第 1 章 数据结构基本概念学习指导

1、时间

2020 年 3 月 17 日周二（1-4 班 3-4 节，5-8 班 5-6 节）
2020 年 3 月 19 日周四（1-4 班 3-4 节，5-8 班 5-6 节）

2、内容

课前观看录播的视频
腾讯会议讲解本章重点和难点
QQ 群交流练习题（中间根据需要采用 QQ 视频或腾讯视频讲解）
投票方式答题（点名、检查上课人数）
问答
具体安排：

3 月 17 日 线上	(1) 数据结构的基本概念和术语； (2) 抽象数据类型的表示和实现；
3 月 19 日 线上	(3) 数据结构与程序设计 (4) 算法和算法分析。

3、本章知识要点

结构之美无处不在！一件事物只要存在，就一定会有自己的结构，数据亦是如此。一个好的算法的基础，是设计一个适合算法实现的数据结构。

本次课学习要点：

- 数据结构的基本概念
- 数据类型和抽象数据类型
- 算法设计与算法分析

3.1 数据结构的基本概念

要设计出一个结构良好而且效率较高的程序，必须研究数据的特性、数据间的相互关系及其对应的存储表示，并利用这些特性和关系设计出相应的算法和程序。数据结构知识无论是对研制系统软件还是对开发应用软件来说，都非常重要，是学习软件知识和提高软件设计水平的重要基础。

3.1.1 数据结构的研究内容

随着计算机应用领域的扩大和软、硬件的发展，非数值计算问题变得越来越重要。据统计，目前非数值计算问题的处理占用了 90%以上的机器时间。这类问题涉及的数据结构更为复杂，数据元素之间的相互关系一般无法用数学方程式来描述。因此，解决这类问题的关键不再是数学分析和计算方法，而是要设计出合适的数据结构。数据结构主要研究非数值计算问题，下面通过具体实例加以说明。

【例 1-1】学生信息检索系统。当系统需要查找某个学生的有关情况，或需要查询某个专业或年级的学生的有关情况时，只要建立了相关的数据结构，按照

某种算法编写了相关程序，就可以实现计算机自动检索。为此，可以在学生信息检索系统中建立一张按学号顺序排列的学生信息表和若干张分别按姓名、专业和年级顺序排列的索引表，如表 1-1～表 1-4 所示。由这 4 张表构成的文件便是学生信息检索系统的数学模型。

表 1-1　学生基本信息表

学号	姓名	性别	专业	年级
2011010001	崔志永	男	计算机科学与技术	2011 级
2011030005	李淑芳	女	软件工程	2011 级
2012040010	陆丽	女	数学与应用数学	2012 级
2012030012	张志强	男	软件工程	2012 级
2012010012	李淑芳	女	计算机科学与技术	2012 级
2013040001	王宝国	男	数学与应用数学	2013 级
2013010001	石国利	男	计算机科学与技术	2013 级
2013030001	刘文茜	女	软件工程	2013 级

表 1-2　姓名索引表

姓名	索引号	姓名	索引号	姓名	索引号
崔志永	1	张志强	4	石国利	7
李淑芳	2，5	王宝国	6	刘文茜	8
陆丽	3				

表 1-3　专业索引表

专业	索引号
计算机科学与技术	1，5，7
软件工程	2，4，8
数学与应用数学	3，6

表 1-4　年级检索表

年级	索引号	年级	索引号
2011 级	1，2	2013 级	6，7，8
2012 级	3，4，5		

诸如此类的还有电话号码查询问题、考试成绩查询问题和企业进销存管理系统等。在这类文档管理系统的数学模型中，计算机处理的对象之间通常存在着一种简单的线性关系，因此，这类数学模型可称为线性的数据结构。

【例 1-2】计算机系统组成结构，如图 1-1 所示。

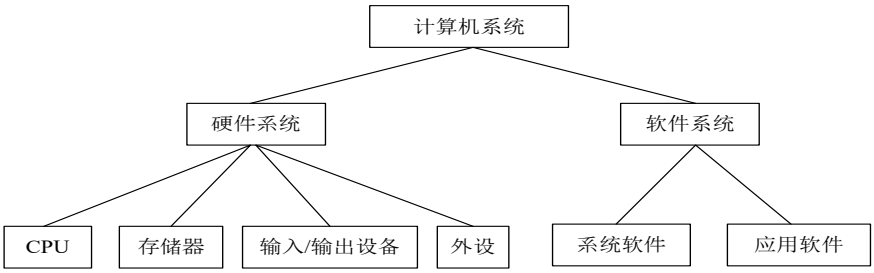


图 1-1　计算机系统组成结构图

计算机系统由硬件系统和软件系统组成，硬件系统由 CPU、存储器、输入/输出设备和外设组成，软件系统由系统软件和应用软件组成。如果把它们视为数据元素，则这些元素之间呈现的是一种层次关系，从上到下按层进行展开，可形成一棵倒立的“树”，最上层是“树根”，依层向下射出“结点”和“树叶”。

