高等财经院校
试用教材

数据结构

杨开汉 主编

中国财政经济出版社

数据结构

高等财经院校试用教材

数 据 结 构

杨开汉 主编

中国财政经济出版社

高等财经院校试用教材

数　据　结　构

杨开汉　主编

*

中国财政经济出版社出版

（北京东城大佛寺东街8号）

新华书店北京发行所发行　　各地新华书店经售

通县西定安印刷厂印刷

*

787×1092毫米　16开19.5印张　477,000字

1989年3月第1版　　1989年3月北京第1次印刷

印数：1—　5000　　定价：3.50元

ISBN 7-5005-0438-8/F·0446（课）

前　言

本书是受财政部委托，按高等财经院校经济信息管理专业本科四年制教学计划数据结构课程教学大纲要求编写的教材，也适应于其它管理专业和计算机应用专业。它还可以作为计算机科技工作者及其有关专业人员的参考书。

“数据结构”发展至今，已成为一门比较成熟的课程。它是计算机系统软件和应用软件研制者的必修课程。数据结构理论的应用范围已经深入到编译系统、操作系统、数据库、人工智能、信息科学、企业管理、系统工程、计算机辅助设计及其它几乎所有工程技术领域。经济信息管理人员必须掌握数据结构知识，以求能用计算机高效率并得心应手地解决常遇到的非数值计算应用问题。以此为目的，本书在阐明各种基本数据结构概念及其主要运算的算法时，还着重叙述其应用，尤其是在经济信息管理方面的应用。

本书简明扼要地叙述了各种基本数据结构的概念，包括数据结构的逻辑定义、物理实现及其运算，并举例说明怎样用这些抽象的概念来解决实际问题。叙述时，由浅入深，由简及繁，回避了复杂的数学定义与推导，力求通俗易懂，以使几乎没有计算机专业基础知识的读者也能顺利地自学全书的内容。通过本书的学习不仅能正确地掌握数据结构的基本理论，并能运用这些理论来解决实际问题。

本书由笔者集多年来计算机软件设计实践及多次讲授数据结构课程的体会，并参考分析国内外数据结构书籍文献编写而成。内容的取舍以重应用为特征而区别于目前已出版的其它数据结构书籍。本书采用易于教学的类PASCAL语言描述算法，并进行了适当的算法复杂性分析。用类PASCAL语言给出的算法虽然不能直接上机执行，但只须稍加修改就可以变成可上机执行的程序。不直接使用PASCAL语言书写算法，仅仅是为了避开标准语言的严格语法规定，而使算法一目了然，清晰易读。考虑到读者可能对BASIC语言较为熟习，有的地方我们还用TRUE BASIC语言举例说明数据结构的应用。供读者了解如何用BASIC语言变通地实现某一种数据结构，这样做也是为了强调应用。

“数据结构”不但是一门理论性很强的课程，同时又是一门实践性很强的课程。在每一章的最后，都安排了适当的习题，供读者练习。教师在教学中还应安排学生适当上机实习，以加强实践。

本书共分十章，介绍了线性表、栈、队列、串、树、图、查找、内部分类、文件、外部分类等基本数据结构及其算法与应用例题。内容丰富，逻辑性强，条理清楚。全书内容按教学大纲规定的72学时安排取材编写，打“*”号的章节供参考与选讲。预期读者学完此书后，对数据结构会有较全面且系统的认识，能大大地提高非数值计算应用软件设计的水平。

为了便于阅读，程序书写采用缩进方式，本书所有例题均在IBM-PC机中通过。

本书由中南财经大学、上海财经大学和东北财经大学的经济信息管理系联合编写。中南财经大学杨开汉编写第一、二、三、五、七章，上海财经大学林立忠编写第六、九、十章、

东北财经大学王焕高编写第八章、中南财经大学王少波编写第四章，杨开汉主编。国防科技大学王广芳副教授仔细地审阅了全书并作了修改。中南财经大学胡久清教授就全书的编写工作提出了指导性意见。吴盘珍、何友鸣、丁云等同志提出了许多宝贵的意见。

