名师策划 名师主理 教改结晶 教材精品

新世纪电子信息与自动化系列课程改革教材

丛书主编 邹逢兴

数据结构

主 编 陆 勤

中国水利水电出版社
www.waterpub.com.cn

新世纪电子信息与自动化系列课程改革教材

数据结构

主 编 陆 勤

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

本书系统地阐述了基本数据结构的多种存储结构和典型算法，以及应用数据结构理论解决实际问题的基本方法和技巧，努力使读者牢固掌握数据结构的理论，培养灵活运用并巧妙解决具体问题的能力，为读者今后进一步地深入学习实践打下坚实基础。

全书内容严谨、编排合理、文字流畅、示例典型、实用性强，书中的程序均已在 Microsoft Visual C++ 6.0 系统下编译运行。全书共分 9 章。第 1 章介绍数据结构的基本概念和算法描述及分析。第 2 章至第 7 章分别介绍线性表、栈和队列、字符串、数组与特殊矩阵、树、图的多种存储结构和典型算法应用示例。第 8 章介绍了线性表的查找、查找树、哈希表查找（杂凑法）方法。第 9 章介绍了插入排序、交换排序、选择排序、二路归并排序、基数排序等多种排序算法。

本书可用作高等学校非计算机专业本科学生数据结构课程的教材。

本书配有电子教案，读者可以到中国水利水电出版社网站和万水书苑免费下载，网址为：http://www.waterpub.com.cn/softdown/和 http://www.wsbookshow.com。

图书在版编目（CIP）数据

数据结构 / 陆勤主编. —北京：中国水利水电出版社，2009

（新世纪电子信息与自动化系列课程改革教材）

ISBN 978-7-5084-6611-8

Ⅰ. 数… Ⅱ. 陆… Ⅲ. 数据结构—高等学校—教材 Ⅳ. TP311.12

中国版本图书馆 CIP 数据核字（2009）第 109883 号

策划编辑：杨庆川　责任编辑：李　炎　加工编辑：徐　雯　封面设计：李　佳

书　名	新世纪电子信息与自动化系列课程改革教材 **数据结构**
作　者	主　编　陆　勤
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（营销中心）、82562819（万水）
经　售	全国各地新华书店和相关出版物销售网点
排　版	北京万水电子信息有限公司
印　刷	北京市天竺颖华印刷厂
规　格	184mm×260mm　16 开本　17.5 印张　455 千字
版　次	2009 年 9 月第 1 版　2009 年 9 月第 1 次印刷
印　数	0001—4000 册
定　价	28.00 元

新世纪电子信息与自动化系列课程改革教材

编审委员会

新世纪电子信息与自动化系列课程改革教材

总　　序

电子信息与自动化系列课程是专业适用面很广的课程系列。随着电子信息时代的到来，特别是进入21世纪之后，我国各级各类本科院校相当多的理工科专业都或多或少地开设了该系列课程中的课程。因此，提高该系列课程的教学水平、教学质量，对于提高我国高等教育水平和质量，增强当代大学生应用先进的信息技术解决专业领域问题的能力和业务素质，具有特殊重要的意义。而教材是课程内容和课程体系的知识载体，对课程改革和建设既有龙头作用，又有推动作用，所以要提高课程教学水平和质量，关键是要有高水平、高质量的教材。

正是基于上述认识，中国水利水电出版社推动成立了“新世纪电子信息与自动化系列课程改革教材”编审委员会，在经过近两年时间的深入调查研究的基础上，策划提出了本系列教材的编写、出版计划。

本系列教材总的定位是面向各级各类高等院校的本科教学，重点是一般本科院校的教学。整个教材系列大体分为电子信息与通信、计算机基础教育和测控技术与自动化三类，共约50本主体教材，它们既自成体系，具有信息类学科的系统性、完整性，又有相对独立性。参加本系列教材编写的作者全部是一些重点大学长期从事相关课程教学的教授、副教授，大多是所在单位的学科学术带头人或学术骨干，不少还是全国知名专家教授、国家级教学名师和教育部有关“教指委”专家、国家级精品课程负责人等，他们不仅有丰富的教学经验，而且有丰富的相关领域的科研经验，对有关课程的内涵、特点、内容相关性及应用等都有较深刻的认识和切身体验。这对编写、出版好本系列教材是十分有利的条件。

