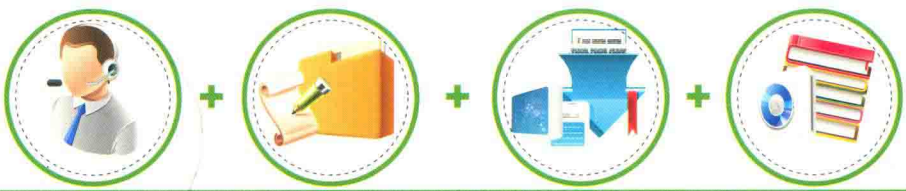
全国高等院校应用型**创新规划**教材 计算机系列

数据结构

（C语言版）

温永刚 王琬茹 王向华 主 编



清华大学出版社

全国高等院校应用型创新规划教材·计算机系列

数据结构(C 语言版)

温永刚　王琬茹　王向华　主编

清华大学出版社
北 京

内 容 简 介

本书是作者根据多年教学的经验，并参考了近几年出版的国内外大学多种数据结构教材和书籍编写而成的。本书内容可以分为三个部分：第 1 部分包含第 1 章，对数据结构进行概要性说明；第 2 部分包含第 2 章至第 6 章，具体介绍线性表、堆栈、队列、串、数组、矩阵、广义表、二叉树、树和森林、图等内容；第 3 部分包含第 7 章和第 8 章，介绍各种数据的查找和排序方法。本书不仅内容广泛、涵盖的知识点全面，而且条理清晰、通俗易懂、图文并茂，有利于学生进行系统性的学习。

本书可以作为高等院校计算机及相关专业本、专科生“数据结构”课程的教材，也可作为从事各种程序设计和计算机应用工作人员的参考书。

图书在版编目(CIP)数据

数据结构(C 语言版)/温永刚，王琬茹，王向华主编. —北京：清华大学出版社，2019
(全国高等院校应用型创新规划教材 • 计算机系列)
ISBN 978-7-302-52901-9

Ⅰ. ①数… Ⅱ. ①温… ②王… ③王… Ⅲ. ①数据结构—高等学校—教材 ②C 语言—程序设计—高等学校—教材 Ⅳ. ①TP311.12 ②TP312.8

中国版本图书馆 CIP 数据核字(2019)第 083523 号

责任编辑：汤涌涛
封面设计：杨玉兰
责任校对：吴春华
责任印制：刘海龙
出版发行：清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn, 010-62791865
印 装 者：三河市君旺印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　印　张：19.25　　字　数：468 千字
版　　次：2019 年 7 月第 1 版　　印　次：2019 年 7 月第 1 次印刷
定　　价：49.00 元

产品编号：067258-01

前　言

当用计算机来解决实际问题时，就要涉及数据的表示和数据的处理，而数据表示和数据处理正是数据结构课程的主要研究对象。学生通过这两方面的学习，可为学习后续课程特别是软件方面的课程打下坚实的基础，并得到必要的技能训练。因此数据结构课程是计算机或信息类等相关专业的一门重要的专业基础课程，对专业的学习起着举足轻重的作用，而数据结构也往往是研究生入学考试的必选科目之一。

本书根据应用型高等院校“计算机应用技术”专业的“数据结构”课程教学大纲编写，在教学内容上结合了应用型高等院校的实际教学情况。针对数据结构课程理论性强以及应用实践性较为突出的特点，本书在整个编写过程中做了以下几个方面的努力。

