全国高职高专计算机系列精品教材

数据结构导论

主　编/蔡厚新　肖守柏

中国人民大学出版社

全国高职高专计算机系列精品教材

数据结构导论

主　编　蔡厚新　肖守柏
副主编　吴金舟　熊　蕾　齐兴敏

中国人民大学出版社
·北京·

图书在版编目（CIP）数据

数据结构导论/蔡厚新主编．
北京：中国人民大学出版社，2010
（全国高职高专计算机系列精品教材）
ISBN 978-7-300-12430-8

Ⅰ．①数…
Ⅱ．①蔡…
Ⅲ．①数据结构-高等学校：技术学校-教材
Ⅳ．①TP311.12

中国版本图书馆 CIP 数据核字（2010）第 133440 号

全国高职高专计算机系列精品教材
数据结构导论
主　编　蔡厚新　肖守柏
副主编　吴金舟　熊　蕾　齐兴敏

出版发行　中国人民大学出版社
社　　址　北京中关村大街 31 号　　**邮政编码**　100080
电　　话　010－62511242（总编室）　　010－62511398（质管部）
010－82501766（邮购部）　　010－62514148（门市部）
010－62515195（发行公司）　　010－62515275（盗版举报）
网　　址　http://www.crup.com.cn
http://www.ttrnet.com(人大教研网)
经　　销　新华书店
印　　刷　北京东方圣雅印刷有限公司
规　　格　185 mm×260 mm　16 开本　　**版　　次**　2010 年 8 月第 1 版
印　　张　23　　**印　　次**　2010 年 8 月第 1 次印刷
字　　数　560 000　　**定　　价**　39.80 元

前　言

计算机科学技术以惊人的速度迅猛发展，它的应用范围已渗入到社会和生活的各个领域。相应地，数据处理的对象也从简单的数值发展到字符、表格和图形等带有结构的数据。在这里要解决的关键问题是：针对每一种新的应用领域的处理对象，如何选择合适的数据表示（结构），如何有效地组织数据、处理数据。数据结构就是研究数据以及数据之间关系的一门学科，主要研究数据之间的逻辑结构及其基本操作在计算机中的表示和实现。数据结构课程不仅是计算机专业重要的专业基础课，也是从事计算机软件开发所必备的专业知识。本教材主要面向高职高专院校或应用性本科的计算机类专业的学生，培养技术应用性人才。内容的构造力求体现“以应用为主体”，强调理论知识的理解和运用，实现教学以实践体系为主及以技术应用能力培养为主的培养目标。

案例教学是计算机语言教学最有效的方法之一，好的案例对学生理解知识、掌握如何应用知识都十分重要。本书围绕教学内容组织案例，对学生的知识和能力训练具有很强的针对性。全书共十二章，大体上可看成为由四个部分组成，基本的线性结构及有关的典型应用是第一部分（第二章到第六章）；具有广泛应用价值的树形结构在第七、八章讲述，这两部分占据了本书的主要篇幅；第九章及第十章介绍复杂数据结构，如图、稀疏矩阵及广义表等；有关外存储器中的数据结构和文件组织放在第四部分。

数据结构是实践性很强的课程，本书注重理论与实践相结合，为此我们编写了与本书配套的指导书《数据结构导论学习指导》。书中每章都给出了精心挑选，难易搭配，给出了大量的不同层次、不同难度的思考题，通过习题与实训，使学生掌握所学知识，并能灵活运用所学知识解决实际问题。书中的所有程序都在 Turbo C 2.0 环境下调试通过。

本教材讲课时数为 50 学时左右。上机实训学时数为 20 学时以上。教师可根据学时数、专业和学生的实际情况选讲应用举例中一些较难的例子。

本书可作为高等院校高职高专计算机软件专业的教材或参考书，也可供广大从事计算机软件工作的科技人员自学参考。

本书由蔡厚新、肖守柏担任主编，吴金舟、熊蕾、齐兴敏担任副主编。本书的第 1 章由肖守柏编写，第 2、4 章由邓田编写，6、7、10 章由吴金舟编写，第 3 章由蔡厚新编写，第 9、11 章由熊蕾编写，第 5、8、12 章由齐兴敏编写。蔡厚新教授统编全书。在本书编写过程中，得到南昌大学信息工程学院计算机科学与技术系姚立文教授的大力帮助，在此表示深深的谢意。