同样是树结构的还有某个单位的组织机构、国家行政区域规划、书籍目录等。在这类问题中，计算机处理的对象是树结构，元素之间是一对多的层次关系，这类数学模型被称为树的数据结构。

【例 1-3】最短路径问题。从城市 A 到城市 B 有多条线路可达，但每条线路的交通成本不同，那么，应怎样选择一条线路，使得从城市 A 出发到达城市 B 所花费的费用最低呢？可以将这类问题抽象为图的最短路径问题。如图 1-2 所示，图中的顶点代表城市，有向边代表两个城市之间的通路，边上的权值代表两个城市之间的交通费。求解 A 到 B 的最低费用，就是要在有向图从 A 点到 B 点的多条路径中，寻找到一条各边权值之和最小的路径，即求该图的最短路径。

同样是图结构的还有网络工程图、教学计划编排问题和比赛编排问题等。在这类问题中，元素之间是多对多的网状关系，这类数学模型被称为图的数据结构。

由以上 3 个例子可见，描述这类非数值计算问题的数学模型不再是数学方程，而是诸如表、树、图之类的数据结构。因此，可以说“数据结构”课程主要是在研究非数值计算的程序设计问题中所出现的计算机操作对象以及它们之间的关系和操作的学科。

数据结构在计算机科学中是一门综合性较强的专业基础课，是操作系统、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。数据结构涉及的知识面十分广，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。数据结构与其他课程间的关系如图 1-3 所示。

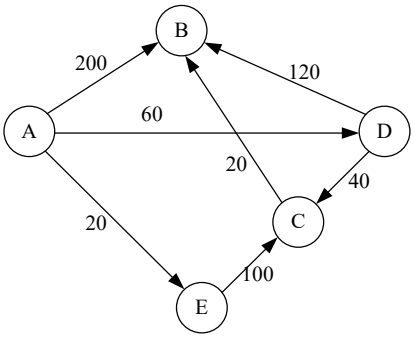


图 1-2　最短路径问题

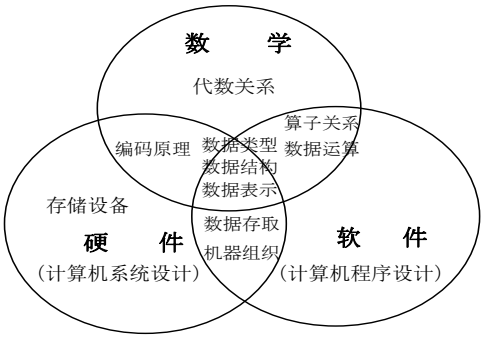


图 1-3　数据结构与其他课程的关系

学习数据结构的目的是为了了解计算机处理对象的特性，将实际问题中所涉及的处理对象在计算机中表示出来，并对它们进行处理。对于计算机专业的学生，不学习数据结构，几乎无法继续前行，因为几乎所有的程序和软件都要用到某种或某些数据结构。例如，在面向对象程序设计中，一个对象在严格意义上来说就是一个数据结构，而哪个程序不使用对象呢？可以这样说，不懂数据结构，就编不出什么像样的程序和软件。

此外，数据结构在软件工程和计算机学科的其他领域也发挥着非常重要甚至是极为关键的作用。例如，对大型数据库的管理、为互联网提供索引服务、云计算和云存储等都需要广泛使用数据结构。在软件工程领域，数据结构被单独提取出来，作为软件设计与实现过程的一个阶段。

3.1.2 基本概念和术语

在系统地学习数据结构知识之前，先来学习一下数据、数据元素、数据项等基本概念和术语的确切含义。

数据（Data）是信息的载体，能够被计算机识别、存储和加工处理。它是计算机程序加工的原料，应用程序处理各种各样的数据。计算机科学中，数据就是计算机加工处理的对象，它可以是数值数据，也可以是非数值数据。数值数据是一些整数、实数或复数，主要用于工程计算、科学计算和商务处理等；非数值数据包括字符、文字、图形、图像和语音等。

数据元素（Data Element）是数据的基本单位。在不同的条件下，数据元素又可称为元素、结点、顶点和记录等。例如，学生信息检索系统里学生信息表中的一个记录、计算机系统组成结构中状态树的一个状态以及最短路径问题中的一个顶点等，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组成。例如，学生信息检索系统中学生信息表的每一个数据元素都是一个学生记录，它包括学生的学号、姓名、性别、专业和年级数据项。这些数据项可以分为两种：一种叫做初等数据项，如学生的性别、年级等，这些数据项是数据处理时不能再分割的最小单位；另一种叫做组合数据项，如学生的成绩，它可以再划分为由多门不同课程成绩组成的更小项。

数据项（Data Item）是组成数据元素的有独立含义且不可分割的最小单位，如表 1-1 中的学号、姓名和年级等都是数据项。数据项有名和值之分，数据项名是一个数据项的标识，用变量定义，而数据项值是它的一个可能取值。例如，表 1-1 中的 2011010001 是数据项“学号”的一个取值。数据项具有一定的类型，依数据项的取值类型而定。