本系列教材在编写时均遵循了以下指导思想：

（1）正确处理先进性和基础性的关系，努力实现两者的统一。

作为进入新世纪的新编信息类教材，既注意在原有同类教材的基础上推陈出新，努力反映学科技术的最新成就，使之具有鲜明的时代特征和先进水平，又注重符合教学规律、教学特点，突出基本原理、基本知识、基本方法和基本技术技能的阐述，着力培养学生应用基础知识分析、解决问题的创新思维能力和将来独立获取、掌握新知识，跟踪相关学科技术发展的能力。

（2）正确处理理论与实践的关系，切实贯彻理论与实践紧密结合的原则。

本系列教材绝大多数都是理论与实际结合紧密、实用性很强的课程教材，因此特别强调从应用的角度组织内容，在重视理论系统性的同时，尤其突出实践性、应用性，使学生学了以后懂得有什么用、怎么用。在教材内容阐释时，积极引入“案例”，将基本知识单元、知识点的讲解融入典型案例的解决和研究过程中，以培养学生解决工程实际问题的能力作为突破口。

（3）遵循“宽编窄用”的内容选取原则和模块化的内容组织原则。

凡教育部课程“教指委”制定了教学基本内容及要求的课程，所编教材均覆盖基本内容，

满足基本要求；其他教材的内容选取也都尽量符合多数学校和国内外同行专家的共识。在此基础上再改革创新，努力从继承与发展的结合上来准确把握（取舍）内容。模块化的内容组织主要有利于适应不同专业、不同层次、不同学时数的教学组织和安排。

（4）努力贯彻素质教育与创新教育的思想，尽量采用“问题牵引”、“任务驱动”的编写方式，融入启发式教学方法。

各知识单元尽量以实际问题、工程实例引出相关知识点，在启发学生分析、解决问题及实例的过程中，讲清原理和概念，提炼解决问题的思路和方法，着力培养学生的创新思维意识、习惯和能力，提高学生思考、分析、解决工程实际问题的素质和能力。

（5）注重内容编排的科学严谨性和文字叙述的准确生动性，务求好教好学。

在内容组织上，除条理清晰、逻辑严谨外，还尽量做到重点突出、难点分散、循序渐进，使学生易于理解。在文字叙述上，不仅概念准确、语言流畅，而且力求富有启发性、互动性、感染性、思想性，重视运用形象思维方法和通俗易懂语言，深入浅出地叙述复杂概念，说明难点问题。

（6）立足于形成立体配套的教材体系，以适应现代化教育教学方法手段的需要。

每本教材编写出版后都配套制作有 PowerPoint 电子教案，可从中国水利水电出版社网站上免费下载。大部分主教材出版后还将相继出版配套的辅助教材（包括教学辅导、习题解答、实验教程等），有的还将推出相应的多媒体教学资源库、CAI 课件和课程网站，为教师备课、教学和学生自主性、个性化学习提供更多更好的支持。

总之，本系列教材是近年来各位作者及所在学校、学科课程教学改革和科学研究成果的结晶，在内容上、体系上、模式上有一定创新。我相信，它的出版将对推动我国高校电子信息与自动化系列课程的改革发挥积极的作用。

但是，由于电子信息与自动化类学科的内涵十分丰富，课程覆盖面很广，在组织策划本系列教材时难免有挂一漏万和不妥之处，所编教材质量也未必都能如愿，恳请广大读者多提宝贵意见，以使本系列教材渐趋合理、完善。

邹逢兴

2005 年 6 月

前　言

数据结构是一门讨论“描述现实世界实体的数学模型（非数值计算）及其上的操作在计算机中如何表示和实现”的学科。随着计算机硬件、软件技术的飞速发展和计算机系统在各行业的广泛应用，有关数据结构的理论和技术也成为了计算机应用技术教育的重要部分。

数据结构是计算机技术应用方面的主要基础课程之一，它引导学生学会从实际应用问题入手，分析研究计算机加工的数据结构的特性，以便为应用所涉及的数据选择适当的逻辑结构、存储结构及其相应的操作算法，并初步掌握算法的时间和空间分析技术。另一方面，本课程的学习过程也是进行复杂程序设计、调试并排除错误的训练过程，要求学生编写的程序代码结构清晰、正确易读，具有良好的可维护性。

本书按照非计算机专业计算机课程基本要求中所规定的数据结构课程的教学内容，并参考教育部制定的计算机基础教学主要课程教学大纲编写。全书共分 9 章。第 1 章介绍数据结构的基本概念和算法描述及分析。第 2 章至第 7 章分别介绍线性表、栈和队列、字符串、数组与特殊矩阵、树、图的多种存储结构和典型算法应用示例。第 8 章介绍了线性表的查找、查找树、哈希表查找（杂凑法）方法。第 9 章介绍了插入排序、交换排序、选择排序、二路归并排序、基数排序等多种排序算法。

