高等学校21世纪计算机教材

数据结构

(C++版)

杨谊　喻德旷　李光明　编著

冶金工业出版社

高等学校21世纪计算机教材

数据结构

（C++版）

杨谊　喻德旷　李光明　编著

北　京

冶金工业出版社

2003

内 容 简 介

本书根据高校计算机《数据结构》课程教学大纲编写而成。本书将数据结构和算法实现相结合，系统地介绍了各种类型的数据结构。对每一种数据结构都从逻辑结构、抽象运算、存储结构以及运算的实现上作了详细的介绍，并列举了大量的实例。内容包括线性表、栈和队列、串和矩阵、递归、树和二叉树、图、排序和查找等。本书设计的程序代码用C++语言实现。

本书条理清晰，重点分明，表达严谨，逻辑严密，通俗简洁，可作为计算机专业、信息专业或其他相关专业的本、专科教材，也是广大参加自学考试的人员和软件开发工作者的参考资料。本书既可作为“数据结构”课程的教材，也可作为其他程序设计类课程的辅导教材。

图书在版编目（CIP）数据

数据结构：C++版 / 杨谊等编著. —北京：冶金工业出版社，2003.6
ISBN 7-5024-3288-4

I. 数… II. 杨… III. ①数据结构②C语言—程序设计 IV. ①TP 311.12 ②TP312

中国版本图书馆CIP数据核字（2003）第035113号

出版人 曹胜利（北京沙滩嵩祝院北巷39号，邮编100009）
责任编辑 程志宏
中山市新华印刷厂有限公司印刷；冶金工业出版社发行；各地新华书店经销
2003年8月第1版，2003年8月第1次印刷
787mm×1092mm 1/16； 21.25印张； 518千字； 332页； 1-5000册
30.00元
冶金工业出版社发行部 电话：（010）64044283 **传真：**（010）64027893
冶金书店 地址：北京东四西大街46号（100711） **电话：**（010）65289081
（本社图书如有印装质量问题，本社发行部负责退换）

前　言

一、关于数据结构

计算机的应用领域日益扩大，其应用早已不再局限于简单的数值运算，而涉及到问题的分析、数据结构框架的设计以及插入、删除、排序、查找等复杂的非数值处理和操作。数据结构的学习就是为以后从事非数值处理的程序设计打下坚实的理论、方法和技术基础。

数据结构是计算机专业的核心课程，是从事计算机软件开发、应用人员必备的专业知识。随着计算机的日益普及，掌握数据结构知识已经成为计算机相关专业的基本要求。

数据结构旨在分析研究计算机加工的数据对象的特性，以便选择适当的数据结构和存储结构，从而使建立在其上的解决问题的算法达到最优。

二、结构安排

全书共分为 9 章，各章主要内容安排如下：

第 1 章：概述。主要介绍了数据结构的基本概念及研究对象，结合 C++语言介绍了面向对象设计以及算法的相关概念。

第 2 章：线性表。主要介绍了线性表的概念、线性表的抽象数据类型、循环链表等内容。

第 3 章：栈和队列。主要介绍了栈和队列的定义及实现等内容。

第 4 章：串和矩阵。主要介绍了串和矩阵的特殊性质和表示方法，以及在各种表示方法上的相关操作。

第 5 章：递归。主要介绍了递归的概念、算法设计和实现以及广义表等内容。

第 6 章：树和二叉树。主要介绍了树的有关概念、二叉树的相关概念及应用、树的表示、树的遍历，以及森林的二叉树的表示和遍历等内容。

第 7 章：图。主要介绍了图的基本概念、图的存储结构及 C++实现、图的遍历、图的连通性问题、最短路径、AOV 网络与拓扑排序以及 AOE-网与关键路径等内容。

第 8 章：排序。主要介绍了排序的基本概念，常用的 5 种排序方法（插入、选择、交换、归并和基数）及它们的实现算法。

第 9 章：查找。主要介绍了查找的基本概念，以及几种最常用的查找方法：顺序表查找、索引表查找、树表查找和哈希表查找。

附录中收录了两套模拟试题，供读者进行自我测试，并附有参考答案和解析。

三、本书特点

本书是作者在多年的教学实践的基础上，参考了近年出版的多种数据结构类书籍编写而成的。本书具有以下特点：

（1）内容全面、系统，实用性强。

本书对每一种数据结构的不同存储方法及有关算法进行了详细的介绍。介绍了抽象数据结构的使用与基本的算法分析技术。书中使用了参数化的模板，提高了算法中数据类型的通用性，支持高效的代码重用。本书内容全面、系统，覆盖了数据结构的主要研究内容，概念清晰，章节合理，逻辑性强。