由于编者水平有限，加上时间仓促，书中错误和不妥之处在所难免，敬请广大同行和读者批评指正。

编　者

2010 年 6 月

目　录

第1章 绪　　论

内容提要及教学目标

本章首先介绍了数据结构中的几个基本概念和术语，然后以一个简单的C语言程序入手，复习了C语言的基本内容，进而对程序的算法特性、评价标准和时间复杂度分别进行介绍。通过本章学习，读者应该掌握以下内容：

- 了解数据结构中常用的基本概念和术语，掌握基本概念。
- 掌握抽象数据类型的定义、表示和实现方法。
- 熟悉C语言的书写规范。
- 理解算法5个要素的确切含义。
- 掌握计算语句频度和估算算法时间复杂度的方法。

本章重点及难点

理解数据结构的逻辑结构、存储结构和数据运算3个方面的概念及相互之间的关系；熟悉算法时间复杂度的分析。

1.1 数据结构

程序设计是计算机学科各个领域的基础。在计算机发展的早期，程序设计所处理的数据都是整型、实型等简单数据，绝大多数的应用软件都是用于数值计算的。随着信息技术的发展，计算机逐渐进入到金融、商业、管理、通信以及制造业等各个行业，广泛地应用于数据处理和过程控制，计算机加工处理的对象也由纯粹的数值型数据发展到字符、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了一些新的问题，数据结构的概念就是在这种背景下产生的。

1.1.1 学习数据结构的必要性

众所周知，数据结构不仅仅是计算机专业教学计划中的核心课程之一，而且是其他非计算机专业的主要选修课之一。数据结构的研究不仅涉及计算机硬件的研究范围，而且

与计算机软件的研究也有着密切的联系，无论在编程还是在操作系统，它都涉及数据元素在存储器中的分配问题，以及如何组织数据，以便查找和存取数据元素更为方便。学习数据结构这门课程的目的有3个。第一是讲授常用的数据结构，这些数据结构形成了程序员基本数据结构工具箱（toolkit）。对于许多常见的问题，工具箱里的数据结构是理想的选择。就像.NET Framework中Windows应用程序开发中的工具箱那样，程序员可以直接拿来或经过少许的修改就可以使用，非常方便。第二是讲授常用的算法，算法和数据结构一样，是人们在长期实践过程中的总结，程序员可以直接拿来或经过少许的修改就可以使用，我们可以通过算法训练来提高程序设计水平。第三是通过程序设计的技能训练来促进程序员综合能力的提高。

1.1.2 数据结构的基本概念和术语

1. 数据结构实例

为了使大家对数据结构有一个感性的认识，我们先举以下几个例子来说明什么是数据结构。

[例1.1] 高校教师信息管理，如表1.1所示。

表1.1　　教师信息表

工　号	姓　名	性　别	授课课程	课程编号	住　址	电　话
10001	匡青青	女	数据结构导论	2142	北京西路411号	6262111
10002	肖子皓	男	网络操作系统	2335	南京东路999号	5966201
10003	谭辉明	男	C语言程序设计	0342	西安路688号	8138793
10004	彭明明	男	软件工程	2333	上海路168号	8138927
10005	章小花	女	网络技术	2141	八一大道21号	8174110
……	……	……	……	……	……	……

在表1.1中，一行代表一位教师的信息，每位教师的信息由工号、姓名、性别、授课课程、课程编号、住址、电话等组成，一列为一个特征，整个二维表按教师的工号顺序排列，每个教师的信息依据工号的大小存在着前后关系，即工号较小者在前，工号较大者在后，形成线性关系，这是一种典型的数据结构，这种关系可称为线性数据结构。有了模型以后，就可以围绕该模型设计算法，即实现教工信息的添加、修改、删除、检索等操作。

