国家精品课程配套教材

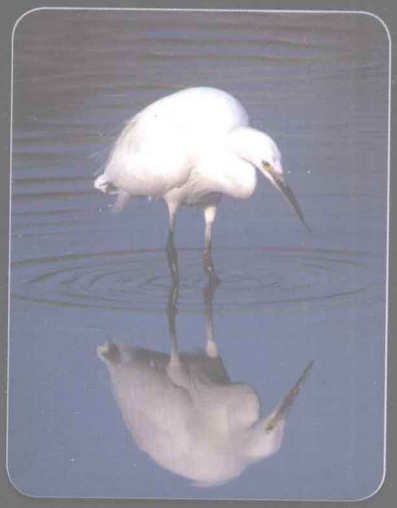
21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

数据结构（C++描述）

Data Structures In C++

胡学钢 张晶 主编



精品系列

人民邮电出版社
POSTS & TELECOM PRESS

国家精品课程配套教材

21世纪高等学校计算机规划教材

数据结构（C++描述）

Data Structures In C++

胡学钢 张晶 主编

精品系列

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

数据结构 : C++描述 / 胡学钢, 张晶主编. -- 北京
人民邮电出版社, 2011.8
21世纪高等学校计算机规划教材
ISBN 978-7-115-25078-0

Ⅰ. ①数… Ⅱ. ①胡… ②张… Ⅲ. ①数据结构－高等学校－教材②C语言－程序设计－高等学校－教材 Ⅳ. ①TP311.12②TP312

中国版本图书馆CIP数据核字(2011)第083605号

内 容 提 要

“数据结构”是计算机类各专业重要的专业基础课程，是提高软件设计水平以及学习后续课程所必需的基础。课程中涉及软件设计中常见的几种数据结构及其在计算机内存中的表示(即存储)形式和各种操作的实现，以及软件设计中常用的排序和查找运算。

本书共11章，内容包括概述、顺序栈、顺序队列、链栈和链队列、线性表、数组和广义表、递归技术、树和二叉树、图、查找、排序等内容，并配有相关的习题。全书内容安排由易到难，逐步深化，便于学习；内容组织上，以特定的知识框架组织知识，便于学生的学习、复习和主动学习；针对难度较大的章节，以模块化方法分解和组织教材内容，降低学习难度；以丰富的例题讨论来加强算法和程序设计的分析，引导和加深学生对技术的理解；以通俗的语言讲解内容，便于学生的理解。全部内容安排避免了概念和理论的平铺直述，因而容易激发学生的学习兴趣，具有较好的学习效果。

本书是工程、应用型计算机类相关专业的“数据结构”课程教材，也可作为其他专业学生学习“数据结构”课程的教材或参考书。

21世纪高等学校计算机规划教材

数据结构（C++描述）

◆ 主　　编　胡学钢　张　晶
责任编辑　邹文波

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街14号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京昌平百善印刷厂印刷

◆ 开本：787×1092　1/16
印张：14　　2011年8月第1版
字数：366千字　　2011年8月北京第1次印刷

ISBN 978-7-115-25078-0

定价：29.00元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前 言

在计算机科学与技术专业以及相关专业的课程中，“数据结构”是重要的专业基础课程，是提高软件设计水平以及学习后续课程所必需的基础。课程中介绍软件设计中常见的线性表、串、栈和队列、数组、树和二叉树、图、文件等数据结构及其在计算机中的存储结构和各种操作的实现，介绍软件设计中常用的排序和查找方法，并讨论有关运算的性能。通过对这些内容的学习，将使学生能熟练地掌握各种常用结构的特性，各种运算的实现方法及其性能，并能在实际应用中根据具体问题的要求设计出合理的数据结构和运算。

然而，由于该课程中的内容多，而且许多内容较抽象，特别是其中大量的算法、所用到的递归技术，以及缺乏有效的实验条件，使学习“数据结构”课程的难度较大。

为此，本书重新组织了课程内容，尽量以通俗易懂的方式介绍有关知识和技术，并给出必要的例题及其分析，以展示课程中知识的应用及其学习方法，因而易于激发学生的学习兴趣，收到良好的学习效果。这是本书的特色之一。