（2）叙述翔实，讲练结合。

本书对重点设计思想介绍和分析时，结合了大量的图解和具体的实例分析，使抽象的内容变得具体生动，浅显易懂，学习者能够联系实际，深刻地掌握数据结构的实质内容。

数据结构的学习过程也是进行复杂程序设计的训练过程，因此在整个教学过程中，习题和上机练习是两个不可缺少的环节。

本书在每章都设计了大量的有代表性的习题。并配有答案和解析。题目分为选择、填空、问答和编程四种类型，题型全面，覆盖面广。对每道题目都有详细的解析，有利于学习者巩固所学的基本概念、基本方法，以及进行基本的程序设计技能训练。通过学习和练习，使读者能够学会正确地选择数据类型，针对实际问题编写出符合程序规范的代码，为应用程序的开发打下良好的基础。

四、适用对象

本书可作为计算机专业、信息专业或其他相关专业的本、专科教材，也是广大参加自学考试的人员和软件开发工作者的参考资料。本书既可作为“数据结构”课程的教材，也可作为其他程序设计类课程的辅导教材。

本书主要由杨谊、喻德旷、李光明编写，由于时间仓促，水平有限，书中难免有不足之处，敬请读者指正。读者如果有好的意见或建议，可以发 E-mail 到 service@cnbook.net，也可以登录网站 **http：//www.cnbook.net**，在该网站的论坛进行探讨。

编　者

2003 年 5 月

目　录

第1章 概 述

计算机的普及和广泛应用，带来了对应用软件的大量需求。软件的开发不再像过去那样，是单纯个人的才智的结果，而必须遵循软件工程的原则，把软件结构和质量标准化、开发和维护体系工程化。对于软件来说，最重要的就是建立合理的软件体系结构，设计高效的适当的数据结构。各种数据的存储和组织成为人们研究的主要目标。面向对象技术的出现，为软件开发提供了更加有效的工具。

本章从数据结构的基本概念入手，介绍数据结构的研究对象，详细阐述抽象数据类型的必要性和使用方法。接着，结合常用的C++语言，介绍了面向对象程序设计的基本机制。最后，提出算法的概念，并讨论了算法的时间和空间复杂度。

1.1 什么是数据结构

1.1.1 数据结构的概念

数据结构是计算机软件的一门基础课程，计算机科学的各个领域及有关的应用软件都会用到各种数据结构。例如，程序设计语言的编译要使用栈、散列表及语法树；操作系统中使用到了队列、堆栈存储管理表及目录树等；在人工智能领域，依求解问题性质的差异将涉及到各种不同的数据结构，如广义表、集合、搜索树及各种有向图等；数据库系统利用线性表、多链表及索引树等进行数据管理。

数据结构这门课程的目的是介绍一些最常用的数据结构，阐明数据结构内在的逻辑关系，讨论它们在计算机中的存储表示，并结合各种典型应用说明它们在进行各种运算和操作时的性质，介绍算法的设计和执行。数据结构是信息的组织方式。对于相同的算法，用不同的数据结构表示其中的抽象数据类型会造成不同的执行效率。这就有必要研究各种抽象数据类型用不同的数据结构表示的效率差异，以及其适用场合。通过数据结构课程的学习，通过对不同存储结构和相应算法的对比，根据问题的性质，选择合理的数据结构，并控制求解算法的空间、时间及复杂性，使得有一定计算机基础的人员达到提高软件设计和编程水平，增强求解的能力的目的。

1.1.2 数据结构基本术语

数据结构（Data Structure）是数据存在的形式，是指互相之间存在着特定关系的数据元素的集合。数据元素之间的关系称为结构。

数据结构分为逻辑上的数据结构和物理上的数据结构。逻辑上的数据结构反映成员数据之间的逻辑关系，而物理上的数据结构反映成员数据在计算机内部的存储安排。数据结构研究的主要内容是：数据的逻辑结构；数据的物理存储结构；对数据的操作即算法。算法的设计取决于数据的逻辑结构，算法的实现取决于数据的物理存储结构。所有的定义在数据结构上的操作必须保持数据结构的逻辑和物理结构。

