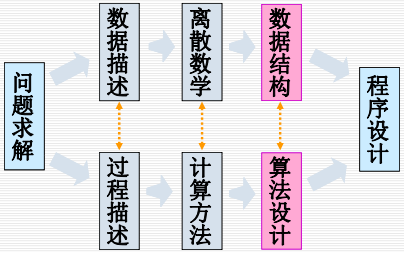
数据结构

西南交通大学
信息科学与技术学院

数据结构定位

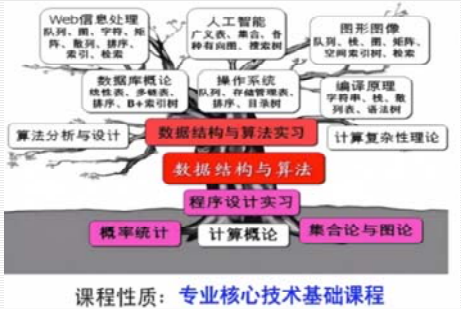
- 例如：在一张图片内搜索某个特定图形。



数据结构的历史

- 数值计算
  - 特点：以数学方程为基础，数据量不太大，但计算量大，数据之间的关系不太复杂，容易处理。
  - 提高计算速度和计算精度
- 非数值计算
  - 特点：数据之间关系复杂，数据量巨大，但计算量不大。主要工作是在大量数据中找到用户所需数据。
- 数据结构起源于程序设计
  - 1968年年克努教授著《计算机程序设计艺术》第一卷《基本算法》第1次较系统地阐述了数据的逻辑结构和存储结构及其操作。
- 数据结构是研究非数值计算的程序设计问题中，计算机的操作对象以及它们之间关系和操作的科学。

数据结构课程的地位



课程性质：**专业核心技术基础课程**

学习要求

- 掌握各种数据结构及基本操作的算法实现
  - 理解和掌握数据结构特点
  - 理解和掌握相关算法原理
  - 了解如何实现算法，具备用C语言编写经典算法程序的能力
- 学会分析数据对象的特征，选择适当的数据结构及相应处理算法求解问题
- 掌握一般算法的复杂度分析
- 上机实习

第1章 绪论

- **主要内容：**
- 数据结构
- 数据类型
- 抽象数据类型
- 算法和算法分析

§1.1 基本概念和术语

- 1. 数据和数据结构
- (1) 基本概念
- **数据**：是描述客观事物的数字、字符以及所有能够输入到计算机中并被计算机处理的信息的总称。

001	高等数学	樊映川	S01	...
002	理论力学	罗远祥	L01	...
003	高等数学	华罗庚	S01	...
004	线性代数	栾汝书	S02	...
...	...	...	...	...

基本概念(.续)

- **数据项**：是有独立含义的数据的最小单位，有时也称为字段。
- **数据元素**：数据的基本单位，通常在计算机中作为一个整体进行考虑和处理。可以是一个数字或字符串，也可以由一个或多个数据项组成
- **数据对象**：性质相同的数据元素的集合。

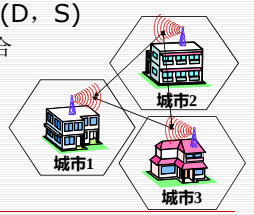
序号	X坐标	Y坐标	颜色
001	12	数据项	255,255,0
002	25	1021	108,134,1
003	103	56	20,13,470
⋮	⋮	⋮	⋮

数据元素

数据对象

(2) 数据结构的定义

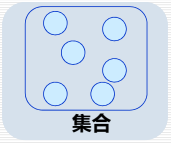
- **数据结构**：相互之间存在一种或多种特定关系的数据元素的集合。
- 形式定义：
  - Data_Structure=(D，S)
    - D——数据元素集合
    - S——关系集合

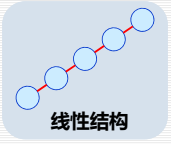


(3) 数据结构的四种基本结构

- **集合：仅同属一集合**
- **线性结构：一对一**
- 例1：电话号码查询系统
- (a_1, b_1) (a_2, b_2)……(a_n, b_n)
- 例2：图书馆书目自动检索问题

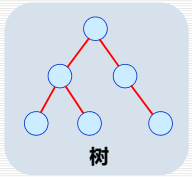
001	高等数学	樊映川	S01	...
002	理论力学	罗远祥	L01	...
003	高等数学	华罗庚	S01	...
004	线性代数	栾汝书	S02	...
...	...	...	...	...



