Data Structure & Algorithm Analysis

Introduction

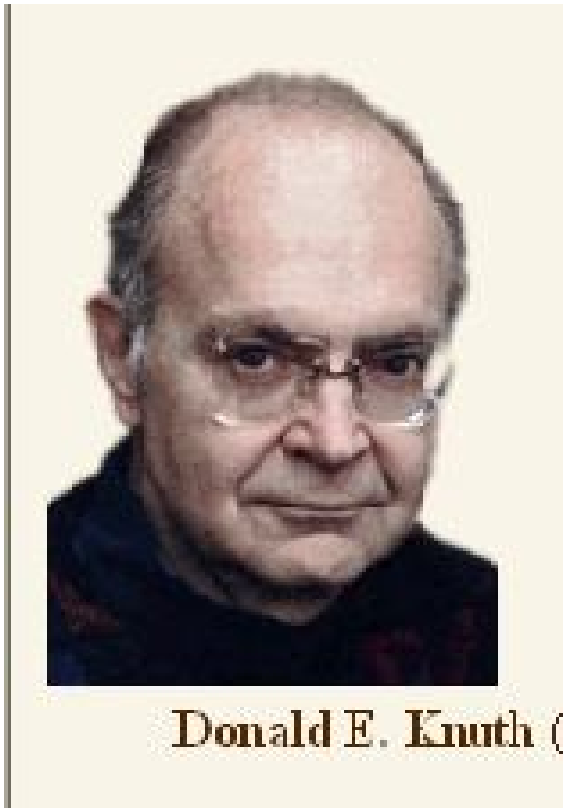
Zibin Zheng（郑子彬）

School of Data and Computer Science，SYSU

http://www.inpluslab.com

课程主页：http://inpluslab.sysu.edu.cn/dsa2016/

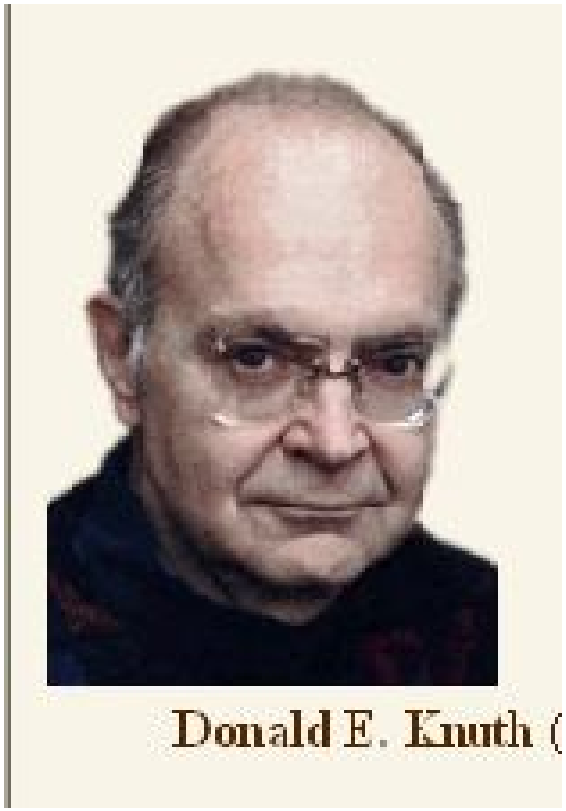
数据结构的创始人——唐纳德·克努斯



- 算法和程序设计技术的先驱者
- 计算机排版系统TEX和METAFONT的发明者

1938年出生，25岁毕业于加州理工学院数学系，博士毕业后留校任教，28岁任副教授。30岁时，加盟斯坦福大学计算机系，任教授。从31岁起，开始出版他的历史性经典巨著：*The Art of Computer Programming*，他计划共写7卷，然而出版三卷之后，已震惊世界，使他获得计算机科学界的最高荣誉——图灵奖，此时，他年仅36岁。

数据结构的创始人——唐纳德·克努斯



《计算机程序设计的艺术》

开始于他念博士期间，计划出七卷，已出版《基本算法》《半数字化算法》《排序与搜索》三卷，第四卷《组合算法》尚在写作之中。

以其内容的丰富和深刻喻为经典，被称为“计算机的圣经”，其发行量创造了计算机类图书的最高记录，直至20世纪80年代中期，都一直保持着月销售量每卷达2000册的势头，成为Addison-Wesley出版社成立以来销路最好的图书。

数据结构的创始人——唐纳德·克努斯

TEX排版软件 & METAFONT字型设计软件

- 对整个西文印刷行业带来了革命性变革
- 1986年度的软件系统奖(Software System Award)
- 作为自由软件无偿提供给用户
- TEX的版本号从3开始，不断地逼近圆周率 （3.14，3.141…目前最新版本是3.14159265）。小修小补，趋近完美。
- 专门设立奖金：谁发现TEX的一个错误，就付他2.56美元，第二个错误5.12美元，第三个10.24美元…以此类推。直到今天，他也没有为此付出多少钱。
- “为了这个TEX，不妨再给克努斯一个图灵碗吧。”

数据结构的创始人——唐纳德·克努斯

乌托邦84 & 作文式程序设计(literate programming)

- “乌托邦84‘’(Utopia 84)：一种理想语言，希望它有更好的数据结构和控制结构，更符合结构化程序设计的思想等 。
- “作文式程序设计”(literate programming)：所谓作文式程序设计就是要像写作文那样进行编程，从完成的“源程序”中既可提炼出可执行的程序代码，又可生成程序文档。
- 第一个作文式程序设计系统WEB：一个子系统抽出算法部分，编译得到可执行代码。另一个子系统加工得到具有高度可读性的完整的程序文档。此外，WEB能够自动产生目录和索引。