本书中各章的主要内容如下。

第 1 章介绍“数据结构”课程的研究内容、特点以及一些所需的基础知识和学习方法，有助于整个课程的学习。

第 2 章介绍顺序栈的逻辑结构、运算定义、顺序存储形式、C++描述及相应的运算实现，最后给出了栈的应用实例。

第 3 章介绍顺序队列的逻辑结构、运算定义、顺序存储形式与循环队列、C++描述及相应的运算实现，最后给出了队列的应用实例。

第 4 章介绍栈和队列的链式存储结构，即链栈和链队列，讨论了存储形式与运算实现，并分析了相关的性能。

第 5 章介绍线性表和串的逻辑结构与运算，重点讨论线性表的顺序表和链表这两种存储结构及其基本运算实现，并给出了大量例题及分析讲解。

第 6 章重点讨论递归技术，介绍递归的应用场合、内部实现原理、递归程序的阅读、递归程序的正确性证明和递归程序的模拟等，并给出了递归技术在算法设计中的应用实例。

第 7 章简要讨论数组和广义表的相关内容与应用。

第 8 章讨论树和森林的逻辑结构、存储结构和运算，介绍二叉树的概念、性质、存储结构，重点讨论二叉树的遍历运算方法及其应用，讨论线索二叉树的概念、应用及相关算法，并给出了相应的例题及分析，另外还介绍了哈夫曼树的有关知识及其应用。

第 9 章介绍图的有关概念、运算和应用，介绍图的两种常用的存储结构，重点讨论了两种遍历算法的方法和应用，最后以通俗的方式讨论了最小生成树、拓扑排序、最短路径等几个图的应用问题及求解算法。

第 10 章所讨论的查找是软件设计中常用的基本运算，讨论了在顺序表、树表和散列表等几种表结构上的查找方法及其性能。

第 11 章重点讨论了各种排序的方法、算法及其性能分析。

全书由几位作者合作完成，其中合肥工业大学计算机与信息学院胡学钢负责整书的策划、组织，并具体编写部分内容；另外参与编写的有合肥工业大学的张晶、张玉红和吴共庆。

虽然本书是为应用型本科计算机类专业编写的，但由于所具有的系统性和通俗性，也可以作为相近层次及专业以及非计算机专业本科“数据结构”课程的教材和有关人员的教学参考书。

由于编者水平及时间所限，书中难免存在错误和不足之处，恳请读者批评指正。

胡学钢

2011 年 5 月于合肥工业大学

目 录

第1章 概论

内容概要

数据结构是计算机类专业重要的专业技术基础课程，重点研究和讨论程序设计中常用的基本技术，因此，本课程学习的效果直接关系到后续课程的学习和软件设计水平的提高。本章首先通过实例说明数据结构课程在软件设计中的作用以及在计算机类专业课程中的地位，然后介绍与整个课程有关的概念、术语、算法及其描述语言和算法分析等。

1.1 数据结构课程的研究内容

数据结构这门课程在计算机专业中有什么作用？下面从运用计算机解决实际问题的过程来讨论本课程在软件设计中的作用。

1.1.1 从几个程序设计问题的讨论开始

下面从几个较为典型的程序设计问题的讨论开始，以进一步揭示程序设计技术的主要内容。

例1.1 设计出求解两个整数 m 和 n 的最大公因子，记为(m,n)。例如，(1000,350)=50，即1000 和 350 的最大公因子为 50；(10000,197)=1，即 10000 和 197 的最大公因子为 1，两个数互质。

解：有关最大公因子的概念及其求解方法在小学数学课程中已有介绍，因而这是一个众所周知的问题。现在的问题是：**如何更快地求解**？下面讨论几种求解方法，并分析相应的时间性能，以展示求解方法的时间性能的差异。