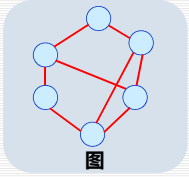


数据结构的四种基本结构(.续)

- **树形结构：一对多**
- 例如：家谱



- **图状或网状结构：多对多**
- 例如：网络



(4) 数据的两种结构

- **数据结构定义2**：数据及数据之间的相互关系。一般包括：
  - 数据的逻辑结构
  - 数据的物理结构
  - 数据的运算

注意：相同的逻辑结构，定义的运算不同，也为不同的数据结构。

- **逻辑结构**：指数据元素之间的相互关系，即数据的组织形式，我们把数据元素间逻辑上的联系，称之为数据的逻辑结构。也就是数据结构的形式定义所描述的结构，即所谓“关系”。
  - 特点：体现数据元素间的抽象化相互联系，并不涉及数据元素在计算机中具体的存储方式，独立于计算机。

数据的两种结构(.续)

- **物理结构**：数据在计算机中的具体表示，也称为存储结构或映象。
  - **顺序映象**→顺序存储结构：借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系
  - **非顺序映象**→链式存储结构：借助指示元素存储地址的指针表示数据元素之间的逻辑关系
  - **元素(结点)**：表示数据元素的若干位组成的位串
- 虚拟存储结构
  - 数据结构在高级程序语言中借助"数据类型"进行的表示

注意：数据的运算定义在数据的逻辑结构上，运算的实现取决于数据的存储结构

数据的操作

- 常见操作
  - 插入操作、删除操作、更新操作、创建操作、查找操作、排序操作、遍历操作、撤销操作。
- 分类
  - **加工型操作**：操作改变了(操作之前的)结构的值。
  - **引用型操作**：不改变值，只是查询或求得结构的值。

总结

- 数据的逻辑结构
  - 线性结构：线性表；栈 队列；串 数组 广义表
  - 非线性结构：树 图；集合
- 数据的存储结构：顺序存储；链式存储
- 数据的操作：插入、删除……

2. 数据类型

- (1) 数据类型(Data Type)
- **数据类型**：是一个值的集合和定义在这个值集上的一组操作的总称。通常数据类型可以看作是程序设计语言中已实现的数据结构，用来刻划程序操作对象的特性，是高级程序设计语言中的一个基本概念
- 形式化定义：Data—Type=(D, S, P)
  - D是数据对象，S是D上的关系集，P是对D的基本操作集

注意：数据结构强调数据元素间的关系。数据类型强调数据对象的特征

数据类型(.续)

- 从用户角度看：数据类型是一组值的集合和定义在这个值集上的一组操作的总称。
- 从机器具体实现看：每类数据都有一个值的集合，规定了值在计算机内的表示形式，规定了施加于值集上的一组运算和运算规则。
- 分类：
  - **原子类型**：值不可分解
  - **结构类型**：值由若干成分按某种结构组成
- 例如：C语言中的原子类型int，结构类型数组。

(2) 抽象数据类型

- **抽象数据类型**(Abstract Type简称ADT)：是指一个数学模型以及定义在该模型上的一组操作。
- 任何一个程序设计语言所提供的数据类型从用户角度看都是抽象数据类型。
  - 例如：所有计算机都拥有的"整型"类型。
- 特点：仅取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关。可以看作是数据的逻辑结构及其在逻辑结构上定义的操作。

注意：抽象数据类型和数据类型实质上是同一个概念，但抽象数据类型的范畴比数据类型更广，它还包括用户在设计软件系统时自己定义的数据类型

抽象数据类型分类

- **原子类型**：属于原子类型的变量值是不可分解的。
  - 例如：整数
- **固定聚合类型**：属于该类型的变量，其值由确定数目的成分按某种结构组成。
  - 例如：复数
- **可变聚合类型**：和固定聚合类型相比，构成可变聚合类型“值”的成分的数目不确定。
  - 例如：矩阵

19

抽象数据类型的定义

```
ADT 抽象数据类型名{
        数据对象：<数据对象的定义>
        数据关系：<数据关系的定义>
        基本操作：<基本操作的定义>
}ADT抽象数据类型名
```

注意：所定义的数据类型的抽象层次越高，含有该抽象数据类型的软件复用程度就越高

20

举例：复数的抽象数据类型定义

```
ADT complex{
    数据对象：D={c1,c2   c1,c2∈FloatSet}
    数据关系：R={< c1,c2 >  c1表示实部，c2表示虚部}
    基本操作：
        creatc(a)；   /*创建一个复数
        outputc(a)；/*输出一个复数
        addc(a,b)；  /*求两个复数相加之和
        subc(a,b)；  /*求两个复数相减之差
        mulc(a,b)； /*求两个复数相乘之积
        ……等等；
        }ADT complex；
```

21

抽象数据类型和“类”

- 抽象数据类型可以看作是描述问题的模型，它独立于具体实现
  - 优点：封装数据和操作，使用户程序只能通过在ADT里定义的操作来访问其中的数据，实现了信息隐藏
- C++中，可用类(包括模板类)的说明表示ADT，用类的实现来实现ADT
- 反映了程序设计的两层抽象：
  - ADT相当于在概念层(抽象层)上描述问题
  - 类相当于在实现层上描述问题
  - 通过操作对象来解决实际问题，可看作应用层

22

§1.2 算法和算法分析

1. 算法(Algorithm)

