第1章　概论

数据结构是计算机专业重要的专业基础课程，直接关系到后续课程的学习和软件设计水平的提高。本章首先通过实例来说明《数据结构》课程在软件设计中的作用以及在计算机专业课程中的地位，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》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谈谈本课程在软件设计中的作用。

1.1.1 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，最后编出程序上机调试，直至得到最终的解答。数值计算问题的数学模型一般可由数学方程或数学公式来描述。但是，对非数值计算问题，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、字符串及其它用编码表示的信息。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。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，例如我们所熟悉的工资表的处理问题，学生成绩管理问题，电话号码查询问题等。这些问题无论是所涉及到的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，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看作一个整体，则这两个表结构之间就有了某些共性。从操作方面来看，

虽然对这两种表的操作存在差异，但也存在一些相同的基本操作。例如，查询一个人的工资信息和成绩信息，修改有关信息等。

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。建立问题模型有这样的好处：因为所涉及到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。《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，我们可以采用《数据结构》和《离散数学》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，我们可以采用《数据结构》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，也相对易于构造求解方法。

构造求解算法：在建立了模型之后，一个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（例如数据结构中有关图结构的基本知识），我们可以相对容易地描述出原问题的求解方法即算法。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。

选择存储结构：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，其一是选择合适的存储结构，以便将问题所涉及到的数据（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。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。存储形式和问题要求决定了编写程序的方法。

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。

例如，假定要编写一个计算机程序以查询某市或单位的私人电话。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。

解决此问题时，首先要构造一张电话号码登记表，表中每个登记项有二个信息：姓名及电话号码。在将众多的登记项合在一起构成表时，有多种不同的组织形式，查找的速度取决于表的结构及存贮方式。

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）依次存贮在计算机内一组连续的存贮单元中。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划排列，并另造一张姓名索引表。这有点象汉语字典的形式。对这样的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需查找其它姓氏的名字了。因此，在这种新的结构上产生的查找方法就会更有效。这两张表便是我们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。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，故这类数学模型可称为线性的数据结构。

《数据结构》课程涉及到上述求解过程中的大多数步骤：

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。

与选择存储结构的关系：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，在第二章中所介绍的对顺序表作插入和删除运算，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，可使学生熟练地选择合理的存储结构。

与编程之间的关系：在实现各结构的算法编写时，涉及到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》对提高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。几乎绝大多数学校和研究单位的计算机专业研究生入学考试都将这一课程定为考试课程之一。

1.1.2 学习《数据结构》的意义

《数据结构》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它课程中有所阐述。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贮结构，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、存贮装置和存取方法等）的研究范围，而且和计算机软件的研究有着密切的关系，无论是编译程序还是操作系统都涉及到如何组织数据，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、数据库系统及其它系统程序和大型应用程序的重要基础。

目前在我国，《数据结构》不仅是计算机专业的核心课程之一，而且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斯（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一种好的数据结构，并设计一个好的算法。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，则难以应付众多复杂的课题，且不能有效地利用计算机。

1.1.3 学习数据结构的四种境界

第一种境界，能看懂各种数据结构的逻辑结构和存储结构，以及相应的运算。

第二种境界，能用算法描述语言描述各种数据结构及运算。

第三种境界，至少能用一种编程语言实现各种数据结构，验证简单应用。

第四种境界，能为实际应用程序自由的选择或设计合适的数据结构，不管什么语言和环境。

我们学习数据结构课程一般要求达到第三种境界，至少要达到第二种境界。如果达到第四种境界，那么你已经是程序设计高手了。

1.2 基本术语

数据是指信息的载体，是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，例如我们前面所述的工资报表、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 1－1 所示。

