中等专业学校计算机专业教材

数据结构与算法入门

王庆瑞 陈卫卫 编著

科学出版社

中等专业学校计算机专业教材

数据结构与算法入门

王庆瑞　陈卫卫 编著

科学出版社

2000

内 容 简 介

本书指导读者如何设计求解一般问题和算法，并用PASCAL语言编程实现，是一本带有“手册”性质的中级科技读物。包括线性表、栈和队，链表，树，排序等章节内容。本书以基本数据结构——表结构和树结构，以及基本运算——查找、插入、删除为基础，着力向读者介绍算法设计中最基本的概念和方法，选用算法设计中最常见的实用性问题作为研究对象，用通俗的语言和结构优美的程序，深入浅出地阐明算法设计常用的方法和技巧，旨在培养广大读者朋友的程序设计爱好，提高程序设计能力，使他们逐步学会编写具有一定难度的高质量程序。

本书主要用作中等专业学校计算机系列课程教材，也可作为广大电脑爱好者学习程序设计方法的参考书。

图书在版编目（CIP）数据

数据结构与算法入门/王庆瑞等编著. -北京：科学出版社，2000

（中等专业学校计算机专业教材）

ISBN 7-03-008277-X

Ⅰ.数… Ⅱ.王… Ⅲ.①数据结构-专业学校-教材②电子计算机-计算方法-专业学校-教材 Ⅳ.TP 301.6

中国版本图书馆CIP数据核字（2000）第61143号

科学出版社 出版

北京东黄城根北街16号

邮政编码：100717

北京双青印刷厂 印刷

科学出版社发行 各地新华书店经销

2000年8月第 一 版 开本：787×1092 1/16

2000年8月第一次印刷 印张：9 1/2

印数：1—4 500 字数：222 000

定价：15.00元

(如有印装质量问题,我社负责调换<环伟>)

中等专业学校计算机专业教材

编 委 会

出版说明

在计算机科学技术飞速发展、广泛应用、深入普及的今天，计算机科学技术图书的出版发行轰轰烈烈、规模空前。但是，在浩瀚的计算机出版物中，人们很难寻觅到适合当前中等专业技术学校使用的计算机专业及专业基础教材，更难发现适合具有高中文化程度的计算机爱好者、“发烧友”系统学习计算机基础知识的图书。因此，我们组织了部分具有丰富中等专业技术教育经验的优秀教师和部分计算机技术专家，编写了这套中等专业学校计算机专业教材。

“中等专业学校计算机专业教材”大致可以分为以下三个模块：

（1）专业基础知识模块，包括：

- 计算机应用基础
- 计算机数学初步
- 模拟和数字电路基础
- 计算机英语阅读
- 数据结构与算法入门

（2）专业知识模块，包括：

- PASCAL 程序设计实践
- 数据库应用基础
- 计算机操作系统基础
- 数据通信基础
- C 语言程序设计
- 微型计算机原理入门
- 计算机网络技术基础
- 多媒体技术基础

（3）实用技术模块，包括：

- 视窗系统及办公应用软件
- 微型计算机系统维护与故障诊断
- 因特网应用入门
- 计算机绘图

我们衷心感谢南京市中等专业（走读）学校的教师在教材编撰过程中所给予的大力支持和关心指导，衷心感谢中国人民解放军理工大学计算机与指挥自动化学院专家、教授们的精心组织和辛勤工作，衷心感谢南京同创计算机学校各级领导和老师们，感谢他们的通力协作和在部分书稿试用阶段的卓有成效的实践。

中等专业学校计算机专业教材

编委会

1999 年 7 月

前　言

本书以中等专科学校、职业高中学生以及广大青年电脑爱好者作为主要读者，旨在培养他们的计算机程序设计能力和初级算法设计能力。与目前市面上各种计算机软件应用方面的书籍不同，《数据结构与算法入门》不但要求读者能熟练地操作计算机，更强调亲自编写性能良好的计算机程序。

考虑到本书主要读者对象的文化基础，在内容编选上，立足于那些入门性的知识，即以最简单和最常用的表结构和树结构为主，介绍数据结构和算法的基本概念，忽略其中理论性较强的内容，算法描述工具也采用能够上机操作的TURBO PASCAL 语言，而不像一般数据结构书籍那样选用类PASCAL 或类C 语言，使读者"离计算机更近一些"。同时也要求读者具有一定的程序设计经验和上机练习的经验，特别是要掌握PASCAL 程序设计等有关知识。

