线性表

# 综述

线性表（Linear List）是数据结构中的一种基本结构，它表示数据元素之间存在一对一关系的数据结构。线性表由一组数据元素构成，数据元素的位置只取决于它们的序号，数据元素之间的相对位置是线性的，即存在唯一的第一个元素和唯一的最后一个元素，除第一个元素外，序列中的每个元素有且只有一个前驱，除最后一个元素外，序列中的每个元素有且只有一个后继。

线性表有两种基本的实现方式：顺序存储和链式存储。

1. 顺序存储：把逻辑上相邻的元素存储在物理位置相邻的存储单元里，元素之间的关系由存储单元的邻接关系来体现。顺序存储结构的主要优点是存储密度大，每个元素占用的存储空间少，且可以快速地存取表中任一位置的元素。主要缺点是插入和删除操作需要移动大量的元素。
2. 链式存储：通过存储在数据元素中的指针来表示元素之间的逻辑关系。链式存储结构的主要优点是在插入和删除元素时，只需要修改被插入或删除元素及其前驱或后继元素的指针，而不需要移动其他元素，因此链式存储结构在进行插入和删除操作时更加方便和灵活。主要缺点是每个元素需要额外的存储空间来存储指针，且存取表中任一位置的元素都需要从头指针开始遍历链表，因此存取效率较低。

线性表在许多实际应用中都有重要的用途，如搜索引擎中的索引、数据库中的记录等。

# 顺序表

顺序表，也被称为数组列表，是一种线性数据结构，它在计算机内存中以数组的形式保存数据元素。顺序表使用一段物理地址连续的存储单元来依次存储数据元素，并且逻辑上相邻的数据元素在物理存储单元上也是相邻的。这种存储方式使得顺序表在访问元素时具有很高的效率，因为可以通过计算得到每个元素的内存地址。

顺序表可以分为静态顺序表和动态顺序表两种：

1. 静态顺序表：使用定长数组来存储元素，这意味着在创建顺序表时需要预先分配固定大小的内存空间。如果实际存储的元素数量超过了数组的长度，就会导致数据溢出。因此，静态顺序表适用于存储元素数量已知且不会改变的场景。
2. 动态顺序表：使用动态开辟的数组来存储元素，这意味着在创建顺序表时不需要预先分配内存空间，而是根据实际需要动态地分配和释放内存。动态顺序表在存储元素数量不确定或可能变化的情况下更加灵活和高效。

顺序表的基本操作包括初始化、扩容、尾插、尾删、头插、头删、删除指定位置的元素以及查找等。这些操作可以通过对数组进行相应的修改来实现。

需要注意的是，虽然顺序表在访问元素时具有高效率，但在插入和删除元素时可能需要移动大量的元素，从而导致较低的效率。因此，在选择使用顺序表还是其他数据结构时，需要根据具体的应用场景和需求来权衡利弊。

# 链表

链表（Linked List）是一种常见的基础数据结构，它也是一种线性表，但与数组不同的是，链表中的数据元素在内存中不是顺序存储的，而是通过存储在元素中的指针链接在一起。链表由一系列节点（Node）组成，每个节点包含两个部分：数据域和指针域。数据域用于存储数据元素，指针域则用于存储指向下一个节点的指针。

链表有多种分类，主要包括：

1. 单链表：每个节点只有一个指向下一个节点的指针，链表的头节点指向第一个元素，尾节点的指针指向空（null）。单链表只能从头节点开始顺序访问元素。
2. 双链表：每个节点有两个指针，一个指向前一个节点，另一个指向下一个节点。双链表可以从头节点或尾节点开始双向访问元素。
3. 循环链表：循环链表的尾节点的指针指向头节点，形成一个环状结构。循环链表可以从任意一个节点开始循环访问所有元素。

链表的主要优点是可以在不移动其他元素的情况下插入和删除元素，这使得链表在某些场景下比数组更加灵活和高效。然而，链表也有一些缺点，例如需要额外的存储空间来存储指针，以及访问特定位置的元素时需要从头节点开始遍历链表，导致访问效率较低。

链表在实际应用中有许多用途，例如实现LRU（Least Recently Used）缓存淘汰算法等。LRU算法是一种常用的缓存置换策略，它根据数据元素最近被访问的时间来决定哪些元素应该被替换出缓存。通过使用链表来存储缓存元素，可以方便地实现LRU算法。