数据是指由有限的符号组成的元素的集合。结构是元素之间的关系的集合。一般地，一

个数据结构 DS 可以表示为一个二元组：

DS=(D，S)

在这个二元组中，D 是数据元素的集合（或者是“结点”，可能还含有“数据项”或“数据域”），S 是定义在 D（或其他集合）上的关系的集合，S = { R | R : D×D × ...}，称之为元素的逻辑结构。

1. 数据的逻辑结构

根据数据元素间关系的不同特性，一般有四种基本类型的逻辑结构：

（1）集合结构：数据元素间的关系是“属于同一个集合”，元素关系松散。

（2）线性结构：数据元素之间存在着一对一的关系。

（3）树型结构：数据元素之间存在着一对多的关系。

（4）图形结构：数据元素之间存在着多对多的关系，图形结构也称作网状结构。

表和树分别是线性结构和树型结构的基本实现形式，是最常用的两种数据结构，许多高效的算法可以用这两种数据结构来设计实现。

有如表 1-1 所示的人员信息表。

表 1-1 人员信息表

编号	姓名	性别	职务	科室
001	郑清远	男	主任	
002	许闻及	男	科长	电信
003	刘风名	女	科员	材料
004	赵欣成	女	科长	材料
005	崔　立	男	科员	电信
006	钟与和	男	科员	电信

在表 1-1 中，第一行为目录行，给出了表中每条记录的结构。表中一共有 6 条记录，记录给出了人员的有关信息：每列为一个数据项，描述人员的一个属性。每条记录都由 5 个数据项组成：编号、姓名、性别、职务、科室。数据项的数据类型根据实际需要，可以是字符型，也可以是数值型或其他类型。

在表 1-1 中，所有记录的编号的值都是不相同的，或者说是惟一的，因此，我们把这样的数据项叫做关键项，关键项的值叫做关键字或关键码（Key）。当然，一个表中的记录还可以有多个关键字，也可以没有关键字。

下面，我们用表 1-1 中的关键字来代表整个记录，构造一种数据结构 PersonDS1：

D={001,002,003,004,005,006}

R={}

在数据结构 PersonDS1 中，只存在元素的集合，不存在关系的集合，表示只需要考虑表中的记录（数据），并不考虑它们之间的结构关系。具有这种松散的组织特点的数据结构叫做集合结构。集合结构也可以看做是元素按照任一顺序排列的线性结构，在存储器中可以按照任何方式存储。

根据表 1-1 所示，再构造一种数据结构 PersonDS2：

D={001,002,003,004,005,006}

R={<005,003>,<003,002>,<002,004>,<004,001>,<001,006>}

这个数据结构对应的图形如图 1-1 所示。

图 1-1 PersonDS2 对应的图形

经过分析便可以发现，图 1-1 中关系 R 是根据人员姓名的汉语拼音升序来构造的。

在 PersonDS2 中，每个数据元素有且仅有一个直接前驱元素（除了第一个元素 005），有且只有一个直接后继元素（除了最后一个元素 006）。这种数据结构被形象地称为线性结构。

根据表 1-1 所示，再构造一种数据结构 PersonDS3：

D={001,002,000,000,005}

R={<001,002>,<001,004>,<002,005>,<002,006>,<004,003>}

这个数据结构对应的图形如图 1-2 所示。

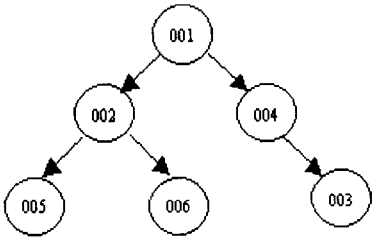

图 1-2 PersonDS3 对应的图形

显然，如图 1-2 所示的关系 R 表示的是表 1-1 中人员之间的领导级别关系。

图 1-2 像一棵倒画的树，最上面的一个结点没有前驱，只有后继，叫做树的根结点。最下面一层结点只有前驱没有后继，叫做树叶结点，其他结点叫做树枝结点。每个结点有且只有一个前驱结点（除了根结点），可以有任意多个后继结点（0 个或多个）。这种具有层次关系的数据结构叫做树形结构，简称为树。