- 编程时先要把程序逻辑弄好，在程序中加入必要的注释和说明，以便阅读和理解。

数据结构的创始人——唐纳德·克努斯

数据结构 & 算法

- 提出计算机科学技术中两个最基本概念：“算法”(Algorithm)和“数据结构”(Data Structure)
- 首创双向链表
- 在编译器设计方面，发明LR(k)文法、属性文法(attribute grammar)等
- Knuth-Bendix算法：考察数学公理及其推论是否“完全”而构造标准重写规则集(rewriting rule set)的算法，曾成功地用它解决了群论中的等式的证明问题，是定理机器证明的一个范例。
- Knuth-Morris-Pratt算法(KMP)：查找字符串的简单高效算法
- 设计与实现过最早的随机数发生器(random number generator)

数据结构在程序设计中的作用

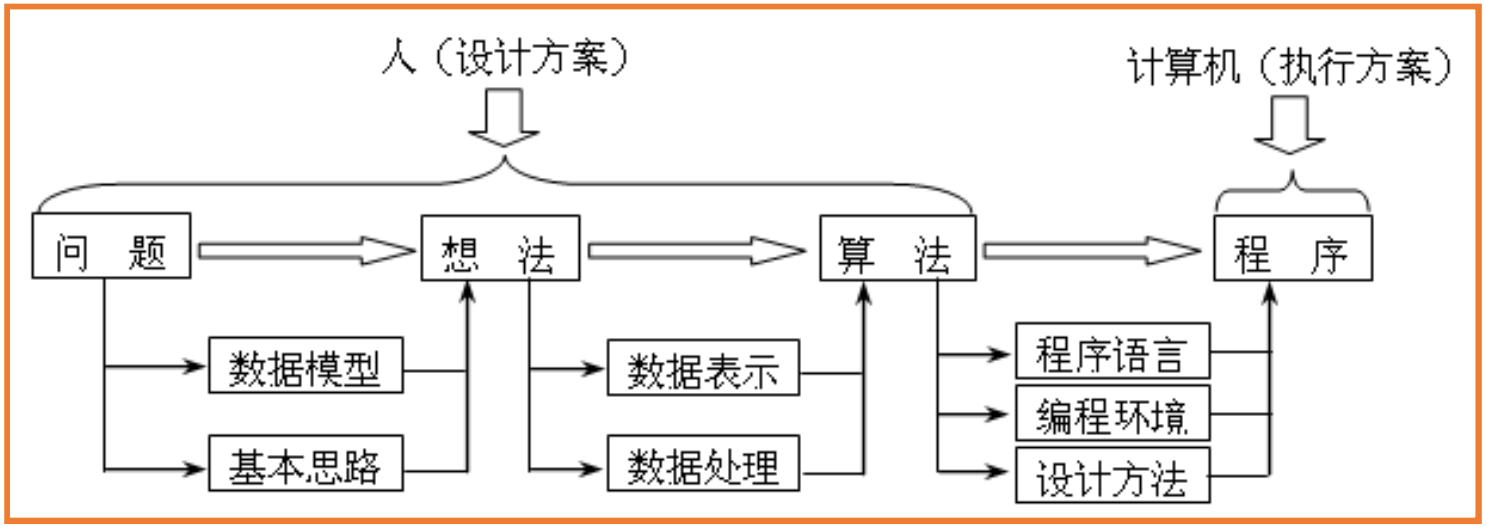
程序设计的实质是什么?

数据表示：将数据存储在计算机（内存）中
数据处理：处理数据，设计方案（算法）

数据结构问题起源于程序设计

数据结构在程序设计中的作用

利用计算机求解问题的一般过程?



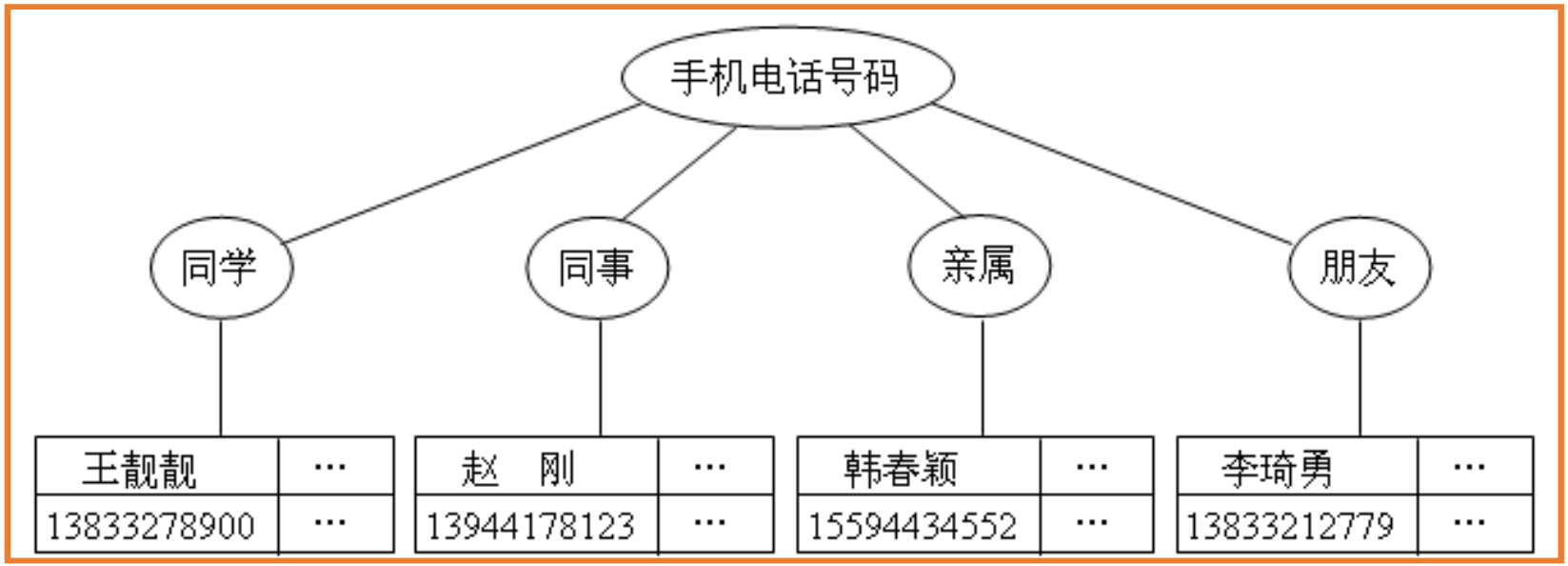
计算机不能分析问题并产生问题的解决方案，必须由人来分析问题，确定问题的解决方案，编写程序，然后让计算机执行程序最终获得问题的解。