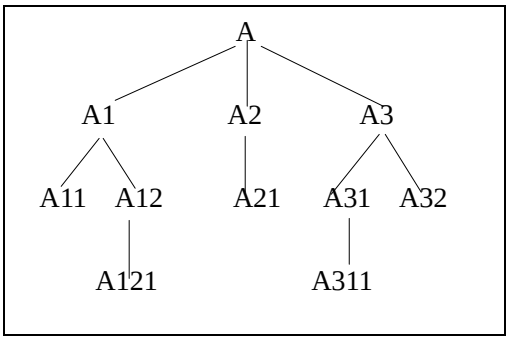
虽然这些数据的形式及运算存在较大的差异，但可以找出其中的共性部分：都是由若干个具有独立意义的个体所组成的，个体之间存在着某些关系。对这些数据的运算也有某些相似部分。例如，在家族关系数据中，组成数据的基本个体是人，其中的人与人之间存在着多种关系，例如父子关系、兄弟关系、祖先－后代关系等，其中有些关系是直接表示出来的，还有一些关系则是隐含的。对家族关系数据，通常要涉及到查询特定个体间的关系，插入和删除个体等。

编号	姓名	基本工资	奖金	…	…

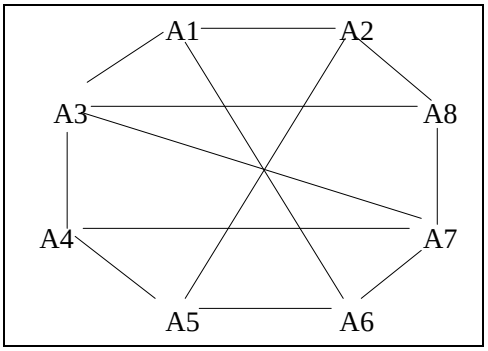
(a) 工资表示例

序号	学号	姓名	成绩	备注

(b) 成绩表示例



(c) 家族关系示例



(d) 群体间关系示例（连线表示相互认识关系）

图 1－1　数据示例

《数据结构》课程中主要要讨论这些具有共性的内容。为便于讨论，先给出有关概念。

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，因而涉及到了元素的字段信息。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、备注五个字段，分别描述了每个学生的有关信息。

数据结构：在图 1－1 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 中，(a),(b)中的元素依次排列，构成**线性关系**；(c)中的元素是按树的形式构成**树型结构**；(d)中的元素构成**图结构**。线性结构、树型结构和图结构是《数据结构》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**，因为仅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。

在用计算机解决涉及到数据结构的问题时，需要完成如下任务：

（1）数据结构的存储：为所涉及的数据结构选择一种存储形式，并将其存储到计算机中，这样就得到了数据结构在内存中的**存储结构**（有时也称为**物理结构**）。一种逻辑结构可能会有多种存储结构。例如，可以采用顺序存储，也可采用连接形式的存储。不同存储结构上所实现的运算的性能可能有一定的差异。

（2）数据结构**运算**的实现：在选择了数据结构的存储结构之后，就可以实现所给出的运算了。在本课程中，运算的实现一般是以算法的形式给出的，而算法大多以某种描述语言中的子程序的形式给出。由于本书主要以 C 语言作为算法描述语言，因而算法就以 C 语言中的函数形式给出。

由此可见，对一种数据结构，需要涉及到其逻辑结构、存储结构和运算**三个方面**。也就是说，对每种结构都要注意这三方面的联系。

由于不同的存储形式对算法的时间性能、空间性能等有较大影响，即使是相同的存储结构也可能会存在不同的算法实现，为此，需要解决这样的问题：究竟何种存储结构更为合适？什么算法更有效？为此，需要对算法进行分析，有关算法分析部分将在本章的后面部分讨论。通过分析，可以知道所实现的算法的性能及所选择的存储结构是否符合要求。

1.3 算法描述及分析

1.3.1 算法描述语言概述

如前所述，对数据结构的运算实现是以算法的形式来描述的。什么是算法？这很难给出一个严格的定义。简单地说，算法就是某类问题的求解方法。然而，这一描述太粗略。下面给出一个关于算法的描述：算法就是一段程序，该程序段对给定的输入可在有限的时间内产生出确定的输出结果。

算法可采用多种描述语言来描述，例如自然语言、计算机语言或某些伪语言。各种描述语言在对问题的描述能力方面存在一定的差异。例如，自然语言较为灵活，但不够严谨，而计算机语言虽然严格，但由于语法方面的限制，使得灵活性不足。因此，许多教材中采用的是以一种计算机语言为基础，适当添加某些功能或放宽某些限制而得到的一种类语言，这些类语言既具有计算机语言的严格性，又具有某些灵活性，同时也容易上机实现，因而被广泛接受。目前的许多《数据结构》教材采用类 PASCAL 语言、类 C++或类 C 语言作为算法描述语言。