数据对象（Data Object）是相同性质的数据元素的集合，是数据集合的一个子集。在某个具体问题中，数据元素具有相同的性质（但元素值不一定相等），属于同一个数据对象，数据元素是数据元素类的一个实例。例如，在最短路径问题中，所有的顶点都是一个数据元素类，顶点 A 和顶点 B 各自代表一个城市，是该数据元素类中的两个实例，其数据元素的值分别为 A 和 B。

数据结构（Data Structure）是指互相之间存在着一种或多种特定关系的数据元素的集合。在计算机中，数据元素不是孤立的，它们之间存在着这样或那样的关系，这种数据元素之间的关系称为结构。一个数据结构包含两个要素：一个

是数据元素的集合；另一个是关系的集合。在形式上，数据结构通常可以采用一个二元组来表示。

数据结构的形式定义为一个二元组：

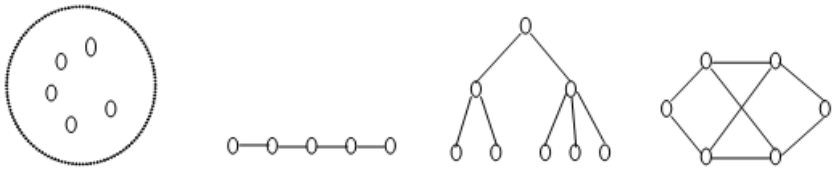
Data_Structure =(*D*,*R*)

其中，*D* 是数据元素的有限集，*R* 是 *D* 上关系的有限集。

数据结构包括数据的逻辑结构和数据的存储结构。

（1）逻辑结构

数据的逻辑结构可以看作是从具体问题抽象出来的数学模型，与数据的存储形式无关。根据数据元素间关系的不同特性，通常有下列 4 类基本的逻辑结构，如图 1-4 所示。



（a）集合结构　（b）线性结构　（c）树结构　（d）图结构或网状结构

图 1-4　4 类基本逻辑结构示意图

1）集合结构。结构中数据元素间的关系是“属于同一个集合”。集合是元素关系极为松散的一种结构。

2）线性结构。结构中数据元素之间存在着一对一的线性关系。

3）树结构。结构中数据元素之间存在着一对多的层次关系。

4）图结构或网状结构。结构中数据元素之间存在着多对多的任意关系。

【例 1-4】有一数据结构采用二元组描述为 D_S=(*D*,*R*)，其中：

D={ *a*,*b*,*c*,*d*,*e*,*f*,*g*}

R={<*e*,*d*>,<*d*,*c*>,<*c*,*a*>,<*a*,*b*>,<*b*,*f* >,<*f*,*g*>}

根据已知条件，对应的图形如图 1-5 所示。

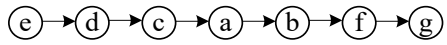


图 1-5　对应 D_S 的逻辑结构示意图

从例 1-4 可以看出，一个数据元素有且只有一个前驱（除第 1 个结点外），有且仅有一个后继（除最后一个结点外）。数据元素之间为一对一的关系，即线性关系。这种数据结构就是线性结构。

由于集合是数据元素之间关系极为松散的一种结构，因此也可用其他结构来表示。故数据的 4 类基本逻辑结构可概括如下：

- 线性结构——线性表、栈、队、串、数组、广义表
- 非线性结构——集合结构、树、图

（2）存储结构

研究数据结构的目的是为了在计算机中实现对它的操作，为此还需要研究如

何在计算机中表示一个数据结构。数据结构在计算机中的表示（又称为映像）称为数据的存储结构（或称物理结构）。它所研究的是数据结构在计算机中的实现方法，包括数据结构中元素的表示及元素间关系的表示。数据的存储结构可采用顺序存储或链式存储的方法。

1）顺序存储结构。是把逻辑上相邻的元素存储在物理位置相邻的存储单元中，由此得到的存储表示称为顺序存储结构。顺序存储结构是一种最基本的存储表示方法，通常借助于程序设计语言中的数组来实现。

2）链式存储结构。对逻辑上相邻的元素不要求其物理位置相邻，元素间的逻辑关系通过附设的指针字段来表示，由此得到的存储表示称为链式存储结构。链式存储结构通常借助于程序设计语言中的指针来实现。

如图 1-6 所示为复数 5.0-5.3i 的两种存储结构示意图。

除了通常采用的顺序存储方法和链式存储方法外，有时为了查找的方便，还会采用索引存储方法和散列存储方法。

（3）数据运算

讨论数据结构的目的是为了在计算机中实现操作运算。为了能有效地处理数据，提高数据运算的执行效率，应按一定的逻辑结构把数据组织起来，并选择适当的存储方法将数据存储到计算机内，然后对其进行运算。