本书共分五章，第一章介绍数据结构和算法的有关概念。第二章介绍线性表在顺序存储下，查找、插入、删除等运算算法，以及进栈、退栈、进队、出队等运算算法。第三章介绍链表的程序设计方法，主要是链表的查找、插入、删除、合并等运算算法。第四章介绍树结构的概念和运算算法，主要有二叉树的递归遍历算法，检索树和哈夫曼树等有关算法。第五章介绍几种排序算法。

对于中等学校在校生来说，本书确有一定的难度。为此，我们特地将书中内容分成几个层次，标有"△"的内容为重点内容，标有"＊"的内容为选讲内容，标有"＊＊"的内容为阅读参考内容，其它内容为一般性内容。这些标记符号，有的标在小节处，有的标在一小节的某个算法处。标在小节处的"△"，表示整个小节都是重点内容，标在某个算法处的"△"，表示该算法是本段内容的典型算法。标记"＊"和"＊＊"的作用类似，即某小节或某个算法为选讲内容或阅读参考内容。

一般性内容主要是一些概念性知识和简单算法，是学习其它内容的基础，应当划作教学范围之内。除第一章外，每章列出一些重点内容，这部分内容主要涉及到一些性能较好的查找、插入、删除、排序算法，以及二叉树的构造和遍历算法，这些内容应当作为教学的重点。

选讲内容通常都有较大的难度，应根据实际情况，从中选择一部分作为教学内容。阅读参考性内容难度更大一些，可以不作为教学内容，而留给有余力的读者自学。

为了使本书具有一定的系统性和完整性，书中含有一部分数学公式和数学证明。这些内容大多涉及对算法性能的评价（主要是算法时间耗用量的计算）。对于中等学校在校生来说，即使不理解公式的推导过程和计算过程，也希望他们能记住计算结果，以便对算法的性能有一个感性的认识。

要学好数据结构和算法，上机练习是相当重要的。建议将习题中标有"△"的题作为上机练习题，而且每章（第一章除外）至少选作一题作为上机练习题。习题中，标有"＊"和"＊＊"的题目是一些难度较大的习题，可作为"学习兴趣小组"的讨论题。其它习题，可选其中一部分作为平时的作业题。

数据结构和算法课程，牵涉内容广泛的计算机软硬件知识和数学知识，所以对于中等学校的学生，即使再浅些，再通俗一些，若真正使他们“入门”，仍然会有一定的难度。本书作者从传统的数据结构和算法内容中，编选一些最基本、最核心的内容，力图将困难降低到最小。

为中学生写数据结构和算法方面的教材是一种尝试。希望本书的面市，对正日益发展的中等学校计算机教学起到应有的推动作用。

作　者

2000 年 1 月

目　录

第一章 引 论

1.1 基本概念

当人们准备为求解某给定问题设计计算机程序时，通常首先要对这个问题进行认真细致的分析，头脑里不时地思考着下列问题：

要求解的这个问题涉及到哪些有关数据，这些数据之间有什么样的相互联系，用什么样的方法去处理这些数据才能达到求解目的，如何有效地存储这些数据、并且体现它们之间的关系，才能使程序处理起来更加方便，效率更高。

经过一番深思熟虑之后，开始起草解题方案，确定求解问题的算法。在正式动手编程之前，还要对算法的可行性和算法效率进行论证和评估。在确认“问题不大”后，着手编程，并上机调试。这中间可能还要几经反复，最后方能得到比较满意的结果——一个性能良好的完整的计算机程序，交付使用。

要做好上述工作，数据结构和算法知识是必不可少的。

数据结构和算法（data structures and algorithms）是研究数据和数据之间的关系（relationship）、数据集合上的运算（operation）、数据和数据之间关系的存储形式、完成运算的算法设计方法，以及对算法性能进行综合评价的一门学问。

一般说来，数据是指对客观事物的名称、数量、特征、性质的描述，并且对这种描述加工整理，进行必要的编码，再交给计算机进行处理的一切符号的总称。因而数据是计算机加工的对象，也是计算机生产出来的产品（即计算结果）。

我们通常不研究那些单个的孤立的数据，而着重研究由众多数据组成的数据集合，研究数据之间的关系，以及在不同的数据集合上能够进行哪些运算（即对数据进行的处理），如何提高运算效率等等。

