数 据 结 构

徐跃东 主讲

信息科学与工程学院

第一章 绪论

- 数据结构概念
- 算法及其描述
- 算法效率分析
- 小结

数据结构概念 - 前传

- 无论你是电工、通信、微电子、计算机等方面的专业人才，是否都要与数据及信息打交道？
- 什么样的问题需要涉及到数据结构及算法？

数据结构概念 - 前传

- i) 表格问题
  - 统计男女比例、年龄、学历

表1.1 某机构的人员情况表

编号	姓名	性别	民族	出生日期	工作时间	文化程度	专业
28011	李立	男	汉	1964	1988.9	大学	图书管理
29121	张媛媛	女	汉	1970	2002.8	大学	机械制造
…	…	…	…	…	…	…	…
23579	王云坤	男	满族	1960	1981.9	硕士	数学

数据结构概念 - 前传

- ii) 排序问题
  - 对多个元素从大到小排列

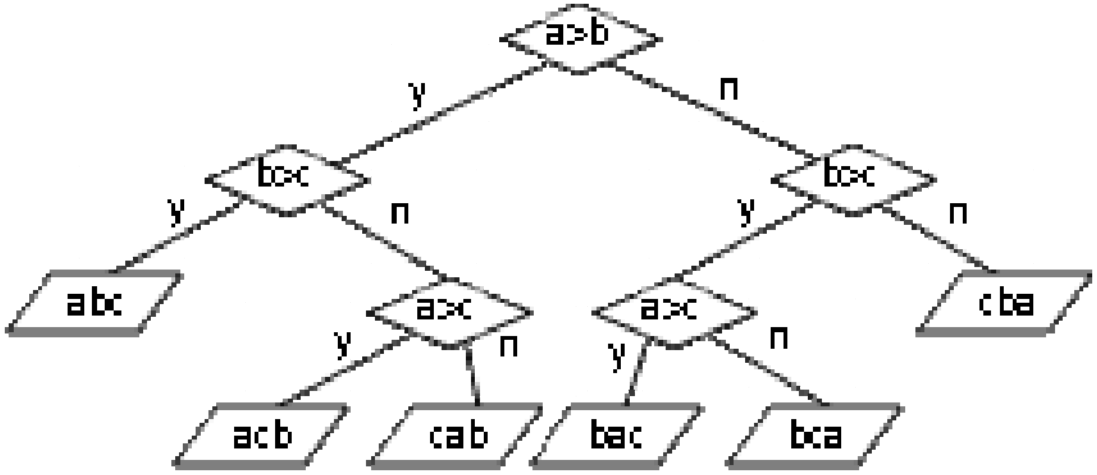
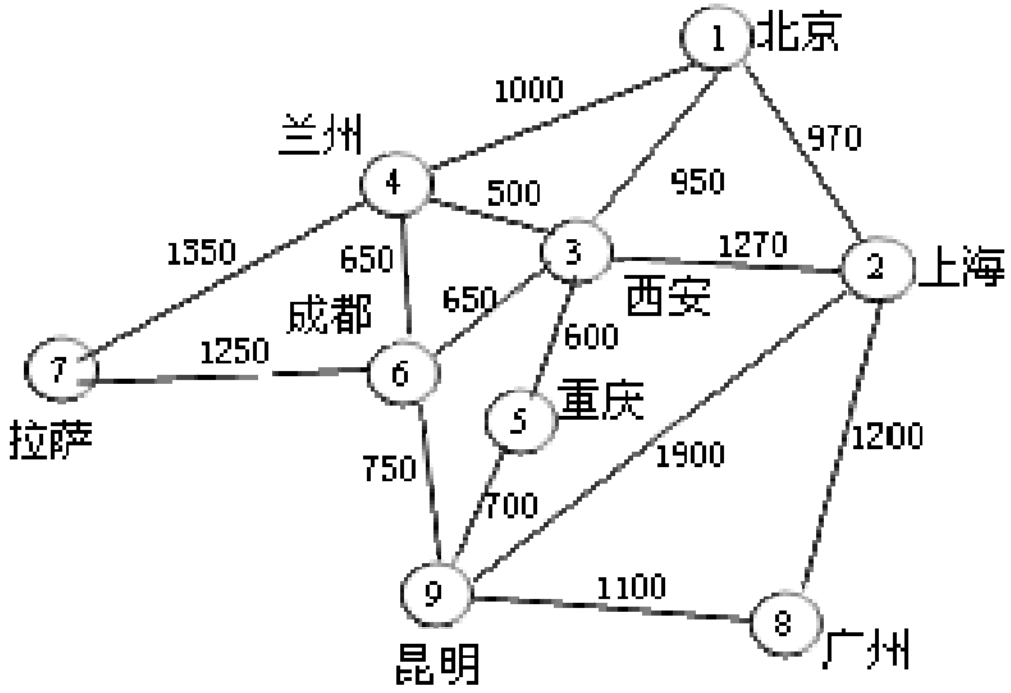