本教材中选用的是以 C 语言为主体的算法描述语言，在 C 语言的基础上适当扩充一些功能或放宽某些限制。所涉及到的算法以 C 语言的函数形式给出。

考虑到读者已经学过 C 语言，因而不再对 C 语言部分作详细描述。下面补充一下所扩充的一些功能描述。

1．输入和输出语句：由于 C 语言中的输入和输出语句的形式对数据的类型有一定的限制，因此，本书中采用 C++中的独立于数据类型的输入和输出语句。

（1）输入：cin>>x; 其功能是读入从键盘输入的一个数，并赋给相同类型的变量 x.。其中变量 x 的类型可以是整型、浮点型、字符型等不同类型。

该语句可用下面的形式同时输入多个不同类型的变量。

```
cin>>x1>>x2>>x3>>x4>>x5;
```

（2）输出：cout<<exp; 其功能是将表达式 exp 的值输出到屏幕上。其中表达式 exp 的类型也可以是整型、浮点型、字符型等不同类型。

该语句可用下面的形式同时输出多个不同类型的表达式的值。

```
cout<<exp1<<exp2<<exp3<<exp4<<exp5;
```

2．最大和最小值函数 min 和 max

datatype min(datatype exp1, datatype exp2,…, datatype expn)：

返回表达式 expi(i=1,2,…,n)中的最小的值。其中元素类型 datatype 可以是各种类型。

datatype max(datatype exp1, datatype exp2,…, datatype expn)：

返回表达式 expi(i=1,2,…,n)中的最大的值。

3．交换变量的值：x1<==>x2;　交换变量 x1 和 x2 的值。

4．注释：为简捷起见，本书中程序的注释采用 C++中的注释形式，即在双斜线“//”后面的内容就是注释的内容。例如，下面的语句的右面就是一个注释。

```
A[I]=I*I;                    //此处为注释内容
```

1.3.2 算法分析

如前所述，为了了解算法的有关性能，需要对算法进行分析。通过分析，不仅可以知道算法的有关性能，而且由此还可知道所选择的存储结构是否符合要求。

衡量算法的主要性能指标包括时间性能、空间性能等，其中时间性能是指运行算法所需的时间的度量，而空间性能则是指运行算法所需要的辅助空间的规模。在《数据结构》中，大多数算法分析是针对算法的时间性能进行的。算法的时间性能以时间复杂度来衡量。

为了使算法的时间复杂度便于比较，不宜采用在某个具体机器上所运行的时间的形式来表示，一般是以算法中**基本语句的执行次数**来衡量的。然而，在实际应用时，基本语句的执行次数的精确计算是困难的，同时也是不必要的，因为许多算法中的语句的执行次数要取决于输入数据，可能会有多种复杂的情况。为便于计算，对这一时间复杂度大多采用一种近似的形式来描述，即采用基本语句执行次数的**数量级**来表示。此处所谓数量级是这样定义的：

如果变量 n 的函数 f(n)和 g(n)满足：$\lim_{n\to\infty}\frac{f(n)}{g(n)}$=常数 k(k≠∞,0)，则称 f(n)和 g(n)是同一数量级的，并用 f(n)=O(g(n))的形式来表示。

由定义可知，两个函数为同一数量级，强调的是在 n 趋向无穷大时，两者“接近”。例如，$\frac{n(n+1)}{2}$ 和 n^2 是同一数量级的，因为$\lim_{n\to\infty}\frac{\frac{n(n+1)}{2}}{n^2}$=1/2 为一个常数。

通过求解算法中语句执行次数的数量级所得到的时间复杂度忽略了其中的“细微”部分，得到了时间性能的近似描述，这一描述有助于对算法时间性能的简要了解。

例 1.1：求解下列各程序段的时间复杂度。

(1)　For (I=1;　I<n; I++)　x++;

解：虽然从循环语句的内部执行来看，该程序段中基本语句的执行数多于循环体的执行次数，但两者是同一数量级的，因此，该问题的求解可以以其循环体的执行次数的数量级的求解来实现。由于循环体执行 n-1 次，因此其时间复杂度为 O(n)。