方法 1 直接试探法

依次取 2～min(m,n)中的每个数来判断是否同时是 m 和 n 的公因子，最后一个满足条件的数即为所求的解。若没有找到这样的数，则其最大公因子为 1。

对应的求解函数描述如下：

```
int hcf(int m, int n)                    // 返回整数 m 和 n 的最大公因子(m,n)
{ int h,i;
  h=1;
  for (i=2; i<=min(m,n); i++)            // 试探 2～min(m,n)中的每个数是否是公因子
      if(m % i==0 && n % i==0)
        h=i;                             // 记录每个公因子的值
```

```
  return h;                                    // 所记录的最后的公因子即为最大公因子
}
```

性能分析：这种求解方法基于相关的概念，较为直观，因而易于被初学者接受和采用。然而，这种方法的时间性能不太好：由于试探 2～min(*m*,*n*)中的每个数是否是公因子，因此，在 *m* 和 *n* 都比较大时，试探的次数较大。另外，当有多个公因子时，仅最后求出的公因子是所需要的解。例如，在求解（1000,150）时，求出的因子有 2、5、10、50，其中最后求出的 50 是本题的解。

针对后面的这一问题，可通过改变试探的次序来加以改进，函数如下：

```
int hcf(int m, int n)                          // 返回整数 m 和 n 的最大公因子(m,n)
{ int h,i;
  h= min(m,n);
  while (m % i!=0 || n % i!=0)                 // 试探 2～min(m,n)中的每个数是否是公因子
        h--;                                   // 记录每个公因子的值
  return h;                                    // 所记录的最后的公因子即为最大公因子
}
```

这种方法对 *m* 和 *n* 存在较大公因子时的求解较快，如求解(1000,200)时，仅需一次试探就可以。然而，在更多的情况下，需要试探的次数较多，如求解(1000,150)就需 101 次试探才能求出其解为 50。

总体来说，这种直接试探法仅仅从定义的角度出发来求解，没有从内在的关系上作更深入的讨论，因此，求解性能不理想。

方法 2 质因子分解法

这种方法在中小学数学课本中应当介绍过，描述如下：

从 2 开始，依次寻找整数 *m* 和 *n* 的当前值的公因子，每当找到一个公因子 h_i (i=1,2,⋯,d)，则将 *m* 和 *n* 都整除 h_i，从而得到新的 *m* 的 *n* 值，然后对新的 *m* 的 *n* 值，重复上述操作，直到确定当前的 *m* 和 *n* 不存在公因子为止。此时，各 h_i 的积就构成了本题的解，即最大公因子。

例如，(1000,150)的求解的图示如下：

2	1000	150	
	500	75	

线上的两个数是 *m*, *n* 当前的值，左边的数是当前两个数的最小的公因子，线下的两个数是当前 *m* 和 *n* 分别整除左边的公因子所得到的商。

下面给出完整的求解示意图。

2	1000	150
5	500	75
5	100	15
	20	3

由此可得两个数的最大公因子是 2×5×5=50。

对应的函数如下：

```
int hcf(int m, int n)                          // 返回整数 m 和 n 的最大公因子(m,n)
{ int h=1; k=2;
```

```
  while (k<=min(m,n))                    // 搜索公因子
    if(m % k==0 && n % k==0)
       { m=m/k; n=n/k; h=h*k; }          // 求解出一个公因子，并去除两个数中的公因子
       else  k++;                        // 搜索后续的公因子
  return h;                              // 返回求解出的最大公因子
}
```

性能分析：如果 m 和 n 有多个公因子，则函数求解中的比较次数会急剧减少。例如，求解(1000,150)的过程中，试探的数的序列如下：

(m,n)…(1000,150)	(500,75)	(500,75)	(500,75)	(500,75)	(100,15)	(20,3)
k　2	2	3	4	5	5	共 6 个数
√				√	√	

表示对应的数是公因子

然而，如果 m 和 n 都较大，并且公因子数较少，特别是在两数互质时，则需要试探的数的个数将达到 min(m,n)。

方法 3 辗转相除法。

求解方法描述如下：

对给定的整数 m 和 n，有如下唯一的公式

$$m=n*q+r \qquad (0\leqslant r\leqslant n-1)$$

其中，q 为 m 除以 n 所得到的整数商，r 为对应的余数。

基于这一公式，可得到关于(m,n)的求解规则如下：

$$(m,n)=(n,r)$$

当 r=0 时，(m,n)=n

例如，(1000,150)和(1000,377)的求解分别如下：

(1000,150) = (150,100)
=(100,50)
=(50,0)=50　　解为 50

(1000,377) = (377,246)
=(246,131)
=(131,115)
=(115,16)
=(16,3)
=(3,1)
=(1,0)=1　　解为 1，即两个数互质