图 1.2 三元素从大到小排序的程序流程图

数据结构概念 - 前传

- iii) 路由问题
  - 从北京到昆明旅游，如何选路？



数据结构概念 - 前传

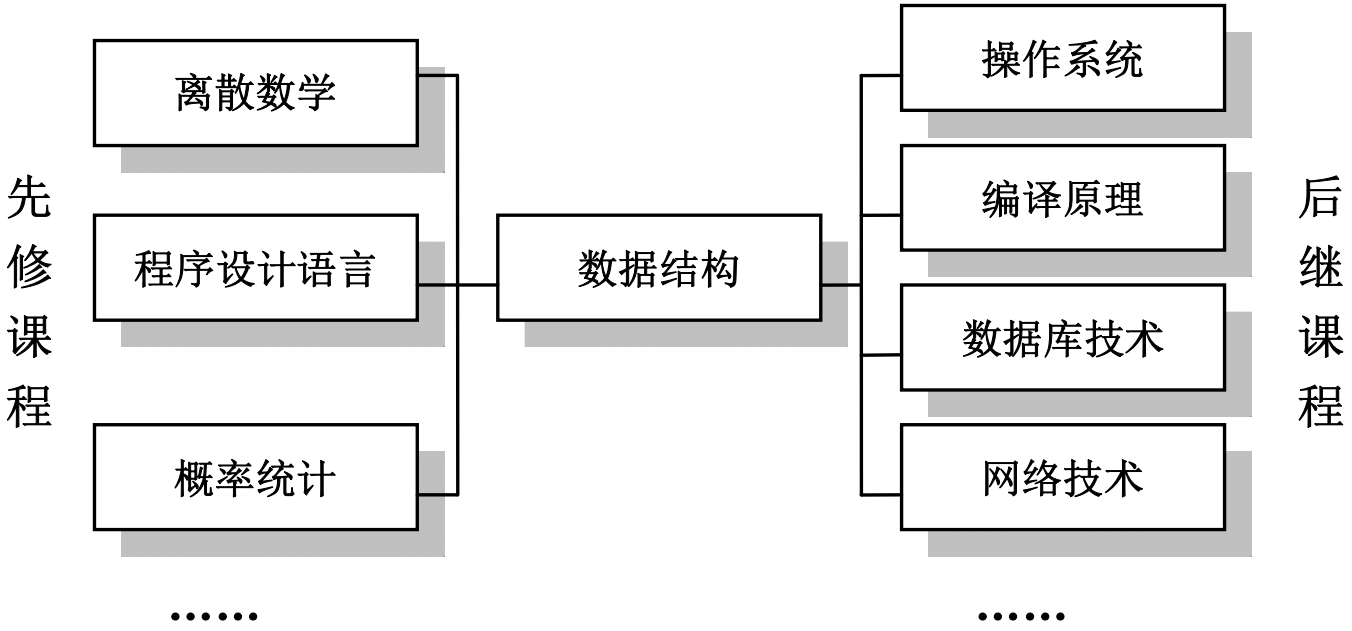
- iv) 七桥问题
  - 在哥尼斯堡的一个公园里，有七座桥将普雷格尔河中两个岛及岛与河岸连接起来(如图)。问是否可能从这四块陆地中任一块出发，恰好通过每座桥一次，再回到起点？欧拉于1736年研究并解决了此问题。(开创了图论与几何拓扑学)