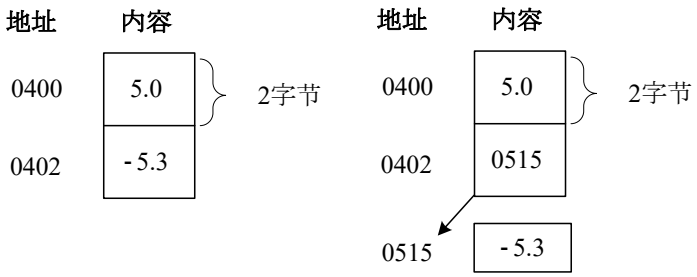


图 1-6　复数 5.0-5.3i 的两种存储结构示意图

数据的运算是定义在数据的逻辑结构之上的，每一种逻辑结构都有一个运算的集合，如插入、删除和修改等。这些运算实际上是在数据元素上施加一系列抽象的操作（只考虑这些操作要做什么，而无须考虑如何做），只有在确定了存储结构后，才能具体实现这些运算。

数据的运算主要有修改、插入、删除、查找和排序等。其中，查找运算是一个很重要的运算过程，修改、插入、删除和排序中都包含着查找运算。排序本身就是元素之间通过查找相互比较的过程，修改、插入和删除则要通过查找来确定其操作的位置。

3.1.3　数据结构课程的内容

数据结构与数学、计算机硬件和软件有十分密切的关系。数据结构技术也广泛应用于信息科学、系统工程、应用数学及各种工程技术领域。

数据结构课程集中讨论软件开发过程中的设计阶段，同时涉及编码和分析阶段的若干基本问题。此外，为了构造出好的数据结构及其实现，还需要考虑数据

结构及其实现的评价与选择。因此，数据结构的内容可归纳为 3 个部分：逻辑结构、存储结构和数据运算。简而言之，按某种逻辑关系组织起来的一批数据，按一定的存储方式将其存入计算机的存储器中，并在这些数据上定义一个运算集，是数据结构课程的基本内容，如表 1-5 所示。

表 1-5　数据结构课程的基本内容

内容 / 层次	数 据 表 示	数 据 处 理
抽象	逻辑结构	基本运算
实现	存储结构	算法
评价	不同数据结构的比较及算法分析	

数据结构主要研究怎样合理地组织数据，建立合适的结构，提高执行程序所用的时空效率。

数据结构的核心技术是分解与抽象。通过对问题的抽象，舍弃数据元素的具体内容，从而得到逻辑结构；同样，通过分解，将数据处理划分成各种功能实现，再通过抽象舍弃实现细节，就得到数据运算的定义。由此可将许多具体问题转换为数据结构，这是一个从具体（即具体问题）到抽象（即数据结构）的过程。然后，通过增加对实现细节的考虑，进一步得到存储结构和实现运算，从而完成设计任务，这是一个从抽象（即数据结构）到具体（即具体实现）的过程。熟练地掌握这两个过程，是数据结构课程在专业技能培养方面的基本目标。

数据结构课程不仅讲授数据信息在计算机中的组织和表示方法，同时也重在培养高效解决复杂问题的能力。不同的数据结构适用于不同的应用，例如，B 树就特别适用于数据库和文件系统，而哈希表则常常在编译器里面使用等。

3.2　数据类型和抽象数据类型

运用抽象数据类型来描述数据结构，则在设计一个软件系统时，不必首先考虑其中包含的数据对象以及操作在不同处理器中的表示和实现细节，而可以在构成软件系统的每个相对独立的模块上定义一组数据和相应的操作（把这些数据的表示和操作细节留在模块内部解决），在更高的层次上进行软件的分析和设计，从而提高软件的整体性能和利用率。

数据结构是一种抽象，它将数据的个体属性去除，只考虑数据元素之间的关系。

通过步步抽象，可不断地突出“做什么”，而将“怎么做”隐藏起来，即将一切用户不必了解的细节封装起来，从而简化了问题。所以，抽象是程序设计中最基本的思想方法。

3.2.1　数据类型

数据类型（Data Type）是一个值的集合和定义在这个值集上的一组操作的总称。数据类型中定义了两个集合，即该类型的取值范围及该类型中可允许使用

的一组运算。

数据类型是和数据结构密切相关的一个概念。在用高级语言编写的程序中，每个变量、常量或表达式都有一个它所属的确定的数据类型。数据类型显式或隐含地规定了在程序执行期间变量或表达式所有可能的取值范围，以及在这些值上允许进行的操作。

在高级程序设计语言中，数据类型可分为两类：一类是原子类型，另一类则是结构类型。原子类型的值是不可分解的，例如，C 语言中的整型、字符型、浮点型和双精度型等基本类型，分别用关键字 int、char、float 和 double 表示。而结构类型的值是由若干成分按某种结构组成的，因此是可分解的，并且它的成分可以是原子的，也可以是结构的。例如，数组的值由若干分量组成，每个分量可以是整数，也可以是数组等。在某种意义上，数据结构可以看成是一种数据类型，而数据类型则可以看成是由一种数据结构和定义在其上的一组操作所组成的。

3.2.2 抽象数据类型

抽象就是抽取出实际问题的本质，将无限多的关系种类里面的非关键属性去除，只取其中的共性来设计数据结构。例如，对于数据元素 A、B、C 而言，它们之间的关系可以是 A 在 B 的前面，B 在 C 的前面；或者 C 在 B 的前面，B 在 A 的前面。显然，对于个体的数据元素而言，这是两种不同的结构。但抛开个体数据元素就会发现，这两种结构实际上是一种数据结构类型——线性结构。