性能分析：这种方法无论 m,n 的大小以及是否互质，都能快速地求解出结果，因而是较为理想的求解方法。

总结

由上面的 3 种方法及对照可知，一个问题可能存在多种求解方法，但不同的求解方法的性能（如时间性能、空间性能以及其他方面的性能）可能有较大的差异。在许多情况下，如果某项性能（如时间性能）不当，将难以在计算机上实现求解。

例1.2　幻方问题。设计程序完成如下问题的求解：将 $1\sim n^2$（n 为奇数）放在一个 $n\times n$ 的方阵中，并要求每行、每列一级两个对角线上的元素的和相同。例如，n=5 时的方阵如下所示。其中，每行每列及两个对角线上的元素之和都是 65。

15	8	1	24	17
16	14	7	5	23
22	20	13	6	4
3	21	19	12	10
9	2	25	18	11

解：如果简单地试探，则由于其可能的放置方法有 $n^2!$ 种，因此，在 n 稍微大一点，如 n=9 时，就会有 81！个试探，显然难以实现。因此，必须要有可行的求解方法。针对这一类幻方问题，已经有了一个可行的求解方法，介绍如下。

将 1 放在(1,(n+1)/2)的位置上，然后往左上的位置上放入下一个数。若左上的位置已有数据，则将其放入该数的下一行中的同一列的位置上。若已是最左或最上面位置上的元素，则其下一个位置的寻找方法是：将方阵卷成一个纸筒，然后依此再按同样的方向再找下一个位置，直到 $n\times n$ 个元素全部放入为止。

依照这一方法，得到求解函数如下：

```
void magic(int A[][], int n)
{ int i,j,k,i1,j1;
  i=0; j=n/2;
  for (k=1; k<=n*n; k++)
      { A[i][j]=k;
        if (k %n==0)
            i++;
        else { i=(i-1+n)%n; j=(j-1+n)%n; }
      }
  }
```

性能分析：该函数没有用到试探，而是直接求解指定位置，因而其时间性能只与元素个数有关，非常理想。这体现了好的求解方法的重要性。

例1.3　“韩信点兵”问题。有一队士兵，确切人数不知，但若每 3 人一组，则余 2 人；每 5 人一组，余 3 人；每 7 人一组，余 5 人；每 11 人一组，余 4 人。编程解答下列问题：

① 至少有多少人？

② 若已知人数为 5 000～10 000，问共有多少个答案？

解：本题的求解可有两种方法。

一种方法是采用试探法，即对所有可能的整型值判断是否满足条件。显然，这种方法太费时间。

另一种方法是：逐个满足各条件，并在寻找满足下一个条件的解时仍保证前面条件成立。

下面给出第 2 种方法所对应的函数。

```
void nums()
  {
     int n=2;                              // 满足第 1 个条件的最小的解
     while (n % 5!=3)
```

```
  n=n+3;                                  // 保证第 1 个条件成立的前提下寻找满足 n%5=3 的 n 值
while (n%7!=5)
  n=n+15;                                 // 保证前两个条件成立的前提下寻找满足 n%7=5 的 n 值
while (n%11<>4)
  n=n+105;                                // 保证前 3 个条件成立的前提下寻找满足 n%11=4 的 n 值
cout<<n;                                  // 输出求解结果
while (n<5000)
  n:=n+1155;                              // 开始寻找 5000 以上的满足上述各条件的所有 n 值
cout<<"between 5000,10000:";
while (n<=10000)                          // 寻找 5000～10000 中的满足上述各条件的所有 n 值
  { cout<<n; n:=n+1155 };
}
```

