# 第2节 顺序表

课程介绍

* 1. 什么是线性表;
* 2. 为什么需要线性表;（了解）
* 3. 顺序表的实现;（掌握）
* 4. 顺序表的含义及操作;(了解)

## 0 线性表

### 0.0 学习目标：

- 什么是线性表? （了解）

- 了解线性表的特征

- 了解线性表的几种基本操作

### 0.1 什么是线性表？

在程序中，通常需将一组数据元素作为一个整体来管理和使用; 最常用的解决方案是将这一组数据元素看作一个序列. 一个线性表（linear list）是n个**数据元素**的有限**序列**：().

换句话说, 一个线性表是某类**元素**的一个**集合**, 加上元素之间的一种**顺序**. 线性表, 简称**表**, 是一种**线性结构**.

线性表是最基本的数据结构之一, 它常常被用作更复杂的数据结构的实现基础. 所以, 线性表是一种最常用的数据结构.

分析：

- **数据元素**,也称**记录**，其具体含义在不同的情况下一般不同.数据元素由**数据项**组成.

- 称为**起始结点**

- 为**终止结点**

- 称为的**前驱**

- 称为的**后续**

线性表的示例：

1. alphabet = (a, b, c, d, ..., z)

结点是一个英文字母; a, b等分别是一个数据元素. 每个结点只有一个数据项.

1. odd = (1,3,5,7,9)

结点是整型数，3, 5等数就是一个数据元素.

1. 学生成绩表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学号 | 姓名 | 语文 | 数学 | 物理 |
| 20189001 | 李小梅 | 91 | 95 | 93 |
| 20189002 | 张军军 | 61 | 78 | 99 |
| 20189003 | 钱豆豆 | 83 | 98 | 90 |

上表中每一行作为一个结点 .每一个结点有5个数据项.

4) 不含任何元素的线性表,称为**空表**.

### 0.2 线性表的特征

1) 一个表中包含的元素的个数,称为表的**长度**.

2) 表元素之间有一个基本关系存在: **“下一个”**关系. 表可以写成

”下一个”关系可以写成:

简记为

下一个关系, 可以看成某种**二元组的集合**. 例如 我们由一个线性表 L = (李梅, 张军军, 钱豆豆). 这个线性表的”**下一个”**关系就是下面的表格中的彩色的二元组的集合.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **李梅** | **张军军** | **钱豆豆** |
| **李梅** | (李梅, 李梅) | (**李梅**, **张军军**) | (李梅, 钱豆豆) |
| **张军军** | (张军军, 李梅) | (张军军, 张军军) | (**张军军**, **钱豆豆**) |
| **钱豆豆** | (钱豆豆, 李梅) | (钱豆豆,张军军) | (钱豆豆,钱豆豆) |

2) 在数据元素的集合中:

- 存在唯一的一个首元素.

- 存在唯一的一个末元素 .

- 除首元素外，每个元素均只有唯一的前驱元素.

- 除末元素外，每个元素均只有唯一的后续元素

### 0.3 线性表的基本操作

- 计算表的长度 求表中的元素个数

- 存取 存取表中第个数据元素

- 插入　在之后插入一个新数据元素

- 删除　删除第个元素

- 合并　将多个线性表合成一个线性表

- 分解　将一个线性表拆成多个线性表

- 查找　查找满足条件的数据元素

- 排序　对数据元素按照某数据项的值递增或递减的次序重新排列

0.4 应用场景

将一组数据元素作为整体来管理的情形，就可以用线性表；

线性表是很多其他数据结构（如队列和栈）的基础。

## 1 顺序表的形式

### 1.0 学习目标

1．掌握顺序表概念

2. 掌握顺序表的两种基本形式

3. 了解顺序表的优点

### 1.1 什么是顺序表?

将线性表中元素依次地存储在一组地址连续的存储单元中，这种存储结构称为**顺序结构**.　用顺序存储结构的线性表称为**顺序表**.

注意：地址连续

### 1.2 顺序表的基本形式

1. 数据元素连续存储的顺序表:

分析：数据元素本身连续存储,可用来存储相同类型的数据.

2. 特点：

- 每个数据元素所占的存储空间的**大小相同**. 设每个数据元素需要占用个存储单元, 也就是说, 一个数据元素所需的存储单元的数目为 t

- 顺序表存储结构示意：设第0个元素在内存上的存储地址为. 第个数据元素的存储位置,或称存储地址为

|  |  |  |
| --- | --- | --- |
| 存储地址 (物理地址) | 元素 | 元素在顺序表中的位置(逻辑地址) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

- 只要有了第一个结点的起始地址，可以按照相同的速度访问线性表中的任意一个结点i.