数据结构概念 - 前传

- v) 互联网视频业务
  - 根据用户观看记录做视频推荐
  - 根据用户观看记录预测视频流行度
  - 挑战：2亿条观看记录/天，数千万活跃用户，近千万条视频。用什么样的编程语言、构建何种数据结构、设计何种查找及排序算法？

数据结构概念 - 前传

- 数据结构的地位
  - 在整个课程体系中处于承上启下的地位，是计算机科学的核心课程。



数据结构概念

- **信息：**是人类为了生存，对各种有用的知识、技能、劳作的记录的总结。它们或使用有形的形式，如图形、文字；或使用无形的东西，如音乐、语言等。如农历的24节气、历书、甲骨文、恒星运动、太阳的东升西落、力学、数学等，都是信息的典型表现形式。今天，信息无所不在，无所不容。

数据结构概念

- **数据：**是对信息的一种符号表示，是指所有能输入到计算机中并被计算机程序处理的符号的总称。它们包括文字、数字、声音、图像等。数据与计算机的处理密切相关，是那些能够编码而被计算机处理的信息。数据有不同的等级，如位(bit)、字节(byte)、字(word)、数据项(data item)、记录(record)、文件(file)和数据库(database)等。

数据结构概念

- **数据元素：**是数据(集合)中的一个“个体”,是数据的基本单位。例如公司名册的每一条记录就是一个数据元素。
- **数据对象：**是具有相同性质的若干个数据元素的集合。数据对象可以是有限的，也可以是无限的。如：正整数的数据对象：{1，2，3，4，……}是无限的；我国行政省份和直辖市的名称构成的数据对象{北京，天津，上海，河北，……，台湾}是有限的；

数据结构概念

- **数据结构：**是相互之间存在一种或多种特定关系的数据元素的集合。或定义为：按照一定逻辑关系组织，并按照一定存储方法存储在计算机中的，且需要定义一系列运算的数据的集合。
- 数据的**逻辑结构**、**存储结构**和**运算**合称数据结构的三要素。
- 数据结构的形式定义：

$$Data\_Structure = (D, R)$$

其中：D --- 是数据元素的有限集合；

R --- 是D上关系的有限集合。

数据结构概念

- 逻辑结构
  - 例1：写出如下学生表的逻辑结构

学号	姓名	性别	班号
1	张斌	男	9901
8	刘丽	女	9902
34	李英	女	9901
20	陈华	男	9902
12	王奇	男	9901
26	董强	男	9902
5	王萍	女	9901