数据结构在程序设计中的作用

例1-1 手机电话号码查询问题

将电话号码集合组织成线性结构和树结构，查找操作的效率不同，当数据量较大时差别就更大。

姓名	王靓靓	赵　刚	韩春颖	李琦勇	……	张　强
电话	13833278900	13944178123	15594434552	13833212779	……	13331688900



本课程讨论的主要内容

- 计算机求解问题:

  问题→抽象出问题的模型→求模型的解

- 问题——数值问题、非数值问题

  数值问题→数学方程

  非数值问题→数据结构

本课程讨论的主要内容

例1-2 学籍管理问题

完成什么功能?各表项之间是什么关系？

学　号	姓　名	性　别	出生日期	政治面貌
0001	陆　宇	男	1986/09/02	团员
0002	李　明	男	1985/12/25	党员
0003	汤晓影	女	1986/03/26	团员
⋮	⋮	⋮	⋮	⋮

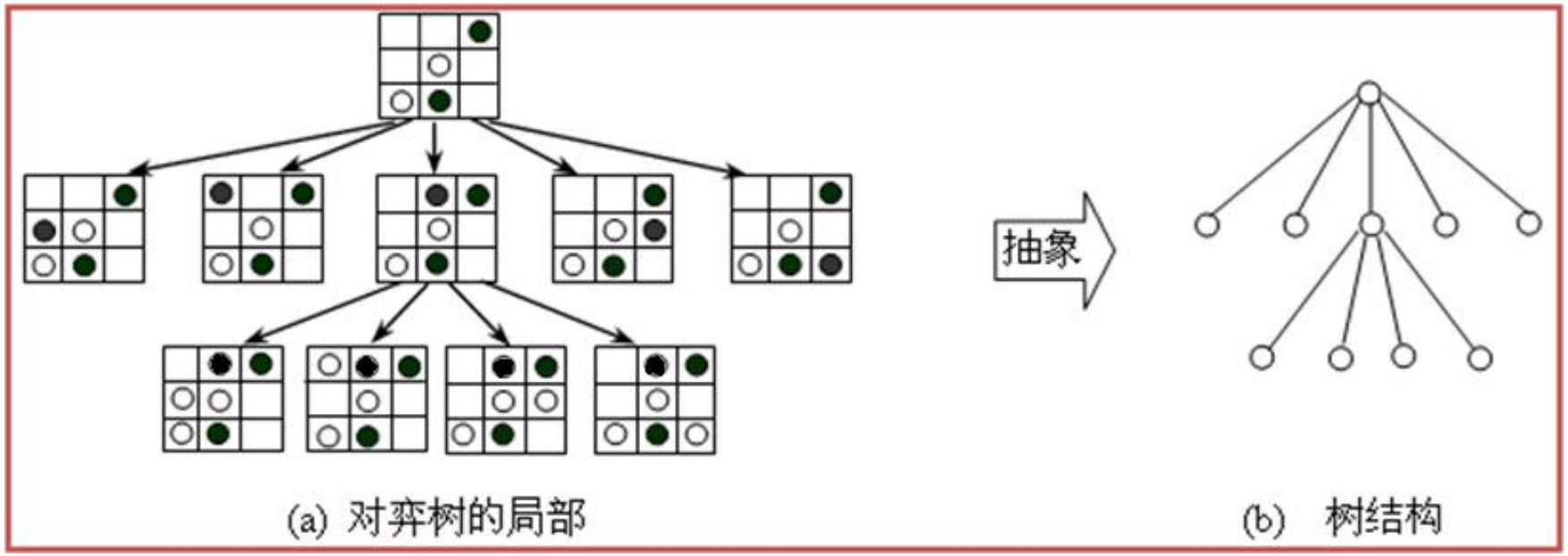
(a) 学生学籍登记表

(b) 线性结构

本课程讨论的主要内容

例1-3 人—机对弈问题

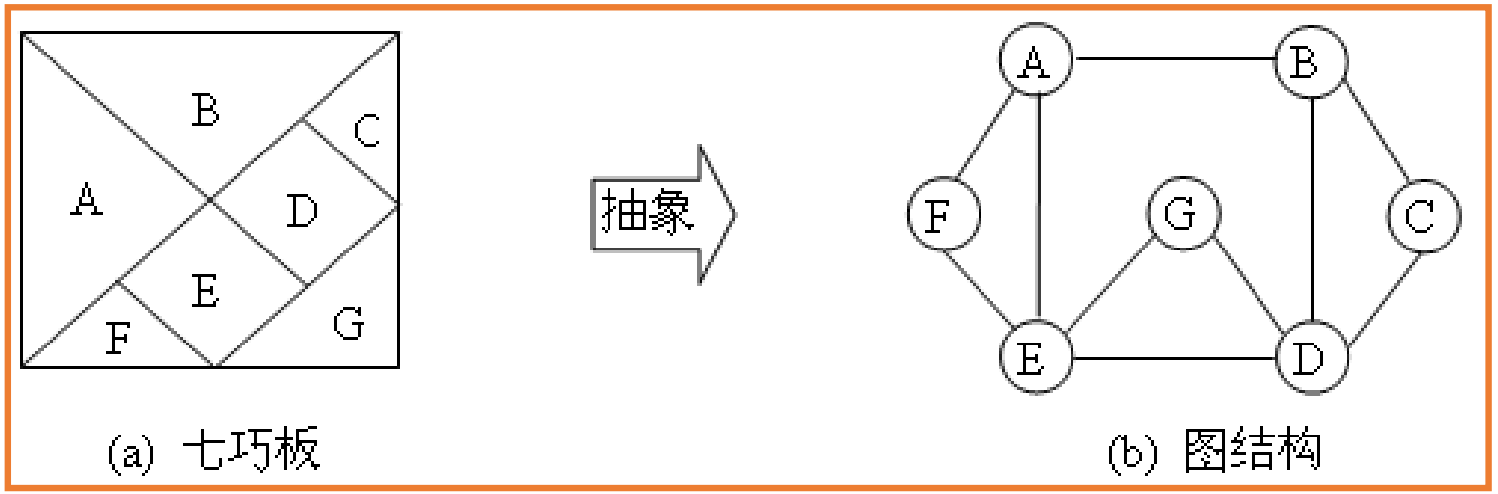
如何实现对弈？各格局之间是什么关系？



