数据结构与算法分析

Data Structures and Algorithm Analysis

唐宁九 主编

四川大学出版社

四川省精品课程、四川大学精品课程建设资助项目

数据结构与算法分析

唐宁九　主编

唐宁九　游洪跃　朱　宏　杨秋辉　编

四川大学出版社

责任编辑:毕　潜　廖庆扬
责任校对:任德斌
封面设计:吴　强
责任印制:杨丽贤

图书在版编目(CIP)数据

数据结构与算法分析 / 唐宁九主编. —成都：四川大学出版社，2006.8
ISBN 7-5614-3484-7

Ⅰ. 数...　Ⅱ. 唐...　Ⅲ.①数据结构-高等学校-教材②算法分析-高等学校-教材③C语言-程序设计-高等学校-教材　Ⅳ.TP311.12

中国版本图书馆CIP数据核字（2006）第094900号

书名　**数据结构与算法分析**

主　　编　唐宁九
出　　版　四川大学出版社
地　　址　成都市一环路南一段24号（610065）
发　　行　四川大学出版社
印　　刷　成都蜀通印务有限责任公司
成品尺寸　185 mm×260 mm
印　　张　22
字　　数　523千字
版　　次　2006年8月第1版
印　　次　2006年8月第1次印刷
印　　数　0 001～2 000册
定　　价　32.50元

◆读者邮购本书,请与本社发行科联系。电话:85408408/85401670/85408023　邮政编码:610065
◆本社图书如有印装质量问题,请寄回出版社调换。
◆网址:www.scupress.com.cn

前　言

《数据结构与算法分析》是计算机程序设计的主要理论技术基础，它已成为计算机及其相关专业的核心课程。本课程研究的数据是非数值性、结构性的数据。学习本课程要求掌握各种主要数据结构的特点，在计算机内的表示方法，处理数据的算法设计，对于算法所花费的时间和空间代价的分析，以及在计算机科学中最基本的应用。通过本课程的学习，要求学生能够掌握组织、处理数据的理论和方法，培养训练学生选用合适的数据结构，编写质量高、风格好的应用程序，及初步评价算法程序的能力。

本书采用C++面向对象的观点介绍数据结构与算法分析，这与传统采用面向过程的观点相比优势较大。面向对象程序设计有三个基本原则：封装性、继承性和多态性。在C++中通过类实现了封装，恰如其分地实现数据结构；通过继承性反映、通过旧概念来引入新概念的思想；使用多态性直接实现抽象数据类型。全书已将面向对象程序设计的思想融合到数据结构与算法分析中，因此通过学习可进一步提高面向对象程序设计的能力。

全书共分为10章，第1章是基础知识，介绍了基本概念及其术语，抽象数据类型的实现，本书涉及的C++主要知识点，最后还讨论了算法的概念和算法分析的简单方法。

第2章引入线性表，详细介绍了线性表的顺序存储结构与链式存储结构。在讨论链式存储结构时，首先仿照传统的方法实现线性表，然后在此基础上，在链表结构中保存当前位置和元素个数，在难度增加不大的情况下提高算法效率，使学生逐步体会改进算法的途径与方法。所有类及算法都在Visual C++6.0环境下进行了严格的测试。

第3章介绍了栈和队列，讨论了栈和队列的顺序存储结构与链式存储结构，用栈实现了表达式求值，通过学习能掌握各种栈的队列的实现和使用方法，对后继课程（如操作系统原理和编译原理）的学习打下良好的基础，所有类及算法都通过了测试。

第4章介绍串，详细讨论了串的各种存储结构与模式匹配算法，为开发串应用软件（如文本编辑软件）打下坚实的基础。

第5章介绍数组和广义表，详细讨论了数组、特殊矩阵、稀疏矩阵和广义表的存储结构

及实现方法，通过模板、继承等技术，使读者对数据抽象概念有更进一步的认识。

第 6 章和第 7 章介绍了最重要的非线性结构——树和图，对其中的二叉树、图的实现及相关算法（如 Kruskal 算法）都通过测试程序加以实现。

第 8 章和第 9 章介绍查找和排序，以简洁方式实现各种查找与排序算法，每个算法都用测试程序进行了严格的测试。

第 10 章介绍了文件，讨论了磁盘结构，各种文件的结构和实现方式，特别还介绍了在大型机中经常采用的 ISAM 文件和 VSAM 文件，对读者学习大型机的文件结构有较大的帮助。