例1.4　工作安排问题。有一项工作需要 A、B、C、D、E、F 中的若干人去做，由于某些原因，这些人之间存在一定的约束关系：

① A、B 中至少有一人要去。　　② A、B 不能同去。

③ A、E、F 中去两人。　　④ B、C 同去或不去。

⑤ C、D 中有一人去。　　⑥ D 若不去，则 E 也不去。

试编程求解出所有可能的安排。

解：本题要解决几个问题：如何表示一个人去或不去？如何表示约束关系（条件）？

由于每个人均可能去或不去，因此，可用布尔型变量来表示：若某人去，则相应的变量为真，否则为假。在这种表示下，几个约束条件就变成了逻辑表达式，原问题的求解就变成了求解能使逻辑表达式同时成立的方程的求解问题了。

由于采用布尔型变量，因此，所限定的条件也就容易用布尔表达式表示了。对此，可以采用逻辑表达式的推导求解，也可以采用程序试探的方法来求解，实现过程中，用整数代替布尔数。

另外，由各条件可知各变量之间的一些关系：由①、②可知 A=not B，由④知 B=C，由⑤知 C=not D，条件③可用 A、E、F 的与或，条件⑥为蕴含式。在此讨论的基础上，用试探法进行求解。由此可得程序如下：

```
void emp();
{  int A,B,C,D,E,F;
   cout<<"A  B  C  D  E  F";
   for (A=0;A<=1A++)
     { B=1-A; C=B; D=1-C;
       for ( E=0; E<=1; E++)
        for (F=0; F<=1; F++)
         if (A * E * (1-F)+ (A*(1-E) *F)+( (1-A)*E *F))* (D + (1-E))
          cout<<A<<B<<C<<D<<E<<F;
}
```

问题：①在程序中找出各条件的具体实现。

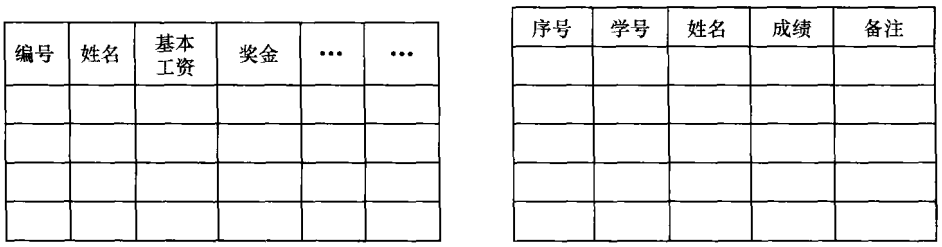
②若用 0，1 分别表示去或不去，则各条件应如何表达？试写出完整的程序。

1.1.2　用计算机解决实际问题的过程

在用计算机解决实际问题时，一般要经过以下几个步骤：首先对具体问题抽象出数学模型，

然后针对数学模型设计出求解算法，选择或设计合适的数据结构存储相关数据，最后编出程序上机调试，直至得到最终的解答。下面简述各环节的有关内容。

建立模型：一般情况下，实际应用问题可能会各式各样，如工资表的处理问题，学生成绩管理问题，电话号码查询问题等。这些问题无论所涉及的是数据还是其操作要求都可能存在一定的差异。尽管如此，许多应用问题之间还是具有一定的相似之处的。例如，如图 1-1 所示，虽然工资表和学生成绩表的具体信息（栏目）不同，但如果将两个表中的每个人的工资信息和成绩信息分别看做一个整体，则这两个表结构之间就有了某些共性，即各元素之间构成一个元素序列，如图 1-2 所示。从操作方面来看，虽然对这两种表的操作存在差异，但也存在一些相同或相似的基本操作。例如，查询一个人的工资信息和成绩信息，插入或删除元素，修改有关信息等。