本课程讨论的主要内容

例1-4 七巧板涂色问题

如何表示区域之间的邻接关系？



本课程讨论的主要内容

本课程讨论非数值问题的数据组织和处理，主要内容如下：

（1）数据的逻辑结构：线性表、树、图等数据结构，其核心是如何组织待处理的数据以及数据之间的关系；

（2）数据的存储结构：如何将线性表、树、图等数据结构存储到计算机的存储器中，其核心是如何有效地存储数据以及数据之间的逻辑关系；

（3）算法：如何基于数据的某种存储结构实现插入、删除、查找等基本操作，其核心是如何有效地处理数据；

（4）常用数据处理技术：查找技术、排序技术、索引技术等。

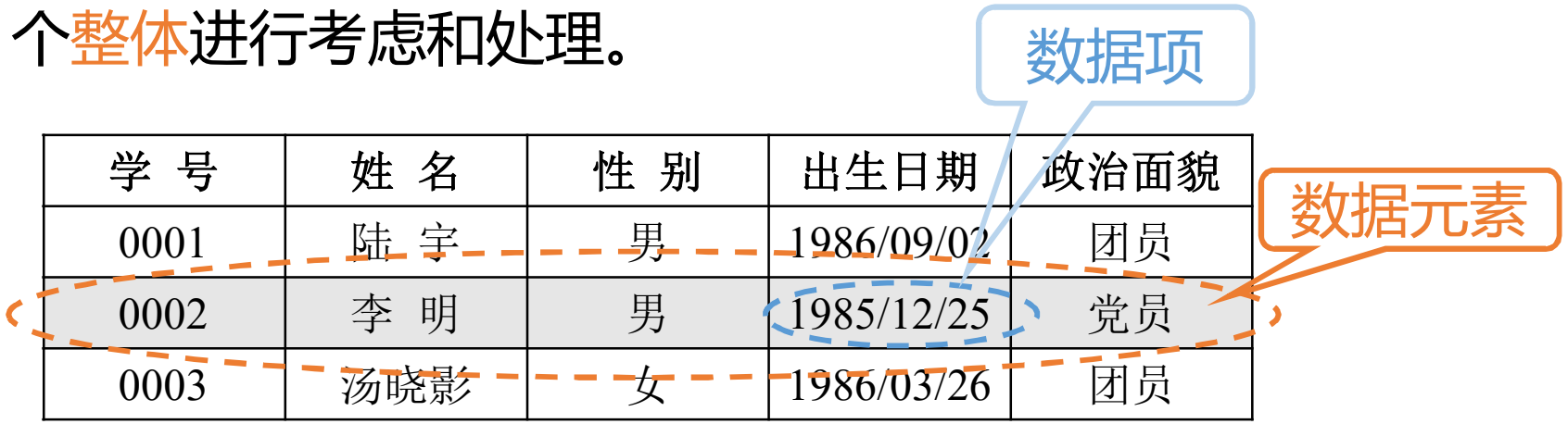
数据结构的基本概念

- 数据：所有能输入到计算机中并能被计算机程序识别和处理的符号集合。

  数值数据：整数、实数等

  非数值数据：图形、图象、声音、文字等

- 数据元素：数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。



学 号	姓 名	性 别	出生日期	政治面貌
0001	陆 宇	男	1986/09/02	团员
0002	李 明	男	1985/12/25	党员
0003	汤晓影	女	1986/03/26	团员