第 11 章和第 12 章介绍了算法设计技术、分析技术与可计算问题。详细讨论了各种算法设计技术（如贪心算法、分治算法、回溯算法）的使用方法，对算法分析技术和可计算问题也进行了深入浅出的讨论。对读者的算法设计和分析的理论和实践会有极大的帮助。

本书内容丰富，读者可根据时间和能力对全书内容加以剪裁，如第 11 章和第 12 章可作为选学内容。

本书第 1、2、3 章由唐宁九编写，第 4、5、6 章由杨秋辉编写，第 7、8、9 章由游洪跃编写，第 10、11、12 章由朱宏编写，全书由唐宁九主编、统稿。

由于作者水平有限，书中难免有不妥之处，敬请读者不吝赐教，以便再版时修正。

编者

2006 年 8 月 1 日

目　录

第1章　绪　论 …… (1)
1.1　数据结构讨论的范畴 …… (1)
1.2　基本概念及术语 …… (2)
1.2.1　数据 …… (2)
1.2.2　数据元素 …… (3)
1.2.3　数据对象 …… (3)
1.2.4　数据结构 …… (3)
1.2.5　数据类型 …… (4)
1.2.6　抽象数据类型 …… (4)
1.3　抽象数据类型的实现 …… (5)
1.3.1　C++的简单程序 …… (5)
1.3.2　C++作用域的说明 …… (6)
1.3.3　C++的类和对象 …… (6)
1.3.4　C++ 的参数传递 …… (8)
1.3.5　C++ 的输入输出 …… (10)
1.3.6　C++ 的动态存储分配 …… (11)
1.3.7　C++的友元函数 …… (13)
1.3.8　C++的运算符重载 …… (13)
1.3.9　结构与类 …… (15)
1.3.10　C++的模板 …… (15)
1.4　算法和算法分析 …… (18)
1.4.1　算法 …… (18)
1.4.2　算法分析 …… (19)
第2章　线性表 …… (23)
2.1　线性表的逻辑结构 …… (23)
2.2　线性表的顺序存储结构 …… (25)
2.3　线性表的链式存储结构 …… (33)
2.3.1　单链表 …… (33)
2.3.2　循环链表 …… (41)

2.3.3 双向链表……（45）
2.3.4 保存当前位置和元素个数……（48）
第3章 栈和队列……（61）
3.1 栈 ……（61）
3.1.1 栈的基本概念……（61）
3.1.2 顺序栈……（62）
3.1.3 链式栈……（67）
3.1.4 栈的应用……（74）
3.2 队 列 ……（77）
3.2.1 队列的基本概念……（77）
3.2.2 链式队列……（79）
3.2.3 循环队列——队列的顺序存储结构……（82）
第4章 串……（88）
4.1 串的定义 ……（88）
4.2 串的存储表示 ……（90）
4.2.1 定长顺序存储表示……（90）
4.2.2 堆分配存储表示……（92）
4.2.3 链表存储表示……（95）
4.3 串的模式匹配算法 ……（96）
4.3.1 简单算法……（96）
4.3.2 首尾匹配算法……（98）
4.3.3 KMP算法 ……（98）
第5章 数组和广义表……（103）
5.1 数 组 ……（103）
5.1.1 数组的基本概念……（103）
5.1.2 数组的存储实现……（104）
5.2 数组的类定义 ……（105）
5.2.1 一维数组的类定义及实现……（106）
5.2.2 二维数组的类定义及实现……（108）
5.3 矩 阵 ……（111）
5.3.1 矩阵的定义和操作……（111）
5.3.2 特殊矩阵……（116）
5.3.3 稀疏矩阵……（120）
5.4 广义表 ……（126）
5.4.1 基本概念……（126）
5.4.2 广义表的存储结构……（128）
第6章 树和二叉树……（135）
6.1 树的基本概念 ……（135）
6.1.1 树的定义……（135）