利用树型结构，可以设计出许多高效的算法，解决大量特殊的复杂问题。树型结构具有递归的性质——每一个叶节点可以被一棵子树所替代，同样，每一棵子树也可以用一个叶结点来抽象表示。每一种递归的结构都可以转化为树形结构。

根据表 1-1 所示，再构造一种数据结构 PersonDS4：

D={001,002,003,004,005}

R={<001,002>,<002,001>,<001,004>,<004,001>,<002,005>,<005,002>,<002,006>,<006,002>,<004,006>,<006,004>,<004,003>,<003,004>,<003,006>,<006,003>}

这个数据结构对应的图形如图 1-3 所示。

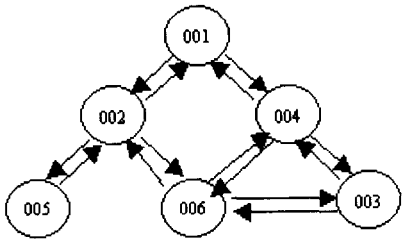

图 1-3 PersonDS4 对应的图形

在这里，关系 R 表示人员之间有业务往来。可以看出，R 呈现对称性，即设 x，y 为图中的结点，则对于任意一个关系<x，y>都有关系<y，x>存在，把<x,y>和<y,x>这两个关系用一个无序对(x,y)来表示。在图形中，把 x 结点和 y 结点之间的两条方向相反的有向边用一条无向边来代替。关系 R 改写为：

R={(001,002),(001,004),(002,005),(002,006),(004,006),(004,003),(003,006)}

对应的图形如图 1-4 所示。

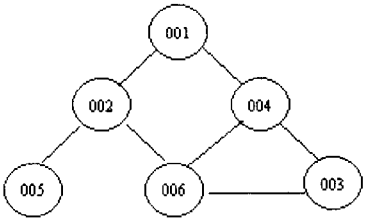

图 1-4　图 1-3 的等价表示

从图 1-3 和图 1-4 可以看到，结点之间的联系是多对多的，即网状关系。每个结点可以有任意多个前驱结点和任意多个后继结点，这种数据结构叫做图形结构，简称为图。

从上面的分析可以知道，树形结构是图形结构的特殊情况，线性结构又是树形结构的特殊情况。树形结构和图形结构统称为非线性结构。

2. *数据的物理存储结构*

数据可以用不同的形式进行描述或存储在计算机存储器中。最常见的数据描述方法有：表达式描述、链接描述、间接寻址和游标。

表达式描述借助数学表达式来确定元素的存储器地址。在高级程序设计语言中，最简单的例子是数组。还可利用函数关系根据元素的某些特征来计算元素在内存中的存储位置——散列(Hash)。

链接描述是指每个元素可以存储在存储器的不同区域中，每个元素都包含一个指向下一个元素的指针，即链表。这种描述方法的优点是，根据第一个元素的位置，就可以依次找到第 n 个元素的位置，而且在其中插入元素非常方便，缺点是由于只知道头（或尾指针）这一个地址，因此查找某个元素要遍历所有在该元素之前的元素。

在间接寻址方式中，元素表中的每个元素也可以存储在存储器的不同区域中，不同的是，此时必须保存一张存储元素地址的表，该表的第 i 项指向元素表中的第 i 个元素。这种描述方法同时具有前两种描述方法的优点，便于确定某个元素的地址，也便于插入和删除元素，但是需要额外的内存空间来保存元素地址。C++中的指针数组就是这种描述法的应用

游标与链接相似，但它用整数代替了指针。集合中的每个元素除了包含自身的数据信息以外，还包括一个指示下一个元素位置的整数，通过指定的函数由该整数计算出下一个元素的存储器地址。

1.2　抽象数据类型及面向对象程序设计

1.2.1　数据类型

高级语言引入了数据类型的概念，即把所有的数据加以分类。数据类型是对数据的取值

范围、数据的结构以及允许对数据执行的操作的描述。每一种高级程序设计语言都有自己的数据类型。一般的高级语言都有整型、实型、字符型、字符串型、指针型、记录型、枚举型、文件型等数据类型。例如，整型数在计算机中一般用两个字节或四个字节表示；对整型数据施加的操作一般有加、减、乘、除、不等于、等于、大于、小于等操作。字符型在机器中常用一个字节表示，常见的运算有赋值运算和关系运算等。