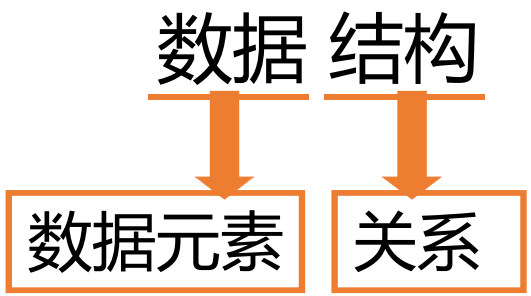
- 数据项：构成数据元素的不可分割的最小单位。

数据结构的基本概念

数据、数据元素、数据项之间的关系

✓包含关系：数据由数据元素组成，数据元素由数据项组成。

✓数据元素是讨论数据结构时涉及的最小数据单位，其中的数据项一般不予考虑。



数据结构的基本概念

数据结构：相互之间存在一定关系的数据元素的集合。
按照视点的不同，数据结构分为逻辑结构和存储结构。

➢逻辑结构：指数据元素之间逻辑关系的整体。

关联方式或邻接关系


学籍管理问题中，表项之间的逻辑关系指的是什么？


人机对弈问题中，格局之间的逻辑关系指的是什么？


七巧板涂色问题中，课程之间的逻辑关系指的是什么？

数据的逻辑结构是从具体问题抽象出来的数据模型

数据结构的基本概念

数据结构：相互之间存在一定关系的数据元素的集合。
按照视点的不同，数据结构分为逻辑结构和存储结构。

- 逻辑结构：指数据元素之间逻辑关系的整体。

数据的逻辑结构在形式上可定义为一个二元组：

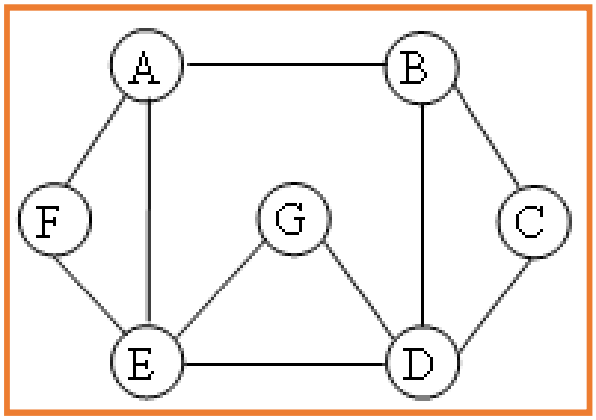
Data_Structure = (D, R)

其中D是数据元素的有限集合，R是D上关系的集合。

数据结构的基本概念

数据结构：相互之间存在一定关系的数据元素的集合。
按照视点的不同，数据结构分为逻辑结构和存储结构。

➢逻辑结构：指数据元素之间逻辑关系的整体。



Data_Structure = (D, R)

其中D = {*A*, *B*, *C*, *D*, *E*, *F*, *G*}

R = {R1} , R1 = {<*A*, *B*>, <*A*, *E*>, <*A*, *F*>, <*B*, *C*>, <*B*, *D*>, <*C*, *D*>, <*D*, *E*>, <*D*, *G*>, <*E*, *F*>, <*E*, *G*>}

?

