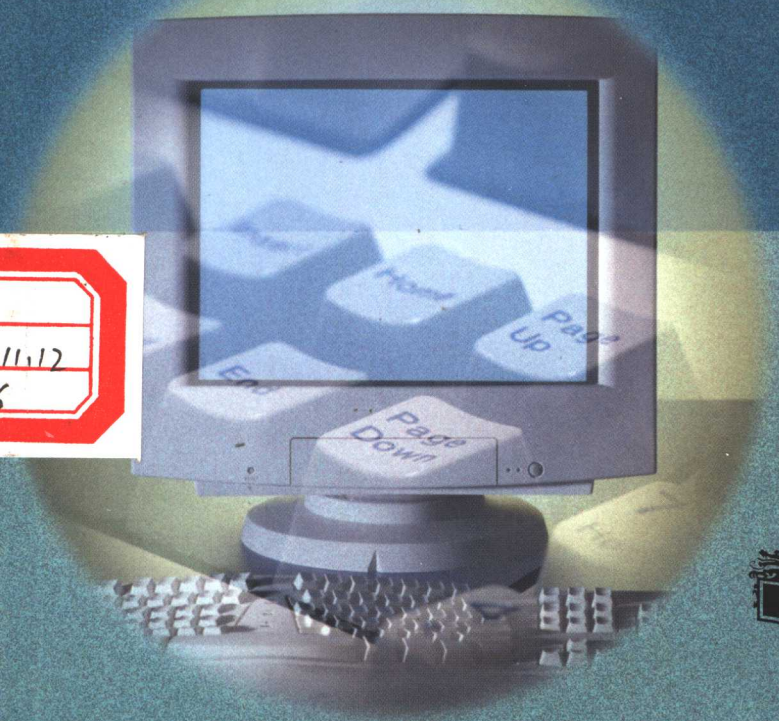
专升本

教育部师范教育司组织编写
中学教师进修高等师范本科(专科起点)教材

数据结构

黄国兴　主编
黄国兴　文忠林　孙　强　编著



高等教育出版社

内容提要

数据结构是计算机科学技术学科最基本的专业基础之一。

本书主要介绍了数据结构的基本内容，包括线性表、多维数组和串、树、图、查找和排序。教材内容的组织力求全面，表达形式注意了可读性和可操作性，便于自学。本书所用到的C语言源程序及部分辅助教学软件，均可在高等教育出版社的网站上下载。

本书可作为计算机科学技术学科“专升本”和一般本科教学教材，也可供相关专业作为信息技术专业基础知识的教学用书。

图书在版编目(CIP)数据

数据结构/黄国兴主编.—北京：高等教育出版社，2001.6.28
专升本及成人本科教材
ISBN 7-04-009441-X

Ⅰ.数…Ⅱ.黄…Ⅲ.数据结构-高等教育-教材 Ⅳ.TP311.12

中国版本图书馆CIP数据核字(2001)第26160号

数据结构
黄国兴 主编

出版发行	高等教育出版社		
社　　址	北京市东城区沙滩后街55号	邮政编码	100009
电　　话	010—64054588	传　　真	010—64014048
网　　址	http://www.hep.edu.cn http://www.hep.com.cn		
经　　销	新华书店北京发行所		
排　　版	高等教育出版社照排中心		
印　　刷	河北新华印刷一厂		
开　　本	787×960　1/16	版　　次	2001年6月第1版
印　　张	9.5	印　　次	2001年6月第1次印刷
字　　数	160 000	定　　价	8.70元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

前　言

“数据结构”是计算机科学技术学科的专业基础课，学好数据结构对学习和深入理解计算机科学技术的其他课程，如“操作系统”、“数据库原理”、“程序设计方法”等都有很重要的作用。

本书遵照教育部师范教育司关于提高信息技术课程师资的学历水平精神，为已具备大专学历的广大中、小学教师进一步学习计算机科学技术课程而撰写。为适应这一需要，本教材的编写力求做到：在内容选择上和全日制本科生教学内容基本相同；在表达形式上注意了教材内容表达方式上的可读性，使之便于自学。教材中所选用的用C语言编写的程序可以在高等教育出版社的网站上下载。这些程序已经调试并可实际运行，这样既可以为读者实际操作提供方便，又可以加快读者的学习进度，提高学习效率。对于某些比较难学难教的章节，本书还附有部分辅助教学的课件（也可以从高等教育出版社网站上下载）供读者参考。本书的学习要求读者所学的前期课程为“C语言程序设计”。