在数据集合中，一个数据元素（data element）又叫做一个数据结点，简称结点（node）。一个数据结点是指用来描述一个独立事物的名称、数量、特征、性质的一组信息。与数据库中记录的概念类似，组成结点的一个数据项叫做结点的一个域（field），能够用来唯一标识结点的域称为关键字（key）。为方便起见，有时我们把关键字域的值称为结点的值。

如果一个结点只含一个数据项（比如一个整数），这种非记录型的结点称为“单值结点”。

为了简化程序结构和文字叙述，本书在举例时，通常把结点看成一个整数，或者用关键字代替结点。

比如，图 1.1 所示的学生成绩表中，每行数据构成一个结点，它包括一个学生的姓名、学号，以及语文、数学、外语三门功课的成绩和成绩等级等数据项，其中学号可作

为关键字。学生们的这些数据信息组成一个结点集合，各结点既互相独立，又互相关联在一起。

一个结点集合，以及该集合中各结点之间的关系，组成一个数据结构。

如果各结点之间仅具有某种先后次序关系，或者说，结点集合构成一个结点序列，那么这样的数据结构就属于一种表结构。图 1.1 所示的学生成绩表就是一个表结构。

姓　名	学　号	语　文	数　学	外　语	等　级
赵小苏	991101	76	80	64	C
钱　宁	991102	92	86	82	B
孙金陵	991103	63	74	73	C
李长江	991104	89	91	97	A
周　珊	991105	70	55	53	D
⋮	⋮	⋮	⋮	⋮	⋮
陈　洁	991150	87	77	84	B

图 1.1　学生成绩表

如果各结点之间的关系是一种比较复杂的层次关系，那么这样的数据结构就属于一种树结构。

表结构和树结构都是最基本的数据结构。

简单地说，数据结构就是指数据之间的关系，研究数据结构就是研究如何确定数据之间的关系才能提高计算机的处理能力和效率。不同的研究对象其数据之间有着不同的关系，这些关系大多是由事物本身内在因素所决定的。有时，同一研究对象的数据之间可以含有多种不同的关系，在处理问题时，就要从中确定一种既简单又便于处理的关系。

数据以及数据之间的关系在计算机内的存储形式叫作数据的存储表示，也称为存储结构。在存储数据的同时，还必须体现出数据之间的关系。本书在描述算法时，大多使用 PASCAL 的数组、链表等结构数据类型表示数据的存储结构。

数据结构的种类繁多，总体上可划分成表结构、树结构、图结构和文件结构等几大类，限于篇幅，本书只讨论其中最简单的表结构（及其变种）和树结构。

我们把对数据和数据结构的处理操作统称为对这个数据结构进行的运算。不同的数据结构有不同的运算。比较简单而又常用的运算有查找、插入、删除、排序和二叉树的遍历运算等等。

本书重点研究表结构和树结构上的查找、插入和删除运算，讨论表结构和树结构的特点及存储方法，介绍实现各种运算的算法，以及对算法的性能作简单的评价。本书的最终目的是使大家了解什么情况下选用什么样的数据结构和算法，可获得较高的解题效率。

1.2　算法的描述和实现

算法是人们求解问题时所采用的方法和步骤，或者说，算法就是“程序流程”。这里所说的“求解问题”是指为解此问题而设计计算机程序。因为有了程序，就能让计算机

“计算出”问题的答案，所以从这个意义上来说，程序就是问题的解答。

算法有两种形式，一种是程序形式，另一种是描述形式。

算法的程序形式（比如PASCAL程序，C程序等）又称为算法的实现形式，是算法的最终形式。

算法的描述形式是向人们介绍算法时使用的一种形式，或者说是算法的记述形式。可以有多种描述形式，最常使用的有三种形式：文字叙述形式，框图（即流程图）形式和类语言（又称伪代码）形式。

文字叙述形式简单直观，易于理解。

例如，将数组a[1..n]的元素重新排列，使之呈由小到大的排列形式，即排序。求解排序问题的方法很多，其中有一种自然选择排序算法，这个算法用文字叙述如下：

先从a的n个元素中选择一个最小元素（比如某个a[j]），将它与a[1]交换位置；

再从剩下的n－1个元素中选择一个最小元素，将它与a[2]交换位置；

再从剩下的n－2个元素中选择一个最小元素，将它与a[3]交换位置；

……

反复进行上述选择和交换工作，直到选择出最后一个元素，从而完成了排序工作。

这样的描述形式很简单，即使“外行”也能看懂。但是这种描述形式离算法的最终形式（程序）相差太远，文字叙述又不太“规范”，用来描述复杂算法就显得不够精确。

若用类PASCAL语言（即伪代码）描述，可以写成：

```
for  i:=1  to  n-1  do
  begin
    从a[i]到a[n]选择最小元素a[j];
    交换a[i]与a[j]的值
  end;
```