6.1.2 基本术语 …… (136)
6.2 二叉树 …… (137)
6.2.1 二叉树的定义 …… (137)
6.2.2 二叉树的性质 …… (138)
6.3 二叉树的存储结构 …… (139)
6.3.1 顺序存储结构 …… (140)
6.3.2 链式存储结构 …… (141)
6.4 二叉树遍历 …… (145)
6.4.1 遍历的定义 …… (145)
6.4.2 遍历算法 …… (146)
6.4.3 二叉树遍历的应用 …… (151)
6.5 线索化二叉树 …… (153)
6.5.1 线索化的概念 …… (153)
6.5.2 线索化算法 …… (156)
6.5.3 遍历线索化二叉树 …… (157)
6.6 树和森林 …… (159)
6.6.1 树的存储表示 …… (159)
6.6.2 树和森林的遍历 …… (161)
6.6.3 树和森林与二叉树的转换 …… (163)
6.7 哈夫曼树与哈夫曼编码 …… (164)
6.7.1 哈夫曼树的基本概念 …… (164)
6.7.2 哈夫曼树构造算法 …… (165)
6.7.3 哈夫曼编码 …… (168)
6.8 树的计数 …… (170)
第7章 图 …… (175)
7.1 图的定义和术语 …… (175)
7.2 图的存储表示 …… (179)
7.2.1 邻接矩阵 …… (179)
7.2.2 邻接表 …… (185)
7.3 图的遍历 …… (193)
7.3.1 深度优先搜索 …… (194)
7.3.2 广度优先搜索 …… (195)
7.4 图的最小代价生成树 …… (197)
7.4.1 Prim 算法 …… (197)
7.4.2 Kruskal 算法 …… (200)
7.5 有向无环图及应用 …… (203)
7.5.1 拓扑排序 …… (204)
7.5.2 关键路径 …… (206)
7.6 最短路径 …… (211)

7.6.1 单源点最短路径问题 …… (211)
7.6.2 所有顶点之间的最短路径 …… (214)
第8章 查 找 …… (219)
8.1 查找的基本概念 …… (219)
8.2 静态表的查找 …… (220)
8.2.1 顺序查找 …… (220)
8.2.2 有序表的查找 …… (221)
8.2.3 分块查找 …… (224)
8.3 动态查找表 …… (225)
8.3.1 二叉排序树 …… (225)
8.3.2 二叉平衡树 …… (233)
8.3.3 B树和 B^+ 树 …… (252)
8.4 散 列 表 …… (254)
8.4.1 散列表的概念 …… (254)
8.4.2 构造散列函数的方法 …… (254)
8.4.3 处理冲突的方法 …… (255)
8.4.4 散列表的实现 …… (256)
8.4.5 散列法性能分析 …… (261)
第9章 排 序 …… (263)
9.1 概 述 …… (263)
9.2 插入排序 …… (264)
9.3 希尔排序 …… (266)
9.4 交换排序 …… (268)
9.4.1 起泡排序 …… (268)
9.4.2 快速排序 …… (269)
9.5 选择排序 …… (272)
9.5.1 简单选择排序 …… (272)
9.5.2 堆排序 …… (273)
9.6 归并排序 …… (277)
9.7 基数排序 …… (279)
9.7.1 多关键排序 …… (280)
9.7.2 基数排序 …… (280)
9.8 外部排序 …… (283)
9.8.1 外部排序基础 …… (283)
9.8.2 外部排序的方法 …… (283)
第10章 文 件 …… (287)
10.1 主存储器和辅助存储器 …… (287)
10.2 磁 盘 …… (288)
10.2.1 磁盘结构 …… (288)