(1) 在内容上基本满足了研究生考试中对数据结构课程提出的要求。

(2) 在引入新概念或讲述新算法的过程中，尽可能使用简洁、清晰的语言进行表达，通过尽可能多的示例对教学内容进行直观的解释和说明，以便能够帮助读者正确地理解。

(3) 对于书中涉及的相关算法，尽可能地从“算法描述”“算法分析”和“算法讨论”三个方面进行全方位的讲述。

(4) 每章的后面，都附有大量的习题，以供教师备课及学生复习所学知识。

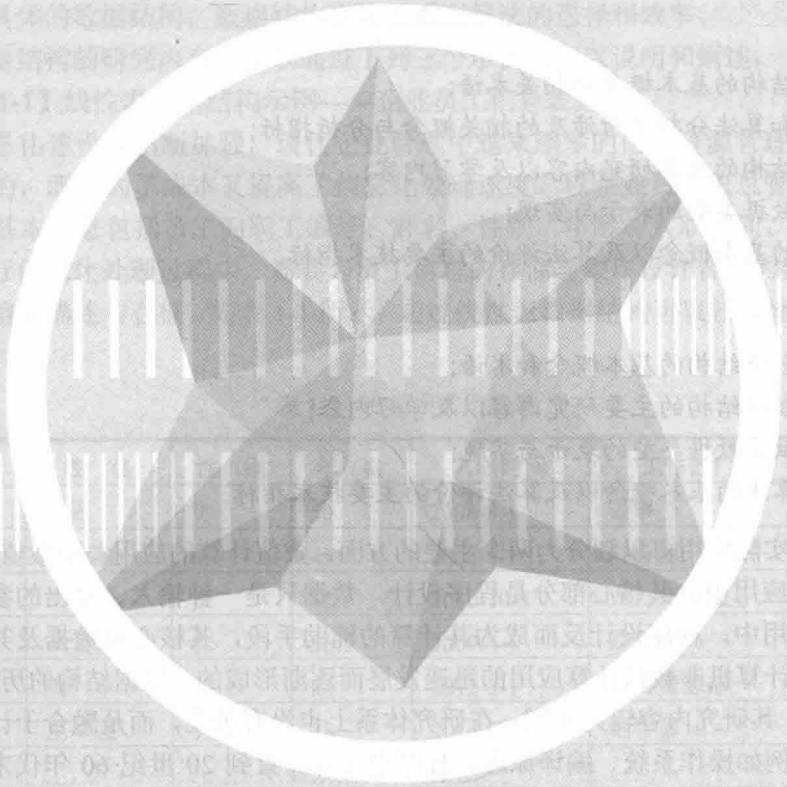
本书的完成，是基于作者多年教学经验的总结，更是参与写作的每一位同仁齐心协力的成果。本书由温永刚、王琬茹、王向华担任主编，本书第 1 章至第 3 章由王琬茹完成，第 4 章至第 7 章由温永刚完成，第 8 章由王向华完成。全书由温永刚和王向华总纂定稿。

本书在编写过程中参考了大量的相关著作、网络资料、教材和文献，吸取和借鉴了同行的相关成果，在此谨向有关作者表示诚挚的谢意和敬意!

限于编者水平，书中难免有不妥和疏漏之处，敬请读者批评指正。

编　者

目　录



第1章

绪　论

本章要点

(1) 数据结构的基本概念和相关术语;
(2) 算法和算法分析中所涉及的相关概念与分析指标;
(3) 数据结构的主要研究内容以及学习内容;
(4) 抽象数据类型的表示与实现;
(5) 算法的基本概念以及算法评价的主要技术指标。

学习目标

(1) 理解数据结构的基本概念和术语;
(2) 了解数据结构的主要研究内容以及学习内容;
(3) 理解抽象数据类型的表示与实现;
(4) 掌握算法的基本概念以及算法评价的主要技术指标。

计算机的实际应用可以划分为两个主要的方面：数值计算的应用与非数值计算的应用。在数值计算的应用中，其核心部分是程序设计，数据只是一种输入、输出的参数。而在非数值计算的应用中，程序设计反而成为其计算的辅助手段，其核心是数据及其结构。数据结构正是随着计算机非数值计算应用的迅速发展而逐渐形成的。数据结构的历史始于 20 世纪 60 年代初，其研究内容较为模糊，在研究体系上也没有独立，而是融合于计算机科学的其他学科中，例如操作系统、编译原理、程序设计等。直到 20 世纪 60 年代末，《计算机程序设计技巧》的问世才为数据结构奠定了基础，较为全面系统地阐述了数据结构的研究对象，明确定义了不同类型的数据结构，数据结构作为一门独立的计算机学科才真正地发展起来，并开始转入方法体系。而进入新的历史发展时期，随着计算机应用的广泛深入，数据结构作为非数值计算应用的重要辅助工具，其地位与作用也日益明显和突出。本章主要介绍数据结构的相关概念与算法分析方法，是后序章节学习的基础。

1.1 数据结构的研究内容

计算机作为一种数据计算工具，其应用范围已经深入到人类社会的各个方面，既有纯粹的数值计算应用，也涉及越来越多的非数值计算应用。在非数值计算应用过程中，计算机解决实际问题时通常包含四个基本阶段。