（a）工资表示例　　（b）成绩表示例

图 1-1　数据表示例

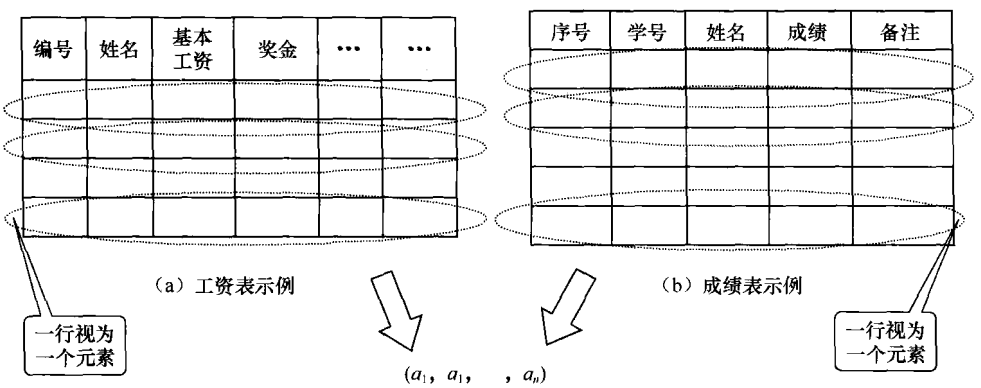


图 1-2　抽象的数据表示例

正因为许多不同的问题之间存在着的某些共性，使我们可以将一个具体的问题用这些共性的形式描述出来，这就是通常所说的**建立模型**。建立问题的模型通常要包括所描述问题中的数据对象及其关系的描述、问题求解的要求及方法等方面。建立问题模型有如下的好处：因为所涉及的许多基本模型在有关的课程中已有介绍，因而通过建立模型，就可以将一个具体的问题转换为所熟悉的模型，然后借助于这一模型来实现，并且易于保证其正确性。数据结构、离散数学及许多数学课程中就介绍了许多模型。例如，要描述一个群体中个体之间的关系时，可以采用数据结构和离散数学课程中所介绍的图结构。要描述一个工程内的关系或进展情况时，可以采用数据结构课程中所介绍的 AOV 网或 AOE 网等。即使所建立的模型没有现成的求解方法，借助于已有模型的适当组合也相对易于构造求解方法。

数值计算类问题的数学模型一般可由数学方程或数学公式来描述。然而，对于非数值计算问题，如图书资料的检索、职工档案管理、博弈游戏等问题，它们的数学模型是无法用数学方程或数学公式来描述的。在这类问题的处理对象中的各分量不再是单纯的数值型数据，更多的是字符、字符串以及其他用编码表示的信息。因此，首要的问题是把处理对象中的各种信息按照其逻辑特性组织起来，再存储到计算机中，然后设计出求解算法，并编写出相应的程序。

构造求解算法：在建立了模型之后，一个原始、具体的问题就转变成了一个用模型所描述的抽象的问题。借助于这一模型以及已有的知识（如数据结构中有关图结构的基本知识），可以相对容易地描述出原问题的求解方法，也就是后面将要介绍的算法。从某种意义上说，该算法不仅能实现原问题的求解，而且还可能实现许多类似的具体问题的求解，尽管这些具体问题的背景及其描述形式可能存在较大的差异。

设计存储结构：在构造出求解算法之后，就需要考虑在计算机上实现求解了。为此，需要做两方面的工作，其一是设计合适的存储结构，以便将问题所涉及的数据（包括数据中的基本对象及对象之间的关系）存储到计算机中。通过本书后续各章节的学习可知，不同的存储形式对问题的求解实现有较大的影响，所占用的存储空间也可能有较大的差异。

编写程序：在选择了存储结构之后，就可以做实现求解的另一项工作，即编写程序了。程序设计中涉及许多相关的技术，如递归技术。程序设计的风格也是重要的方面，良好的程序设计风格可以有效提高程序设计的质量，便于维护等。另外，存储形式和问题要求决定了编写程序的方法，如线性结构、树形结构所对应的程序设计就有较大的差异，本课程中就是以数据结构为主题来讨论程序设计。

测试：在编写出完整的程序之后，需要经过测试才能交付使用。按照软件工程的描述，测试的目的不是为了证明正确，而是为了发现错误。因此，充分地测试数据是实现测试目标的保证。

例如，假定要编写一个计算机程序以查询某市或某单位内的私人电话，要求对任意给定的一个姓名，若该人装有电话，则要求迅速找到其电话号码，否则指出该人没有装电话。

为解决这一问题，首先要构造一张电话号码登记表，表中每个登记项有两个信息：姓名和电话号码。在将众多的登记项合成一个数据表时，有多种不同的组织形式，查找的速度取决于表的结构及存储方式。