[例1.2] 某高校专业设置情况，如图1.1所示。

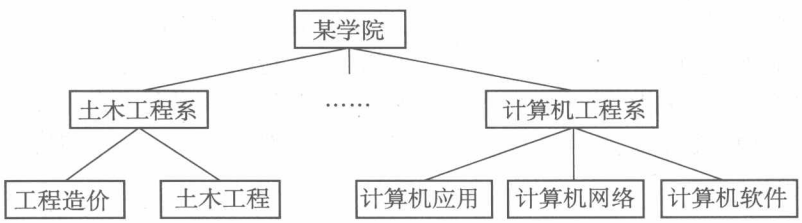

图1.1　高校专业设置

在图1.1中可以把“某学院”看成是树根，把下设的若干个系看成是它的树枝中间结点，把每个系的若干专业方向看成是树叶，这就形成一个树形结构。树形结构通常用来表示

结点的分层组织，结点之间是一对多的关系，它也是一个典型的结构，称为树形结构。

［例 1.3］ 7个城市间建立通信网络，用顶点表示城市，边上的权值表示两个城市之间建立通信线路所需花费的代价，如图 1.2 所示。

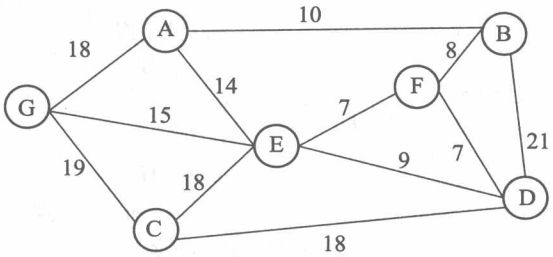

图 1.2 城市间通信网络

在此结构中，数据之间呈现多对多的非线性关系，这也是常用的一种数据结构，我们将它称为图形结构。

2. 基本概念和术语

上述 3 个例子都是数据结构的具体实例，那么，数据结构的定义是什么呢？下面介绍数据结构的基本概念和术语。

（1）数据。

数据（Data）是对信息的一种符号表示，是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。通俗地说，凡是能被计算机识别、存储和加工处理的符号，如字符、图形、图像、声音、视频信号等一切信息都可以称为数据。数据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，其处理的数据是整数和实数；一个文字处理器（如 Word）处理的数据是字符串。

（2）数据元素。

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。数据元素有时也被称为元素、结点、顶点、记录等。一个数据元素可由若干个数据项组成。数据项是数据的不可分割的最小单位。例如，在表 1.1 中，一行是作为一个数据元素来看待的，工号、姓名、性别、授课课程、课程编号、住址、电话等被称为数据项（Data Item），数据项有时也称为字段（Field）或域（Domain）。

（3）数据对象。

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，整数数据对象的集合可表示为 N＝{0，±1，±2，…}，字母字符数据对象的集合可表示为 C＝{‘a’，‘b’，‘c’，… }。

3. 数据结构

数据结构（Data Structure）是指数据元素之间存在着的一种或多种特定关系的集合，这是对数据结构一种简单的定义。具体来说，数据结构是按某种逻辑关系组织起来的一批数据（或称带结构的数据元素的集合），它们应用计算机语言并按一定的表示方式存储在计算机的存储器中。数据结构包括数据的逻辑结构、数据的存储结构、数据的运算三个方面的内容，如图 1.3 所示。