本书在编写和出版过程中曾得到财政部教育司和中南财经大学各级领导的支持和鼓励。对此表示衷心的感谢。

由于水平有限，疏漏之处再所难免，望读者悉心指正。

编　者

1987年12月

目　录

第一章 绪 论

自从世界上第一台电子计算机问世以来，在短短的四十余年时间中，计算机科学的发展之快，已远远超过人们原来对它的估计。计算机的应用范围不仅已深入到各个领域，而且在具体应用上也不再是限于计算，而是更多地应用到数据处理和实时控制方面，也就是已从数值计算为主转到非数值计算。在非数值计算应用中，计算机加工的数据对象往往是大量的数据。这些数据存取、查找、插入和删除等等操作效率的提高，仅仅依赖程序设计的技巧已经无法达到目的，必须对这些被加工数据的组织形式加以研究，找出最佳的数据组织形式，并与好的程序设计技巧相配合，才能达到提高效率的目的。其实，有时还不只是一个效率的问题；在一些情况下，若没有好的数据组织形式，就根本无法完成所需要做的工作。这也是我们为什么要学习并研究数据结构这门科学的原因。

§1.1 什么是数据结构

为了说明这一问题，我们先回顾一下我们现实生活中的两个例子。

当你拿起一本厚厚的汉语字典查找某一个汉字时，你首先必须知道你使用的字典的编码方法，然后才能按照偏傍部首、四角号码或者拼音等相应的编码方法较快地查到你所需要查找的汉字。你为什么能如此顺利地在几万个汉字中找到你所需要的汉字呢？这是由于字典中的每一个汉字都是按偏傍部首或四角号码或者拼音的规律严格地安排在它应处的页行(编码)上。查找时也必须遵守相同的规律。倘若不按某一规律，将几万个汉字任意安排，你为了查找某一个汉字就不得不从字典的第一页开始逐页地查找了。由这个例子我们可以看出，高的查找效率是与字典中汉字的安排规律——也就是组织形式——密切相关的。自然，对于不同组织形式必须采用相应的查找办法才能达到提高效率的目的。就如同四角号码字典，采用偏傍部首方法无从查找一样。

另一个例子是，当你要给你的朋友打电话时，你不知道他的电话号码，可并不着急，翻开桌上的电话号码本你很快找到他们单位的电话号码，你给这个单位打了电话，请该单位的人帮你找到你的朋友和你通话。在这个例子中，首先你按系统或街道等规律，在厚厚的电话号码簿上很快地找到了你朋友所在单位的电话号码，然后又通过你朋友单位的人再找你的朋友通话。

这是两个并不陌生的例子。字典和电话号码簿，是按不同的规律将汉字和电话号码这些数据进行组织与安排的，以适应不同的目的和要求。这样可以在实现与组织形式相应的目的时，大大地提高操作效率。请注意在第二个例子中，为了与你的朋友通话，经历了两个查找过程，查电话簿和传呼人找你的朋友。

上面说的是日常生活中的“数据结构”的例子。这类例子想必大家都可以遇到。在计算机中，与此类似，大量的数据存贮在存储器中（主存和辅存），等待加工。为了查找的方便，

应将这些数据按某些规律存放在确定的存储单元中。这样才能在数据的加工过程中采用相应的办法高效率地进行数据的访问、插入和删除等运算，从而也提高了数据加工的效率。这种数据存放规律，就是指数据的组织形式即数据结构形式。

综上所述，我们可以这样给数据结构下个定义：**数据结构就是研究计算机中大量数据存储的组织形式，并定义相应的运算以提高计算机的数据处理的能力的一门科学。**

§1.2 基本术语介绍

在本章中，我们将用到许多术语和名词。我们现在给出确切的定义，以便在今后的学习中能有统一的概念。

数据（data）：在计算机中，数据这个名词的含义异常广泛，可以认为它是描述客观事物的数字、字符以及所有能输入到计算机中并能为计算机所接受的符号的集合。

数据元素（data element）：它是数据的基本单位，是数据这个集合中的个体。在数据存储组织中，它是基本的处理单位，正由于此，它的含义也十分广泛。根据不同的需要它可以是一个数，一个字符，一个字符串，一个描写客观事物的记录，甚至可以是一篇文章。

数据对象（data object）：它是具有相同特性的数据元素的集合，是数据的一个子集。例如，当我们研究整数时，其数据对象是{0，±1，±2，±3，…}，而当我们研究字母字符时，其数据对象是{A，B，C，…}。