最简单的方式是把表中的信息，按照某种次序（如登记的次序）依次存储在计算机内一组连续的存储单元中。用高级语言表述，就是把整个表作为一个数组，表的每项（即一个人的姓名和电话号码）是数组的一个元素。查找时从表的第一项开始，依次查对姓名，直到找出指定的姓名或是确定表中没有要找的姓名为止。这种查找方法对于一个规模不大的单位或许是可行的，但对于一个有几十万乃至几百万私人电话的城市就不实用了。因此，一种常用的做法是把这张表按姓氏或姓氏笔画排列，并另造一张姓名索引表，这类似于汉语字典的组织形式。对这种表的查找过程可以先在索引表中查对姓氏，然后根据索引表中的地址到登记表中核查姓名，这样查找登记表时就无须查找其他姓氏的名字了。因此，在这种新的结构上产生的查找方法就会更有效。这两张表便是为解决电话号码查询问题而建立的数学模型。这类模型的主要操作是按照某个特定要求（如给定姓名）去对登记表进行查询。诸如此类的还有人事档案管理、图书资料管理等。在这类文档管理的数学模型中，计算机处理的对象之间通常存在一种简单的线性关系，故称这类数学模型为线性的数据结构。

数据结构课程涉及上述求解过程中的大多数步骤如下。

与建立模型的关系：数据结构课程中介绍了许多基本的数据结构模型及其运算实现。例如，线性表、栈和队列、树和二叉树、图、二叉排序树、堆等。通过学习，不仅可以掌握这些基本内容及其应用，还能根据实际问题选择或设计合适的模型。

与算法设计的关系：课程中对每种结构都讨论了相应的基本运算的实现，并且其中的一些算法是非常经典的，掌握这些基本运算的实现方法有助于进行更为复杂的算法设计。

存储结构：课程中对每种结构都讨论了其具体存储结构及其对运算实现的影响。例如，在第2章中所介绍的对顺序表做插入和删除运算，平均需要移动表中一半的元素，而采用链表结构则不需要移动元素。通过对这些内容的学习和比较，可以培养学生设计存储结构的方法和能力，从而使学生在实际应用时能根据问题的要求熟练地设计合理的存储结构。

与编程之间的关系：在实现各结构的算法编写时，涉及许多具有代表性的设计方法。通过对这些方法的学习有助于编程技术的提高。

综上所述，数据结构课程对提高软件设计人员的软件设计水平有较大的影响。也正因为如此，这一课程在计算机专业课程中具有极其重要的作用，是高校和研究单位的计算机专业研究生入学考试课程之一。

1.1.3 学习数据结构课程的意义

数据结构课程作为一门独立的课程在美国是从1968年开始的。在这之前，它的某些内容曾在其他课程中有所阐述。1968年，美国一些大学计算机系的教学计划中，虽然把数据结构规定为一门课程，但对其内容并未作明确规定。当时，数据结构几乎和图论，特别是和表、树的理论为同义语。随后，数据结构这个概念被扩充到包括网络、集合代数、格、关系等方面，从而变成现在称之为“离散结构”的内容。然而，由于数据必须在计算机中进行处理，因此，不仅要考虑数据本身的数学特性，而且还要考虑数据的存储结构，这就进一步扩大了数据结构的内容。