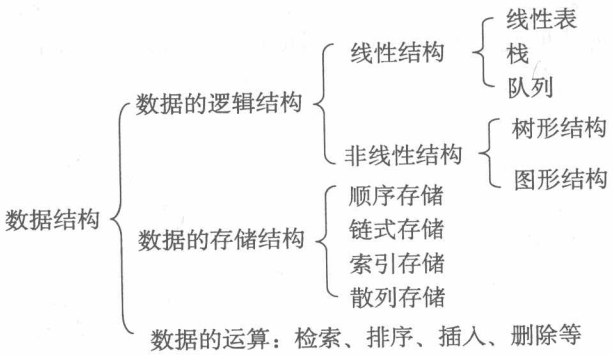

图 1.3　数据结构的内容

（1）数据的逻辑结构。

逻辑关系是指数据元素之间的关联方式。数据元素之间逻辑关系的整体称为逻辑结构。数据的逻辑结构就是数据的组织形式。从逻辑上划分，数据结构分为线性结构和非线性结构。

①线性结构。数据元素之间为一对一的线性关系，第一个元素无直接前趋，最后一个元素无直接后继，其余元素只有唯一的一个前趋和唯一的一个后继。

②非线性结构。数据元素之间为一对多或多对多的非线性关系，每个数据元素有多个直接前趋或者多个直接后继。

然而又可根据数据元素之间关系的不同特性，将数据结构划分为 4 种类型，如图 1.4 所示。

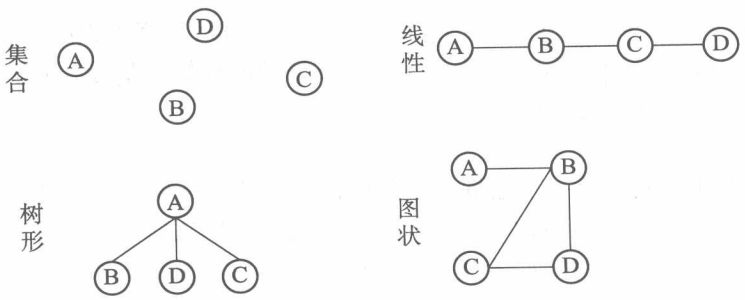

图 1.4　4 种基本数据结构

①集合（Set）：结构中的数据元素除了存在“同属于一个集合”的关系外，不存在任何其他关系。

②线性结构（Linear Structure）：结构中的数据元素之间存在一对一的关系，即有且仅有一个开始和一个终端结点，并且所有结点最多只有一个直接前趋和一个后继。例如，线性表、栈、队列都属于线性结构。

③树形结构（Tree Structure）：结构中数据元素之间存在一对多的关系。

④图状结构（Graphic Structure）：结构中数据元素间存在多对多的关系。

由于集合中元素的关系极为松散，可用其他数据结构来表示，数据结构的形式化定义为：

数据结构（Data Structure）简记为 DS，是一个二元组，DS=（D，R）。其中，D 是数据元素的有限集合，R 是数据元素之间关系的有限集合。

（2）数据的存储结构。

数据的逻辑结构是不能描述计算机是如何操作数据的。研究数据结构的目的是为了在计算机中实现对它的操作，因此必须研究如何在计算机中表示数据结构，即数据的存储结构，又称物理结构，它包括数据元素的表示和关系的表示。数据的逻辑结构可以通过映像得到与它对应的存储结构。数据元素在计算机中的映像是元素（Element）或结点（Node），数据项的映像称为数据域（Data Filed）。数据元素间关系的表示是存储结构的主要部分，它由存储结点之间的关联方式表达，有以下4种关联方式。

①顺序存储。每个存储结点只含一个数据元素，所有存储结点相继存放在一个连续的存储区里，逻辑上相邻的数据元素存放到计算机内存中仍然相邻。借助数据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，用这种方式表示逻辑关系的存储结构称为顺序存储结构。

②链式存储。每个存储结点不仅含有一个数据元素，还包含一组指针，每个指针指向一个与本结点有逻辑关系的结点。该方法不要求逻辑上相邻的数据元素在物理位置上也相邻，数据元素之间的逻辑关系是由附加的指针表示的。

③索引存储。每个存储结点只包含一个数据元素，所有存储结点连续存放，此外使用该方式存放元素的同时，还需建立附加的索引表，索引表中的每一项称为索引项，索引项的一般形式是：（关键字，索引），其中的关键字是能唯一标识一个结点的数据项，索引指示各存储结点的存储位置或位置区间端点。

④散列存储。每个存储结点只包含一个数据元素，各个结点均匀分布在存储区里。该方法的基本思想是通过构造散列函数，根据结点的关键字直接计算出该数据元素的存储地址。

上述4种基本的存储方法既可以单独使用，也可以组合起来对数据结构进行存储映像。同一种逻辑结构采用不同的存储方法，可以得到不同的存储结构。选择何种存储结构来表示相应的逻辑结构，可以视具体要求而定，主要考虑运算方便及算法的时空要求。

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内存中的物理位置，但由于我们是在高级程序语言的层次上讨论数据结构的操作，因此，不直接以内存地址来描述存储结构，而借用高级语言中提供的“数据类型”来描述它。例如，顺序存储结构用数组来描述，链式存储结构利用指针来描述。在实际程序设计过程中，针对一种数据的逻辑结构，所选择的存储结构会影响到具体算法的设计。也就是说，在设计具体算法之前，必须先确定存储结构。在C语言中，一般使用typedef语句来为存储结构定义新的数据类型名字。