本书可用作高等学校非计算机专业本科学生数据结构课程的教材，旨在培养学生运用数据结构的基本概念和方法解决实际问题的能力。一般情况下，课堂讲授学时数应安排为 40～60 学时，集体上机实践时间应安排为 30 学时，可根据具体条件适当增减教学内容和学时数。多上机实践是学好本书内容的捷径，希望读者通过学习本书尽快掌握数据结构的基本应用技能。

本书由陆勤主编，特别感谢国防科学技术大学邹逢兴教授对本书的出版所给予的巨大帮助。此外，作者参阅了国内外一些有关数据结构的教材、书籍，受益匪浅。在此，谨向这些教材、书籍的作者表示感谢。

由于时间仓促及作者水平有限，书中难免存在错误或不当之处，敬请广大读者批评指正。

编　者

2009 年 5 月

目　　录

第1章

绪论

本章讨论的都是一些关于数据结构的基本概念，要求读者通过本章的学习，熟悉主要专用名词、术语的含义，了解抽象数据类型的定义、表示和实现方法，理解算法的要素和算法正确性的确切含义，掌握算法的时间复杂度、空间复杂度的估算方法。

1.1 数据结构讨论的范畴

数据结构在程序设计中占有重要地位，可以说，程序设计=算法+数据结构。程序设计的实质即为计算机处理问题编制一组“指令”，高效优质的程序设计首先需要解决两个问题：寻找简洁高效处理问题的策略——算法，设计与算法匹配良好的数据结构（data structure）。

计算机技术的迅猛发展不仅表现在计算机本身运算速度不断提高、信息存储量日益扩大、价格逐步下降，更重要的是计算机越来越广泛地应用于情报检索、企业管理、系统工程等领域，已远远超出了科学计算的范围，而渗透到了人类社会活动的众多领域。相应地，计算机的处理对象也从简单的纯数值性信息发展到非数值性的和具有一定结构的信息。

大量非数值计算问题的数学模型正是“数据结构”学科要讨论的内容。求解非数值计算问题主要考虑的是设计出合适的数据结构及相应的算法。即首先要考虑对相关的各种信息如何表示、组织和存储？计算机内的非数值运算（如表、树、图等）要依靠数据结构。同样的数据对象，用不同的数据结构来表示，运算效率可能有明显的差异。因此，简单地说，数据结构是一门讨论“描述现实世界实体的数学模型（非数值计算）及其上的操作在计算机中如何表示和实现”的学科。

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操作对象以及它们之间的关系和操作等的学科。它主要研究如何应用计算机语言按某种逻辑关系组织数据并将数据存储在计算机中，而且在其上定义运算集合。具体来说，数据结构包含三个方面的内容，即数据的逻辑结构，数据的存储结构和对数据所施加的运算（操作）。这三个方面的关系为：

（1）数据的逻辑结构独立于计算机，是数据本身所固有的。

（2）数据的存储结构是数据逻辑结构在计算机存储器中的映像，必须依赖于计算机。

（3）运算是指针对某种数据结构所施加的一组操作的总称。运算的定义直接依赖于数据的逻辑结构，但运算的实现必须依赖于数据的存储结构。

1.2 数据结构的基本概念

1.2.1 基本术语

1. 数据

数据是人们利用文字符号、数字符号以及其他规定的符号对现实世界的事物及其活动所做的描述。在计算机科学中，数据的含义非常广泛，可以把一切能够输入到计算机中并被计算机程序处理的信息，包括文字、表格、图像等，统称为数据（Data），而计算机则是加工处理数据（信息）的工具。也就是说数据是计算机程序加工的对象和计算的结果。例如，一个处理学生成绩的程序所要处理的数据可能是一张如表 1-1 所示的表格。

表 1-1 学生成绩表

学号	姓名	课程	成绩
5001	赵伟	001	94
5002	孙文强	001	88

续表

学号	姓名	课程	成绩
5003	马文林	001	78
5004	罗涛涛	001	84
5005	曾起雄	001	85
5001	赵伟	002	83
5002	孙文强	002	86
5003	马文林	002	76
5004	罗涛涛	002	74
5005	曾起雄	002	92