10.2.2 磁盘访问代价分析 …… (292)
10.2.3 磁盘缓存 …… (293)
10.3 文件结构 …… (295)
10.3.1 顺序文件 …… (295)
10.3.2 索引文件 …… (297)
10.3.3 ISAM 文件和 VSAM 文件 …… (298)
10.3.4 散列文件 …… (301)
10.3.5 多关键字文件 …… (302)
第 11 章 算法设计技术 …… (305)
11.1 算法设计 …… (305)
11.2 贪心算法 …… (308)
11.2.1 算法思想 …… (308)
11.2.2 应用 …… (308)
11.3 分治算法 …… (311)
11.3.1 算法思想 …… (311)
11.3.2 应用 …… (311)
11.4 回溯算法 …… (315)
11.4.1 算法思想 …… (315)
11.4.2 应用 …… (315)
11.5 分支定界 …… (318)
11.5.1 算法思想 …… (318)
11.5.2 应用 …… (318)
11.6 动态规划 …… (322)
11.6.1 算法思想 …… (322)
11.6.2 应用 …… (322)
第 12 章 分析技术与可计算问题 …… (325)
12.1 分析技术 …… (325)
12.1.1 求和分析 …… (325)
12.1.2 递归分析 …… (326)
12.1.3 均摊分析 …… (328)
12.1.4 算法分析举例 …… (329)
12.2 可计算问题 …… (332)
12.2.1 归约 …… (332)
12.2.2 难解问题 …… (334)
12.2.3 不可解问题 …… (338)

第1章 绪 论

早期的计算机主要应用于数据计算，一般要经过如下几个步骤：

(1) 从具体问题抽象出数据模型。

(2) 设计解决此问题的算法。

(3) 编写程序。

(4) 测试、调试直到解决问题。

当前计算机主要应用于非数值计算，包括处理字符、表格和图像等各种具有一定结构的数据。为编写良好的程序，必须分析和处理对象的特性以及对象之间的关系，并具体分析各种算法的时间性能，这就是数据结构与算法分析主要研究的问题。

1.1 数据结构讨论的范畴

数值计算问题主要通过数学方程建立数学模型。例如，结构静力学的数学模型为线性代数方程组；全球天气预报的数学模型为二阶椭圆偏微分方程；预测人口增长情况的数学模型为常微分方程。求解这些数学方程的算法是计算数学研究的范畴，比如采用高斯消元法、差分法、有限元法、无限元法等算法。

而数据结构主要研究非数值计算问题，非数值问题无法用数学方程建立数学模型，下面通过实例加以说明。

例 1.1 在教务系统中，包含有“学生基本信息”表格，包括了许多学生的基本信息记录（如表 1.1 所示）。这些记录按照不同的顺序号，依次存放在“学生基本信息”表格中，每个学生基本信息记录按顺序号排列，形成了学生基本信息记录的线性序列，呈一种线性关系。

表 1.1 学生基本信息

学号	姓名	性别	籍贯	生日
0201001	刘思源	男	北京	1985. 12. 18
0201002	朱晓敏	女	成都	1986. 6. 28

续表 1.1

学号	姓名	性别	籍贯	生日
0201003	张世民	男	天津	1983. 10. 16
0201004	陈建杰	男	杭州	1982. 11. 29
0201005	游洪杰	男	苏州	1988. 6. 8
0201006	林德凯	男	青岛	1986. 2. 28
0201007	李靖	女	太原	1985. 3. 19
0201008	刘茜	女	广州	1981. 8. 18

例 1.2 一个典型的UNIX文件系统结构如图 1.1 所示。这是一棵倒置的“树”，“树根”代表整个系统，用根目录“/”表示；下一层代表各个子系统，如“bin”、“lib”、“user”等；“叶子”就是文件，如“list.h”、“stack.h”、“queue.h”等。