- 例如：　在32／64位系统,**整型数据占4个字节**,　如果第一个结点的起始地址为#0120，　顺序表(100, 200, 400, 800)的存储结构如下：

|  |  |  |
| --- | --- | --- |
| 存储地址 | 元素 | 元素在顺序表中的位置 |
| #0120 | **100** | 0 |
| #0124 | **200** | 1 |
| #0128 | **400** | 2 |
| #0132 | **800** | 3 |

- 这样的顺序表可用来存储相同类型的数据.

### 1.3 顺序表的形式二：元素外置的顺序表

举例如下, 线性表种的数据元为一个存储地址:

|  |  |  |
| --- | --- | --- |
| 存储地址 | 元素 | 元素在顺序表中的位置 |
| #0120 | **#100** | 0 |
| #0124 | **#808** | 1 |
| #0128 | **#400** | 2 |
| #0132 | **#56** | 3 |

以这些存储地址为线索, 可以找到实际数据. 如下表格所示:

|  |  |  |
| --- | --- | --- |
| 存储地址 | 元素 | 元素在顺序表中的位置 |
| #100 | **1999** | 0 |

|  |  |  |
| --- | --- | --- |
| 存储地址 | 元素 | 元素在顺序表中的位置 |
| #400 | **“March”** | 0 |

等等.

1) 含义：数据在顺序表之外, 可以存储不同类型的数据

2) 特点：

- 顺序表中各单元保存数据地址,每个地址所需要的存储量相同

- 可以用来保存数据元素的**大小不同**的数据对象，例如python中列表li =[1999 2018,"Mar"]

3) 连续存储形式和元素外置形式的区别：

- 连续存储形式下，每个存储空间所存的是**数据**本身

- 元素外置形式下，每个存储空间存储的是数据的**地址**，数据存储在内存的其他位置

### 1.3 为什么需要顺序表？(顺序表的优点)

- 顺序表中的任意数据元素可由公式直接导出，因此顺序表是随机存取的存储结构;

- 利用顺序表，无需为表示结点间的逻辑关系增加额外的存储空间；

- 可方便地随机存取表中的任意数据元素(结点)

## 2 顺序表的结构与实现

### 2.0 学习目标

1. 掌握顺序表的结构

2. 了解顺序表的两种实现方式

3. 了解顺序表存储区的扩充方式

### 2.1 顺序表的结构

1. 数据区：用于存储数据本身

2. 表头信息:关于整个顺序表的详情

- 容量

- 元素个数

3. 作图讲解：

内存地址

容量

元素个数

数据元素

|  |  |
| --- | --- |
| #100 | 8 |
| #104 | 4 |
| #108 | 20 |
| #112 | 40 |
| #116 | 60 |
| #120 | 80 |
| #124 |  |
| #128 |  |
| #132 |  |
| #136 |  |

### 2.2　顺序表的两种实现方式

1. **一体式结构**:表头信息与数据区连续存储

如上图所示.

2. **分离式结构**:表头信息与数据区分别存储

作图讲解与一体式结构的区别.

|  |  |
| --- | --- |
| #100 | 8 |
| #104 | 4 |
| #106 | #200 |
|  |  |
| … |  |
| #200 | 20 |
| #204 | 40 |
| #208 | 60 |
| #212 | 80 |
| #216 |  |
| #220 |  |
| #224 |  |
| #228 |  |

3. 两种实现方式的区别

- 为考虑数据的动态变化，往往用分离式结构

- 若更换数据，一体式结构需整体搬迁,而分离式结构中只需改变表头中链接到数据区的地址

### 2.3　顺序表存储区的扩充

1.**扩充固定数目的存储空间**

- 优点:节省存储空间

- 缺点:操作往往非常频繁

2.**倍增其存储空间**

- 优点: 减少操作次数

- 缺点: 浪费存储空间

## 3 顺序表的操作

### 3.0学习目标

- 掌握插入操作和删除操作的时间复杂度

### 3.1 插入元素

1. 插入元素: 在顺序表的第个数据元素之后插入一个新的数据元素x.

- 插入前，顺序表长度为,顺序表为

()

- 插入后，顺序表长度为,顺序表为

()

- 顺序表进行插入操作后，向后平移一个存储空间

2. 插入元素的效率：

- 最坏情形,,顺序表中结点移动次数是

- 插入算法的最坏时间复杂度为:

- 最好情形,,无须移动结点

- 在顺序表末尾插入元素的时间复杂度:

### 3.2删除元素

1. 顺序表中删除第个元素,

- 删除前，顺序表长度为, 顺序表为

();

- 删除后，顺序表长度为, 顺序表为

();

- 删除元素时，必须将第到第个元素依次向前移动一个位置.

