Chapter 2 线性表

线性表概念

线性表：n 个数据元素的**有限**序列

表中各个表项相继排列，每两个相邻表项之间都有直接前驱和直接后继的关系。线性表存在唯一的第一个表项和最后一个表项。

线性结构中表项和表项的邻接关系是一对一的，而所有结点按一对一邻接关系构成的整体就是线性结构

顺序表

顺序表：把线性表中的所有表项按照其逻辑顺序依次存储到从计算机存储中指定存储位置开始的一块**连续**的存储空间中。是线性表基于数组的存储表示

若有 n 个表项，每个表项的数据类型是 T ，则顺序表占据的空间为 $n \times sizeof(T)$

顺序表的特点：

- 逻辑顺序与物理顺序一致
- 既可以顺序访问，也可以随机访问
- 存储利用率高，无需为表项间的逻辑关系增加额外的存储空间，存取操作的开销都是常数级
- 插入和删除开销较大
- 预先进行存储分配在表长变化时难以确定合适的大小，若按最大长度分配则有大量闲置空间，若分配空间不足在插入元素时可能会溢出，使用动态分配的方法在扩充空间时时间开销比较大（将所有表项复制一遍）

静态存储与动态存储

顺序表与一维数组的差异：一维数组只有两个操作，按下标存和按下标取，所以一维数组中的数据可能是跳跃式的，不连续的，而顺序表的存储则是在一个连续的空间，不能跳跃地存储

顺序表的性能分析

搜索：最坏情况要进行 n 次比较，一般情况下平均比较次数为 $\frac{n+1}{2}$ 次

插入：在表头插入需要移动 n 个表项，在表尾插入需要移动 0 个表项，平均要移动 $\frac{n}{2}$ 个表项（共有 $n+1$ 个插入位置）

删除：在表头删除需要移动 $n-1$ 个表项，在表尾删除需要移动 0 个表项，平均要移动 $\frac{n-1}{2}$ 个表项（共有 n 个删除位置）

若对顺序没有要求则可在插入时插至表尾，删除时用最后一个元素代替被删除的元素，时间复杂度为常数

单链表

单链表：用指针表示结点的逻辑关系，一个结点包含两个域：数据域和指针域，数据域存放数据元素，指针域指向下一个结点的存储地址

单链表的特点：

- 长度方便扩充
- 数据元素的顺序与其物理顺序可能不一致，用指针将其按逻辑顺序连接起来

单链表的类定义

包含两个类：用于表示结点的 ListNode 类和表示链表的 List 类

定义方式：

- 复合方式：将 List 类声明为 ListNode 的友元
- 嵌套方式：在 List 类内嵌套定义 ListNode 类
- 继承方式：List 类继承 ListNode 类
- 结构方式：将 ListNode 声明为结构

插入与删除操作的特点

- 不用移动元素，只用修改指针，较为方便
- 情况复杂，空表和在表头插入需要额外判断并处理
- 寻找插入/删除位置需要遍历表项

带附加头结点的单链表

附加头结点：位于表最前端的结点，不存储数据，仅表示表的开始

统一了非空表和空表的插入与删除操作

单链表的建立

头插法：将新结点插入表前端，逻辑顺序与插入顺序相反

尾插法：将新结点插入链表尾部，顺序保持不变，需要设置尾指针

线性链表的变形

循环链表：表尾结点存储表头结点的地址。为了插入操作方便可以不保存表头指针而是保存表尾指针

双向链表：每个结点有两个指针域，指向其直接前驱和直接后继，这样向前和向后搜索的时间复杂度都为 $O(1)$。插入/删除修改四个指针。通常采用附加头结点，减少操作复杂度

单链表的应用：多项式

多项式可使用数组或链表存储，需考虑到稀疏多项式存储时的内存开销（数组存储系数，数组存储项数和系数，链表存储）

多项式加法：类似归并排序的 merge 操作（时间复杂度为 $O(m+n)$，m,n 为两个链表的长度）

多项式乘法：使用 result 数组暂存系数的积的结果，最终非零的项可加入结果多项式

静态链表

为数组的每一个元素增加一个链接指针，重新链接便可改变逻辑顺序（不改变物理位置）。

优点：可将建立好的链表数组作为文件储存起来，以后读入可不用重新建立链表