目录

[**1、** **数据结构** 2](#_Toc497249561)

[**1.1** **线性表** 2](#_Toc497249562)

[**定义** 2](#_Toc497249563)

[**特征** 2](#_Toc497249564)

[**基本操作** 2](#_Toc497249565)

[**存储结构** 2](#_Toc497249566)

[**结构特点** 3](#_Toc497249567)

[**线性表的推广** 3](#_Toc497249568)

[**1.2** **栈** 3](#_Toc497249569)

[**定义** 3](#_Toc497249570)

[**算法** 4](#_Toc497249571)

[**实现** 4](#_Toc497249572)

[**1.3** **队列** 4](#_Toc497249573)

[**定义** 4](#_Toc497249574)

[**种类** 5](#_Toc497249575)

[**实现** 5](#_Toc497249576)

[**运算** 6](#_Toc497249577)

[**1.4** **xx** 7](#_Toc497249578)

1. **数据结构**
   1. **线性表**

**定义**

线性表（linear list）是数据结构的一种，一个线性表是n个具有相同特性的数据元素的有限序列。数据元素是一个抽象的符号，其具体含义在不同的情况下一般不同。

在稍复杂的线性表中，一个数据元素可由多个数据项（item）组成，此种情况下常把数据元素称为记录（record），含有大量记录的线性表又称文件（file）。

线性表中的个数n定义为线性表的长度，n=0时称为空表。在非空表中每个数据元素都有一个确定的位置，如用ai表示数据元素，则i称为数据元素ai在线性表中的位序。

线性表的相邻元素之间存在着序偶关系。如用（a1，…，ai-1，ai，ai+1，…，an）表示一个顺序表，则表中ai-1领先于ai，ai领先于ai+1，称ai-1是ai的直接前驱元素，ai+1是ai的直接后继元素。当i=1,2，…，n-1时，ai有且仅有一个直接后继，当i=2，3，…，n时，ai有且仅有一个直接前驱

**特征**

集合中必存在唯一的一个“第一元素”。

集合中必存在唯一的一个 “最后元素” 。

除最后一个元素之外，均有 唯一的后继(后件)。

除第一个元素之外，均有 唯一的前驱(前件)。

**基本操作**

1）MakeEmpty(L) 这是一个将L变为空表的方法

2）Length（L） 返回表L的长度，即表中元素个数

3）Get（L，i） 这是一个函数，函数值为L中位置i处的元素（1≤i≤n）

4）Prior（L，i） 取i的前驱元素

5）Next（L，i） 取i的后继元素

6）Locate（L，x） 这是一个函数，函数值为元素x在L中的位置

7）Insert（L，i，x）在表L的位置i处插入元素x，将原占据位置i的元素及后面的元素都向后推一个位置

8）Delete（L，p） 从表L中删除位置p处的元素

9）IsEmpty(L) 如果表L为空表(长度为0)则返回true，否则返回false

10）Clear（L）清除所有元素

11）Init（L）同第一个，初始化线性表为空

12）Traverse（L）遍历输出所有元素

13）Find（L，x）查找并返回元素

14）Update（L，x）修改元素

15）Sort（L）对所有元素重新按给定的条件排序

16) strstr(string1,string2)用于字符数组的求string1中出现string2的首地址

**存储结构**

线性表主要由**顺序表**示或**链式表**示。在实际应用中，常以栈、队列、字符串等特殊形式使用。

**顺序表**示指的是用**一组地址连续的存储单元依次存储线性表的数据元素**，称为线性表的**顺序存储结构**或**顺序映像**（sequential mapping）。它以“物理位置相邻”来表示线性表中数据元素间的逻辑关系，可随机存取表中任一元素。

优点：具有简单、运算方便等优点，特别是对于小线性表或长度固定的线性表，采用顺序存储结构的优越性更为突出

缺点：

1.顺序存储插入与删除一个元素，必须移动大了的数据元素，以此对大的线性表，特别是在元素的插入和删除很频繁的情况下，采取顺序存储很是不方便，效率低；

2.顺序存储空间容易满，出现上溢，程序访问容易出问题，顺序存储结构下，存储空间不便扩充；

3.顺序存储空间的分配问题，分多了浪费，分少了空间不足上溢

对于大的线性表，特别是元素变动频繁的大线性表不宜采用顺序存储空间，而采用链式存储结构

**链式表**示指的是用**一组任意的存储单元**存储线性表中的数据元素，称为线性表的**链式存储结构**。它的存储单元可以是连续的，也可以是不连续的。在表示数据元素之间的逻辑关系时，除了存储其本身的信息之外，还需存储一个指示其直接后继的信息（即直接后继的存储位置），这两部分信息组成数据元素的存储映像，称为结点（node）。它包括两个域；存储数据元素信息的域称为数据域；存储直接后继存储位置的域称为指针域。指针域中存储的信息称为指针或链