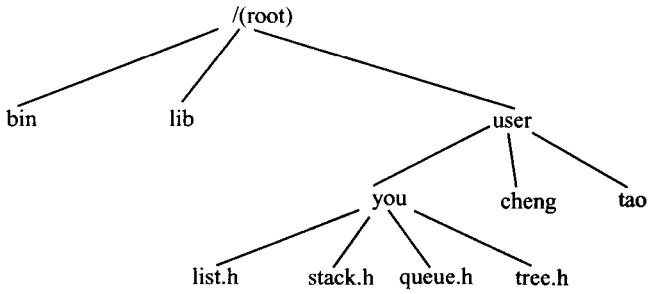

图 1.1 UNIX 文件系统图

例 1.3 不同网站之间可能有通信线路直接相连，图 1.2 是 6 个网站之间的通信联络图，它是一种“图”的数据结构。

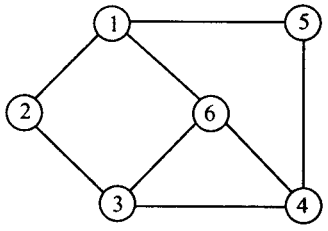

图 1.2 6 个网站之间的通信联络图

从上面三个实例可以看出，非数值计算问题的数学模型不再是数学方程，而是诸如线性表、树和图的数据结构。简单地说，数据结构是一门讨论描述现实世界中非数值计算的操作对象，及这些对象之间的关系、操作和在计算机中的表示及实现的学科。

1.2 基本概念及术语

本节将介绍一些基本概念及术语，对于初学者可能会感到难于理解，但一定会随着课程的不断学习逐步加深理解。

1.2.1 数据

数据在哲学意义上讲就是信息的载体，是对客观事物的符号表示。在计算机学科中则表示一切能输入到计算机中，并能被计算机处理的符号的总称。

数据主要分成两类：一类是数值性数据，比如整数、实数和复数等，这些数据主要应用

于工程计算；另一类数据就是非数值型数据，主要包括文字、图形、语音等数据。

1.2.2 数据元素

数据元素是数据的基本单位，在计算机编程中一般将数据元素作为一个整体进行处理。有时一个数据元素由若干数据项所组成，比如学生基本信息可作为一个数据元素，而学生基本信息可由学号、姓名、性别、籍贯、生日……等内容组成。数据项是不可分割的最小单位。

1.2.3 数据对象

人们习惯于在解决所遇到的问题时，把数据按其性质归类。数据对象是指性质相同的数据元素的集合，如偶数数据对象为 EVEN= {0，±2，±4，±6，…}，英文小写字母数据对象为 LETTER= {'a'，'b'，…，'z'}。

1.2.4 数据结构

数据结构是由某个数据对象及此对象包含的数据元素之间的关系组成。根据数据元素之间关系的特性，有如下四类基本结构：

(1) 集合结构。集合结构的数据元素之间只存在“同属于一个数据对象”的关系，如图 1.3 所示。

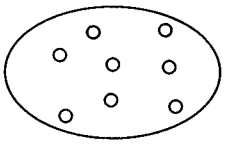

图 1.3 集合结构示意图

(2) 线性结构。线性结构中的数据元素之间存在一个对应一个的关系，如图 1.4 所示。

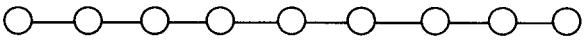

图 1.4 线性结构

(3) 树形结构。树形结构中的数据元素之间存在着一个对应多个的关系，及数据元素之间存在着层次关系，如图 1.5 所示。

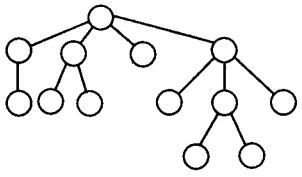

图 1.5 树形结构

(4) 图状结构。图状结构中的数据元素之间存在多个对应多个的关系，如图 1.6 所示。

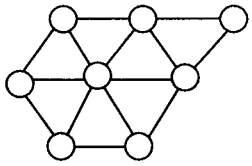

图 1.6 图状结构

数据结构从形式上可定义为如下的二元组：

$$\text{Data\_Structure}=(D, S)$$

其中 D 是一个数据对象，S 为定义在 D 中的数据元素之间的关系的有限集合。下面通过实例加以说明。

例 1.4 在计算机科学中，数列可定义成如下形式的数据结构：

$$\text{Sequence}=\{D, S\}$$

$$\text{其中 } D=\{a_i \mid 1\leqslant i\leqslant n\},\ S=\{R\},\ R=\{<a_{i-1}, a_i> \mid 2\leqslant i\leqslant n\}。$$

