarrow 35+4*2- \\
(3+5)*(4-2) &\Longrightarrow 35+42-*
\end{aligned}
$$