数据结构（data structure）：简单说来，数据结构是指计算机处理的数据元素的组织形式和相互关系。在第一节中我们举了汉语字典和电话号码本的两个例子，每一个汉字在字典中的位置安排，某一电话号码在电话号码本上的位置安排都不是孤立的，它遵循某种规律，相互间有着某种联系，可以认为它是某种数据结构形式。计算机中数据元素的存储，元素间也有着类似的组织关系，即它遵循某种组织规律，我们称它为数据结构。而在讨论时，又分为数据的逻辑结构和数据的物理结构。所谓数据的逻辑结构是从逻辑上来抽象地描述数据元素间的结构关系。而数据的物理结构（又称存贮结构）是数据的逻辑结构在物理存贮器中的映象，即逻辑结构在物理存贮器中的具体实现。对程序设计者来说，数据的逻辑结构和物理结构是密切相连的两个方面。

数据类型（data type）：它是程序设计语言中所允许的变量的种类，也是变量可以取的值和可以进行运算的集合。每一种程序设计语言都有一组它所允许使用的基本数据类型。如在FORTRAN语言中允许五种基本数据类型：整数、实数、双精度数、复数和布尔量。而每一种数据类型都有它取值的范围和所允许的运算。在PASCAL语言中，除提供整数等四种标准类型外，还有其它非标准类型，甚至允许自己定义数据类型和它的取值范围。各种语言所能提供的数据类型的多少决定了该语言的功能的强弱。我们可以把数据类型看成程序设计语言中已经实现了的数据结构。如复数、数组等。因此，数据类型也可以认为是数据结构（包括逻辑结构和物理结构）的另一种称呼。

算法（algorithm）：算法是非空的、有限的指令序列，遵循它就可以完成某一确定的任务。算法这个词并不是计算机出现后才有的。早在古代就有欧几里德算法（求两个数的最大公因子）和孙子算法（求若干个数最小公倍数）。只是随计算机科学的发展，对算法的研究也飞跃发展。算法并不是指通常所说的程序，它有五大特征：

（1）有穷性。一个算法在执行有限步骤之后必须终止。

（2）确定性。一个算法所给出的每一个计算步骤必须是精确定义的，无二义性。

（3）可行性。算法中要执行的每一个计算步骤都可以在有限时间内完成，可行性与有穷性和确定性是相容的。

（4）输入。一个算法一般都要求有一个或多个输入信息（个别的算法可能不要求输入信息）。这些输入量是算法所需的初始数据。它取自某一特定的集合。

（5）输出。算法一般有一个或多个信息输出，它是算法对输入信息的执行结果。

由以上五个特征可以看出，算法是不同于一般的程序的。例如，程序可以在无外来干涉情况下一直执行下去，程序可以既无输入又无输出信息。

还有一点需要说明的是，数据结构的学习是在学员已经熟悉地掌握了一门高级语言之后进行的。所以高级语言中已经学习了的一些术语和概念还将用到数据结构的学习中，但有时同一个术语在高级语言和数据结构中又会稍有区别，这些区别主要是由于问题提出的角度不同而带来的，高级语言中谈到的数据类型（即数据结构）主要是从应用角度提出的，而本书讨论数据结构时提到的同一种术语往往是从逻辑结构和物理实现上来讨论。请读者在遇到这些问题时联系前后文，体会其中的细微区别，这有利于我们搞清问题。例如，我们在第二章讨论数组（概念与实现）之前已经用到了高级语言中数组的概念了，在讨论分类、查找之前我们也不得不用到一些简单的分类、查找技术来说明问题，这是难以避免的。数据结构的学习是把问题研究得更深入、广泛而已。

§1.3 数据结构的发展和它在计算机科学中的地位

“数据结构”最早是1968年在美国被确定为一门独立的课程的。在此之前，数据结构课程的一些内容曾在其它课程中，如表处理语言中讲述。1968年，美国一些大学虽将数据结构规定为一门课程，但该课程的内容并未作明确的规定。其后发表的一些文件和出版的书籍中，对数据结构这个术语有各种不同的解释和理解。最初，数据结构几乎是图论，特别是表和树的理论的同义语，随后这个概念又扩充到包括网络、代数、集合论、关系等现在称之为“离散数据结构”的那些内容，它与现在的“数据结构”的内容结合在一起，统称为“数据结构”。

由于数据需要在计算机中处理，因此不能局限于数据本身的数学概念的研究，还必须考虑到数据的物理结构，即数据在存贮器中如何存贮的问题，这就进一步扩大了数据结构的研究范围。