R={r}={<1,8>,<8,34>,<34,20>,…,<26,5>}

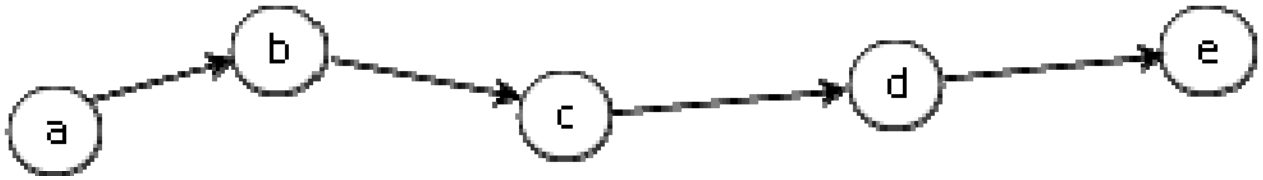
数据结构概念

- 逻辑结构
  - 例2：数据结构 A=（D，R）

    D={a,b,c,d,e }

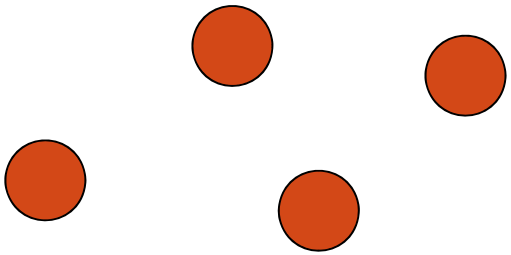
    R={r}={<a,b>,<b,c>,<c,d>,<d,e>}

    请画出此数据结构的逻辑图形。

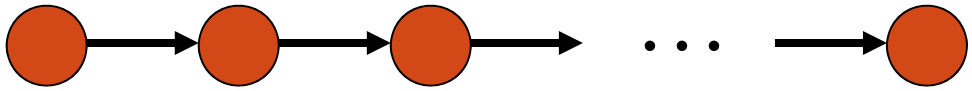


数据结构概念

- 逻辑结构
  - 集合结构

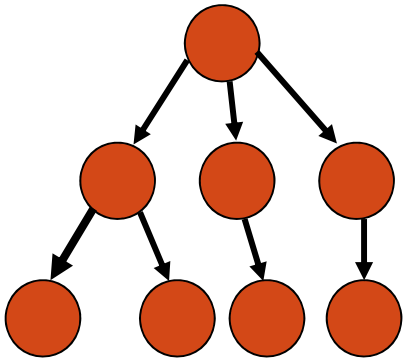


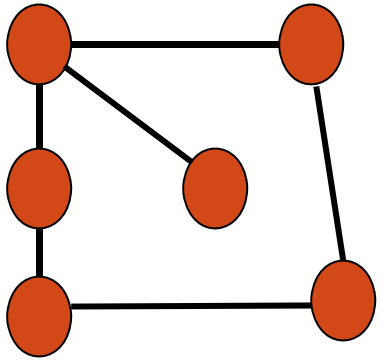
  - 线性结构：开始节点和终端节点都是唯一的，除了开始节点和终端节点以外，其余节点都有且仅有一个前驱节点，有且仅有一个后继节点。



数据结构概念

- 逻辑结构
  - 树形结构：开始节点唯一，终端节点不唯一。除终端节点以外，每个节点有一个或多个后续节点；除开始节点外，每个节点有且仅有一个前驱节点。





  - 图形结构：没有开始节点和终端节点，所有节点都可能有多个前驱节点和多个后继节点。

数据结构概念

- 逻辑结构
  - 例3：有如下数据即一个矩阵，对应的二元组$B=(D,R)$ 如何表示？

$$\begin{bmatrix} 2 & 6 & 3 & 1 \\ 8 & 12 & 7 & 4 \\ 5 & 10 & 9 & 11 \end{bmatrix}$$

答：$D=\{2,6,3,1,8,12,7,4,5,10,9,11\}$

$R=\{r1,r2\}$　其中，$r1$表示行关系，$r2$表示列关系

$r1=\{<2,6>,<6,3>,<3,1>,<8,12>,<12,7>,<7,4>,<5,10>,$
$<10,9>,<9,11>\}$

$r2=\{<2,8>,<8,5>,<6,12>,<12,10>,<3,7>,<7,9>,$
$<1,4>, <4,11>\}$

数据结构概念

- 线性 vs 非线性 数据结构
  - 数据元素之间存在“一一对应”关系：唯一的第一元素、唯一的最后元素、唯一的前驱节点、唯一的后继节点
  - 常用的**线性结构**有：线性表，栈，队列，双队列，数组，串。
  - 常见的**非线性结构**有：二维数组，多维数组，广义表，树(二叉树等)，图。

数据结构概念

- 存储结构

1. **顺序存储：**逻辑上相邻的数据元素存储在物理位置上相邻的存储单元中，这是一种最基本的存储表示方法。
2. **链式存储：**逻辑上相邻的数据元素不要求其物理位置上相邻，数据元素间的逻辑关系通过附设的指针字段值来指示，这是数据结构中最常见的一种存储表示方法。
3. **索引存储：**用上述基本方法存储数据元素同时，还建立一个附加的索引表提高检索的速度和性能。
4. **散列存储：**依据数据元素的关键字值，用散列函数计算出该数据元素的对应的存储地址，依据冲突处理办法实施对数据元素的存储和检索。