(1) 分析问题阶段：对具体问题进行输入输出的边界界定，抽取问题的实质信息，对问题进行抽象，进而形成适当的数学模型。

(2) 数据结构设计阶段：设计合适的数据结构对象，实现解决问题所需的数据及数据之间的关系的描述与存储。

(3) 算法设计阶段：根据问题要求和数据结构的特点来选择和设计算法，同时要考虑算法的效率和占用的内存空间。

(4) 程序设计阶段：对设计的算法进行编程实现，最终开发调试出解决实际问题的应用程序软件。

在这四个阶段中，数据结构设计阶段与算法设计阶段之间的关系尤为密切相关，算法

无不依附于具体的数据结构，数据结构直接关系到算法的选择和效率。

对于数据结构的研究内容，可以通过下列三个示例来加以说明和阐述。

【实例 1-1】线性表数据结构示例——企业员工信息表。

随着信息化建设的不断加强，现代企业管理中越来越多的信息资源管理是通过电子化手段来完成的，既节约了成本又提高了信息化处理速度。以企业员工信息管理为例，企业员工的相关基本信息包括员工的职工编号、姓名、性别、年龄、入职时间、工作部门、职务等。在进行信息化处理过程中，需要为员工信息的存储与操作设计合适的数据结构对象，实现相关的信息描述与存储。如表 1-1 所示，通过线性表数据结构可以较好地描述和存储员工信息。

表 1-1　企业员工信息表

职工编号	姓　名	性　别	年　龄	入职时间	工作部门	职　务
A10010	李光	男	50	2008-8-1	董事会	董事长
B10011	赵小渝	男	45	2008-6-1	财务部	财务总监
B10021	胡春婷	女	42	2010-10-1	人力资源部	人力总监
C10011	刘爱云	女	36	2009-4-1	财务部	员工
C10021	周一卓	男	34	2012-3-1	人力资源部	员工
C10031	王华强	男	28	2014-8-1	销售部	员工
……	……	……	……	……	……	……

通过该线性表，对企业中所有员工的信息依次进行了存放，较好地解决了数据的描述与存储。同时通过对该线性表进行相关的操作，例如查找、插入与删除等操作，即可以实现对企业员工的相关数据处理。

该线性表数据结构被广泛应用于图书馆图书信息管理、仓储信息管理、学生信息管理等诸如此类的数据管理应用中。在这类应用中，信息管理与数据操作的数据结构对象都是这种结构简单的线性表。

【实例 1-2】树状数据结构示例——家谱信息。

家谱用于记录某家族历代家族成员的情况与关系，其管理主要是实现对家庭成员的登记、查询以及维护。在使用计算机解决这一问题时，对其数据结构的设计一方面要求家谱信息的描述与存储要简洁直观，另一方面则要求可以较好地支持对家谱的存储、更新、查询、统计等操作。在具体的设计与实现上，可以采用如图 1-1 所示的树状数据结构进行描述与存储。

在本例中为了便于描述，假定只考虑家庭中的父子关系。在这个数据结构中，每个数据元素成员代表的是每个家庭成员，数据元素使用记录来表示，每个记录中包含所代表的家庭成员的姓名、出生日期、性别、死亡日期等。数据元素之间的关系是父子关系，用无向连线来表示，位于某数据元素下方的与其相连的各个数据元素，表示该家庭成员的子女。这样，一个家族中的各个家庭成员信息通过连线就构成了一个层次结构，这种分层的数据结构称为树状数据结构。

用这种数据结构存储家庭成员信息的同时，也可以方便地实现相关的数据操作，例如

家庭成员的查找(个人查找、前辈查找、后代查找等)，家庭成员的添加、删除以及修改等操作。除了确定表示方式外，数据结构的任务还包括对这棵树的计算机物理存储、操作的抽象化。其中操作的抽象化旨在建立存取/访问树结构的基本计算机程序，以支持其他各种操作的实现。

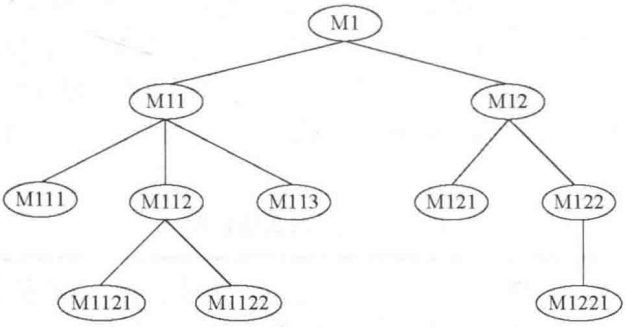
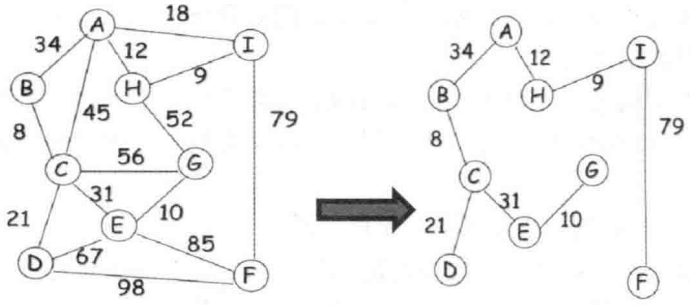