单向链表、双向链表、循环链表

**结构特点**

1.均匀性：虽然不同数据表的数据元素可以是各种各样的，但对于同一线性表的各数据元素必定具有相同的数据类型和长度。

2.有序性：各数据元素在线性表中的位置只取决于它们的序号，数据元素之前的相对位置是线性的，即存在唯一的“第一个“和“最后一个”的数据元素，除了第一个和最后一个外，其它元素前面均只有一个数据元素(直接前驱)和后面均只有一个数据元素（直接后继）。

**线性表的推广**

时间有序表、排序表、和频率有序表都可以看做是线性表的推广。如果按照结点到达结构的时间先后，作为确定结点之间关系的，这样一种线性结构称之为时间有序表。例如，在红灯前停下的一长串汽车，最先到达的为首结点，最后到达的为尾结点；在离开时最先到达的汽车将最先离开，最后到达的将最后离开。这些汽车构成理一个队列，实际上就是一个时间有序表。栈和队列都是时间有序表。频率有序表是按照结点的使用频率确定它们之间的相互关系的，而排序表是根据结点的关键字值来加以确定的。

* 1. **栈**

**定义**

栈是限定仅在表头进行插入和删除操作的线性表

栈作为一种数据结构，是一种只能在一端进行插入和删除操作的特殊线性表。它按照先进后出的原则存储数据，先进入的数据被压入栈底，最后的数据在栈顶，需要读数据的时候从栈顶开始弹出数据（最后一个数据被第一个读出来）。栈具有记忆作用，对栈的插入与删除操作中，不需要改变栈底指针。

栈是允许在同一端进行插入和删除操作的特殊线性表。允许进行插入和删除操作的一端称为栈顶(top)，另一端为栈底(bottom)；栈底固定，而栈顶浮动；栈中元素个数为零时称为空栈。插入一般称为进栈（PUSH），删除则称为退栈（POP）。栈也称为后进先出表。

栈可以用来在函数调用的时候存储断点，做递归时要用到栈！

在计算机系统中，栈则是一个具有以上属性的动态内存区域。程序可以将数据压入栈中，也可以将数据从栈顶弹出。在i386机器中，栈顶由称为esp的寄存器进行定位。压栈的操作使得栈顶的地址减小，弹出的操作使得栈顶的地址增大。

栈在程序的运行中有着举足轻重的作用。最重要的是栈保存了一个函数调用时所需要的维护信息，这常常称之为堆栈帧或者活动记录。堆栈帧一般包含如下几方面的信息：

1．函数的返回地址和参数

2． 临时变量：包括函数的非静态局部变量以及编译器自动生成的其他临时变量。

**算法**

1．进栈（PUSH）算法

①若TOP≥n时，则给出溢出信息，作出错处理（进栈前首先检查栈是否已满，满则溢出；不满则作②）；

②置TOP=TOP+1（栈指针加1，指向进栈地址）；

③S(TOP)=X，结束（X为新进栈的元素）；

2．退栈（POP）算法

①若TOP≤0，则给出下溢信息，作出错处理(退栈前先检查是否已为空栈， 空则下溢；不空则作②)；

②X=S(TOP)，（退栈后的元素赋给X）：

③TOP=TOP-1，结束（栈指针减1，指向栈顶）。

**实现**

* 1. **队列**

**定义**

队列是一种特殊的线性表，特殊之处在于它只允许在表的前端（front）进行删除操作，而在表的后端（rear）进行插入操作，和栈一样，队列是一种操作受限制的线性表。进行插入操作的端称为队尾，进行删除操作的端称为队头。队列中没有元素时，称为空队列。

队列的数据元素又称为队列元素。在队列中插入一个队列元素称为入队，从队列中删除一个队列元素称为出队。因为队列只允许在一端插入，在另一端删除，所以只有最早进入队列的元素才能最先从队列中删除，故队列又称为先进先出（FIFO—first in first out）线性表。

**种类**

1. 顺序队列

建立顺序队列结构必须为其静态分配或动态申请一片连续的存储空间，并设置两个指针进行管理。一个是队头指针front，它指向队头元素；另一个是队尾指针rear，它指向下一个入队元素的存储位置，如图所示

每次在队尾插入一个元素是，rear增1；每次在队头删除一个元素时，front增1。随着插入和删除操作的进行，队列元素的个数不断变化，队列所占的存储空间也在为队列结构所分配的连续空间中移动。当front=rear时，队列中没有任何元素，称为空队列。当rear增加到指向分配的连续空间之外时，队列无法再插入新元素，但这时往往还有大量可用空间未被占用，这些空间是已经出队的队列元素曾经占用过得存储单元。

顺序队列中的溢出现象：

（1） "下溢"现象：当队列为空时，做出队运算产生的溢出现象。“下溢”是正常现象，常用作程序控制转移的条件。

（2）"真上溢"现象：当队列满时，做进栈运算产生空间溢出的现象。“真上溢”是一种出错状态，应设法避免。

（3）"假上溢"现象：由于入队和出队操作中，头尾指针只增加不减小，致使被删元素的空间永远无法重新利用。当队列中实际的元素个数远远小于向量空间的规模时，也可能由于尾指针已超越向量空间的上界而不能做入队操作。该现象称为"假上溢"现象。