- **算法**：是对特定问题求解步骤的一种描述，是指令的有限序列。
- 例：欧几里德算法：给定两个正整数M和N，求它们的最大公因子？输入整数M和N，R存放M除N的余数
  - 求余数：R=M / N余数
  - 判断余数：R=0吗？不是，则转到第4步
  - 余数R=0，输出最大公因子N，运算结束
  - R，N互换数据，返回1，继续循环。
- 一个算法就是一个有穷规则的集合，规则规定了解决某特定问题的运算序列。

23

2. 算法的特性

- **有穷性**——算法必须在执行有穷步之后结束，而且每一步都可在有穷时间内完成。
- **确定性**——每条指令无二义性。
- **可行性**——算法中描述的每一操作，都可以通过已经实现的基本运算来实现。
- **输入**——算法有零至多个输入。
- **输出**——算法有一个至多个输出。

24

3. 算法的描述工具

- 自然语言
- 计算机程序语言
- 类程序语言(例如：类C语言)
- 其他语言

注意：唯一的要求是该说明必须精确地描述计算过程

25

4. 算法设计的要求

- 正确性(Correctness)
  - 程序不含语法错误
  - 程序对于几组输入数据能得出满足要求的结果
  - 程序对于精心选择的典型、苛刻的输入数据能够得出满足要求的结果
  - 程序对于一切合法的输入数据都能够得出满足要求的结果。
- 可读性(Readability)
- 健壮性(Robustness)
- 效率与低存储量要求

26

5. 算法效率的度量

- **效率**：指时间和空间的利用率。
- 算法效率的衡量
  - 事后统计/事前分析估计
  - 与效率相关的因素
    - 算法策略的选择
    - 问题的规模
    - 程序语言的选择
    - 编译程序产生机器代码质量
    - 机器执行指令速度
- 算法效率的衡量指标：时间复杂度和空间复杂度

27

时间复杂度

- **语句的频度**：规模为n的算法重复执行该语句的次数。一般记为f(n)。
  - n为问题的规模或大小，一般指算法中数据元素的数量
- 每条语句的执行时间=语句的执行次数(频度)×语句执行一次所需时间
  - 为独立于软、硬件系统，可设每条语句执行一次所需的时间均是单位时间
- 算法时间复杂度=算法中所有语句频度之和
  - 一般记为T(n)。

28

渐进时间复杂度

- 一般算法的时间度量依据算法中**最大语句频度**来估算。
- **渐进时间复杂度**
- T(n)=O(f(n))
  - 若T(n)和f(n)是定义在正整数集合上的两个函数，则T(n)=O(f(n))表示存在正的常数c和n，当$n \geq n_0$时，都满足$0 \leq T(n) \leq c \cdot f(n)$；即，这两个函数当整型变量n趋于无穷大时，两者的比值是一个不等于0的常数。
  - 例如：$f(n)=3n^2$，$T(n)=O(f(n))=n^2$

29

空间复杂度

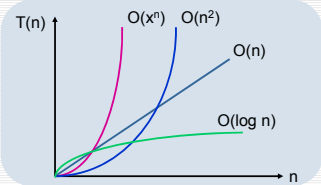
- **空间复杂度**：当问题的规模以某种单位由1增至n时，解决该问题的算法实际所使用的空间也以某种单位由1增至f(n)，则称该算法的空间代价是f(n)，或空间复杂度为f(n)
- S(n)=O(f(n))

注意：在讨论算法的空间复杂度时，一般只分析算法执行过程中所需要的辅助空间，即不包括存储输入数据和输出数据所需要的空间

30

时间复杂度的等级

- O(1)、O(log n)、O(n)、O(nlog n)、$O(n^2)$、$O(n^k)$、$O(x^n)$
- 关系：
  - $O(1)<O(\log n)<O(n)<O(n\log n)<O(n^2)<O(n^3)$
  - $O(2^n)<O(n!)<O(n^n)$



举例1

- { x=x+1; s=0 }　　复杂度：O(1) 常量阶
- for(i=1; i<=n; i++)
  { x=x+1; s+=x; }　　复杂度：O(n) 线性阶
- for(i=1; i<=n; i++)
  for(j=1; j<=n; j++)
  { x=x+1; s+=x; }　　复杂度：$O(n^2)$ 平方阶
- i=1;
  while(i<=n) i=5*i;　　复杂度：$O(\log_5 n)$
- for(i=1; i<=n; i++)
  for(j=1; j<=n; j++)
  { k=1;
  while(k<=n) k=5*k; }　　复杂度：$O(n^2\log_5 n)$

举例2

```
void bubble-sort(int a[], int n)
{  for(i=n-1, change=TRUE; i>1 && change; --i)
   {  change=false;
      for(j=0; j<i; ++j)
        if (a[j]>a[j+1])
          {  a[j]←→a[j+1];
             change=TRUE;  }
   } }
```

- 最好情况：0次(交换)，n-1次(比较)
- 最坏情况：1+2+……+n-1=n(n-1)/2(比较/交换)
- 平均时间复杂度为：$O(n^2)$