抽象数据类型（Abstract Data Type，ADT）是指一个数学模型及定义在该模型上的一组操作。抽象数据类型的定义取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，就不会影响到其外部的使用。

抽象数据类型和数据类型实质上是一个概念。例如，各种计算机都拥有的整数类型就是一个抽象数据类型，尽管在不同处理器上的实现方法可能不同，但由于其定义的数学特性相同，在用户看来都是相同的。因此，“抽象”的意义在于数据类型的数学抽象特性。

但在另一方面，抽象数据类型的范畴更广，它不再局限于前述各处理器中已定义并实现的数据类型，还包括用户在设计软件系统时自己定义的数据类型。为了提高软件的重用性，在近代程序设计方法学中，要求在构成软件系统的每个相对独立的模块上，定义一组数据和应用于这些数据上的一组操作，并在模块的内部给出这些数据的表示及其操作的细节，而在模块的外部使用的只是抽象的数据及抽象的操作。这也就是面向对象的程序设计方法。

抽象数据类型的定义可以由一种数据结构和定义在其上的一组操作所组成，而数据结构又包括数据元素及元素间的关系，因此，抽象数据类型一般可以由数据对象、数据对象上关系的集合以及对数据对象的基本操作的集合来定义。

抽象数据类型的特征是使用与实现相分离，实行封装和信息隐蔽。也就是说，在设计抽象数据类型时，要把类型的定义与其实现分离开来。

和数据结构的形式定义相对应，抽象数据类型可用以下三元组表示：

$$\mathrm{ADT}=(D,S,P)$$

其中，D 是数据元素的有限集；S 是 D 上的关系集；P 是对 D 的基本操作集。

抽象数据类型的定义格式如下：

ADT 抽象数据类型名{

数据对象：<数据对象的定义>

结构关系：<结构关系的定义>

基本操作：<基本操作的定义>

}ADT 抽象数据类型名

【例 1-5】给出线性表的抽象数据类型的定义。

ADT List {

数据元素：所有 a_i 属于同一数据对象，i=1，2，…，n，n≥0；

结构关系：所有数据元素 a_i（i=1，2，…，n-1）存在次序关系$<a_i,a_{i+1}>$，a_1 无前趋，a_n 无后继；

基本操作：设 L 为 List，则有

InitList(L)：初始化线性表；

ListLength(L)：求线性表的表长；

GetData(L,i)：取线性表的第 i 个元素；

InsList(L,i,b)：在线性表的第 i 个位置插入元素 b；

DelList(L,i)：删除线性表的第 i 个数据元素；

}ADT List;

注意，所有上述操作一般都以函数形式实现，是非标准的，需要用户自己去实现。ADT 操作的实现依赖于数据结构，我们需要为不同类型的数据结构实现 ATD 操作。

3.3 算法和算法分析

著名的计算机科学家 N.Wirth 教授给出了一个对计算机科学的发展影响深远的公式：算法+数据结构=程序，足以说明算法和数据结构关系紧密，是程序设计的两大要素，二者相辅相成，缺一不可。

在进行算法设计时，先要确定相应的数据结构；而在讨论某一种数据结构时，也必　　然要涉及相应的算法。下面就从算法特性、算法描述和算法性能分析 3 个方面对算法进行介绍。

3.3.1 算法特性

算法（Algorithm）是为解决特定问题而规定的一系列操作。

一个算法应该具有下列 5 个重要特性：

（1）有穷性。一个算法必须在执行有穷步之后结束，即必须在有限时间内完成。

（2）确定性。算法的每一步必须有确切的定义，无二义性。算法的执行对应着的相同的输入仅有唯一路径。

（3）可行性。算法中的每一步都可以通过已经实现的基本运算执行有限次得以实现。

（4）输入。一个算法具有零个或多个输入，这些输入取自特定的数据对象集合。

（5）输出。一个算法具有一个或多个输出，这些输出同输入之间存在某种特定的关系。

算法的含义与程序十分相似，但又有区别。一个程序不一定满足有穷性，例如操作系统，只要整个系统不遭破坏，它将永远不会停止，即使没有作业需要处理，它仍处于动态等待中。因此，操作系统不是一个算法。另一方面，程序中的指令必须是机器可执行的，而算法中的指令则无此限制。算法代表了对问题的解，而程序则是算法在计算机上的特定实现。一个算法若用程序设计语言来描述，则它就是一个程序。

【例 1-6 】不符合有穷性。

```
void test(void){
    int n=8;
    while(n%8==0)
        n+=8;
    printf("%d\n",n);
    }
```

【例 1-7】无输出的算法没有任何意义。

```
GetSum(int num){
    int sum=0;
    for(i=1;i<=num;i++)
    sum+=i;
}
```