1968年，著名的美国计算机科学教授唐·欧·克努特所著《计算机程序设计技巧》的第一卷《基本算法》，是第一本系统地阐述数据的逻辑结构和存贮结构以及运算的著作。从60年代末到70年代初出现了大型程序，软件与数据相对独立，结构程序设计成为程序设计方法学的主要内容，人们越来越感到数据结构的重要。认为程序设计的实质是对确定问题选择一种较好的数据结构加之一种好的算法。至今，对数据结构的研究仍在发展中。

由于数据结构是高级程序设计语言、操作系统、数据库、人工智能等课程的基础，同时，数据结构技术也广泛地应用于信息科学、系统工程、应用数学以及各种工程技术领域，所以数据结构不仅仅是计算机专业和信息管理专业的核心课程之一，而且也是其它与计算机（应

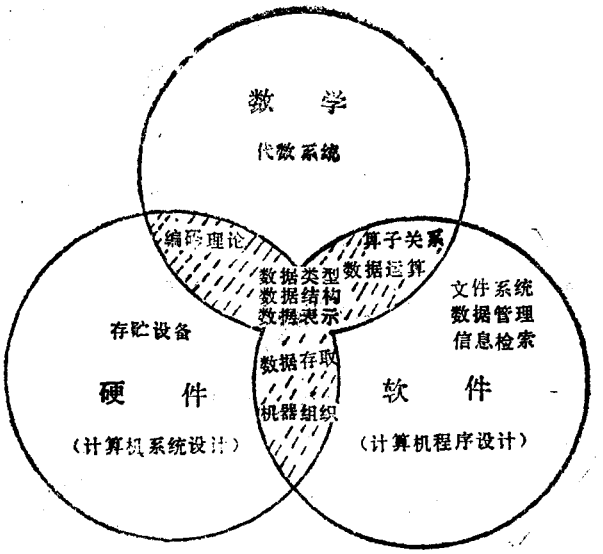

图1-1 数据结构与其它课程的关系

题时减少时间和空间（存贮单元）的开销。

用）有关的专业的必修课程。

数据结构的研究涉及的知识面十分之广，可以认为它是介于数学、计算机硬件和软件之间的一门核心课程。图 1-1 表示出数据结构与其它课程的关系。

数据结构的侧重点在于实践技术。我们研究各种数据结构的性质，定义相应的算法并分析算法的效率，同时研究各种数据结构的应用范例，其目的全在于引导我们设计出好的数据结构与程序，提高计算机的数据处理能力。换句话说，数据结构的研究目标在于在解决具体数据处理问

§1.4 学习数据结构的基本知识

一、类 PASCAL 语言语法说明

本书采用类 PASCAL 语言为教学语言，之所以用类 PASCAL 语言来描述算法而不直接采用 PASCAL 语言技术，是为了避开 PASCAL 语言程序所必须遵循的繁琐的规定，使读者更加一目了然，清晰地阅读算法。为此，我们需要先介绍一下这种假想的类 PASCAL 语言的语法规定。

1. 本书中所有的算法都以过程和函数形式表示，它们是：

```
procedure  过程名（参数）；
  begin
      语句组
  end; {过程名}
function 函数名（参数表）：类型名；
  begin
      语句组
  end; {函数名}
```

其中，参数表中可以有若干实参数和形式参数，语句组由一个或一个以上的语句构成，用“；”号作为两个语句间的分隔符。在正常情况下，过程结束于 END；非正常结束出口语句用 RETURN 表示跳出过程。在函数过程中，函数值直接由已赋参数值的函数名返回。

2. 除过程中的参数表外，所有过程中出现的局部变量允许不加变量类型说明（加说明也可以），并设想该语言允许一切可能出现的变量种类。未加说明的变量类型，读者根据前

后文推断，或听任课老师说明。

3. 基本语句。

（1）赋值语句。

变量名：=表达式；

例：　a：=15；

　　b：="3514"；

（2）条件语句。它有如下两种形式：

(a)　if　条件　then　语句；

(b)　if　条件　then　语句1

　　else　语句2；

这两种形式的条件语句如图1-2所示。

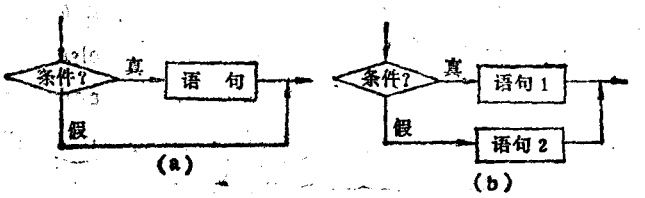

图1-2　条件语句的两种形式

（3）while 循环语句：

```
while　条件　do
　语句；
```

它的含义是当条件为真时，就执行 do 之后的语句，直到条件为假时才结束语句的执行。如果一开始条件就为假，则根本不执行 do 后面的语句。其执行过程如图1-3所示。