图 1-1　家谱关系的树状数据结构表示

【实例 1-3】网状数据结构示例——煤气管道铺设。

在小区的煤气管道建设过程中，n 幢居民楼之间需要铺设 n−1 条管道线路，居民楼之间的管道铺设可以存在多种方案，因此需要施工人员进行合理的规划，使铺设的管道长度最短。在解决这一实际问题过程中，其数据结构的设计与前面的线性表结构和树状结构都有着较大的区别，在这里引入了图状数据结构，将居民楼抽象为其中的若干个结点(node，也称节点，以下统称为结点)，而将管道抽象为连接这些结点的无向边，如图 1-2 所示。



(a) 图状数据结构　　(b) 图的最小生成树

图 1-2　居民楼煤气管道架设示意图

在图 1-2(a)中利用图状数据结构描述出了居民楼之间的所有管道的架设方案，它代表的是一种网状关联。而将管道架设问题可以抽象转化为图结构的最小生成树问题，利用典型的求解算法即可得到管道架设的最优解决方案，如图 1-2(b)所示的最终管道架设方案。

以上所举的三个例子解决的都不是数值计算问题，非数值计算问题所使用的数学模型不再是传统的数学方程，而诸如线性表、树和图的数据结构，则正是数据结构这门学科需要讨论的问题。因此，数据结构可以理解为一门讨论“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。

值得注意的是，“数据结构”一直伴随着计算机科学的发展而不断的发展。一方面，

面向各专门领域中特殊问题的数据结构得到研究和发展，如多维树结构在遗传生物学中的应用；另一方面，随着面向对象技术的不断发展，面向对象程序设计将成为程序设计方法中的主流技术，而作为程序设计与软件技术基础的数据结构，它的方向与内容自然随着面向对象技术相应地进行调整，以保证它仍是程序设计的基础。

1.2　基本概念和术语

本节的主要内容是解释数据结构中常用的相关概念和术语的含义，以便让读者能更好地学习后续章节中的内容。

1.2.1　数据、数据元素、数据项和数据对象

数据(Data)：数据是描述客观事物信息的符号，是计算机系统可加工处理的对象。如数学计算中用的整数和实数，文本处理中用的字符串以及多媒体处理中用的音频、视频、图像等。

数据元素(Data Element)：能独立、完整地描述问题世界中实体的最小数据单位称为数据元素，也称为元素、记录。数据元素可用于完整地描述一个对象，在计算机程序处理中通常作为一个整体进行考虑与处理。例如表 1-1 中的一行记录(A10010, 李光, 男, 50, 2008-8-1, 董事会, 董事长)，代表一个具体企业员工的相关信息。

数据项(Data Item)：构成数据元素的不可分割的数据单位，具有独立的含义，也称为字段。例如表 1-1 中的职工编号、姓名、性别、年龄、入职时间、工作部门、职务都是数据项。

数据、数据元素和数据项三者之间存在着包含关系，即数据由数据元素组成，数据元素由若干数据项组成。

数据对象(Data Object)：同类数据元素的集合称为数据对象，是数据的一个子集。例如，偶数数据对象是集合 B= {0, 2, 4, 6, …}，汉字数据对象是所有汉字的集合，实例 1-1 中的表 1-1 企业员工信息表也可以看作是由若干同类企业员工信息聚集而形成的数据对象。只要集合内元素的性质相同即可以称为一个数据对象。

1.2.2　数据结构

下面给出数据结构的定义。

数据结构(Data Structure)：数据元素之间的关系称为结构，而相互之间存在着一定关系的数据元素的集合及定义在其上的基本操作(运算)称为数据结构。利用集合论的方式也可以将数据结构定义为一个二元组(D, S)，其中 D(Data)是数据元素的有限集，S(Structure)是 D 上的关系的有限集。