当用算法解决某一特定类型问题时，可以选择不同的数据结构，而选择的恰当与否会直接影响到算法的效率。反之，一种数据结构的优劣可由不同算法的执行效果来体现。

一个算法的优劣应从以下几个方面来评价：

（1）正确性。在合理的数据输入下，能够在有限的运行时间内得到正确的结果。

（2）可读性。一个算法应当思路清晰、层次分明、简单明了和易读易懂。可读性强的算法有助于人们对算法的理解，而难懂的算法易于隐藏错误，且难以调试和修改。

（3）健壮性。当输入不合法数据时，应能做出正确反应或适当处理，不致引起严重后果。

（4）高效性。高效性包括时间和空间两个方面。时间高效是指算法设计合理，执行效率高，可以用时间复杂度来度量；空间高效是指算法占用的存储容量合理，可以用空间复杂度来度量。

3.3.2 算法描述

算法可以使用各种不同的方法来描述。

最简单的方法是使用自然语言。用自然语言来描述算法的优点是简单，且便

于人们阅读算法，缺点是不够严谨。

可以使用程序流程图、N-S 图等算法描述工具，其特点是描述过程简洁明了。

用以上两种方法描述的算法不能够直接在计算机上执行。若要将它转换成可执行的程序，还有一个编程的问题。

可以直接使用某种程序设计语言来描述算法，不过直接使用程序设计语言并不容易，而且不太直观，常常需要借助于注释才能使人看明白。

为了解决理解与执行之间的矛盾，常常使用一种称为伪码语言的描述方法来进行算法描述。伪码语言介于程序设计语言和自然语言之间，忽略程序设计语言中一些严格的语法规则与描述细节，因此，它比程序设计语言更容易描述和被人理解，且比自然语言更接近程序设计语言。它虽然不能直接执行，但可以很容易地被转换成程序设计语言。

【例 1-8】设计一个算法，打印出所有的“水仙花数”。“水仙花数”是指一个 3 位数，其各位数字的立方和等于该数本身。

算法设计如下：

```
for(n=100～999) {
    每个数分解出个位，十位，百位;
        令 k=个位数*100+十位数*10+百位数;
         p=(个位数)3+(十位数)3+(百位数) 3;
    if(k==p)
        printf(n);
   }
```

3.3.3 算法性能分析

通过算法分析，可判断一个算法实际是否可行，并可在同一问题存在多个算法时帮助选择性能较优的算法。评价一个算法的性能，主要从算法执行时间与占用存储空间大小两方面考虑，即从算法执行所需的时间和存储空间来判断一个算法的优劣。

当然，设计者希望选用一个占用存储空间小、运行时间短并且其他性能也好的算法，但在现实中很难做到十全十美，因为上述要求有时会相互抵触。要节约算法的执行时间，往往要以牺牲更多的空间为代价；而为了节省空间，又可能要以牺牲更多的时间为代价。因此，只能根据具体情况有所侧重。若该程序使用次数较少，则应力求算法简明易懂，易于转换为上机的程序。

当一个算法被转换成程序并在计算机上执行时，其运行所需要的时间主要取决于下列因素：

- 硬件的速度；
- 书写程序的语言；
- 编译程序所生成的目标代码质量；码优化较好的编译程序，其生成的程序质量较高。
- 问题的规模；例如，求 100 以内的素数与求 1000 以内的素数，其执行时间必然是不同的。

（1）算法耗费的时间和语句频度

一个算法的执行时间是指算法中所有语句执行时间的总和。每条语句的执行时间等于该条语句的执行次数乘以执行一次所耗用的实际时间。

语句频度是指该语句在一个算法中被重复执行的次数。一个算法的时间耗费就是该算法中所有语句频度之和。

【例 1-9】求两个 n 阶矩阵的乘积算法。

```
for(i=1;i<=n;i++)                                    //频度为 n+1
   for(j=1;j<=n;j++)                                  //频度为 n*(n+1)
       {c[i][j]=0;                                    //频度为 n^2
         for(k=1;k<=n;k++)                                 //频度为 n^2*(n+1)
             c[i][j]= c[i][j]+a[i][k]*b[k][j]     //频度为 n^3
```

该算法中，所有语句的频度之和，即算法的执行时间，用 $T(n)$表示，则

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

（2）问题规模和算法的时间复杂度

为便于比较同一问题的不同算法，通常以算法中基本操作重复执行的频度作为度量标准。原操作是指从算法中选取一种对所研究问题是基本运算的操作，用问题规模增加的函数来表征，以此作为时间量度。

算法中，语句的总执行次数 $f(n)$是问题规模 n 的函数，根据 $f(n)$随 n 的变化情况，可确定 $T(n)$的数量级（Order of Magnitude）。这里用 O 来表示数量级，给出算法的时间复杂度概念。算法的时间复杂度 $T(n)$是该算法的时间量度，记作

$$T(n)=O(f(n))$$

它表示随问题规模 n 的增大，算法执行时间的增长率和 $f(n)$的增长率相同，称作算法的渐近时间复杂度，简称时间复杂度。

数学符号 O 的严格数学定义为：若 $T(n)$和 $f(n)$是定义在正整数集合上的两个函数，则 $T(n)=O(f(n))$表示：存在正的常数 C 和 n_o，使得当 $n\geq n_o$ 时，满足 $0\leq T(n)\leq Cf(n)$）。