2. 结点

用于描述一个独立事物的名称、数量、特征、性质的一组相关信息组成一个数据结点，简称结点（node）。结点也称数据元素（data element），是组成数据的基本单位。在程序中通常把结点作为一个整体进行考虑和处理。例如，在表 1-1 所示的学生数据中，为了便于处理，把其中的每一行（代表一名学生的信息）作为一个基本单位来考虑，故该数据由 10 个结点构成。

一般情况下，一个结点中含有若干个字段（也叫数据项）。例如，在表 1-1 所示的表格数据中，每个结点都由学号、姓名、课程、成绩 4 个字段构成。字段是构成数据的最小单位。

3. 关键字

每个数据项叫做结点的一个域（field），唯一标识结点的一组域称为关键字（key）。例如，在设计处理上述学生成绩的程序时，每个学生的数据结点均包括学生的学号、姓名、课程、成绩等，学号和课程可以作为结点的关键字，用以唯一标识学生信息。

4. 逻辑结构

类型相同、内容相关的众多结点构成一个结点集合。结点集合中结点和结点之间的逻辑关系称为数据的逻辑结构。在表 1-1 所示的表格数据中，各结点之间在逻辑上有一种线性关系，它指出了 10 个结点在表中的排列顺序。根据这种线性关系，可以处理表中第 1 位学生的信息、第 2 位学生的信息……等等。

5. 存储结构

数据在计算机中的存储表示和实现称为数据的存储结构或物理结构，也称存储表示。在存储结点集合时，除了存储数据结点外，还必须体现出结点之间的关系。表 1-1 所示的表格数据在计算机中可以有多种存储表示，例如，可以表示成数组，存放在内存中；也可以表示成文件，存放在磁盘上。

用来存储一个数据结点，以及该结点与其他结点之间关系的存储单元称为存储结点。因为一个数据结点对应一个存储结点，所以，在不致混淆时，存储结点也简称为结点。尚未存储数据的存储结点称为空白结点（或空结点、自由结点）。

6. 数据处理

数据处理是指利用程序对数据进行查找、插入、删除、合并、排序、统计以及计算等操作。

数据结构重点研究各种数据通常需要进行哪些操作，如何设计完成相应的算法，从而提高程序运行效率。

对每一个数据结构而言，必定存在与它密切相关的一组操作。若操作的种类和数目不同，即使

逻辑结构相同，数据处理的结果也会不同。

不同数据结构的操作集不同，但下列操作必不可缺：

（1）结构的生成。

（2）结构的销毁。

（3）在结构中查找满足规定条件的数据元素。

（4）在结构中插入新的数据元素。

（5）删除结构中已经存在的数据元素。

（6）遍历。

1.2.2 数据结构

数据结构的形式定义为：

Data_Structure=(D,S)

数据结构是一个二元组，其中 D 是数据元素的有限集，S 是 D 上关系的有限集。数据结构应该包括数据的“逻辑结构”和“物理结构”两个方面（层次）。

1. 逻辑结构

按逻辑结构划分，数据结构可归结为以下四类：表结构（线性结构）、树形结构、图结构和集合结构。

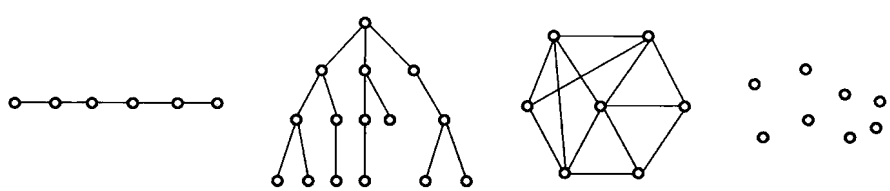
图 1-1 给出了各类数据结构的逻辑关系示意图。图中，圆圈表示结点，线条表示结点之间的逻辑关系。

表结构用于描述结点之间简单的先后次序关系（一对一的关系），比如学生成绩单。

树形结构用于描述结点之间的层次关系（一对多的关系）、分支关系和嵌套关系，比如某部门的组织机构。

图结构用于描述结点之间复杂的多对多关系，比如城市交通网。

在集合结构中，结点之间不存在任何关系（在数学上属于一种“无关关系”），仅仅是松散地聚集在集合中，比如，散列表可以存储具有这种关系的结点集合。



（a）表结构　（b）树结构　（c）图结构　（d）集合结构

图 1-1　常见的数据结构逻辑关系示意图

例 1-1　用逻辑关系示意图表示下列数据结构，并指出它们是属于线性结构还是非线性结构。

（1）S=(D, R)