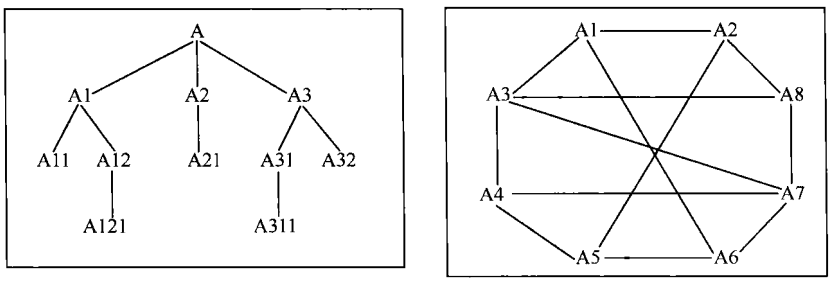
数据结构是计算机科学中一门综合性的专业基础课程。数据结构的研究不仅涉及计算机硬件（特别是编码理论、存储装置、存取方法等）的研究范围，而且和计算机软件的研究有着密切的关系，无论是编译程序还是操作系统都涉及如何组织数据，使检索和存取数据更为方便。因此，可以认为数据结构是介于数学、计算机硬件和软件三者之间的一门核心课程。在计算机科学中，数据结构不仅是一般程序设计的基础，而且是设计和实现编译程序、操作系统、数据库系统及其他系统程序和大型应用程序的重要基础。

目前，数据结构不仅是大学计算机专业的核心课程之一，而且还是一些非计算机专业的主要选修课程之一。

随着计算机应用领域的扩大和软、硬件技术的发展，“非数值性问题” 显得越来越重要。据统计，当今处理非数值性问题占用了90%以上的机器时间。从前面的例子可看到，解决此类问题的关键已不再是数学方法，而是设计出合适的数据结构。瑞士计算机科学家沃思（N.Wirth）曾以“算法 + 数据结构 = 程序”作为他的一本著作的名称。可见，程序设计的实质是对实际问题选择或设计好的数据结构，并设计好的算法。因此，若仅仅掌握几种计算机语言和程序设计方法，而缺乏数据结构知识，则难以应对众多复杂的课题，且不能有效地利用计算机。

1.2 基 本 术 语

数据是信息的载体，是指能够输入到计算机中，并被计算机识别、存储和处理的符号的集合。数据的形式较多，如工资报表、学生成绩表，一个家族关系的表示形式，表示一个群体中个体之间关系的图形描述等，如图 1-3 所示。



（a）家族关系示例　　（b）群体间关系示例（连线表示相互认识关系）

图 1-3　数据示例

虽然这些数据的形式及运算存在较大的差异，但可以找出其中的共性：都是由若干个具有独立意义的个体所组成的，个体之间存在着某些关系。对这些数据的运算也有某些相似之处。例如，在家族关系数据中，组成数据的基本个体是个人信息，其中的个人与个人之间存在着多种关系，如父子关系、兄弟关系、祖先-后代关系等，其中有些关系是直接表示出来的，还有一些关系则是隐含的。对家族关系数据，通常要涉及查询特定个体间的关系、插入和删除个体等。

数据结构课程中主要讨论这些具有共性的内容。为便于讨论，先给出几个有关的概念。

数据元素：数据中具有独立意义的个体。例如，工资表中的个人工资信息，成绩表中的学生成绩信息，家族关系中的个人等。在有些场合下，也称之为**元素**、**记录**、**结点**、**顶点**等。

字段（域）：虽然将具有独立意义的个体用元素来表示，但在许多情况下还需要特定个体的具体信息，因而涉及元素的字段信息。字段是对元素的详细描述，通常情况下，元素可能包含多个字段。例如，在图 1-1 所示的成绩表中，每个元素包括序号、学号、姓名、成绩和备注 5 个字段，分别描述了每个学生的有关信息。

数据结构：在图 1-1 及图 1-3 所示的几个数据示例中，元素之间各自具有一定的构成形式，这些构成形式被称为数据结构。**数据结构**是指组成数据的元素之间的结构关系。在图 1-1（a）、（b）中的元素依次排列，构成**线性关系**；图 1-3（a）中的元素是按树的形式构成**树形结构**；图 1-3（b）中的元素构成**图结构**。线性结构、树形结构和图结构是数据结构课程中的几类常见的数据结构形式。如果数据中的元素之间没有关系，则构成**集合**，这也是一种结构。

通常，称这几类结构为**逻辑结构**，因为只考虑了元素之间的逻辑关系，而没有考虑到其在计算机中的具体实现。

针对给定的逻辑结构，涉及这类结构的基本运算，以及在采用计算机实现时的存储结构的设计及运算的实现几个方面。因此，在用计算机解决涉及数据结构的问题时，需要完成如下任务。