根据不同的观点，数据结构可分为逻辑结构和物理结构。逻辑结构是指从解决问题的实际出发，为实现必要功能建立的数学模型，其结构定义中的关系用于描述数据元素之间的逻辑关系，是面向问题的；数据结构在计算机中的表示称为物理结构或存储结构，是面向计算机的。

通常讨论的数据结构，不但要讨论典型的逻辑结构，还应讨论逻辑结构相应的存储结构，而且还要讨论数据结构的相关操作及其实现。

1.2.5 数据类型

程序设计语言中已学过各种数据类型，比如整型和浮点型，这些数据类型不但规定了使用这些类型时的取值范围，而且还规定类型可以使用的不同操作，比如 32 位长度的整型数据的取值范围是 $-2^{31}\sim 2^{31}-1$，能进行的操作有+、−、*、/。数据类型是一组性质相同的值的集合，及定义在此集合上的一些操作的总称。

1.2.6 抽象数据类型

抽象的本质就是抽取出实际问题的本质。据说在欧洲古代没有 3 及 3 以上的概念，他们数数时，3 及 3 以上的数都称为 many，也就是说，他们没有抽象出 3 及 3 以上的自然数。从这里可以看出，只有抽象才具有普遍性。抽象数据类型一般指由用户定义的，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（也可称为服务或方法）的总称。

1.3 抽象数据类型的实现

抽象数据类型可利用处理器中已有的数据类型来实现，并用已实现的操作来组合成新的操作。本书是在高级程序设计语言的基础上讨论抽象数据类型的实现，为使读者易于上手实践，采用C++程序设计语言是较好的选择，这不但可使读者学到数据结构与算法的知识，同时也可加深对程序设计技术的领悟。

下面对C++语言进行讨论，对于已学过C++的读者快速浏览即可，对于没学过C++的读者，最好将所有的实例上机运行，并且在读后面章节时随时查阅相关内容，这样随着学习进度的深入，会不断加深对C++的理解与掌握。

1.3.1 C++的简单程序

下面是一个典型的C++程序结构：输出 Hellow word。此程序由以下三部分组成。

```
// 文件路径名：e1_1/ hellow.h
#ifndef_HELLOW_H_                    // 如果没定义_HELLOW_H_
#define_HELLOW_H_                    // 那么定义_HELLOW_H_
void hellow (char *);                 // 函数原型
#endif
// 文件路径名：e1_1/ hellow . cpp
#include <iostream. h>                // 包含 cout
#include " hellow . h"                // 包含 hellow 的原型
void hellow (char *name)
{
    cout << " Hellow " << name << endl;     // 用 cout 和<<输出,endl 表示回车换行
}
// 文件路径名：e1_1/main. cpp
#include " hellow. h"                 // 提供 hellow ()函数的原型
void main()
{
    hellow ("world!");                // 调用 hellow ()函数显示：Hellow world!
}
```

C++的程序主要分为两类：头文件（扩展名为“.h”）和源程序文件（扩展名为“.cpp”）。头文件一般用于存放函数原型及类声明，上面实例中，hellow.h 中存储有 hellow ()的函数原型，主程序文件 main.cpp 中通过“#include "hellow.h"”包含了头文件 hellow.h，这样便实现了对函数原型及类的使用。

源程序文件主要存放函数和类的实现，本例中 hellow.cpp 包含了函数 hellow()的实现文件。一般将函数原型与函数实现分别定义在头文件与源程序文件中，这两个文件的文件名