数据结构概念

- 存储结构
  - 索引存储：电话号码簿

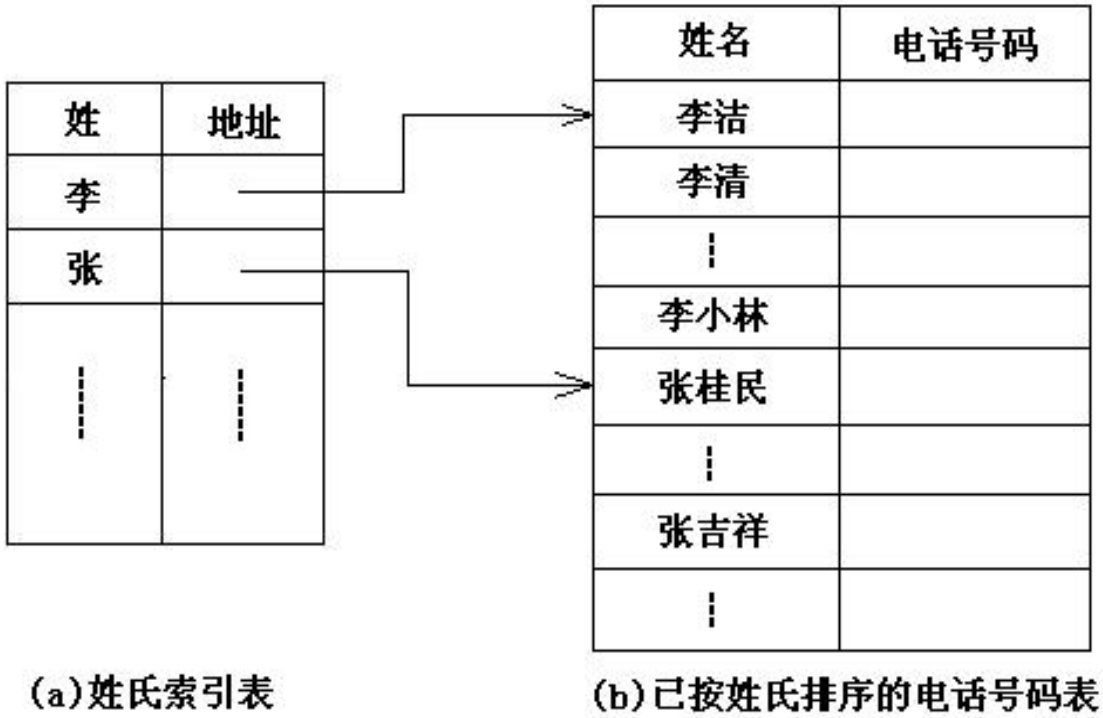


图1.2　电话号码查询问题的索引存储

数据结构概念

- 运算
  - 建立：建立某种指定的数据结构。
  - 清除：把某个指定的数据结构置为空（不存在）。
  - 插入：在数据结构指定位置上插入一个新的数据元素。
  - 删除：在数据结构指定位置上删除一个数据元素。
  - 更新：修改数据结构中某个数据元素的值。
  - 查找：在数据结构中查找满足某种条件的数据元素。
  - 排序：使数据元素按某种指定的次序重新排列。
  - 判空和判满：判定某个数据结构是否为空和判定该数据结构是否已达到逻辑上或存储上的最大允许容量
  - 求长：求指定的数据结构中数据元素的个数。

数据结构概念

- **数据类型：**是一个值的集合和定义在此集合上的一组操作的总称。高级程序语言中,一般须对程序中出现的每个变量、常量或表达式,明确说明它们所属的数据类型。不同类型的变量,其所能取的值的范围不同,所能进行的操作不同。
  - 基本数据类型(int, float, double, bool, char)
  - C/C++指针类型
  - C/C++数组类型
  - C/C++结构体类型
  - C/C++引用类型等

数据结构概念

抽象数据类型（Abstract Data Type简写为ADT）：

指的是用户进行程序设计时从问题的数学模型中抽象出来的逻辑数据结构和逻辑数据结构上的运算，而不考虑计算机的具体存储结构和运算的具体实现算法。

抽象数据类型=逻辑结构＋抽象运算

抽象数据类型实质上就是描述一个求解问题本身（与计算机无关），在理解问题基础上实现用计算机求解问题的过程。

数据结构概念

- **抽象数据类型：**
  - 例4：复数形式：e_1+e_2i或（e_1，e_2）的抽象定义：

ADT Complex
{
数据对象：
D={e_1,e_2 | e_1,e_2均为实数}
数据关系：
R={<e_1,e_2> | e_1是复数的实数部分，e_2 是复数的 虚数部分 }
基本操作：
AssignComplex(&z,v1,v2)：构造复数Z。
DestroyComplex(&z)：复数z被销毁。
GetReal(z,&real)：返回复数z的实部值。
GetImag(z,&Imag)：返回复数z的虚部值。
Add(z1,z2,&sum)：返回两个复数z1、z2的和。
} ADT Complex