全书共分为7章，建议学时数为72学时：第一章概论，讨论数据结构的基本概念和算法描述的一些概念，建议学时数为3学时；第二章线性表，讨论了线性表的顺序表形式和链表形式及有关操作，建议学时数为14学时；第三章多维数组和串，讨论多维数组和串的一些基本内容，建议学时数为8学时；第四章树，主要讨论二叉树的基本内容，建议学时数为16学时；第五章图，讨论图的基本结构和有关算法，建议学时数为12学时；第六章查找，讨论基本查找方法，建议学时数为7学时；第七章排序，讨论各种排序方法，建议学时数为12学时。使用教材时也可以对建议学时数作适当调整和增删。

鉴于在数据库系统教材中会较详细地介绍“文件”的概念和与之有关的数据结构，本书中不再重复选用该部分的内容。

本书每章最后的学后自测思考题和习题，供读者参考。

在本书的编写过程中始终得到高等教育出版社的支持与帮助；薛锦云教授对本书提出了修改建议；范雅军、蔡健、张召、谢孟军、鲍钰等参加了本书的编写工作，并制作了教材中部分重要内容的辅助教学软件，在此一并致谢。

编　者

2001年3月

目　录

第一章 概 论

【学习目标】

通过本章的学习，学生应该能够：

- 掌握数据的逻辑结构、物理结构概念和相关术语
- 熟悉算法的描述，了解算法的分析方法
- 了解什么是数据结构和学习本门课程的意义

学习重点和难点是：

数据的逻辑结构、物理结构

信息的表示是计算机科学的基础。大多数计算机程序的主要目标与其说是完成运算，不如说是存储和检索信息。从存储空间和运行时间的实现角度来看，这些程序必须组织信息，以支持高效的信息处理过程。因此，研究数据结构和算法以有效地支持程序的实现，就成了计算机科学的核心问题。数据结构是计算机科学专业的一门核心课程，它的研究对象为问题求解方法、程序设计方法及一些典型数据结构的算法。

自 1946 年第一台计算机问世以来，计算机科学技术的飞速发展已远远超出人们的预料。计算机的产量大幅增加、价格大幅下降，这都使得它的应用范围迅速扩展。如今，计算机已深入到人类社会的各个领域，计算机的应用已不再局限于科学计算，而更多地用于控制、管理及数据处理等非数值计算的领域。与此相应，计算机加工处理的对象由纯粹的数值发展到字符、表格和图像等各种具有一定结构的数据，这就给程序设计带来一些新的问题。为了编写出一个“好”的程序，必须分析待处理对象的特性以及各处理对象之间存在的关系。这就是“数据结构”这门学科形成和发展的背景。