2. 删除元素的效率

- 最坏情形,,顺序表中结点移动次数是

- 删除算法的最坏时间复杂度：

- 最好情形, ,无须移动结点

- 删除表尾元素:

## 4 Python中的列表

### 4.0 学习目标

1. 掌握列表的特征

2. 了解列表的基本实现技术

3. 了解列表的扩充策略

### 4.1 Python列表的特征

Python的标准类型—列表, 就是一种元素个数可变的线性表. 它满足的约束条件(特征)有:

1. 基于下标,元素访问和更新非常**高效**：

1. 允许任意加入元素，且对象的标识id保持不变

### 4.2 Python列表的基本实现技术

#### 4.2.1 Python列表是采用分离式技术实现的动态顺序表

1. 为了达到常数复杂度O(1), 所以采用**顺序表**技术.

2. 为了允许加任意多入元素，所以采用**动态**顺序表. 也就是说, 必须能更换元素存储区.

3. 为了满足在不断加入元素之过程中, 列表对象的标识id保持不变这一条件, 只能采用**分离式**实现技术.

Python列表的官方实现中,列表就是一种采用分离式技术实现的动态顺序表.

小结：列表是分离式动态顺序表.

#### Python列表的扩充策略

1. 建立空列表或很小的列表时，系统分配一块能容纳**８个元素**的存储区

2. 执行插入操作(insert, append等)时，若元素存储区满，换一块**４倍大**的存储区

3. 若表已经足够大，则改变策略：存储区满时，就将存储区的容量加倍. “很大”的意思: 50000.

通过这一套技术实现的列表, 尾端加入元素操作的平均时间复杂度时O(1).

4.3列表的主要操作的性质

1) len()操作的时间复杂度是O(1). 列表中的元素个数已有记录,可以直接提取.

2) 元素访问,赋值,尾端加入元素,尾端删除元素等操作: 时间复杂度为O(1)

3) 一般位置的元素加入和删除操作: 时间复杂度为O(n)

例. 写一个将Python列表中的元素顺序倒置的操作.

## 5 课后扩展

### 5.1 基本概念

a) 什么是**位**(bit)?

信息量的度量单位，是信息量的**最小单位**.二进制数的一位包含的信息.

例：表现26个拉丁字母必要的信息量是5**比特**(16<26<32).

b) 什么是**字节**(Byte)?

信息量的度量单位. 1 Byte = 8 bit.

例： 除拇指外，双手手指弯曲表示0,伸直表示1，那么所有的信息量，就是1字节.

例： CPU的位是指CPU一次性可处理的数据量. 1字节=8位，**32位处理器可以一次性处理4个字节**的数据量，依次类推. 32位操作系统针对的32位的CPU设计. 64位操作系统针对的64位的CPU设计.

c) 什么是**内存**?

计算机内暂时存储程序和数据的地方.

d) 计算机如何找到内存中的信息?

计算机只能1个字节、1个字节地读取内存中的数据. 这样一个字节的内存区域叫一个**存储单元**.

计算机为每个存储单元分配一个号码，这就是内存的**地址**.

内存地址的范围，就是存储地址空间.类比: 为一层楼的房间命名，0001,0002,0003,0004,0005,0006,0007,0008,内存地址的空间为 9999.

### 5.2注意事项：

- 区分字节和位的概念：1 Byte = 8 bit

- **内存读取和定位的最小单位是字节(Byte).**

- 顺序表是线性表的一种，是一种特殊的线性表：顺序表中的元素存储在连续的存储单元中.

6课后总结

6.1 重点

1. 掌握顺序表的概念和特征.

2. Python列表的实现方式; 列表尾端操作的时间复杂度.

* 1. 难点

1. 顺序表的实现.

7习题：

1. 在一个采用顺序存储方式的线性表中，如果表的第一个元素的存储地址为0100,每一个元素的长度为2,则第4个元素的地址是\_\_\_\_\_. (C)

A. 0110

B. 0108

C. 0106

D. 0120

实现一个操作,将列表的元素倒置.

1. 一个有n个元素的线性表，采用顺序存储方式.若删除第个元素，需要向前移动\_\_\_\_\_个元素；若在第个元素前插入一个元素，需要向后移动\_\_\_\_\_个元素.

解: n-2; n-(i-1).

8课外阅读

8.1参考书

刘清,张小勇,王琼 <<数据结构>>(第三版),电子工业出版社 (2010)

裘宗燕 <<数据结构与算法>> 机械工业出版社 (2015)