第一章 绪论

- 数据结构概念
- 算法及其描述
- 算法效率分析
- 小结

算法及其描述

- **算法：**算法是解决某个特定问题的一种方法或一个过程。
  - 计算机对数据的操作可以分为数值性和非数值性两种类型。在数值性操作中主要进行的是算术运算；而在非数值性操作中主要进行的是检索、排序、插入、删除等。
  - Algorithm（算法）一词最早与欧几里德算法(Euclid's algorithm)联系在一起，这个算法又叫辗转相除法。作为算法的简单引入，我们来讨论欧几里德算法。此算法表述如下：
    - 给定两个正整数m和n，寻找它们的最大公约数，即可以同时整除m和n的最大正整数d。

算法及其描述

- **算法：**

  算法描述如下：

  ①输入两个正整数m,n;

  ②令r等于m除以n的余数；

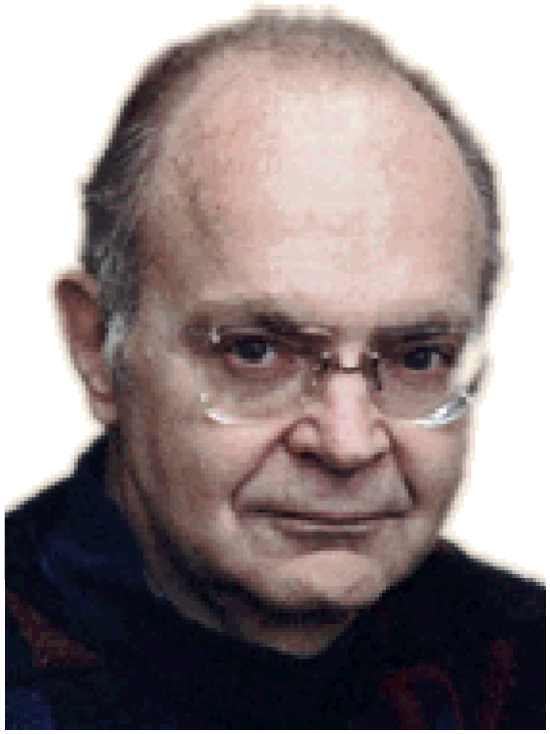
  ③令m←n，n←r， 如果r≠0返回②；否则，运算结束，返回m。

  如果用伪语言描述，则为：

  欧几里德算法

```
Algorithm Euclid(m,n)
  //计算m,n的最大公约数,m,n由调用函数赋值。
do
    r←m mod n
    m←n
    n←r
while(r≠0)
输出 m
```

What is an algorithm?



Donald E. Knuth
(January 10, 1938-)
Turing Award(1974)

“A finite set of rules which gives a sequence of operations for solving a specific type of problem.”

算法是规则的有限集合， 是为解决特定问题而规定的一系列操作。

—— The Art of Computer Programming, Addison-Wesley. Vol 1- *Fundamental Algorithms*

算法及其描述

- 算法的基本特征：
  - **输入：**是指算法要操作的对象。这个输入可以是从键盘输入，也可以是文件读取，或者其它程序的运行结果等多种形式的输入。
  - **有穷性：**一个算法必须在执行了有穷步之后结束。否则陷入了一个死循环，不能称之为一个合理的算法。
  - **确定性：**算法中的每一个指令都必须有确切的含义，不能有二义性。例如分支语句，输入了某个条件，执行语句的哪个分支是明确的，在任何条件下，输入一样，执行的结果也肯定一样。
  - **可行性：**算法中描述的操作是可以通过已实现的操作来完成的。就是使用现有的计算机程序语言是可以完成执行的。
  - **输出：**是指算法执行的结果，与输入存在着某种关联。

算法及其描述

- 算法的基本特征：

描述一

```
void exam1()
 { int  n＝2;
   while (n%2＝＝0)
            n＝n+2;
   printf("%d\n",n);
 }
```