1. 循环队列

在实际使用队列时，为了使队列空间能重复使用，往往对队列的使用方法稍加改进：无论插入或删除，一旦rear指针增1或front指针增1 时超出了所分配的队列空间，就让它指向这片连续空间的起始位置。自己真从MaxSize-1增1变到0，可用取余运算rear%MaxSize和front%MaxSize来实现。这实际上是把队列空间想象成一个环形空间，环形空间中的存储单元循环使用，用这种方法管理的队列也就称为循环队列。除了一些简单应用之外，真正实用的队列是循环队列。

在循环队列中，当队列为空时，有front=rear，而当所有队列空间全占满时，也有front=rear。为了区别这两种情况，规定循环队列最多只能有MaxSize-1个队列元素，当循环队列中只剩下一个空存储单元时，队列就已经满了。因此，队列判空的条件时front=rear，而队列判满的条件时front=（rear+1）%MaxSize。队空和队满的情况

**实现**

1. 数组实现

队列可以用数组Q[1…m]来存储，数组的上界m即是队列所容许的最大容量。在队列的运算中需设两个指针：head，队头指针，指向实际队头元素；tail，队尾指针，指向实际队尾元素的下一个位置。一般情况下，两个指针的初值设为0，这时队列为空，没有元素。数组定义Q[1…10]。Q(i) i=3,4,5,6,7,8。头指针head=2，尾指针tail=8。队列中拥有的元素个数为:L=tail-head。现要让排头的元素出队，则需将头指针加1。即head=head+1这时头指针向上移动一个位置，指向Q(3)，表示Q(3)已出队。如果想让一个新元素入队，则需尾指针向上移动一个位置。即tail=tail+1这时Q(9)入队。当队尾已经处理在最上面时，即tail=10，如果还要执行入队操作，则要发生"上溢"，但实际上队列中还有三个空位置，所以这种溢出称为"假溢出"。

克服假溢出的方法有两种。一种是将队列中的所有元素均向低地址区移动，显然这种方法是很浪费时间的；另一种方法是将数组存储区看成是一个首尾相接的环形区域。当存放到n地址后，下一个地址就"翻转"为1。在结构上采用这种技巧来存储的队列称为循环队列。

队列和栈一样只允许在断点处插入和删除元素。

循环队的入队算法如下：

1、tail=tail+1；

2、若tail=n+1，则tail=1；

3、若head=tail，即尾指针与头指针重合了，表示元素已装满队列，则作上溢出错处理；

4、否则，Q(tail)=X，结束（X为新入出元素）。

队列和栈一样，有着非常广泛的应用。

注意：（1）有时候队列中还会设置表头结点，就是在队头的前面还有一个结点，这个结点的数据域为空，但是指针域指向队头元素。

（2）另外，上面的计算还可以利用下面给出的公式cq.rear=(cq.front+1)/max;

当有表头结点时，公式变为cq.rear=(cq.front+1)/（max+1）。

1. 链表实现

在队列的形成过程中，可以利用线性链表的原理，来生成一个队列。

基于链表的队列，要动态创建和删除节点，效率较低，但是可以动态增长。

队列采用的FIFO(first in first out)，新元素（等待进入队列的元素）总是被插入到链表的尾部，而读取的时候总是从链表的头部开始读取。每次读取一个元素，释放一个元素。所谓的动态创建，动态释放。因而也不存在溢出等问题。由于链表由结构体间接而成，遍历也方便。

**运算**

（1）初始化队列：Init\_Queue(q) ，初始条件：队q 不存在。操作结果：构造了一个空队；

（2）入队操作： In\_Queue(q,x),初始条件： 队q 存在。操作结果： 对已存在的队列q，插入一个元素x 到队尾，队发生变化；

（3）出队操作： Out\_Queue(q,x)，初始条件: 队q 存在且非空，操作结果： 删除队首元素，并返回其值，队发生变化；

（4）读队头元素：Front\_Queue(q,x)，初始条件: 队q 存在且非空，操作结果： 读队头元素，并返回其值，队不变；

（5）判队空操作：Empty\_Queue(q)，初始条件： 队q 存在，操作结果： 若q 为空队则返回为1，否则返回为0。

* 1. **数组**

**定义**

数组是一个钟顺序式的结构，用来存储多个同类型的数据。

数组使用的是一种静态的内存空间配置方式，程序员在使用数组前，需要将数组所需要的空间和数据类型定义出来。

**一维数组运用**

使用一维数组模拟存储大数据实现高位的阶乘。查看样例代码cn.com.demo.array.DemoArray里面的printFactorial方法。

对数组的操作一般都结合下标循环操作，下标从0开始。

**二维数组**

**特殊类型数组**

1. 稀疏数组

在一个二维数组中，大部分的元素都没有使用，值为0，仅仅有少部分空间设置了值，这样造成了很大的空间浪费。为了解决空间浪费问题，同时又不影响保存在数组中的数据，这时候可以一种压缩的方式保存稀疏数组中的数据。

1. 上三角数组
2. 下三角数组