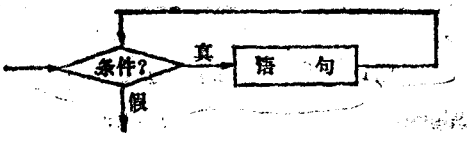

图1-3　while 循环语句执行过程

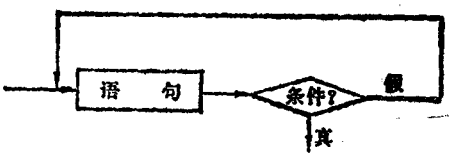

图1-4　repeat 循环语句执行过程

（4）repeat 循环语句。

```
repeat
　语句
until　条件；
```

它的含义是将 repeat 和 until 之间的语句执行若干次（至少一次），直到 until 之后的条件为真时才跳出循环。其语句执行过程如图1-4所示。

（5）for 循环语句。它也有两种不同的形式。

a)　for　$i_1 = e_1$　to　e_2　do

　　语句；

b)　for　$i_1 = e_1$　downto　e_2　do

语句；

在这两种形式的FOR循环语句中，e_1是循环初值，e_2是循环终值。第一种形式的循环增量为+1，而第二种形式的循环增量为-1，因而一般情况下，前者e_1小于e_2，而后者e_1大于e_2。它们的执行情况如图1-5所示。

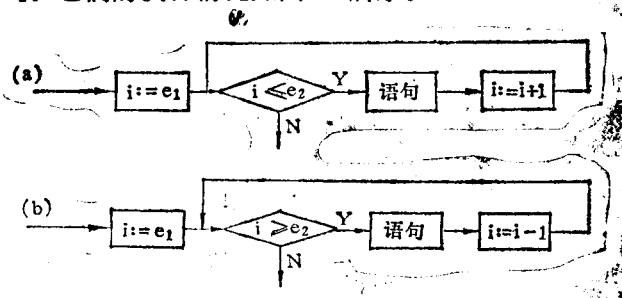

图1-5　两种不同的for循环语句的执行过程

（6）case（情况）语句。

```
case
条件1:语句1;
条件2:语句2;
    .
    .
    .
条件n; 语句n;
else    语句n+1;
end;
```

情况语句的执行过程如图1-6所示。说明一下，就是语句n+1可以没有，即不存在ELSE的情况也是允许的。这时情况语句变成下面形式：

```
case
条件1:语句1;
条件2:语句2;
    .
    .
    .
条件n: 语句n;
end;
```

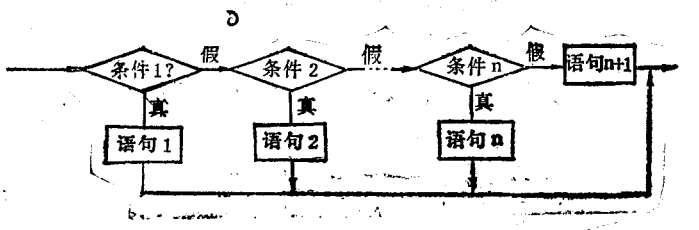

图1-6　Case语句执行过程

另一种形式的条件语句在算法中也允许出现：

```
case  变量名  of
    值1: 语句1;
    值2: 语句2;
      .
      .
      .
    值n: 语句n;
end;
```

（7）exit

```
exit
```

它的作用是用在循环中需要跳出循环的地方。

（8）过程调用语句：

```
call  过程名（参数表）;
```

(9) 函数调用语句：

变量名:=函数名（参数表）；

(10) 出错处理语句：

error （字符串说明）；

(11) 输入/输出语句：

read（变量表）；

write（变量表）；

(12) 复合语句：

```
begin
  语句1;
    .
    .
    .
  语句n;
end;
```

由两个或两个以上的语句构成的语句为复合语句。复合语句中的成份语句组按其书写次序顺序执行，成份语句组写在 begin 和 end 之间实际上 begin 和 end 起着括号的作用。