因此，数据结构研究的是客观事物个体属性在计算机中表达及描述的方法，学习数据结构的内容主要包含三个基本方面。

(1) 数据对象的逻辑结构：数据对象中各数据元素之间的逻辑关系，例如线性表、树、图等。

(2) 数据对象的物理结构：在对数据对象进行访问和处理时，各个数据元素在计算机

物理介质中的实际存储方式，例如顺序存储结构与链式存储结构等。

(3) 数据对象中数据元素的运算操作：例如常用的数据元素检索、排序、插入、删除、修改等。

图 1-3 给出了几种基本的数据结构形式。要设计应用于计算机处理的数据结构形式，上述的定义必须联系于计算机的物理实现才有实际意义。

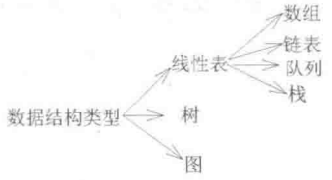


图 1-3 基本的数据逻辑结构类型

数据结构在计算机内存中的表示方法，我们称为数据结构的物理结构，以区别于前者的逻辑结构形式。物理结构有四种基本的形式，如图 1-4 所示。其中，索引结构用于文件操作，散列结构是对数据检索时采用的一种形式。

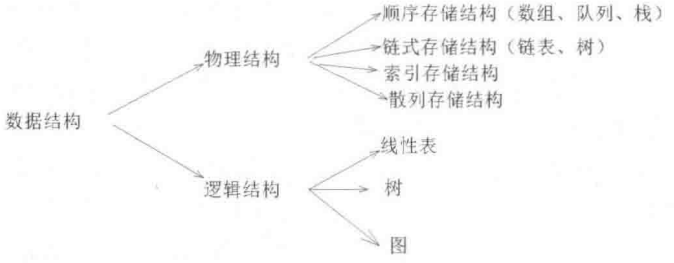


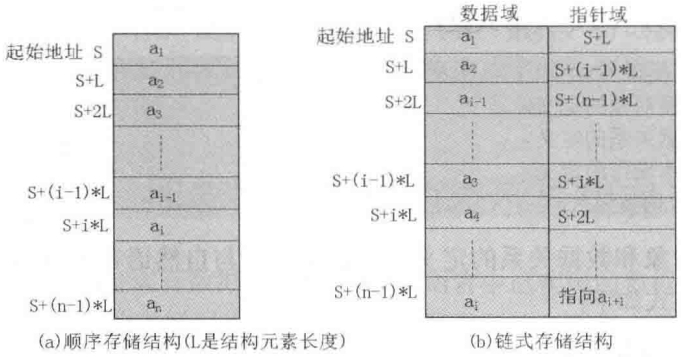
图 1-4 基本的数据物理结构类型

所谓顺序存储结构，是指将数据元素顺序地存放于计算机内存中一个连续的存储区域里，借助元素在存储器中的相对位置来表示元素之间的逻辑关系，也就是用数组描述的一群有限数据元素集合。

链式存储结构的特点，是在每个元素中加入一个指针域，它指向逻辑上相邻接的元素的存放地址，而数据元素在内存中的存放顺序与逻辑关系无关。即链式存储结构是用指针的指向来表达结点的逻辑关系，这也就是 C、Pascal 适用于数据结构设计的原因。图 1-5 给出了顺序、链式存储结构示意。它们都是描述，或者说存储了线性关系$<a_i, a_{i+1}>$，但方式不同。

数据结构有线性与非线性之分。一个数据结构的关系里，除去端点外，每一个结点有且仅有一个前驱和后继时，这个数据结构就是线性的，如数组、链表。如果数据结构关系中，其结点有一个以上的前驱或后继，则称为非线性的数据结构，如树、图。一般情况下，我们讨论非空有限集合 D 上只有单一关系 r 的数据结构。但是，在关系数据库设计时，讨论的则是非空有限集合 D 上的一组关系 R 的数据结构设计问题。

数据结构的物理表达问题，在有关参考书中已经明确给出，读者可以仔细阅读理解，对 C 语言不熟悉的读者要尤其注意指针和链式存储结构。



(a)顺序存储结构(L是结构元素长度)　　(b)链式存储结构

图 1-5　向量的顺序存储结构与链式存储结构

在结束有关数据结构的概念讨论之前，我们再次明确地给出数据结构内容的三要素是：

数据结构=数据逻辑结构+物理结构+数据运算

数据运算是指对数据结构的检索、插入、排序、删除、更新等操作。此外，不同数据结构之间的运算效率也是我们要重点考虑的内容。

1.2.3　数据类型和抽象数据类型

1. 数据类型

数据类型(Data Type)是程序设计中与数据结构密切相关的一个概念，它指的是一个值的集合和定义在该值集上的一组操作的总称。例如程序设计语言中常用的整型变量，它涵盖两个基本范畴，一方面它是所用整数的集合，另一方面则是定义了在整数集合上的相关操作，如加、减、乘、除等运算。