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

R={(a,e), (b,c), (c,a), (e,f), (f,d)}

（2）S=(D,　R)

D={ d_i|1≤i≤5 }

$R=\{(d_i, d_j), i<j\}$

解：（1）表达式可用逻辑关系示意图表示，如图 1-2 所示，其结构为线性的。

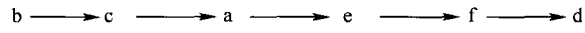


图 1-2　例 1-1 表达式（1）的逻辑关系示意图

（2）表达式可用逻辑关系示意图表示，如图 1-3 所示，其结构为非线性的。

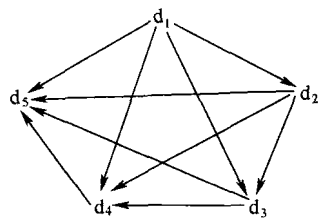


图 1-3　例 1-1 表达式（2）的逻辑关系示意图

2. *存储结构*

数据结构在计算机中的表示（或称映像）称为数据的存储结构，又称为物理结构。同一种逻辑结构可采用不同的存储方法（以上 4 种之一或组合），这主要考虑的是运算方便及算法的时空要求。

有 4 种基本的存储结构：顺序存储结构、链式存储结构、索引存储结构、散列存储结构。

（1）顺序存储结构。顺序存储结构用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，如图 1-4 所示。顺序存储结构常用于线性数据结构，将逻辑上相邻的数据元素存储在物理上相邻的存储单元里。

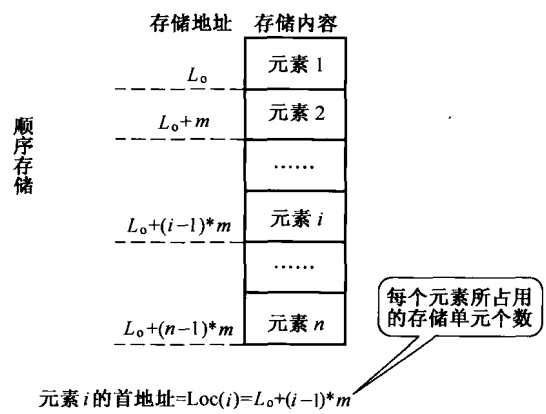


图 1-4　顺序存储结构

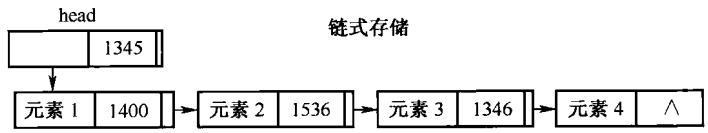
顺序存储结构有 3 个弱点：

1）执行插入或删除操作时，需移动大量元素。

2）长度变化较大时，需按最大空间分配。

3）表的容量难以扩充。

（2）链式存储结构。链式存储结构在每一个数据元素中增加一个存放地址的指针，用此指针来表示数据元素之间的逻辑关系，如图 1-5 所示。链式存储结构的每个结点都由两部分组成：数据域和指针域。数据域存放元素本身的数据，指针域存放指针。数据元素之间逻辑上的联系由指针来体现。



存储地址	存储内容	指针
1345	元素 1	1400
1346	元素 4	∧
……	……	……
1400	元素 2	1536
……	……	……
1536	元素 3	1346

图 1-5 链式存储结构

链式存储结构的特点如下：

1）比顺序存储结构的存储密度小（每个结点都由数据域和指针域组成）。

2）逻辑上相邻的结点物理上不必相邻。

3）插入、删除灵活（不必移动结点，只需改变结点中的指针）。

（3）索引存储结构。在线性结构中，设开始结点的索引号为 1，其他结点的索引号等于其前趋结点的索引号加 1，则每一个结点都有唯一的索引号，索引存储结构就是根据结点的索引号确定该结点的存储地址。

（4）散列存储结构。散列存储的思想是构造一个从集合 K 到存储区域 M 的函数 h，该函数的定义域为 K，值域为 M，K 中的每个结点 k_i 在计算机中的存储地址由 $h(k_i)$确定。

1.2.3 数据类型和抽象数据类型

数据类型是一个“值”的集合和定义在此集合上的“一组操作”的总称。

例如，所有高级语言中都有“整型”数据类型，它们各自的实现方法可以不同，但对程序员而言，它们是“相同”的。程序员使用它们时仅需了解它们的数学特性，而不需要了解它们在语言中是如何实现的。换句话说，各种语言中实现的是同一个“整数类型”，而这个“整数类型”的定义仅对“整数的数学特性”有明确规定，可称这个“整数类型”为“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此数学模型上的一组操作。