相同，只是扩展名不同，在编译时对函数实现进行编译，在连接时实现对函数的引用。对于用模板给出的参数化数据类型的函数，如果将函数体单独存放于一个源程序文件中，由于在编译这个源程序文件时还无法确定模板给出的参数化数据类型的具体类型，无法进一步编译，这样在连接时无法确定函数实现的代码，将会出现连接错误。这时只能将函数声明和实现放在同一个头文件中。

主源程序文件 main. cpp 用于调用函数 hellow()实现打印“Hellow world!”。

1.3.2 C++作用域的说明

在C++中每个变量都有起作用的范围——作用域。在函数定义中声明的变量，只能在此函数内部起作用；在类定义中声明的变量，只能在此类内部起作用——局部变量；在所有类及函数定义之外声明的变量，在源程序文件中全局有效——全局变量。如果一个全局变量在文件1中声明，要在文件2中使用，那么在文件2中必须使用关键字 extern 对此变量进行重新声明；如果在一个程序的两个文件中分别声明了两个具有相同名字的全局变量，而这两个全局变量具有不同的含意，那么需在两个文件中分别使用关键字 static 对变量进行声明。

如果一个文件中全局变量与局部变量同名，在默认情况下为局部变量，如要使用全局变量，应使用域操作符::加在全局变量的前面加以区别。

1.3.3 C++的类和对象

C++主要通过类和对象支持面向对象程序设计，类是C语言中结构体的扩充，对象就是类型为类的变量，类中不但能包含数据成员，也可包含函数成员，同时规定了对类中成员的三级访问权限：public、private 和 protected。

public 中声明的成员在程序中可对其进行直接访问，在 private 和 protected 中声明的成员能由此类的成员函数以及声明为友元（friend）的函数所访问，在 protected 中声明的成员可以被此类派生的类所访问，而在 private 中声明的成员则不能被此类派生的类所访问。下面是复数类的声明和实现，一般将类声明放在头文件中，而将类实现放在源程序文件中，在源程序文件中成员函数实现通过作用域操作符::来表示被归属于某一个类。

```
// 文件路径名:e1_2/complex.h
#ifndef _COMPLEX_H_
#define _COMPLEX_H_
class Complex
{
private:
  double RealPart;                          // 实部
  double ImagePart;                         // 虚部
public:
    Complex(double rp=0, double ip=0);      // 构造函数,构造复数,其实部和虚部分别被赋
                                            // 以参数 rp 和 ip 的值
```

```
    ~Complex() {};                          // 析构函数,复数被销毁
    double GetReal () const;                // 返回复数的实部值
    double GetImage() const;                // 返回复数的虚部值
    void PutReal (double e);                // 将实部赋为 e
    void PutImage (double e);               // 将虚部赋为 e
};
#endif
// 文件路径名:e1_2/complex.cpp
#include "complex.h"
Complex::Complex(double rp, double ip)
{  // 构造函数,用来构造复数,其实部和虚部分别被赋以参数 rp 和 ip 的值
    RealPart=rp;
    ImagePart=ip;
}
double Complex:: GetReal () const
{  // 返回复数的实部值
    return RealPart;
}
double Complex::GetImage() const
{  // 返回复数的虚部值
    return ImagePart;
}
void Complex:: PutReal (double e)
{  // 将实部赋为 e
    RealPart=e;
}
void Complex:: PutImage (double e)
{  // 将虚部赋为 e
    ImagePart=e;
}
// 文件路径名:e1_2/main.cpp
#include <iostream.h>
#include "complex.h"
void main()
{
    Complex z1(2, 3);                       // 通过构造函数自动地生成复数 z=2+3i
    cout<<"z1=" << z1.GetReal () << " + " << z1.GetImage() << "i" << endl;
    Complex z2;                             // 通过构造函数自动地生成复数 z=0+0i=0
    cout << "z2=" << z2. GetReal () << " + " << z2.GetImage() << "i" << endl;
    z2. PutReal (6);                        // 将 z2 实部赋为 6
    z2. PutImage (8);                       // 将 z2 虚部赋为 8
```