【例 1-10】赋值语句。

```
++x;
s=0;
```

语句频度为 1，时间复杂度为 O(1)。

【例 1-11】简单循环。

```
for(i =1;i<=n;++i){
    ++X;S+=X;
}
```

语句频度为 n，时间复杂度为 O(n)。

【例 1-12】双重循环。

```
for(j =1;j<=n;++j)
      for(k=1;k<=n;++k)
      {++x; s+=x;}
```

语句频度为 $n*n$，时间复杂度为 $O(n^2)$。

【例 1-13】有一个双重循环。

```
for ( j=1;j<=n;++j)
        for(k=1;k<=j;++k)
        {++x;s+=x;}
```

语句频度近似于 $n^2/2$，所以时间复杂度仍为 $O(n^2)$。

（3）常用算法的时间复杂度

数据结构中常用的时间复杂度频率计数有以下几种：一个没有循环的算法，其基本运算次数与问题规模 n 无关，记作 $O(1)$，也称作常数阶；一个只有一重循环的算法，其基本运算次数与问题规模 n 呈线性增长关系，记作 $O(n)$，也称作线性阶；其余常用的还有平方阶 $O(n^2)$、立方阶 $O(n^3)$、对数阶 $O(\log_2 n)$和指数阶 $O(2^n)$等。不同数量级对应的时间复杂度值存在着如下关系：

$$O(1) < O(\log_2 n) < O(n) < O(n*\log_2 n) < O(n^2) < O(n^3) < O(2^n) < O(n!)$$

不同数量级对应的时间复杂度曲线如图 1-7 所示。一般情况下，随着 n 增大，$T(n)$增长较慢的算法为最优的算法。显然，时间复杂度为指数阶 $O(2^n)$的算法效率极低，当 n 值稍大时，程序将无法应用。

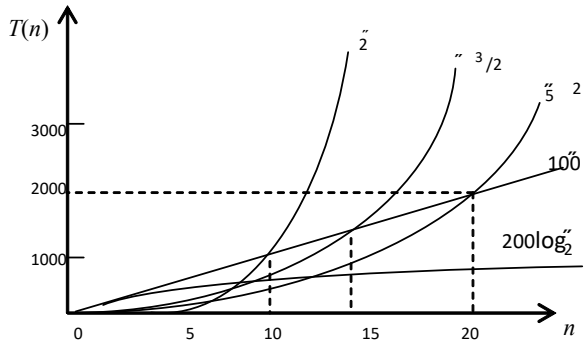


图 1-7　常见数量级的渐近增长趋势

（4）算法的空间复杂度

算法的存储空间需求类似于算法的时间复杂度，采用渐近空间复杂度（Space Complexity）作为量度，简称空间复杂度。它也是问题规模 n 的函数，记作 $S(n)=O(f(n))$，表示的是该算法在运行过程中所耗费的辅助存储空间。如果一个算法所耗费的存储空间与问题规模 n 无关，记作 $S(n)=O(1)$。

一个算法所耗费的存储空间包括 3 类：算法本身所占用的存储空间、算法的输入/输出所占用的存储空间和算法在运行过程中临时占用的辅助存储空间。

算法输入/输出所占用的存储空间，由算法解决的问题规模决定，不随算法的改变而改变。算法本身所占用的存储空间与实现算法的程序代码长度有关，代码越长，占用的存储空间越大。算法在运行过程中临时占用的辅助存储空间随算法的不同而不同，有的不随问题规模的大小改变，而有的与解决问题的规模 *n* 有关，它随 *n* 的增大而增大。

【例 1-14】将一维数组 a 中的 *n* 个数据逆序存放到原数组中，给出实现该问题的两种算法。

算法 1：

```
for(i=0;i<n;i++)
   b[i]=a[n-i-1];
for(i=0;i<n;i++)
   a[i]=b[i];
```

算法 2：

```
for(i=0;i<n/2;i++)
   {
    t=a[i];
    a[i]=a[n-i-1];
    a[n-i-1]=t;
   }
```

算法 1 的空间复杂度为 O(*n*)，需要一个大小为 *n* 的辅助数组 b。

算法 2 的空间复杂度为 O(1)，仅需要一个变量 *t*，与问题规模没有关系。

3.4 本章小结

本章介绍了数据结构的基本概念和术语，以及算法和算法时间复杂度的分析方法。主要内容如下：

（1）掌握数据结构的基本概念

数据结构包括数据的逻辑结构、存储结构和运算集合 3 个部分。

- 数据的逻辑结构
  - 线性结构（线性表、栈、队列、字符串、数组和广义表）
  - 非线性结构（树、图）
- 数据的存储结构
  - 顺序结构
  - 链式结构
- 数据的运算集合

（2）逻辑结构与存储结构的区别

逻辑结构定义了数据元素之间的逻辑关系。存储结构是逻辑结构在计算机中的实现。一种逻辑结构可采用不同的存储方式存放在计算机中，但无论哪种存储方式，都必须能反映出要求的逻辑关系。