数据结构的基本概念

数据结构：相互之间存在一定关系的数据元素的集合。
按照视点的不同，数据结构分为逻辑结构和存储结构。

- 逻辑结构：指数据元素之间逻辑关系的整体。
- 存储结构：又称为物理结构，是数据及其逻辑结构在计算机中的表示。

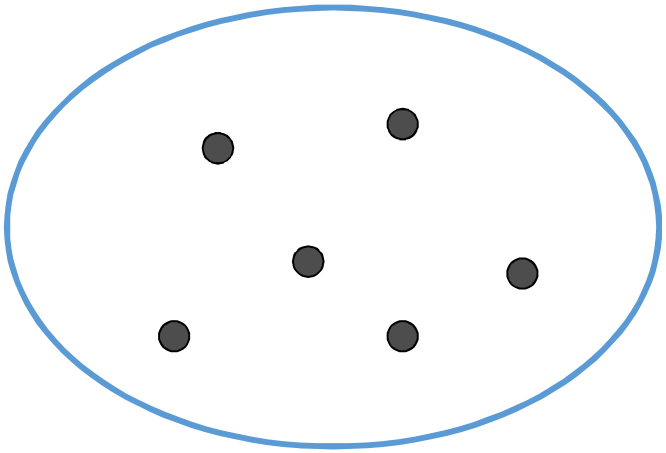
内存

存储结构实质上是内存分配，在具体实现时依赖于计算机语言。

逻辑结构

数据结构从逻辑上分为四类：

- 集合：数据元素之间就是“属于同一个集合”；



逻辑结构

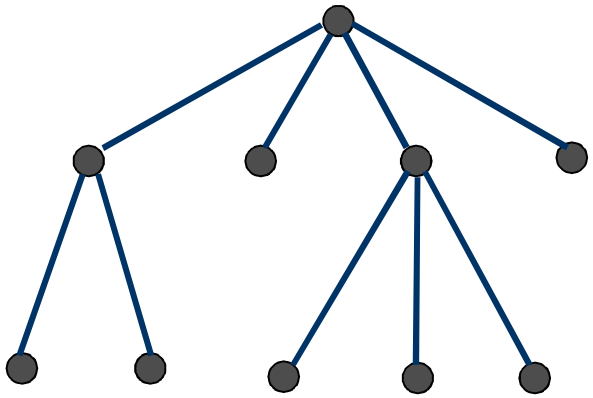
数据结构从逻辑上分为四类：

- 集合：数据元素之间就是“属于同一个集合”；
- 线性结构：数据元素之间存在着一对一的线性关系；

逻辑结构

数据结构从逻辑上分为四类：

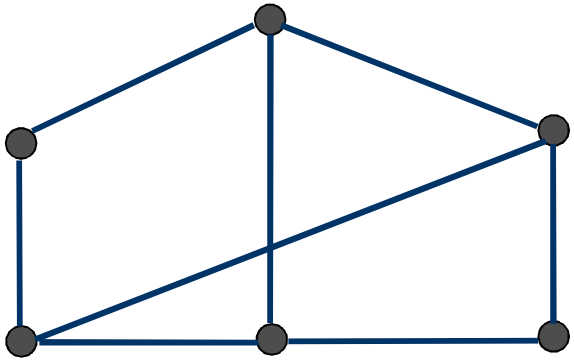
- 集合：数据元素之间就是“属于同一个集合”；
- 线性结构：数据元素之间存在着一对一的线性关系；
- 树结构：数据元素之间存在着一对多的层次关系；



逻辑结构

数据结构从逻辑上分为四类：

- 集合：数据元素之间就是“属于同一个集合”；
- 线性结构：数据元素之间存在着一对一的线性关系；
- 树结构：数据元素之间存在着一对多的层次关系；
- 图结构：数据元素之间存在着多对多的任意关系。

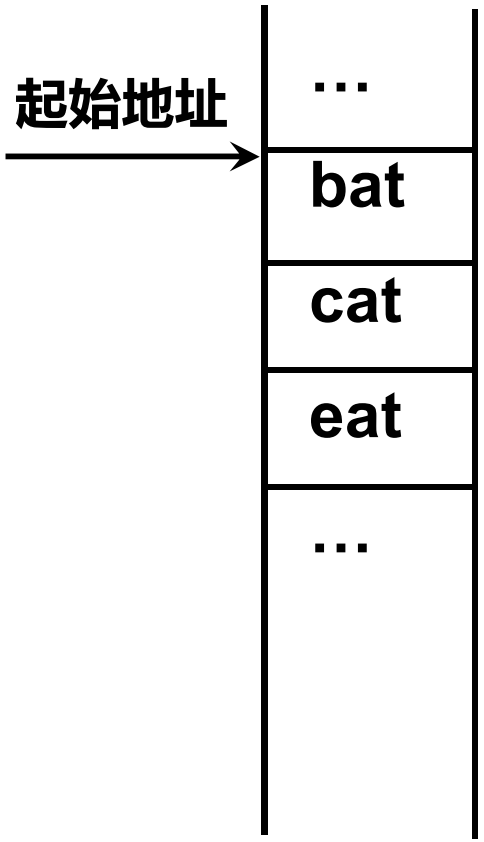


存储结构

通常有两种存储结构：

- 顺序存储结构：用一组连续的存储单元依次存储数据元素，数据元素之间的逻辑关系由元素的存储位置来表示。

例：（bat, cat, eat）

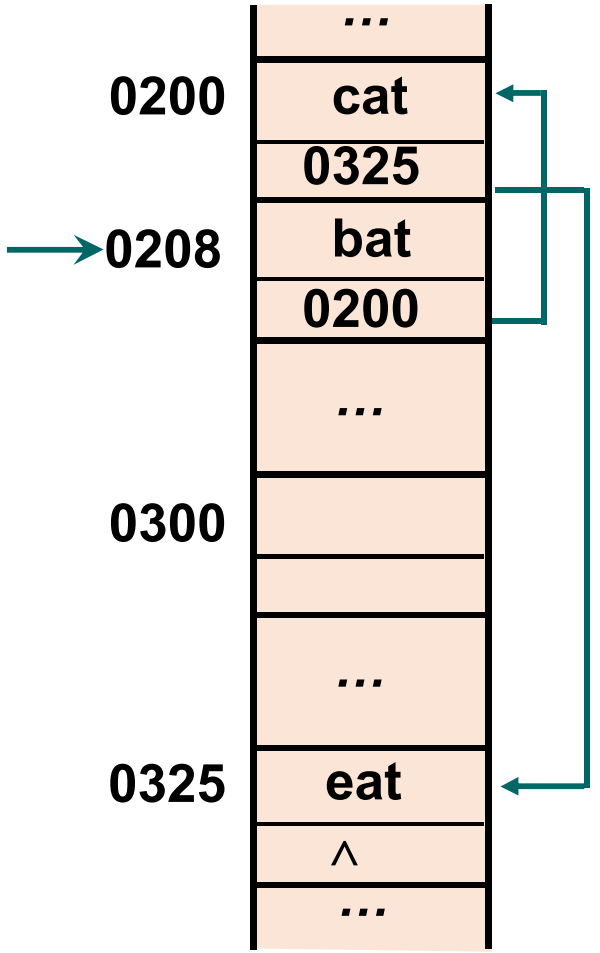


存储结构

通常有两种存储结构：

- 顺序存储结构：用一组连续的存储单元依次存储数据元素，数据元素之间的逻辑关系由元素的存储位置来表示。
- 链接存储结构：用一组任意的存储单元存储数据元素，数据元素之间的逻辑关系用指针来表示 。