数据的逻辑结构和物理结构是密切相关的两个方面，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。

（3）数据的运算。

数据的运算是指对数据施加的操作。运算的定义取决于逻辑结构，运算的实现必依赖于存储结构。由于运算只描述处理功能，不包括处理步骤和方法，因此运算实现的核心是处理步骤的规定，即算法的设计。

1.1.3 数据类型与抽象数据类型

1. 数据类型

数据类型（Data Type）是与数据结构密切相关的一个概念，几乎所有高级语言都提供

这一概念。数据类型是一个值的集合，以及在这个集合上定义的一组操作的总称。例如，C语言中的整型变量，其值集为某个区间上的整数（区间大小依赖于不同的机器），定义在其上的操作为：加、减、乘、除和取模等运算。

按“值”是否可分解，可以把数据类型分为两类：

（1）原子类型：其值不可分解，如C语言的基本类型（如整型、字符型、实型）、指针类型和空类型。

（2）结构类型：其值可分解成若干成分（或称分量），如C语言的数组类型、结构类型等。结构类型的成分可以是原子类型，也可以是某种结构类型。我们可以把数据类型看做程序设计语言已实现的数据结构。

引入数据类型的目的，从硬件角度考虑，是作为解释计算机内存中信息含义的一种手段；对用户来说，实现了信息的隐蔽，即将一切用户不必了解的细节都封装在类型中。例如，用户在使用整数类型时，既不需要了解整数在计算机内如何表示，也不必了解其操作（如两个整数相加）在硬件中是如何实现的。

2. 抽象数据类型

抽象数据类型（Abstract Data Type，ADT）是指一个数学模型以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，都不影响其外部的使用。

抽象数据类型和数据类型实质上是一个概念。例如，整数类型是一个ADT，其数据对象是指能容纳的整数，基本操作有加、减、乘、除和取模等。尽管它们在不同处理器上的实现方法可以不同，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，在现在的程序设计方法学中，要求在构成软件系统的每个相对独立的模块上，定义一组数据和施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组成，而数据结构又包括数据元素间的关系，因此抽象数据类型一般可以由元素、关系及操作3种要素来定义。

3. 抽象数据类型的表示

本书按以下格式表示抽象数据类型：

```
ADT 抽象数据类型名
    数据元素集合:
数据元素集合的定义
    基本操作:
基本操作的定义
```

其中，数据元素用自然语言描述，基本操作用伪码描述，并规定基本操作的格式为：

```
    中文名(操作名):含义
```

[例1.4] 抽象数据类型“字符串”的定义。

ADT String
数据元素集合：
字符的一个有限序列。
基本操作：
求串长（StrLen）：求取字符串中字符的个数。
求子串（SubStr）：获取字符串中的一个连续字符序列。
求位串（Index）：定位子串在主串中的第一次出现的位置。
求连接（Concat）：连接两个字符串形成一个新串。
求比较（StrCmp）：比较两个串的大小。
求空串（StrEmpty）：判断所给字符串是否为空串。
求替换（StrReplace）：替换字符串中指定的所有子串。

1.2　实例：编写 HELLO，WORLD！程序

【实例目的】

了解 C 语言上机环境，掌握 Turbo C 系统中各菜单的使用；熟悉 C 语言的编写风格及 C 语言程序的基本框架。

【实例内容】

编写程序，输出 HELLO，WORLD!。

【实例步骤】

参考《数据结构导论学习辅导》的附录，双击桌面的 Turbo C 快捷方法，输入 HELLO WORLD 的 C 语言应用程序代码，为了使读者更清楚地了解程序的执行过程，下面给出编辑、编译和运行的具体步骤。