描述二

```
void exam2()
 { int x,y;
   y=0;
   x=5/y;
   printf(“%d,%d\n”,x,y);
 }
```

这两段描述均不能满足算法的特征，试问它们违反了哪些特征？

算法及其描述

- （1）算法是一个死循环，违反了算法的有穷性特征。
- （2）算法包含除零错误，违反了算法的可行性特征

算法及其描述

- 算法的基本要求
  - 正确性。一个算法必须是正确的，即保证能得到正确的结果，否则就不能用它来解决提出的问题，得出的结果也没有意义。
  - 可读性。一个好的算法要便于用户理解和交流。可读性好的程序便于读者理解，也便于调试与修改，这有赖于程序设计员平时编程风格的形成和良好编程习惯的培养。
  - 健壮性。也叫容错能力，即对非法的输入的抵抗能力。当输入一个非法的数据时，是能识别并提示，而不是输出一个错误结果，甚至直接瘫痪。
  - 高效率。算法的效率指的是算法的执行时间。对于同一个问题如果有多种算法可以求解，执行时间短的算法效率高。
  - 存储量需求。算法存储量指的是算法执行过程中所需的最大存储空间。

算法及其描述

- 算法的设计过程
  - 流程图。例5：对一批正整数（N0,N1,……Nm）进行分类，要求如下：
    - ① 当Ni≤20时，归类为第一类
    - ② 当20<Ni≤50时，为第二类
    - ③ 当50<Ni≤100时，为第三类
    - ④ 当100<Ni时，为第四类