数据类型是数据或数据变量类型的说明，它指示该数据或数据变量所有可能取的值。例如，在C++中，除了标准的基本数据类型（布尔型、字符型、整型和实型）外，还提供用户可自定义的枚举类型和指针类型。C语言对于有结构的数据，提供了数组、记录、结构和联合等四种标准的结构数据类型。程序设计者可以利用所提供的基本数据类型构造有结构的数据，通过复合或嵌套构造更复杂更高层的结构数据。C++提供了面向对象程序设计的基石——类的机制。

引入数据类型实现了信息的物理存储细节的隐蔽，例如，在使用整数时，人们不需要了解它在机器内存中的具体表示方式，也不需要知道它的运算位操作的实现细节，而把注意力集中到整数的抽象特征以及运算结果上。

1.2.2 抽象数据类型

对于一个明确的数学问题，设计它的算法，总是先选用该问题的一个数据模型。接着，弄清该问题的初始状态和要求的结果状态，以及两个状态之间的关系。然后探索从初始状态到结果状态所必需的运算步骤，即求解算法。

在设计算法时，首先应该考虑算法顶层的运算步骤，它们组成算法的主干部分；然后再考虑底层的运算步骤，运算的具体实现。底层的运算步骤包括数据模型的具体表示和定义在该数据模型上的运算的具体实现。为了将顶层算法与底层算法隔开，使二者在设计时不会互相牵制、互相影响，必须对二者的接口进行一次抽象。底层只通过这个接口为顶层服务，顶层也只通过这个接口调用底层的运算。这个接口就是抽象数据类型（Abstract Data Types，简记ADT）。

使用抽象数据类型将给算法和程序设计带来以下好处：

（1）算法顶层的设计与底层的设计被隔开，使得在进行顶层设计时不必考虑它所用到的数据表示和运算实现；同时，在进行数据表示和运算实现等底层设计时不必考虑它被谁使用。

（2）算法设计与数据结构设计隔开，可优化算法，提高程序运行的效率。

（3）由于顶层设计和底层设计被局部化，设计中的错误是局部的，容易查找和纠正，程序具有好的可维护性。

（4）代码模块化，便于程序正确性的证明和复杂性的分析；抽象的数据类型的表示和实现都可以封装起来，便于移植和重用。

在定义了抽象数据类型之后，算法底层的设计任务就可以明确为：首先用抽象数据类型名构造具体的构造数据类型，然后为抽象数据类型上的每个运算名设计具体的运算内容。

1.2.3 面向对象程序设计

大型软件系统复杂度的不断增加，对软件系统设计方法也提出了新的要求。

传统的模块化设计方法要求有一位软件总体设计师了解整个系统并负责分解任务，如图

1-5 所示简要地表示了这种层次关系。这种自顶向下的设计方法将系统视为分层的子程序集合。在最顶层，主程序通过顺序调用子程序来完成计算并返回信息。子程序还可以进一步分解为完成更小任务的子程序。显然，这对主程序设计者提出了很高的要求，他需要对各个层次的设计要求和细节了如指掌。如果软件系统的庞杂性超出个人能力之外时，自顶向下的方法是不适用的。而且，上层子程序的微小改动都可造成底层程序大量的修改。

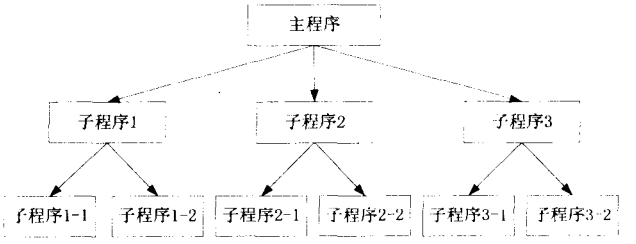

图 1-5 传统的模块化设计方法图解

与传统程序设计风格相对照的是一种崭新的程序设计思想——面向对象程序设计，它提出了一种新的系统设计模型。面向对象的思想不像过去那样让问题去适应程序，而是让程序去适应要解决的问题。它将系统看成通过交互作用来完成任务的对象的集合，每个对象用自己的方法管理数据。