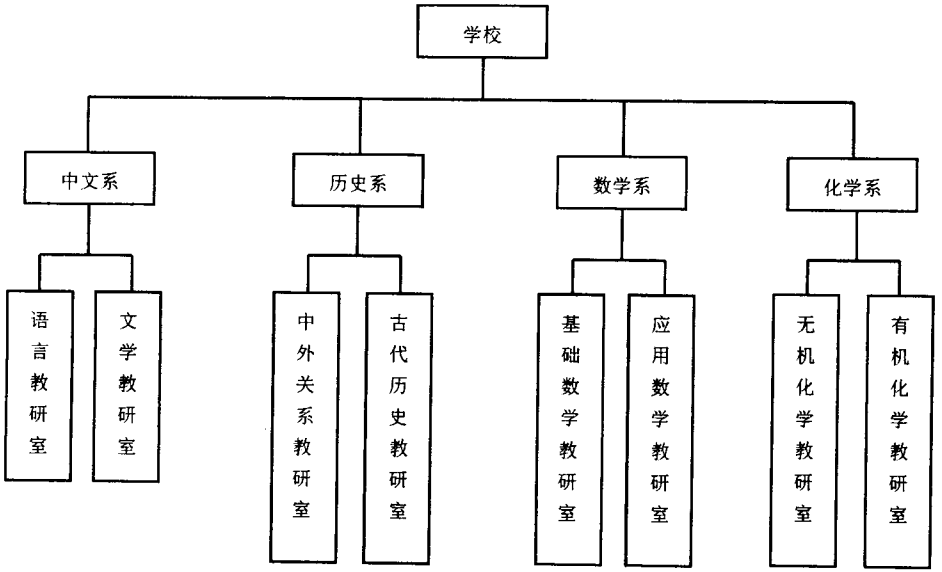
1.1 数据的逻辑结构

1.1.1 数据和数据结构

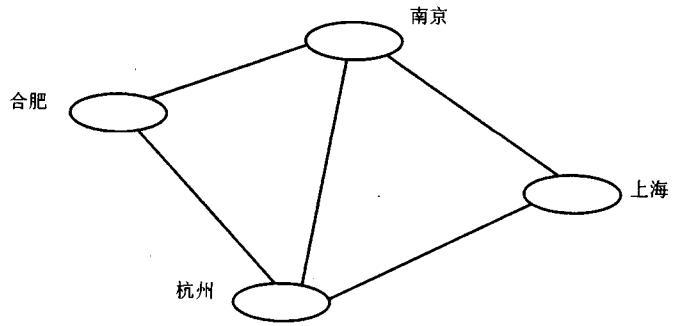
数据(data)是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号的总称。这是计算机程序加工的“原料”。例如，一个利用数值分析方法解代数方程的程序，其处理对象是整数和实数；一

个编译程序或文字处理程序的处理对象是字符串。因此,对计算机科学而言,数据的含义极为广泛,如图像、声音等都可以通过编码而归之于数据的范畴。

数据元素(data element)是数据的基本单位,在计算机程序中通常作为一个整体进行考虑和处理。例如,图 1.1(a)组织结构中的"树"中的一个方框、图 1.1(b)中的一个圆圈,都被称为一个数据元素。有时,一个数据元素可由若干个数据项(data item)组成,例如,一本书可以看作一个数据元素,而书目信息中的每一项(如书名、出版社、作者名等)为一个数据项。数据项是数据的不可分割的最小单位。



(a)



(b)

图 1.1　数据元素组织结构

数据对象(data object)是性质相同的数据元素的集合,是数据的一个子集。例如,整数数据对象是集合 N={0,±1,±2,…},字母字符数据对象是集合 C={'A','B',…'z'}等。

数据结构(data structure)是相互之间存在一种或多种特定关系的数据元素的集合。从上述例子可以看到,在任何问题中,数据元素都不是孤立存在的,它们之间存在着某种关系,这种元素之间的关系称为结构(structure)。

数据项(data item)是一条信息或者是其值属于某个类型的一条记录。数据项是数据类型的成员。整数是简单数据项,因为它不包含子结构,而银行的账户记录是复杂数据项,包含许多项信息,比如,姓名、地址、账号及余额等。

类型(type)是一组值的集合。例如,布尔类型由两个值 TRUE 和 FALSE 组成;整数也构成一个类型。

数据类型(data type)是指一个类型以及定义在这个类型上的一组操作。例如,一个整数变量是整数类型的一个成员。

数据类型的描述与它在计算机程序中的实现有很大的区别。例如,实现线性表数据类型有两种传统的数据结构:链表(linked list)和顺序表(array-based list,基于数组的线性表)。因此,可以在链表或者数组之间选择一种方式来实现线性表数据类型。

抽象数据类型(abstract data type,简称 ADT)通常是指对数据的某种抽象,它定义数据的取值范围及其结构形式,是对数据的操作的集合。抽象数据类型描述数据的构造及使用,并不注重其在程序中如何实现。

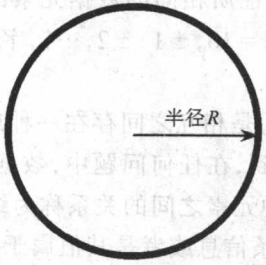
例如,整数的数学概念和施加到整数的运算构成一个抽象数据类型。

又如,圆是平面上与圆心等距离的所有点的集合。由图 1.2 可看出,从图形显示角度看,圆的抽象数据类型包括圆心和半径;而从计量角度看,它所需要的抽象数据类型只需半径即可。如果从计量角度来给出圆的抽象数据类型,那么它的数据取值范围应为半径的取值范围,即为非负实数,而它的操作形式为确定圆的半径(赋值);求圆的面积;求圆的周长。