(13) 结束语句。

```
end;
```

它用以指出算法的结束。

我们在描述算法时，采用上述语句形式的类 PASCAL 语言。可以很方便地将算法改写成真正的 PASCAL 程序。我们书中的实例则用 PASCAL 语言书写，并在 IBM—PC/XT 机上通过。

二、算法分析方法说明

前面已经说到，在编制程序之前，首先应该选择一个恰当的数据结构和一个好的算法。那么，如何衡量一个算法的好坏呢？

显然，首先应确保算法是正确的，此外，通常还有三方面的考虑。

（1）依据算法所编制的程序在计算机中运算时所消耗的时间。

（2）依据算法所编制的程序在计算机中运行时占有内存容量的大小（也要考虑占辅存容量的多少）。

（3）算法是否易读，易于转换成任何其它可运行的语言程序以及是否易于调试。

从主观上说，我们希望选择一个既不占很多存贮单元，运行时间又短且简明易读的算法。然而，实际上不可能做得十全十美。往往是，一个看起来很简单的程序其运行时间要比复杂的程序慢得多，而一个运行时间较短的程序占用内存单元也多。因此，在不同的情况下应有不同的偏重选择。如果算法使用次数较少则应力求简明易读，易于转换成上机程序，若算法转换成的程序反复运行多次，则就选择运行时间尽可能少的算法。若待解决的问题数据量大，而所使用的计算机存贮容量又较小，则相应算法应着重考虑如何节省内存单元。在本书中，

主要讨论算法的时间特性，有时候也考虑一下空间特性。

一个程序在计算机上运行所消耗的时间主要取决下述因素：

（1）程序运行时所需要输入的数据总量。

（2）对源程序进行编译所需要的时间。

（3）计算机执行每条指令所需要的时间。

（4）计算机指令重复执行的次数。

上面四条，前三条取决于执行算法的计算机的硬、软件系统的客观条件。因此，一般把第四条即语句的重复执行次数作为算法的时间变量。

为此，我们引入语句频度的概念（frequency count）。所谓语句频度即为语句重复执行的次数。例如，两个 $n \times n$ 的矩阵相乘，其算法可描述如下：

```
procedure matrix—product (a,b,n) ;
 begin
    for  i:=1  to n do                          n+1
      for  j:=1  to n do                        n(n+1)
        begin
          c[i, j]:=0;                           n²
          for k:=1 to n do                      n²(n+1)
            c[i,j]:=c[i,j]+a[i,k]*b[k,j];       n³
        end;
 end;
```

其中，每一语句的频度如上述算法右列所示。整个算法中所有语句的频度之和可约定作为该算法执行时间的度量，记作：

$$T(n) = 2n^3 + 3n^2 + 2n + 1$$

显然，它是矩阵阶 n 的函数，并且当 $n \to \infty$ 时，有 $T(n)/n^3 \longrightarrow 2$ 若引入 "O" 记号（读作 "大O"），则有 $T(n) = O(n^3)$，其意为在 n 较大时，算法的执行时间与 n^3 成正比。或者说，$T(n)$ 数量级与 n^3 数量级相同，我们称 $T(n)$ 为算法时间复杂性（time complexity）。

一般情况下，n 为问题的规模的度量，如矩阵的阶，多项式的项数，图中顶点的个数等等。一个算法的时间复杂性为 $T(n) = O(f(n))$。因此，通常可以通过判定程序段中重复执行次数最多的语句的频度来估算算法的时间复杂性。

例如，下面有三个简单的程序段。

```
(a)   x:=x+1;
(b)   for i:=1 to n do x:=x+1;
(c)   for i:=1 to n do
          for j:=1 to n do x:=x+1;
```

假定(a)中语句 x:=x+1 不在任何的循环中，则它的语句频度为 1，其执行时间是一个常数，而在(b)中，同一语句执行 n 次，其频度为 n。显然，在(c)中，语句频度为 n^2。因此，这三个语句的时间复杂度分别为 $O(1)$，$O(n)$，$O(n^2)$。它们分别称为常数阶，线性阶和平方阶。算法还可能呈现的复杂度有对数阶 $O(\log n)$，指数阶 (2^n) 等等。图1-6表示出不同数量级的特