面向对象程序设计方法以对象为设计的基础。比如说，在程序中可以建立多种数据结构和方法，把它们组织成具有各种相互关系的层次结构。问题被划分为具有数据的行为的对象。对象以及对象之间的操作组成模块。模块有机地组织起来，实现问题的解决。模块的重新组合和重复利用都是非常方便的。

如图 1-6 所示是一般软件开发方法的图示，它将软件开发过程划分为明显的阶段，即问题分析和程序功能定义、对象设计、编码、测试与维护。

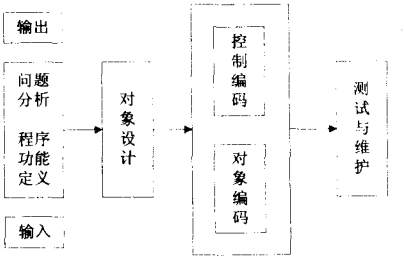

图 1-6 软件开发方法图示

1.3 C++实现面向对象程序设计

C++是一种运用较广泛的程序设计语言，也是数据结构实现的优良语言。C++相对其他语言具有与计算机硬件集合紧密、代码效率高等特点。

1.3.1 C++语言概述

C++语言是一种应用较广的面向对象的程序设计语言，使用它可以实现面向对象的程序设计。

C++语言的主要特点是：

1. C++支持数据封装

支持数据封装就是支持数据抽象。在C++中，类是支持数据封装的工具，对象则是数据封装的实现。在面向对象的程序设计中，将数据和对该数据进行合法操作的函数封装在一起作为一个类的定义，数据将被隐藏在封装体中，该封装体通过操作接口与外界交换信息。C++中的类是数据和函数的封装体。在C++中，结构可作为一种特殊的类，它虽然可以包含函数，但是它没有私有或保护的成员。

2. C++类中包含私有、公有和保护成员

C++类中可定义三种不同访问控制权限的成员。一种是私有（Private）成员，只有在类中说明的函数才能访问该类的私有成员，而在该类外的函数不可以访问私有成员；另一种是公有（Public）成员，类外面也可访问公有成员，成为该类的接口；还有一种是保护（Protected）成员，这种成员只有该类的派生类可以访问，其余的在这个类外不能访问。

3. C++中通过发送消息来处理对象

C++中是通过向对象发送消息来处理对象的，每个对象根据所接收到的消息的性质来决定需要采取的行动，以响应这个消息。响应这些消息是一系列的方法，方法是在类定义中使用函数来定义的，使用一种类似于函数调用的机制把消息发送到一个对象上。

4. C++允许函数名和运算符重载

C++支持多态性，C++允许一个相同的标识符或运算符代表多个不同实现的函数，这称为函数名或运算符的重载，用户可以根据需要定义函数名重载或运算符重载。

5. C++支持继承性

C++中可以允许单继承和多继承。一个类可以根据需要生成派生类。派生类继承了基类的所有方法，另外派生类自身还可以定义所需要的不包含在父类中的新方法。一个子类的每个对象包含有从父类那里继承来的数据成员以及自己所特有的数据成员。

6. C++支持动态联编

C++中可以定义虚函数，通过定义虚函数来支持动态联编。

1.3.2 C++程序结构的组成

C++的程序结构主要包括程序主体、编译预处理和注释。

C++的程序主体由函数组成，每个C++程序都有且只有一个主函数main()，以及若干个其他函数。因此，可以认为C++的程序就是函数串。函数与函数之间是相对的，并且是并行的，函数之间可以调用。C++程序总是从main()开始执行的。函数是由若干条语句组成的。语句是程序的基本单元。语句是由单词组成，单词间用空格符分隔，C++程序中的语句是以分号结束。语句除了有表达式语句和空语句之外，还有复合语句、分支语句、循环语句和转向语句等若干类。程序中一般都需要说明和使用变量。变量也称为一种对象，可将对象看作是类的实例。C++程序中经常使用到输入和输出的语句，实现与程序内部的信息交流。

编译预处理命令是以“#”开头的语句，作用是在编译之前将所引用的文件的内容包含到当前的C++程序中，作为它的一部分来执行。C++提供了三类预处理命令：宏定义命令、文件包含命令和条件编译命令。