所谓的类PASCAL语言，其实就是一种不太标准的PASCAL语言，或者说是一种“假的”程序设计语言，用类PASCAL语言编写的程序是不能上机运行的。由于人们常把程序称做代码，高级语言的源程序叫源代码，所以用类PASCAL语言写的程序叫做“伪代码”。除了类PASCAL外，还有类C语言等等。

用类PASCAL描述算法，比文字叙述清楚，而且在必要时还可以逐步将非标准的PASCAL语句（比如上面算法中的循环语句的循环体）改写成标准的PASCAL语句。比如，可将上面的算法进一步地细化成：

```
for  i:=1  to  n-1  do
  begin
    j:=i;
    for  k:=i+1  to  n  do    if  a[k]<a[j]  then  j:=k;
    t:=a[i];  a[i]:=a[j];  a[j]:=t
  end;
```

经过如此细化之后，与“真”程序已经非常接近了。只要配上必要的变量定义和适当的输入输出语句，就是一个完整的计算机程序了。

画算法流程图（框图）也是描述算法的一种办法。读者对框图并不陌生，本书不再

介绍框图的画法，仅给出一个用框图描述算法的例子。

设整型数组 a[1..n]的元素是由小到大排列的，现在给出一个整型变量 x，要求判断数组 a 中是否有一个元素值等于 x，如果有，则输出该元素的下标 i，如果没有则输出 0。

这个问题叫做查找问题，求解查找问题有多种算法，其中有一个速度非常快的查找算法叫做二分查找算法。

图 1.2 是二分查找算法的框图描述形式。相信读者能够看懂这个框图。如果用文字叙述，则描述如下：

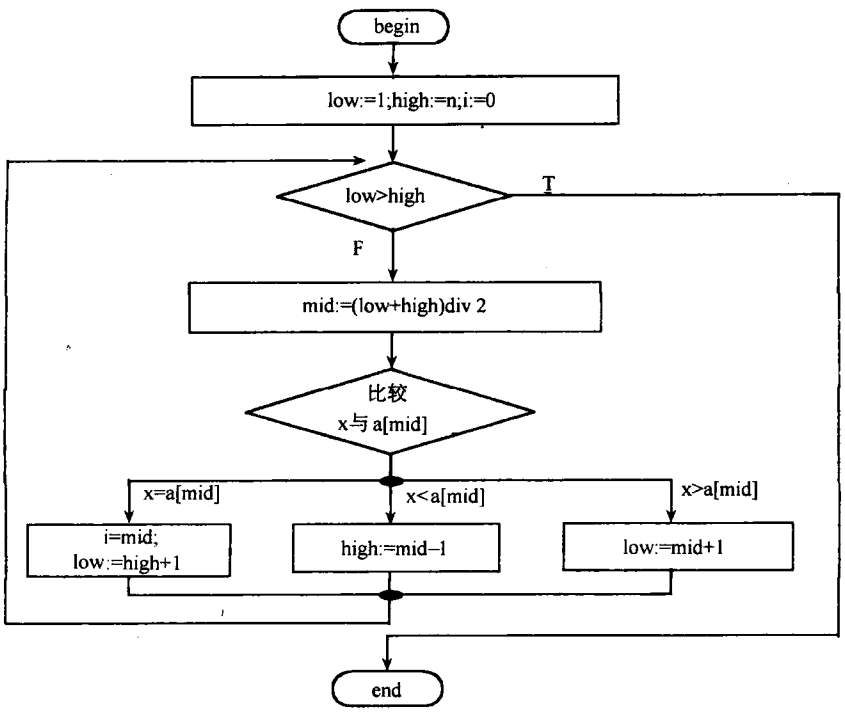

图 1.2　二分查找算法的流程图

第一步　置 low 和 high 的初值，low：=1；high：=n；

第二步　如果 low>high，则查找不成功（即数组 a 中没有 x），输出 0，算法终止，否则；

第三步　计算中值地址，即下标 mid=(low+high) div 2；

第四步　比较 x 与 a[mid]的值；

第五步　如果 x=a[mid]，则找到 x，输出下标 mid，算法终止，否则；

第六步　如果 x<a[mid]，则调整指针 high，使 high：=mid-1，返回第二步，继续查找；

第七步　如果 x>a[mid]，则调整指针 low，使 low：=mid+1，返回第二步，继续查找。

对 PASCAL 语言比较熟悉的读者，根据上述算法的流程图和文字描述，很容易写出关于二分查找的 PASCAL 程序。

不难发现，与单纯的文字叙述相比，用流程图描述的算法结构更清晰，也更容易理解，更接近程序形式。因此，流程图不失为描述算法的好形式，只是画起来比较麻烦，画错了又不容易修改。