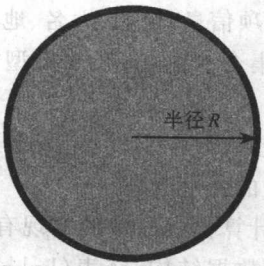
抽象数据类型通过一种特定的数据结构在程序的某个部分得以实现,而在设计使用抽象数据类型的那部分程序时,我们只关心这个数据类型上的操作,而不关心数据结构的具体实现。因此,在思考一个复杂的程序时,首先应考虑能否将它抽象简化,否则很难理解或者实现它。

1.1.2 数据的逻辑结构

数据结构的定义仅是对操作对象的一种数学描述,它是从操作对象抽象出来的数学模型。结构定义中的"关系"描述的是数据元素之间的逻辑关系,因此又称为数据的逻辑结构。然而,讨论数据结构的目的是为了在计算机中实现对



(a) 周长



(b) 面积

图 1.2　圆的抽象数据类型

它的操作,因此还需研究如何在计算机中表示它。

数据项有逻辑形式(logical form)和物理形式(physical form)两个方面。抽象数据类型给出的数据项的定义是它的逻辑形式,数据结构中对数据项的实现是它的物理形式。

图 1.3 说明了数据项的逻辑形式和物理形式之间的关系。

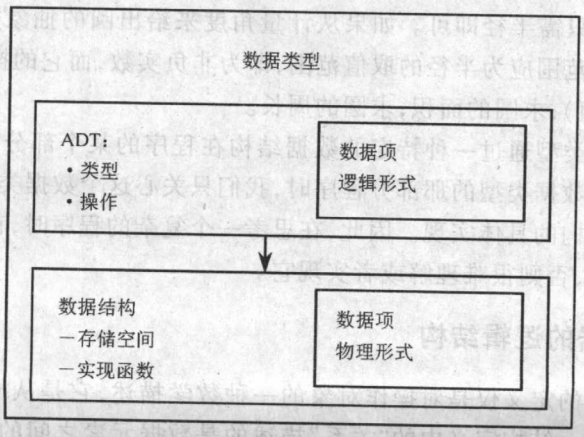


图 1.3　数据项的逻辑形式和物理形式间的关系

1.2 数据的物理结构

1.2.1 物理结构

数据结构在计算机中的表示(又称映象)称为数据的物理结构,又称为存储结构。它包括数据元素的表示和关系的表示。在计算机中表示信息的最小单位是二进制数的一位,叫做位(bit)。在计算机中,可以用一个由若干位组合形成的一个位串表示一个数据元素(如用由若干个位所组成的一个字长的位串表示一个整数,用八位二进制表示一个字符等),通常称这个位串为元素(element)或结点(node)。当数据元素由若干数据项组成时,位串中对应于各个数据项的子位串称为数据域(data field)。因此,元素或结点可看成是数据元素在计算机中的映象。

数据元素之间的关系在计算机中有两种不同的表示方法:顺序映象和非顺序映象,并由此得到两种不同的存储结构:顺序存储结构和链式存储结构。顺序映象的特点是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。数据的逻辑结构和物理结构是密切相关的两个方面,以后读者会看到,任何一个算法的设计取决于选定的数据(逻辑)结构,而算法的实现依赖于采用的存储结构。

如何描述存储结构呢?虽然存储结构涉及数据元素及其关系在存储器中的物理位置,但由于是在高级程序语言的层次讨论数据结构的操作,因此不能直接以内存地址来描述存储结构,可以借用高级程序语言中提供的"数据类型"来描述它们。例如,可以用"一维数组"类型来描述顺序存储结构,以"指针"来描述链式存储结构等。

1.2.2 学习数据结构的意义

"数据结构"在计算机科学中是一门综合性的专业基础课。数据结构的研究不仅涉及到计算机硬件(特别是编码理论、存储装置和存取方法等)的研究范围,而且和计算机软件的研究有着更密切的关系,无论是编译程序还是操作系统,都涉及到数据元素在存储器中的分配问题。在研究信息检索时也必须考虑如何组织数据,以便查找和存取数据元素更为方便。因此,可以认为数据结构是介于数学、计算机硬件和计算机软件三者之间的一门核心课程。在计算机科学中,数据结构不仅是一般程序设计(特别是非数值计算的程序设计)的基础,而且是设计和实现编译程序、操作系统、数据库系统及其他系统程序和大型应用程序的重要基础。