C++的注释是以“//”开头直到换行的内容，也兼容了C中的注释，即一对符号“/*”和“*/”之间的内容。注释部分仅供阅读程序时参考，不参加程序的编译和运行。

1.3.3 类和对象的定义

类是面向对象程序设计的核心，它实际上是一种新的数据类型，也是实现抽象类型的工具，因为类是通过抽象数据类型的方法来实现的一种数据类型。类是对某一类对象的抽象；而对象是某一种类的实例，因此，类和对象是密切相关的。

1. 什么是类

类是一种复杂的数据类型，它是将不同类型的数据和与这些数据相关的操作封装在一起的集合体。

类的结构即类的组成是用来确定一类对象的行为的，而这些行为是通过类的内部数据结构和相关的操作来确定的。这些行为是通过一种操作接口来描述的，即平时所看到的类的成员函数。使用者只关心的是接口的功能，即类的各个成员函数的功能，而对于这些成员函数是如何实现的并不感兴趣。而操作接口又被称为这类对象向其他对象所提供的服务。

2. 类的定义格式

类的定义格式一般地分为说明部分和实现部分。说明部分是用来说明该类中的成员，包含数据成员的说明和成员函数的说明。成员函数是用来对数据成员进行操作的，又称为方法。实现部分是用来对成员函数的定义。概括说来，说明部分将告诉使用者干什么，而实现部分是告诉使用者怎么干。

类的一般定义格式如下：

```
class <类名>
{
    public:
        <成员函数和数据成员的说明>
    private:
        <数据成员和成员函数的说明>
};
<各个成员函数的实现>
```

在上面的定义中，class 是定义类的关键字，<类名>可以是各种标识符，通常用 T 字母开始的字符串作为类名。一对花括号内的部分是类的说明部分，包括前面的类头，用来说明该类的成员。类的成员包含数据成员和成员函数两部分。从访问权限上来分，类的成员又分为：公有的（Public）、私有的（Private）和保护的（Protected）三类。公有的成员用 Public 来说明，公有部分往往是一些操作即成员函数，它是提供给用户的接口功能。这部分成员可以在程序中引用。私有的成员用 Private 来说明，私有部分通常是一些数据成员，这些成员是用来描述该类中的对象的属性的，用户是无法访问它们的，只有成员函数或经特殊说明的函数才可以引用它们，它们是被用来隐藏的部分。保护类（Protected）成员则允许私有成员访问，而不允许普通函数或其他类的成员函数从外部访问。

其中，<各个成员函数的实现>是类定义的实现部分，这部分包含所有在类体内说明的函数的定义。如果一个成员函数在类体内定义了，实现部分将不出现。如果所有的成员函数都在类体内定义，则实现部分可以省略。

下面的例子是一个日期类的定义：

```
//类的成员定义
class TDate
{
```

```
    public:
        void SetDate(int y, int m, int d);
        int IsLeapYear();
        void Show();
    private:
        int year, month, day;
};

//类的成员函数的实现
void TDate::SetDate(int y, int m, int d)
{
    year = y;
    month = m;
    day = d;
}

int TDate::IsLeapYear()
{
    return(year%4==0 && year%100!=0) || (year%400==0);
}

void TDate::Show();
{
    cout<<year<<"."<<month<<"."<<day<<endl;
}
```

这里出现的作用域运算符::是用来标识某个成员函数是属于哪个类的。在上面所举的例子中，SetDate、IsLeapYear 和 Show 这三个成员函数都是属于 Tdate 类的。成员函数定义在类体外时需要在函数头的前面加上该函数所属类的标识（类名），以及作用域运算符“::”。

3. 对象和对象的定义

对象的格式如下：

<类名> <对象名表>

其中，<类名>是待定的对象所属的类的名字，即所定义的对象是该类类型的对象。<对象名表>中可以有一个或多个对象名，存在多个对象名时用逗号分隔。<对象名表>中，可以是一般的对象名，还可以是指向对象的指针名或引用名，也可以是对象数组名。如：

```
TDate date1, date2, *Pdate, date[31];
```

一个对象的成员就是该对象的类所定义的成员。对象成员有数据成员和成员函数，其表示方式如下：

<对象名>. <成员名>

或者

<对象名>. <成员名>(<参数表>)

前者用来表示数据成员，后者用来表示成员函数。如：

```
date1.year, date1.month, date1.day;
date1.SetDate(int y, int m, int d);
```

这里，“.”是一个运算符，该运算符的功能是表示对象的成员。

指向对象的指针的成员表示如下：

<对象指针名> → <成员名>