可以将矩阵的抽象数据类型定义为一个由 *m*n* 个数排成的 *m* 行 *n* 列的表，它可以进行初等变换、相加、相乘、求逆……等运算。

抽象数据类型有两个重要特性：

（1）数据抽象。用 ADT 描述程序处理的实体时，强调的是其本质的特征、所能完成的功能以及它和外部用户的接口（即外界使用它的方法）。

（2）数据封装。将实体的外部特性和其内部实现细节分离，并且对外部用户隐藏其内部实现细节。

抽象数据类型的描述形式为：

ADT=(D,S,P)

其中：

D 是数据对象；S 是 D 上的关系集；P 是 D 的基本操作集。

定义抽象数据类型的描述形式为：

ADT 抽象数据类型名 {

数据对象：数据对象的定义

数据关系：数据关系的定义

基本操作：基本操作的定义

} ADT 抽象数据类型名

数据对象和数据关系的定义用伪码描述。

基本操作的定义格式为：

基本操作名（参数表）

初始条件：<初始条件描述>

操作结果：<操作结果描述>

基本操作有两种参数：赋值参数只为操作提供输入值；引用参数以&开头，除可提供输入值外，还将返回操作结果。

“初始条件”描述了操作执行之前数据结构和参数应满足的条件，若不满足，则操作失败，并返回相应出错信息。

“操作结果”说明了操作正常完成之后，数据结构的变化状况和应返回的结果。

若初始条件为空，则可省略。

例 1-2　抽象数据类型“复数”的定义为：

ADT Complex {

数据对象：D={ e_1,e_2 | e_1,e_2 ∈ RealSet }

数据关系：R={ <e_1,e_2> | e_1 是复数的实部，e_2 是复数的虚部 }

基本操作：

InitComplex(&Z, V_1, V_2)

操作结果：构造复数 Z，其实部和虚部分别被赋以参数 V_1 和 V_2 的值。

DestroyComplex(&Z)

初始条件：复数已存在。

操作结果：复数 Z 被销毁。

GetReal(Z,&realPart)

初始条件：复数已存在。

操作结果：用 realPart 返回复数 Z 的实部值。

GetImag(Z,&ImagPart)

初始条件：复数已存在。

操作结果：用 ImagPart 返回复数 Z 的虚部值。

Add(Z_1,Z_2, &sum)

初始条件：Z_1、Z_2 是复数。

操作结果：用 sum 返回两个复数 Z_1、Z_2 的和。

} ADT Complex

抽象数据类型需要通过高级语言编程实现，本书采用 C++实现 ADT。

例 1-3　用 C++描述复数抽象数据类型 complex 的实现，以执行有关复数和、差、积、输出等操作。

解：在类定义中包含构造函数，每当创建一个复数对象时，通过该构造函数对复数对象初始化，即令其实部与虚部均为 0。

```
#include<iostream>
using namespace std;
class complex
{public:
complex(int=0,int=0);
void setcomp(int,int);
void addcomp(complex,complex);
void subcomp(complex,complex);
void mulcomp(complex,complex);
void printcomp( );
private:
int real;
int imag;};

complex::complex(int re,int im)
{setcomp(re,im);}

void complex::setcomp(int re,int im)
{real=re;
imag=im;}

void complex::addcomp(complex z1,complex z2)
{real=z1.real+z2.real;
imag=z1.imag+z2.imag;}

void complex::subcomp(complex z1,complex z2)
{real=z1.real-z2.real;
imag=z1.imag-z2.imag;}

void complex::mulcomp(complex z1,complex z2)
{real=z1.real*z2.real-z1.imag*z2.imag;
imag=z1.real*z2.imag+z2.real*z1.imag;}

void complex::printcomp( )
{cout<<"("<<real<<","<<imag<<")";}

void main( )
{complex z1,z2(5),z;
z.addcomp(z1,z2);
z1.printcomp( );
cout<<"+";
z2.printcomp( );
cout<<"=";
z.printcomp( );
cout<<endl;
z1.setcomp(8,9);
z2.setcomp(2,10);
z.subcomp(z1,z2);
z1.printcomp( );
cout<<"-";
z2.printcomp( );
cout<<"=";
z.printcomp( );
```