1.3 算法的描述和分析

学习本门课程的读者应已学习过“C 语言程序设计”。为了能使读者有一个回顾,这里先简要介绍 C 语言的主要内容。

1.3.1 C 语言简介

一种语言之所以能存在和发展,并具有生命力,总是有其不同于(或优于)其他语言的特点。C 语言的主要特点如下:

(1) 语言简洁、紧凑,使用方便、灵活。C 语言一共只有 32 个关键字,9 种控制语句,程序书写形式自由,主要用小写字母表示,压缩了不必要的成分。

(2) 运算符丰富。C 语言的运算符共有 34 种,它把括号、赋值、强制类型转换等都作为运算符处理,使运算类型极其丰富,表达式类型多样化。灵活使用各种运算符可以实现在其他高级语言中难以实现的运算。

(3) 数据结构丰富。C 语言具有现代化语言的各种数据结构,它的数据类型有:整型、实型、字符型、数组类型、指针类型、结构体类型、共用体类型等。能用来实现各种复杂数据结构(如链表、树、栈等)的运算,尤其是指针类型数据,使用起来非常灵活、高效。

(4) 具有结构化的控制语句(如 if…else 语句、while 语句、do…while 语句、switch 语句、for 语句等)。C 语言用函数作为程序模块以实现程序的模块化,它是结构化的理想语言,符合现代编程风格要求。

(5) 语法限制不太严格,程序设计自由度大。例如,对数组下标越界不作检查,由程序编写者自己保证程序的正确性。变量的类型使用比较灵活,例如,整型数据与字符型数据以及逻辑型数据可以通用。一般的高级语言语法检查比较严、能检查出几乎所有的语法错误。而 C 语言允许程序编写者有较大的自由度,因此放宽了语法检查。程序员应当仔细检查程序,保证其正确,而不要过分依赖 C 语言的编译程序去查错。“限制”与“灵活”是一对矛盾,限制严格,就失去灵活性;而强调灵活,就必然放松限制。一个不熟练的人,编一个正确的 C 语言程序可能会比编一个其他高级语言程序难一些。也就是说,对用 C 语言的人,要求对程序设计更熟练一些。

(6) C 语言直接访问物理地址,能进行位(bit)操作,能实现汇编语言的大部分功能,可以直接对硬件进行操作。因此 C 语言既具有高级语言的功能,又具有低级语言的许多功能,可以用它来编写系统软件。C 语言的这种双重性,使它既是成功的系统描述语言,又是通用的程序设计语言。有人把 C 语言称为“高级语言中的低级语言”,也有人称它为“中级语言”,这就意味着它兼有高级和低

级语言的特点。

(7) 生成目标代码质量高,程序执行效率高。C 语言生成的目标代码一般只比汇编程序生成的目标代码效率低 10%～20%。

(8) 用 C 语言编写的程序可移植性好。同汇编语言相比,用 C 语言编写的程序基本上不用作修改就能用于各种型号的计算机和各种操作系统。

下面先介绍几个简单的 C 语言程序,然后从中分析 C 语言程序的特性。

例 1.1

```
main ()
{
printf (" This is a c program. \ n");
}
```

本程序的作用是输出以下一行信息:

```
This is a c program.
```

其中 main 表示"主函数",每一个 C 语言程序都必须有一个 main 函数。函数体由大括号"{ }"括起来。本例中主函数内只有一个输出语句,printf 是 C 语言中的输出函数。双引号内的字符串原样输出。"\ n"是换行符,即在输出"This is a c program."后回车换行。语句最后有一分号,它是语句的结束标志,书写程序时应注意不要漏写。

例 1.2

```
main ()                    // 求两数之和
{int a,b,sum               // 这是定义变量
a = 123;b = 456;
sum = a + b;
printf (" sum is %d \ n",sum);
}
```

本程序的作用是求两个整数 a 和 b 之和 sum。"/ * …… * /"或"//"表示注释,为便于理解,用汉字表示注释,当然也可以用英语或汉语拼音作注释。注释只是给人看的,对编译和运行不起作用。注释可以加在程序中任何位置。第二行是变量定义部分,说明 a 和 b 为整型(int)变量。第三行是输入输出的"格式字符串",用来指定输入输出时的数据类型和格式值。printf 函数中括号内最右端 sum 是要输出的变量,现在它的值为 579(即 123 + 456 之值),因此输出一行信息为:

```
sum is 579
```