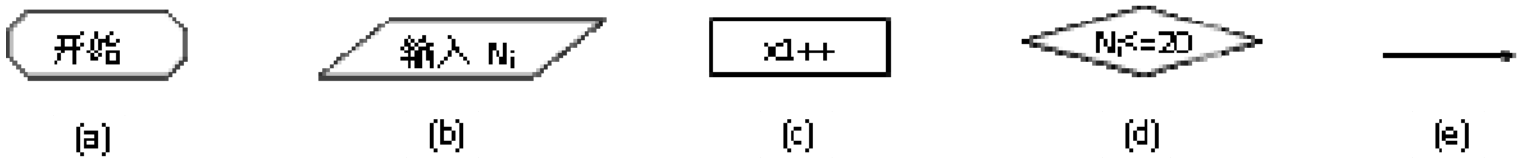


图 1.6　程序流程图符号

算法及其描述

- 算法的设计过程
  - 流程图。

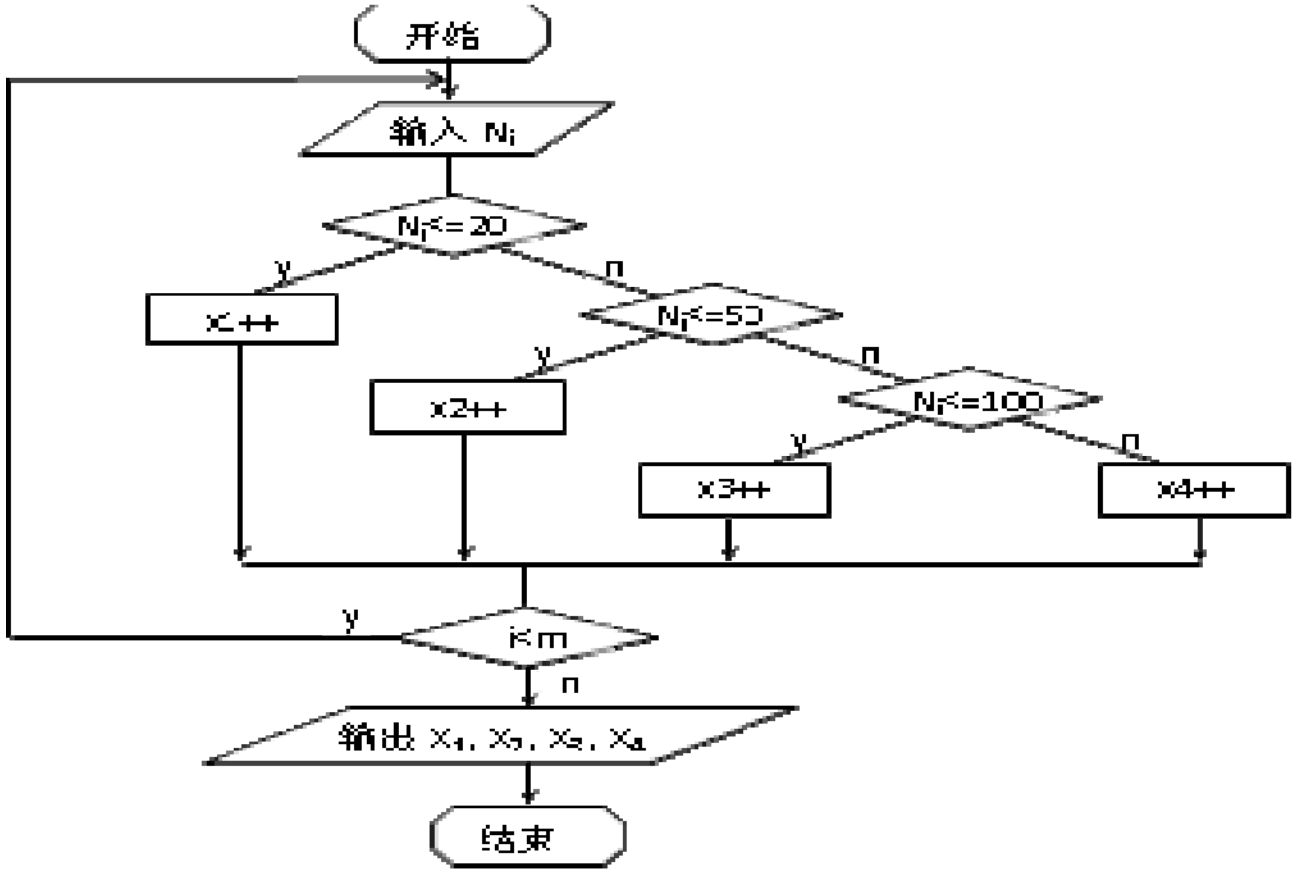


图 1.7 例 1.2 的程序流程图

第一章 绪论

- 数据结构概念
- 算法及其描述
- 算法效率分析
- 小结

算法效率分析

- **算法复杂度分析**：衡量一个算法的优劣主要看算法的执行效率，算法的时间复杂度和空间复杂度分析叫算法分析，其目的是考察算法的运行时间和空间占有情况，以求改进算法或对不同算法的效率进行比较。
  - 可否用算法运行时间来刻画？
    - 算法描述的语言不同
    - 算法执行的环境不同
    - 其他因素

算法效率分析

- **算法执行时间**：一个算法的执行时间大致上等于其所有语句执行时间的总和， 对于语句的执行时间是指该条语句的执行次数和执行一次所需时间的乘积。
- **语句频度**：语句频度是指该语句在一个算法中重复执行的次数。
- 算法执行时间大致为基本运算所需的时间与频度的乘积。

算法效率分析

- 大O表示法：

设函数f(n)和g(n)是正整数n的实函数，如果存在实常数c>0和整常数n0≥1,对于每个n≥n0的整数,满足f(n) ≤cg(n)，则称f(n)是O(cg(n))。这个定义称为大O符号。有时又称函数f(n) 的阶至多是O(g(n))。读作“f(n) 是g(n)的大O”。借助大O符号，我们可以了解当n趋于无穷大时，n的函数f(n) 小于或等于另一个函数的常数倍，即cg(n)。换句话说，**O(g(n))是函数f(n)的一个上界。**

作业

- 1. Given an array of integers, find if the array contains any duplicates. Your function should return true if any value appears at least twice in the array, and it should return false if every element is distinct.

```
bool containsDuplicate(int* nums, int numsSize) {


}
```

- 2. Given an array nums, write a function to move all 0's to the end of it while maintaining the relative order of the non-zero elements. For example, given nums = [0, 1, 0, 3, 12], after calling your function, nums should be [1, 3, 12, 0, 0].

  Note:

  - You must do this **in-place** without making a copy of the array.
  - Minimize the total number of operations.

```
void moveZeroes(int* nums, int numsSize) {

}
```