（1）编辑程序。

在 Turbo C 2.0 编译界面中按 Alt＋F 键，选择 New，然后输入源程序，如图 1.5 所示。

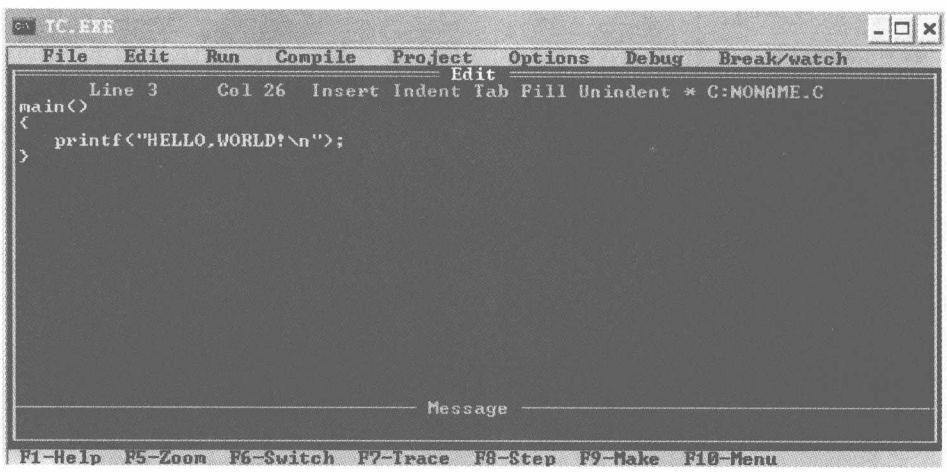

图 1.5　源代码界面

（2）编译和运行程序，运行结果如图 1.6 所示。

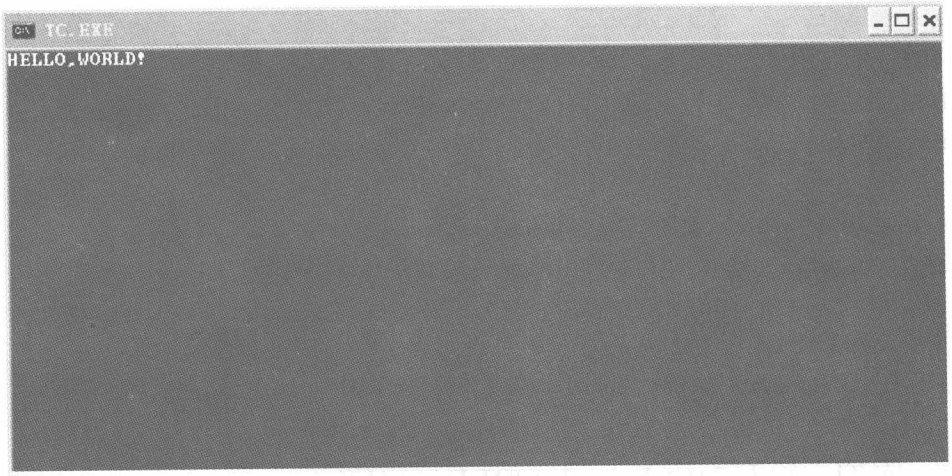

图 1.6　运行结果界面

1.2.1　C 语言的编写风格

一个编写规范的程序便于人们阅读、交流与调试。可读性好有助于人们对算法的理解；晦涩难懂的算法易于隐藏错误难以调试和修改。

在此提供几个在程序编写上提高可读性的方法，以帮助读者建立起良好的编写风格。

1. 注释

一份良好的程序，除了程序本身外，最重要是要有一份完整的程序说明文件。一份没有注释的程序，犹如一部天书，常常会让负责维护的程序员，搞不懂原设计者的设计目的。而一份注释完整的程序，除了自己阅读和排错上的方便外，更容易让人读懂。因此，通常我们选择在重要的程序语句后面加上一个注释内容。

C 语言的注释符为/ * … * /。注释通常用于：

（1）版本、版权声明。

（2）函数接口说明。

（3）重要的代码行或段落提示。如图 1.7 所示给出了一个程序注释的示例。

```
/*
 * 函数介绍：
 * 输入参数：
 * 输出参数：
 * 返回值：
 */
void fun1 (int a, int b)
{
   …
}
```

```
if (…)
{
   …
      while (…)
      {
         …
      }        /* end of while */
   …
}    /* end of if */
```

图 1.7　程序的注释

2. 空行

空行起着分隔程序段落的作用。恰当的空行将使程序的布局更加清晰。通常，在每个类型声明之后和每个函数定义结束之后都要加空行，如图 1.8（a）所示。在一个函数内部，逻辑上密切相关的语句之间不加空行，其他地方应加空行分隔，如图 1.8（b）所示。

```
/* 空行 */
void fun1 (int a, int b)
{
  …
  }
/* 空行 */
void fun2 (int m, int n)
{
  …
}
/* 空行 */
void fun3 (int x, int y)
{
  …
}
```