例 1.3

```
main ()                    // 主函数
{int a,b,c;                // 定义变量
```

```
scanf("%d,%d",&a,&b);      // 输入变量 a 和 b 的值
c=max(a,b);                // 调用 max 函数,将得到的值赋给 c
printf("max=%d",c);        // 输出 c 的值
}
int max(x,y)               // 定义 max 函数,函数值为整型,x,y 为形
                              式参数
  int x,y;                 // 对形参 x,y 作类型定义
    {int z;                // max 函数中用到的变量 z,也要加以定义
     if(x>y)  z=x;
      else  z=y;
     return(z);            // 将 z 的值返回,通过 max 带回调用处
}
```

本程序包括两个函数:主函数 main 和被调用的函数 max。max 函数的作用是将 x 和 y 中较大者的值赋给变量 z。return 语句将 z 的值返回给主调函数 main。返回值是通过函数名 max 带回到 main 函数的调用处。main 函数中的 scanf 是"输入函数"的名字(scanf 和 pritf 都是 C 语言提供的标准输入输出函数)。程序中 scanf 函数的作用是输入 a 和 b 的值。&a 和 &b 地址所标志的单元中,输入变量 a 和 b 的值。这种输入形式是与其他语言不同的。

main 函数中第四行为调用 max 函数,在调用时将实际参数 a 和 b 的值分别传送给 max 函数中的形式参数 x 和 y。经过执行 max 函数得到一个返回值(即 max 函数中变量 z 的值),把这个值赋给变量 c,然后输出 c 的值。printf 函数双引号内的"max=%d"中,"%d"在输出时,将由 c 的值取代,"max="原样输出。程序运行情况如下:

```
8,5↙    (输入 8 和 5 给 a 和 b)
max=8    (输出 c 的值)
```

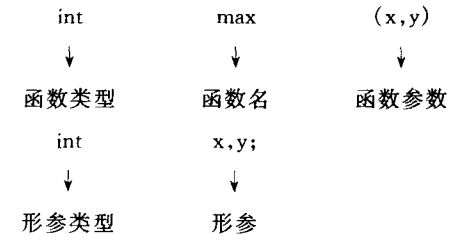
通过以上几个例子,可以看到:

1. C 语言程序是由函数构成的,一个 C 语言源程序至少包含一个函数(main 函数),也可以包含一个 main 函数和若干个其他函数。因此,函数是 C 语言程序的基本单位。被调用的函数可以是系统提供的库函数(例如 printf 和 scanf 函数),也可以是用户根据需要自己编制设计的函数。C 语言的函数相当于其他语言中的子程序。

C 语言的这种特点使得程序很容易实现模块化。

2. 一个函数由两部分组成:

(1) 函数的说明部分,包括函数名、函数类型、函数属性、函数参数(形参)名、形式参数类型。例如,例 1.3 中的 max 函数的说明部分为:



一个函数名后面必须跟一对圆括号。函数也可以没有参数，如 main( )。

(2) 函数体，即函数说明部分下面的大括号“{”和“}”内的部分。如果一个函数内有多个大括号，则最外层的一对括号“{”、“}”为函数体的范围。

函数体一般包括：

① 变量定义。如例 1.3 中 main 函数中的“int a,b,c;”。

② 执行部分。由若干个语句组成。

当然，在某些情况下也可以没有变量定义部分(例如例 1.1)，甚至可以既无变量定义又无执行部分。如：

```
dump()
{ }
```

它是一个空函数，什么也不干，但这是合法的。

3. 一个 C 语言程序总是从 main 函数开始执行的，而不论 main 函数在整个程序中何种位置。

4. C 语言程序书写格式自由，一行内可以写几个语句，一个语句可以分写在多行书写。

5. 每个语句和数据定义的最后必须有一个分号，分号是 C 语言语句的必要组成部分。例如，“c=a+b;”中分号不可少，即使是程序中最后一个语句也应包含分号。

6. C 语言本身没有输入输出语句，输入和输出的操作是由库函数 scanf 和 printf 等函数来完成的。C 语言对输入输出实行“函数化”。

7. 可以用/* …… */或//等符号对 C 语言语句中的任何部分作注释。一个好的、有使用价值的源程序都应当加上必要的注释，以增加程序的可读性。

1.3.2 算法的描述

算法(algorithm)是指解决问题的一种方法或一个过程。算法需要用一种语言来描述，同时，算法可有各种描述方法以满足不同的需求。例如，一个需要在计算机上运行的程序(程序也是算法)必须是按照语法用机器语言或汇编语言或高级程序语言编写，而一个为了便于人们阅读和交流的算法可以用伪码语言或框图等其他形式来描述。由于本书中讨论的算法主要是为读者阅读的，且能容

易地转换成用高级语言书写的程序,因此采用C语言描述。

从直觉上讲,问题(problem)无非是一个需要完成的任务,即对应一组输入有一组相应的输出。问题的定义中不能包含有关怎样解决问题的限制。只有在问题被准确定义并完全理解后才能研究问题的解决方法。然而,问题的定义中应该包含对任何可行方案所需要资源的限制。对于计算机要解决的任何一个问题,总有一些直接的或者间接的限制。例如,任何计算机程序只能使用主存储器和可用磁盘空间,而且必须在合理的时间内完成运行。

从数学角度,也可以把问题看作函数。

函数(function)是输入和输出之间的一种映射关系。函数的输入可以是一个值或是一些信息,这些值组成的输入称为函数的参数(parameters)。不同的输入可以产生不同的输出,但是对于给定的输入,每次计算函数所得到的输出必须相同。

如果将问题看作函数,那么算法就是把输入转化为输出。一个问题可以有多种算法,一个给定的算法解决一个特定的问题。

应知道一个问题多种解法,因为,不同的解法可能对问题的几个特定定量更有效,或者对同一个问题的不同输入更有效。例如,有的排序算法适合于数目较少的序列,有的算法适合于数目较多的序列,而有的算法则适合于可变长度的字符串。

一个算法应该包含以下几条性质:

(1) 正确性(correctness)。也就是说,它必须能完成所期望的功能,把每一次输入转化为正确的输出。

(2) 具体步骤(concrete steps)。一个算法应该由一系列具体步骤组成。"具体"意味着每一步所描述的行为对于必须完成算法的人或机器是可读的、可执行的。每一步必须在有限的时间内执行完毕。因此,算法好像给我们一个通过一系列步骤解决问题的"处方",其中的每一步都是我们力所能及的,是否能够完成每一步,依赖于谁或者用什么方法来执行这个处方。

(3) 确定性(no ambiguity)。下一步(通常是指算法描述中的下一步)应执行的步骤必须明确。选择语句(例如 if 和 case 语句)是任何算法描述语言的组成部分,它允许对下一步执行的语句进行选择,但是选择过程必须是确定的。

(4) 有限性(finite)。一个算法必须由有限步组成。如果一个算法的描述是由无限步组成的,我们就不可能将它写出来,也不可能将它作为计算机程序来实现。大多数算法描述语言均提供一些实现重复行为的方法,如循环。

(5) 可终止性(terminable)。算法必须可以终止,即不能进入死循环。

程序(program)被认为是对一个算法使用某种程序设计语言的具体实现。本书中的几乎所有算法都给出了用C语言描述的程序。当然,由于使用任何一

种现代计算机程序设计语言都可以实现任何一个算法，所以可能有许多程序都是同一个算法的实现。为了简化表达，我们在这本书中常常混用“算法”和“程序”，尽管它们实际上是互相独立的两个概念。在定义一个算法时，必须提供足够多的细节，以便必要时转化为程序。

算法必须强制终止，这意味着不是所有的计算机程序都是算法。操作系统是一个程序，而不是一个算法。然而，我们可以把操作系统的各种任务看成是一些单独的问题，每一个问题由一部分操作系统程序通过特定的算法来实现，得到输出结果后便终止。

这里再强调一下，问题是一个函数，或是输入和输出的一种联系。算法是一个能够解决问题的、有具体步骤的方法。算法步骤必须无二义性，算法必须正确，长度有限，必须对所有输入都能终止。程序在计算机程序设计语言中是算法的实现。