例：（bat, cat, eat）



逻辑结构和存储结构之间的关系

- 数据的逻辑结构属于用户视图，是面向问题的，反映了数据内部的构成方式；
- 数据的存储结构属于具体实现的视图，是面向计算机的。

- 一种数据的逻辑结构可以用多种存储结构来存储，而采用不同的存储结构，其数据处理的效率往往是不同的。

分逻辑结构及存储结构有什么好处？

抽象数据类型

- 数据类型（Data Type）：

  一组值的集合以及定义于这个值集上的一组操作的总称

  例如：C++中的整型变量

- 抽象（Abstract）：

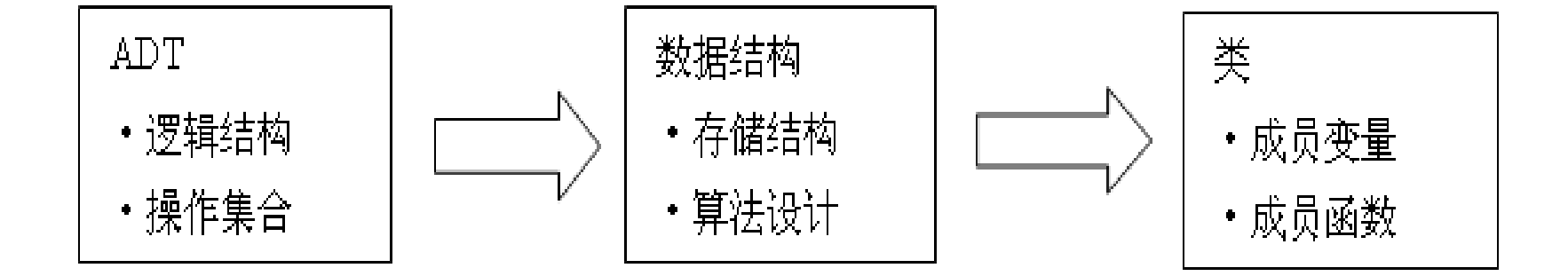
  抽出问题本质的特征而忽略非本质的细节

  例如： 地图、驾驶汽车

- 抽象数据类型（Abstract Data Type，ADT）：

  一个数学模型以及定义在该模型上的一组操作的总称

抽象数据类型



- **在设计ADT时，把ADT的定义、设计和实现分开来**
- **定义部分只包含数据的逻辑结构和所允许的操作集合**
  - **ADT的使用者依据这些定义来使用ADT，即通过操作集合对该ADT进行操作**
  - **ADT的实现者依据这些定义来完成该ADT各种操作的具体实现**

抽象数据类型

ADT 抽象数据类型名
Data
　　数据元素之间逻辑关系的定义
Operation
　　操作1
　　　　前置条件：执行此操作前数据所必须的状态
　　　　输　　入：执行此操作所需要的输入
　　　　功　　能：该操作将完成的功能
　　　　输　　出：执行该操作后产生的输出
　　　　后置条件：执行该操作后数据的状态
　　操作2
　　　　……
　　……
　　操作n
　　　　……
endADT

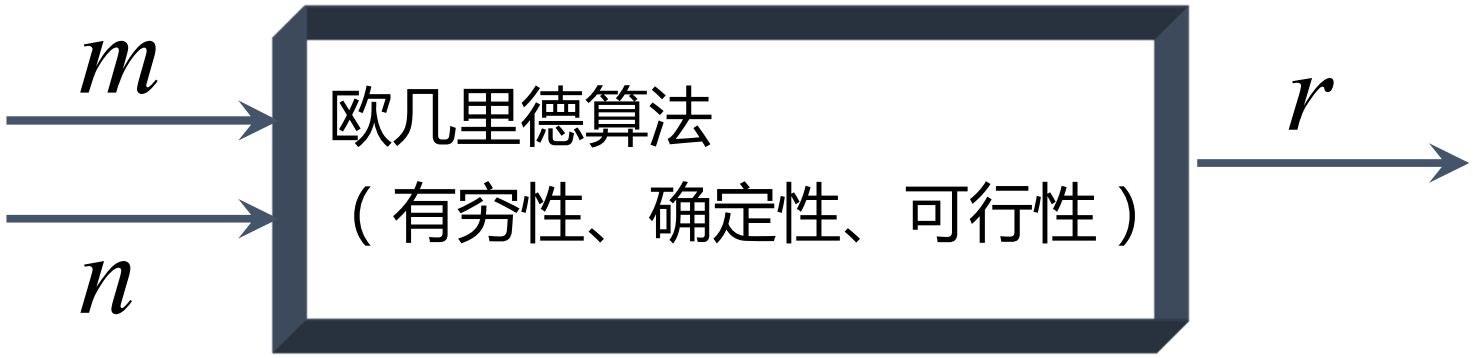
算法的相关概念

- 算法（Algorithm）:是对特定问题求解步骤的一种描述，是指令的有限序列。
- 算法的五大特性：
  - 输入：一个算法有零个或多个输入。
  - 输出：一个算法有一个或多个输出。
  - 有穷性：一个算法必须总是在执行有穷步之后结束，且每一步都在有穷时间内完成。
  - 确定性：算法中的每一条指令必须有确切的含义，对于相同的输入只能得到相同的输出。
  - 可行性：算法描述的操作可以通过已经实现的基本操作执行有限次来实现。

