基本概念

数据的基本单元: **数据元素**.

数据 (元素) 的最小单元: **数据项** (不可分割).

数据结构: 分为**逻辑结构**和**存储 (物理) 结构**.

存储 (物理) 结构: **顺序存储**、**链式存储**、**索引存储**、**散列存储**.

评价一个算法的两个标准: **时间复杂度** $T(n)$、**空间复杂度** $S(n)$.

数据结构研究的三个方面: **逻辑结构**、**存储结构**、**算法**.

线性结构: 元素之间的逻辑关系用线性序列简单表示.

非线性结构: 不是线性结构 (顾名思义).

头指针: 指向链表中的第一个结点 (头结点或首元结点) 的指针.

首元结点: 是第一个数据元素结点 (第一个用于存储数据的结点).

头结点: 首元结点之前的一个结点 (不用于存储数据, 只作为链表的开头).

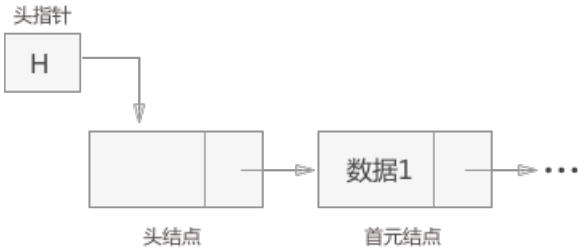

图 1: 头指针, 头结点, 首元结点的区别

指针域: 存储直接后继存储位置的域. (就是常用的 next 指针)

线性表包括: 顺序表 (数组), 链表 (单链表, 双链表). (两者的优缺点可以对照记忆)

顺序优缺点 (a 为优点, b 为缺点):

a.1. 具有**随机存储特性** (注意区分它的**存储单元是连续的一块**), 也可以说是可以**直接访问**数据元素.

随机存储 = 直接访问 ≠ 连续的存储单元.

a.2. **存储密度大**, 无需为线性表中各元素间的逻辑关系而增加存储空间 (省去了链表所需要存储的指针域, 以减少空间).

b.1. **插入和删除操作时间复杂度高**, 需要移动大量元素.

b.2. **初始空间不宜分配** (需要初始化数组必须给定一个大小).

链表优缺点 (a 为优点, b 为缺点):

a.1. 采用**动态分配**的方法, 内存利用率高. (用多少结点就创建多少个, 不多不少)

a.2. **插入和删除操作时间复杂度低**, 只需修改指针域, 无需移动元素. (改下指针的指向即可)

b.1. **存储密度小**, 记录元素之间的逻辑关系需要额外的存储空间. (需要记录指针域)

b.2. **不具有随机存储特性**.

线性表和链表的使用场景: 如需要快速地读取且很少使用插入删除操作, 则使用顺序表; 如遇到需要频繁使用插入删除操作, 则用链表.