如何比较两种算法解决问题的效率呢？一种办法就是使用源程序分别实现这两种算法，然后输入适当的数据运行，测算两个程序各自的开销。但是这个方法并不尽如人意。第一，编写两个程序，测算两个算法将花费较多的时间和精力，而我们至多只需要保留其中之一。第二，仅凭实验来比较两种算法，很有可能因为一个程序比另一个“写得好”，而使得算法的真正质量没有得到很好的体现。第三，测试数据的选择可能对其中的一个算法有利。第四，有时会发现即使是较好的那种算法也超出了预计的开销，这意味着使用者不得不再寻找一种新的算法，编写一个新程序实现它。

有一种方法可以较好地解决上述这些问题，称之为渐近算法分析(asymptotic algorithm analysis)，简称算法分析(algorithm analysis)。它可以估算出当问题规模变大时，一种算法及实现它的程序的效率和。这种方法实际上是一种估算方法，但是在实际应用中，它被证明是很有效的。

运行时间经常是算法代价的一个关键方面，但是也不能片面地注重运行速度，而应该同时考虑其他因素，如运行该程序所需要的空间代价(包括内存和磁盘空间)。通常我们需要分析一种算法(或者是实现该算法的一个程序实例)所花费的时间，以及一种数据结构所占用的空间。

许多因素都会影响程序的运行时间。有些因素与程序的编译和运行环境有关，如计算机主频、总线和外部设备等。程序设计使用的语言和编译系统生成的机器代码的质量会对运行速度产生很大的影响，编程人员用程序实现算法的效率也会在很大程度上影响运行的速度。如果要在一台指定的机器上，在给定的时间和空间限制下运行一个程序，以上这些因素都会对结果产生影响。但是，这些因素与两种算法或数据结构的差异无关。为了公平起见，同一个问题的两种算法所对应的两个程序，应该在同样的条件下用同一个编译器编译，在同一台计

算机上运行。并且,两次编程时所花费的精力也应该尽可能地相等,以使得算法的实现"等效"。做到这几点,上面的那些因素就不会对结果产生影响,因为它们对每一个算法都是公平的。

判断算法性能的一个基本考虑是处理一定"规模"(size)的输入时,该算法所需要执行的"基本操作"数。"基本操作"和"规模"这两个名词的含义都是比较模糊的,而且要视具体算法而定。"规模"一般是指输入量的数目。比如,在排序问题中,问题的规模一般可以用被排序的元素来衡量。一个"基本操作"必须具有这样的性质:完成该操作所需时间与操作数的具体取值无关。在大多数高级语言中,两个整数相加、比较两个整数的大小都是基本操作,而 n 个整数累加就不是基本操作,因为其代价(cost)要由 n 的大小来决定。

例如查找一维 n 元整数数组中最大元素的算法。该算法依次遍历数组中的元素,并保存当前的最大元素,被称为"最大元素顺序检索"。下面就是使用 C 语言编写的程序:

```
int largest(int * array,int n){                    // 找最大值
  int currlarge=0;                                 // 赋初值
  for (int i=0;i<n;i++)                            // 进入循环
    if(array[i]>currlarge                          // 找大元素
        currlarge=array[i];                        // 存储大元素
    return (currlarge);                            // 返回最大元素值
}
```

其中,问题的规模为 n。这 n 个整数放在数组 array 中,基本操作是"检查"一个整数,即把一个存放现有最大值的变量与之作比较。我们可以认为,这样检查数组中的某个整数所需要的时间就是一定的,与该整数的大小或其在数组中的位置无关。

一般来说影响时间代价的最主要因素是输入的规模,我们经常把执行算法所需要的时间代价 T 写成输入规模 n 的函数,记作 $T(n)$。

我们把 largest 函数中检查一个元素所需要的时间记为 C。C 中包括变量 i 增值的时间,找到一个新的最大元素时要做的工作,不考虑 C 的实际值,也不考虑函数初始化时所需要的一小部分额外时间,只想得到执行该算法的一个合理的近似时间。因此,运行 largest 函数的总时间可近似地认为是 Cn(共需要 n 步检查工作,每一步需要时间(C)。我们说 largest 函数(或者说是最大元素顺序检索法)的时间代价可以用下面的等式来表示:

$$T(n)=Cn$$

这个等式表明了最大元素顺序检索时间代价的增长率。

又如,一个整数数组的第一个元素值赋给另一个变量,只要复制这个元素的