（3）抽象数据类型

抽象数据类型是指由用户定义的表示应用问题的数学模型，以及定义在这个模型上的一组操作的总称。具体包括 3 部分：数据对象、数据对象上关系的集合以及对数据对象的基本操作的集合。

（4）算法和算法分析

理解算法的定义、算法的特性、算法的时间代价和算法的空间代价。

4、习题

（1）解释下列术语：数据、数据元素、数据对象、数据结构、存储结构、线性结构、算法和抽象数据类型。

（2）试举一个数据结构的例子，叙述其逻辑结构、存储结构及运算 3 方面的内容。

（3）选择题

1）在数据结构中，从逻辑上可以把数据结构分成（　　）。

A．动态结构和静态结构　　B．紧凑结构和非紧凑结构

C．线性结构和非线性结构　　D．内部结构和外部结构

2）与数据元素本身的形式、内容、相对位置和个数无关的是数据的（　　）。

A．存储结构　　B．存储实现

C．逻辑结构　　D．运算实现

3）通常要求同一逻辑结构中的所有数据元素具有相同的特性，这意味着（　　）。

A．数据具有同一特点

B．不仅数据元素所包含的数据项的个数要相同，而且对应数据项的类型要一致

C．每个数据元素都一样

D．数据元素所包含的数据项的个数要相等

4）以下说法中，正确的是（　　）。

A．数据元素是数据的最小单位

B．数据项是数据的基本单位

C．数据结构是带有结构的各数据项的集合

D．一些表面上很不相同的数据可以有相同的逻辑结构

5）以下数据结构中，（　　）是非线性数据结构。

A．树　　B．字符串　　C．队　　D．栈

6）算法的时间复杂度取决于（　　）。

A．问题的规模　　B．待处理数据的初态

C．计算机的配置　　D．A 和 B

7）算法分析的目的是（①　　），算法分析的两个主要方面是②（　　）。

① A. 找出数据结构的合理性　　B. 研究算法中的输入和输出的关系

C. 分析算法的效率以求改进　　D. 分析算法的易懂性和文档性

② A. 空间复杂性和时间复杂性　　B. 正确性和简明性

C. 可读性和文档性　　D. 数据复杂性和程序复杂性

8）计算机算法指的是（①　），它必具备输入、输出和（②　）等五个特性。

① A. 计算方法　　B. 排序方法

C. 解决问题的有限运算序列　　D. 调度方法

② A. 可行性、可移植性和可扩充性　　B. 可行性、确定性和有穷性

C. 确定性、有穷性和稳定性　　D. 易读性、稳定性和安全性

9）以下任何两个结点之间都没有逻辑关系的是（　　）。

A. 图形结构　B. 线性结构　C. 树形结构　D. 集合

10）每一个存储结点只含有一个数据元素，存储结点存放在连续的存储空间，另外有一组指明结点存储位置的表，该存储方式是（　　）存储方式。

A.顺序　　B. 链式　　C. 索引　　D. 散列

（4）填空题

1）数据结构是一门研究非数值计算的程序设计问题中计算机的___以及它们之间的___和运算等的学科。

2）若数据结构的形式被定义为(*D*,*R*)，其中，*D* 是___的有限集合，*R* 是 *D* 上的____有限集合。

3）数据结构包括数据的____、数据的___和数据的____ 三个方面的内容。

4）线性结构中，元素之间存在___关系；树形结构中，元素之间存在_____关系；图形结构中，元素之间存在___关系。

5）一个算法的效率可分为___效率和___效率。

6）在线性结构中，第一个结点___前驱结点，其余每个结点有且只有___个前驱结点；最后一个结点___后续结点，其余每个结点有且只有___个后续结点。

7）在树形结构中，树根结点没有___结点，其余每个结点有且只有___个直接前驱结点，叶子结点没有___结点，其余每个结点的直接后续结点可以___。

8）在图形结构中，每个结点的前驱结点数和后续结点数可以___。

9）分析下面算法（程序段）给出最大语句频度___，该算法的时间复杂度是___。

```
i=s=0;
while (s<n)
{   i++;
    s+=i;    }
```

10）分析下面算法（程序段）给出最大语句频度___，该算法的时间复杂度是___。

```
i=1;
while (i<=n)
        i=i*2;
```

（5）试分析下面各算法的时间复杂度。

1）

```
x=90; y=100;
while(y>0)
  if(x>100)
     {x=x-10;y--;}
  else x++;
```

2）

```
for (i=0; i<n; i++)
   for (j=0; j<m; j++)
        a[i][j]=0;
```

3）for(int i=1;i<=n,i++)
    for(int j=1;j<=i;j++)
        s++;

4）i=1;
  while(i<=n)
    i=i*2;

5）i=0,s1=0,s2=0;
  while(i++<n){
    if(i%2)s1+=i;
    else s2+=i;
  }

6）x=n;   //n>1
  y=0;
  while(x≥(y+1)* (y+1))
        y++;