(2)　For (I=1;　I<n;　I++)

```
        For (j=1;　j<=I;　j++) {x++;}
```

解：该问题同样以其循环体的执行次数的数量级的求解来实现。由于是双重循环，并且内层循环的循环次数不是常数，因此其计算稍微有些麻烦。计算方法如下：

I=1 时，内层循环执行 1 次（j 从 1 到 1）；

I=2 时，内层循环执行 2 次（j 从 1 到 2）；

I=3 时，内层循环执行 3 次（j 从 1 到 3）；

… …

I=n-1 时，内层循环执行 n-1 次（j 从 1 到 n-11）；

因此，最内层循环体共执行 n(n-1)/2 次，因而其时间复杂度为 $O(n^2)$。

(3) I=1;

while (I<n) I*=2;

解：对许多初学者来说，该问题的求解有些麻烦，求解方法如下：

由语句可知，循环次数和 I 之间的对应关系如下：

次数	*1*	*2*	*3*	*4*	*...*	*...*	*k*
I 值	2^1	2^2	2^3	2^4	…	…	2^k

假设最后一项即 $2^k=n$，则可知 $k=\log_2 n$。也就是说，循环次数 k 是 $\log_2 n$ 的数量级。即使 2^k 不是正好等于 n，k 也与 $\log_2 n$ 几乎相等。由此可知，该语句段的时间复杂度为 $O(\log_2 n)$。

本章小结

《数据结构》研究软件设计中的基本技术，是计算机专业重要的专业基础课程。课程中所研究的内容有助于实际问题求解的许多方面。

数据是计算机的操作对象，可以分解为元素的集合，每个元素中可能包含多个称为字段的描述信息。组成数据的元素之间按一定的结构形式组织起来，从而构成数据结构。

数据结构包含逻辑结构、存储结构和运算三个方面。数据结构的逻辑结构侧重描述元素之间的内在联系，不涉及数据结构的具体存储实现。数据结构的存储结构讨论数据的存储结构，不同的存储结构对运算的实现可能有较大的差异。数据结构的运算又可以分解为运算定义、运算实现即算法和算法分析三项内容。运算定义是从有关领域中提出的具有某些共性的问题描述。运算实现是在选定的存储结构上实现所描述的运算，从而得到算法。算法分析通过对所实现的算法的时间性能和空间性能等方面的分析，来确定算法或存储结构的性能。

有关数据结构几个方面的联系如图 1-2 所示。

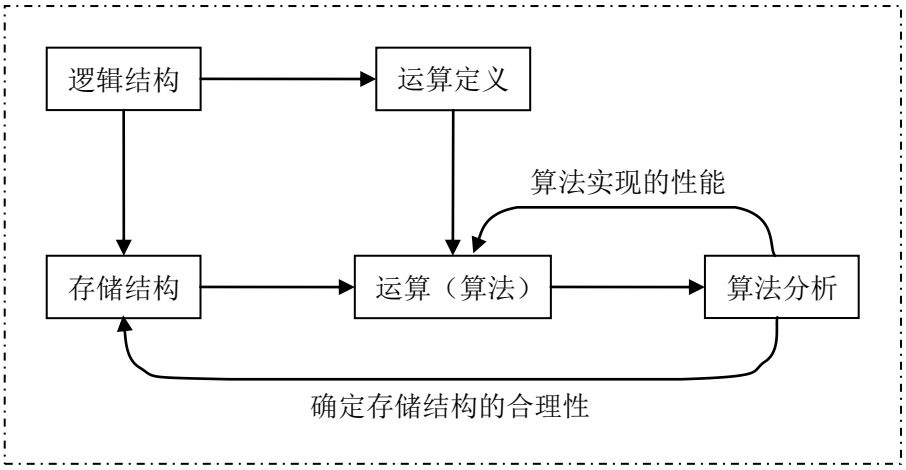


图 1-2 数据结构所涉及的几个方面的关系

算法的时间性能以时间复杂度来描述，是对算法运行时间的近似估计，通常以基本语句的执行次数的数量级来描述，其形式是以数据规模(最常见的是 n)的函数的数量级。