目前，大多数有关算法和数据结构方面的书籍都是用类 PASCAL 语言，或类 C 语言，C++语言描述的。考虑到读者对象，本书不用类 PASCAL 语言描述算法。

文字叙述、伪代码描述、流程图三种形式都可以用来描述算法。为了少占用篇幅，本书在介绍算法时，通常先用文字叙述的方法介绍算法的设计思想，再给出具体的 PASCAL 程序，通常是 PASCAL 过程，或 PASCAL 函数的形式。读者只要配上适当的主程序和必要的数据，就能上机运行。

1.3* 算法性能的评价

一个算法设计完毕后，通常还要对这个算法的性能加以评价，这种评价称为对算法进行分析，也就是要对算法的可行性和实用性作初步的估计。如果评估的结果比较满意，就将该算法编程实现。如果不满意，要重新修改算法，直到满意为止。

可以从几个不同的角度去评价一个算法的综合性能。

首先要考虑算法的正确性。这是最起码的，也是最重要的，因为只有正确的算法才实用。一个正确的算法（或程序）应当对所有合法的输入数据都能得到应该得到的结果。比如一个排序算法，只有对任意 n 个数据都能完成排序工作的算法才是正确的排序算法。

对于一些简单的算法（或程序），可以通过上机调试去验证其正确与否。调试用的数据要精心挑选那些具有“代表性”的，甚至有点“刁钻性”的，以保证算法对“所有的”数据都正确。

但是，一般来说，调试并不能保证算法对所有数据都正确，只能保证算法对部分数据正确。著名的计算机科学家克努特说“调试只能验证算法有错，不能证明算法无错”。就是说只要找出一组数据使算法失败（即计算结果不对），就能否定整个算法的正确性。然而，往往调试并不能穷尽所有可能的情况，所以，即使算法有错，也不一定不能通过调试，在短时间内发现。例如，不少大型软件在使用多年后，仍然还能发现其中的错误。

要严格地阐明一个算法是正确的，通常要用数学归纳法去证明。考虑读者对象，本书将不讨论算法的正确性。而且本书涉及的算法大多是“一目了然”的简单算法，通过对算法设计思想的阐述，其正确性也就在其中了。

评价算法性能另一个要考虑的因素就是算法的运行效率（或运行成本、费用），也就是要估计一下算法投入运行时，大致需要耗用多少时间，占用多少内存单元（即空间），其时间空间需求量能否满足客观要求。其中最主要的是算法运行时间的估算。

一般说来，一个算法的时间耗用量将随输入数据量的增大而增大。比如，排序算法，对 10 000 个元素排序所用时间要比对 100 个元素排序所用时间长。要精确地算出一个算法所用时间是非常困难的，为了简单起见，本书在评价算法时间用量时，主要根据算法的循环次数、递归调用次数等，粗略地估算所用时间与输入数据量之间的比例关系（即函数关系）。

设输入数据量为 n，若某算法的时间耗用函数 T(n)差不多与 n 的平方成正比，我们

就说这个算法的时间耗费是 n 的平方阶的，用符号表示成：

$$T(n)=O(n^2)$$

读作“梯 n 是欧 n 平方的”。

本书在讨论算法的时间耗用函数 T(n)时，通常只估算到 T(n)的最高阶，既不考虑低阶项，也不考虑常系数。比如，有一个算法 A，其时间耗用函数为：

$$T_1(n)=20n^2+100n$$

另一个算法 B，其时间耗用函数为：

$$T_2(n)=0.5n^2-3n+18$$

那么，这两个算法的时间耗用函数的阶是一样的，都是 $O(n^2)$。

一般说来，当 n 很大时，算法时间耗用量的阶越低，算法的运行速度越快，所以低阶算法（指时间耗用量的高低）比高阶算法好。

下面由低到高列出一些经常用于表示算法时间耗用函数的阶：

O(1)	常数时间	注：算法时间用量与输入数据量 n 的大小无关。
O(log n)	对数时间	注：对数阶通常不写底数（对数一般以 2 为底）。
O(n)	线性时间	注：算法时间用量与 n 成正比。
O(n log n)		注：这是一种非常理想的阶。
$O(n^2)$	二次的时间	
$O(n^3)$	三次的时间	
……		
$O(2^n)$	指数时间	注：这是一种非常高的阶，很不实用的算法才具有此阶。
O(n!)	阶乘时间	注：这种阶更高。