A Good Algorithm

- 正确性(Correctness)
  - 指算法至少应该具有输入、输出和加工处理无歧义性，能正确反映问题的需求，能够得到问题的正确答案。
- 可读性(Readability)
  - 算法设计的另一目的是为了便于阅读、理解和交流。
- 健壮性(Robustness)
  - 当输入数据不合法时，算法也能做出相关处理，而不是产生异常或莫名其妙的结果。
- 效率(Efficiency)
  - 具有时间效率高和存储量低的需求。

算法的例：欧几里德算法——辗转相除法求两个自然数 m 和 n 的最大公约数



算法的描述方法

1. 自然语言

优点：容易理解

缺点：冗长、二义性

使用方法：粗线条描述算法思想

注意事项：避免写成自然段

算法的描述方法

例：欧几里德算法—自然语言描述算法

自然语言

① 输入m 和n；

② 求m除以n的余数r；

③ 若r等于0，则n为最大公约数，算法结束；

否则执行第④步；

④ 将n的值放在m中，将r的值放在n中；

⑤ 重新执行第②步。

算法的描述方法

2. 流程图

优点：流程直观

缺点：缺少严密性、灵活性

使用方法：描述简单算法

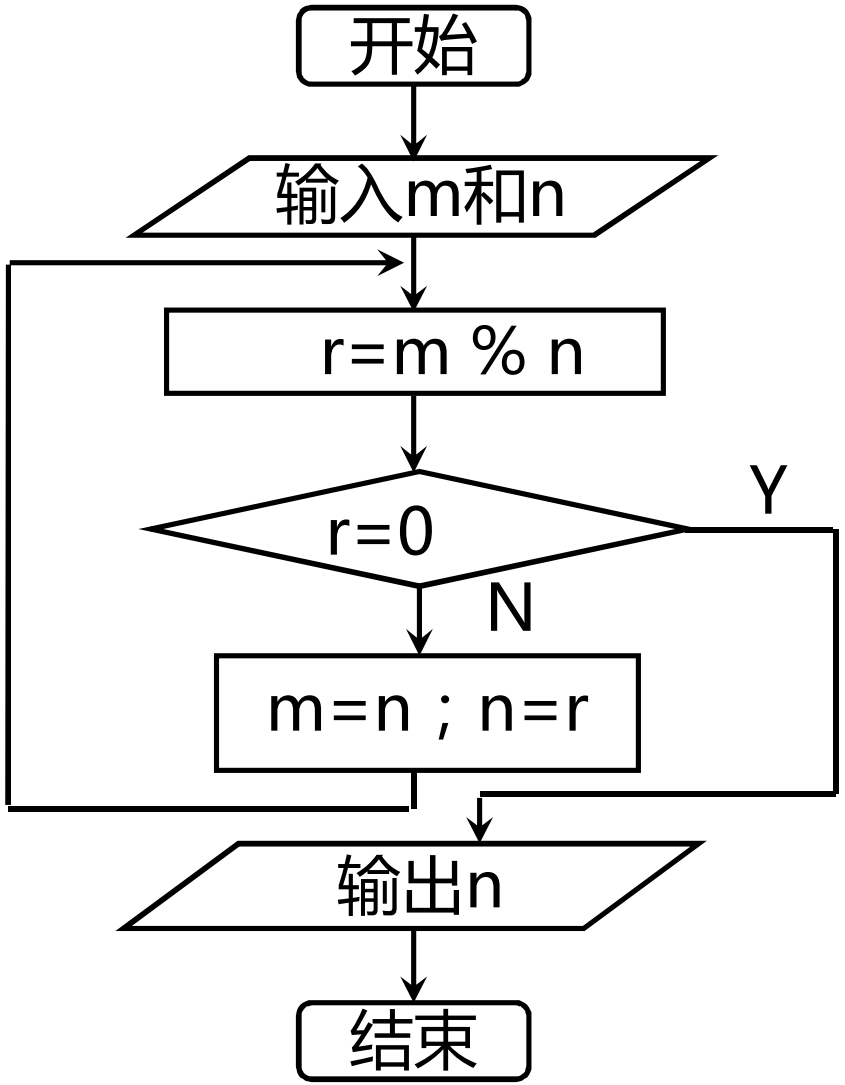
注意事项：注意抽象层次

算法的描述方法

例：欧几里德算法

—流程图描述算法

流程图



算法的描述方法

3. 程序设计语言

优点：能由计算机执行

缺点：抽象性差，对语言要求高

使用方法：算法需要验证

注意事项：将算法写成子函数

算法的描述方法

例：欧几里德算法—程序设计语言描述算法

程序设计语言

```
#include <iostream.h>
int CommonFactor(int m, int n)
{
    int r=m % n;
    while (r!=0)
    {
       m=n;
       n=r;
       r=m % n;
    }
    return n;
}
void main( )
{
   cout<<CommonFactor(63, 54)<<endl;
}
```

算法的描述方法

4. 伪代码

伪代码（Pseudo Code）：介于自然语言和程序设计语言之间的方法，它采用某一程序设计语言的基本语法，操作指令可以结合自然语言来设计。

优点：表达能力强，抽象性强，容易理解

算法的描述方法

例：欧几里德算法

伪代码

1. r = m % n;
2. 循环直到 r 等于0
   2.1 m = n;
   2.2 n = r;
   2.3 r = m % n;
3. 输出 n ;

上述伪代码再具体一些，用C++的函数来描述。

谢谢！