（a）函数之间的空行

```
/* 空行 */
 while (…)
   {
    statement1;
    /* 空行 */
    if (…)
     {
        statement2;
    }
    else
     {
        statement3;
    }
     /* 空行 */
    statement4;
  }
```

（b）函数内部的空行

图 1.8 空行的使用

3. 代码行

一行代码只做一件事情，例如，只定义一个变量，或只写一条语句。这样的代码容易阅读，并且方便于写注释。if、for、while、do 等语句自占一行，执行语句不得紧跟其后。不论执行语句有多少都要加 { }，这样可以防止书写失误。

代码行示例如图 1.9 所示，其中（a）为风格良好的代码行示例，（b）为风格不良的代码行示例。

4. 对齐

程序的分界符｛和｝应独占一行并且位于同一列，同时与引用它们的语句左对齐。{ } 之内的代码块在｛右边数格处左对齐。

对齐的应用示例如图 1.10 所示，其中（a）为风格良好的对齐示例，（b）为风格不良的对齐示例。

5. 变量命名

变量是程序中用于记录中间变量、输入值或输出结果的一个内存位置，变量所象征的意义正如同该变量在程序中所代表的意义。如果想要编写一个计算学生成绩的程序，程序中需

要用户输入学生学号、语文成绩、英语成绩、数学成绩，最后再计算出 3 门学科的平均成绩。此时如果程序中的变量声明为：

```
int a;
int b1,b2,b3;
int c;
```

以上的定义没有人会看懂这几个变量所代表的意义，就算在程序之初已注明 a 是代表学生学号、b1 代表语文成绩、b2 代表英语成绩、b3 代表数学成绩，c 代表 3 门学科的平均成绩，在程序中，也会很容易忘记某变量代表什么。如果把这 4 个变量声明为：

```
int StudentNum;     /* 学生学号 */
int Chinese;        /* 语文成绩 */
int English;        /* 英语成绩 */
int Math;           /* 数学成绩 */
int Average;        /* 科平均 */
```

这样定义就比较清楚易懂。因为变量的声明通常会在某一个特定的区域（如程序开头），如果在变量之后再加上一些注释说明，这样在编写或修改程序之际，变量声明的区域就像是一个小字典，提供给我们所有在此程序中的输入和输出信息。

int width; /* 宽度 */ int height; /* 高度 */ int depth; /* 深度 */	int widt, height, depth; /* 宽度高度深度 */

a=i+j;
b=m+n;
c=x+y;
	a=i+j; b=m+n; c=x+y;

if (x<y)
{
…
}
	if (x<y) { … }

for (…)
{
…
}
/* 空行 */
Other ();
	for (…) { … }
	Other ();

(a) 风格良好的代码行　　(b) 风格不良的代码行

图 1.9　代码行示例

(a) 风格良好的对齐	(b) 风格不良的对齐

void fun1 (int x)
{
… /* program code */
}
	void fun1 (int x) {
	… /* program code */
	}

if (x>y)
{
… /* program code */
}
else
{
… /* program code */
}
	if (x>y) {
	… /* program code */
	}
	else {
	… /* program code */
	}

for (i=0; i<n; i++)
{
… /* program code */
}
	for (i=0; i<n; i++) {
	… /* program code */
	}

while (x>y)
{
… /* program code */
}
	while (x>y) {
	… /* program code */
	}

如果出现嵌套的 { }，则使用缩进对齐，比如：
{
…
{
…
}
…
}
	出现嵌套的 { }：
	{
	…
	{
	…
	}
	…
	}

图 1.10 对齐的应用示例

1.2.2 C语言预备知识

1. 指针与引用

指针变量是一种特殊的变量，该变量用来保存一个指针值。定义一个指针变量的格式为：

数据类型符 *指针变量名称；

其中，数据类型符是指该指针变量可以用来保存相同类型变量的指针。

&运算符也称为地址运算符，在一个变量前加&运算符，表示该变量的指针；*运算符称为指针运算符，在一个指针变量前加*，表示该指针所指向的内存单元的值。也就是说，指针变量保存的是一个内存的起始地址值，指针变量加*后表示该地址对应的存储单元的值。

C语言规定，指针变量也可以定义为void型。例如：

```
void *p;
```

这里p仍然是一个指针变量，有自己的内存空间，占用2个字节。但是不指定p指向哪