除最后两项外，这些阶都是多项式的。如果一个算法时间用量超过多项式，比如指数、阶乘等，那么这个算法就是一个不可使用的无效算法，因为只要输入数据量 n 稍大一点，算法就不能够在“有限的”时间内完成。

我们在设计算法时，应当尽量设法降低它的时间耗用量，以提高算法的运行速度。

有的算法对不同的输入数据（即使输入数据量都是 n）耗用时间会有所不同，而且往往相差很大。在这种情况下，为使评价更客观，更有说服力，我们要分别估算最坏情况下（worst-case）和平均情况下（expected-case）算法的时间耗用量。

所谓最坏情况是指，对于具有相同输入数据量的不同输入数据，算法时间用量的最大值。

比如一个排序算法，对于一组（n 个）待排序数据，算法耗用时间会少些，而对另外一组（也是 n 个）待排序数据，算法耗用时间会多些。这样，至少存在一组数据，算法耗用时间最多（最坏情况），我们就以这组耗用时间最多的数据为准，估算算法时间耗用量。

为醒目起见，最坏情况下的时间耗用函数用 $T_W(n)$表示，其中下脚标“W”指的是最坏情况。

所谓平均情况是指，对于所有相同输入数据量的各种不同数据，算法耗用时间的平均值（有时还要考虑各种不同数据出现的可能性，即概率）。

再比如排序算法，如果把每 n 个待排序的输入数据看作一组，输入数据不同“杂乱程度”将有 n! 种可能。每种可能（的输入数据）耗用时间不同，我们把这 n! 种可能情况耗用的时间加起来，除以 n!，算作该算法平均时间耗用量。

用 $T_E(n)$ 表示平均情况下算法耗时量函数，其中下脚标“E”代表平均情况。

当然，如果某算法最坏情况和一般情况时间用量相差不多，就没有必要区分两种不同情况了。这时我们会笼统地用 T(n) 表示算法时间用量，即 T(n) 不带下脚标。

在设计算法时，有时为了节省时间而多使用一些辅助变量（主要是数组）。这种做法实际上是“以牺牲空间换取时间”。

考虑到读者对象，本书并不打算介绍如何求出一个算法的时间用量，提出上述概念的目的只是让大家懂得怎样鉴别算法的优劣，要想改善算法的性能，应当从哪些方面着手。

本章小结

这一章，我们简单地介绍了有关数据结构和算法的基本概念，主要有：

（1）什么是算法，什么是数据结构，什么是运算。

（2）数据结构的种类（表，树，图，文件等），运算的种类（查找，插入，删除等）。

（3）算法的描述形式（文字叙述，流程图，伪代码，PASCAL 程序等）。

（4）对算法评价（时间耗用量等）方法。

有些概念比较抽象，尤其是关于算法性能评价（即算法分析）的内容，对于初学者来说，是很难理解的。教学时，可以跳过这些难于理解的内容。

这些概念不要去“死记”，要了解其大体的含义，为学习后面的“正式”内容作理论上的准备。

习 题 一

1.1 什么叫数据结构，数据结构和算法课程要研究哪些问题？

1.2 什么叫算法，什么是算法的描述形式和实现形式，算法的描述形式有哪些？

1.3 什么是算法时间耗用量的最坏情况和平均情况？试举例说明某算法（程序）在最坏情况和平均情况时间用量的差别。

1.4 试分别用文字叙述和画流程图两种形式，描述求两个正整数 m 和 n 的最大公约数的欧几里德算法。

1.5 试用流程图描述计算多项式

$$p(x)=a_nx^n+a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+\cdots+a_1x^1+a_0$$

之值的 Hanoi 算法，并将该算法用 PASCAL 语言实现。

1.6 试用流程图描述将正整数 n（大于 1）作质因子分解的算法，并将这个算法用 PASCAL 语言实现。

1.7 已知数组 A [1..m] 和数组 B [1..n] 的元素都是由小到大排列的。试写一个算法，将数组 A 和数组 B 合并成数组 C [1..m+n]，使数组 C 的元素也由小到大排列。先画出算法的流程图，再编制 PASCAL 程序。