用户可以使用的数据类型是由具体的程序设计语言进行定义并规范的，程序设计语言支持的数据类型反映了该设计语言的数据描述与数据处理能力。C 语言中除了支持常用的整型、实型、字符型等基本数据类型外，还允许用户创建自定义数据类型，例如数组类型、结构体类型等。

2. 抽象数据类型

当程序设计语言中所支持的数据类型在解决实际问题时无法正确、全面地进行数据描述、存储或处理时，用户可以使用现有的数据类型抽象构造出解决实际问题的高级数据类型，即抽象数据类型。

抽象数据类型(Abstract Data Type，ADT)：是一种更高层次的数据抽象，它是由用户定义，用以表示应用问题的数据模型，它由基本的数据类型组成，并包括一组在该模型上的相关操作。例如，用户可以使用整型、实型、字符型等 C 语言中已有的数据类型构造出线性表、栈、队列、树、图等更为复杂的抽象数据类型。

在具体组成上，可以将抽象数据类型划分为三个基本部分，分别是数据对象、数据对象上关系的集合以及数据对象基本操作集合。

抽象数据类型的一般定义形式是：

```
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>
```

其中，数据对象和数据关系的定义采用数学符号与自然语言进行描述，对于其中的基本操作，其定义格式如下：

```
<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>
```

相关说明：

(1) 在基本操作的定义中有两种参数：赋值参数与引用参数。赋值参数主要用于为操作数赋值，而引用参数除了可以赋值以外还可以实现操作结果返回，并在参数名前添加“&”符号以示区别。

(2) 定义中的“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，如果不满足条件，则操作失败并返回相应提示信息。

(3) 定义中的“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。

【实例 1-4】一个复数抽象数据类型的定义部分。

```
ADT Complex {
数据对象：D={c1, c2, c1, c2∈R }
数据关系：R={<e1, e2> | c1 是复数的实数部分，c2 是复数的虚数部分 }
基本操作：
AssignComplex( &Z, x, y )
操作结果：构造复数 Z,其实部和虚部,分别被赋予参数 x 和 y 的值。
DestroyComplex( &Z)
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 GetReal 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 GetImag 返回复数 Z 的虚部值。
Add( z1,z2,&sum )
初始条件：z1, z2 是复数。
操作结果：用 sum 返回两个复数 z1,z2 的和。
} ADT Complex
```

有关抽象数据类型的相关说明：

(1) 抽象数据类型和数据类型本质上是一个概念。只是抽象数据类型所覆盖的范畴更广，它除了包含系统已定义并实现的数据类型之外，还包括用户自己定义的数据类型。

(2) 抽象数据类型最重要的特点是抽象和信息隐蔽。抽象的本质就是抽取反映问题本

质的东西，忽略非本质的细节，使所设计的结构更具有一般性，可以解决一类问题。信息隐蔽就是对用户隐藏数据存储和操作实现的细节，使用户了解抽象操作或界面服务，通过界面中的服务来访问这些数据。

1.3　抽象数据类型的表示与实现

抽象数据类型需要通过固有数据类型(高级编程语言中已实现的数据类型)来实现。由于我们在高级程序设计语言的虚拟层次上讨论抽象数据类型的表现和实现，并且讨论的数据结构及其算法主要是便于理解，故采用伪码和 C 语言之间的类 C 语言作为描述的工具，有时也采用伪码描述一些只含抽象操作的抽象算法。这使得数据结构和算法的描述和讨论简明清晰，不拘泥于 C 语言的细节，又能容易转换成 C 或 C++程序。

本书采用的类 C 语言精选了 C 语言的一个核心子集，同时结合实际的教学内容进行了相关的扩充修改，增强了算法描述功能，其核心的语法内容介绍如下。

(1) 预定义常量和类型：

```
//函数结果状态代码
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define INFEASIBLE -1
#define OVERFLOW -2
//Status 是函数的类型，其值是函数结果状态代码
Typedef int Status;
```

(2) 数据结构的表示(存储结构)用类型定义(typedef)描述。数据元素类型约定为 ElemType，由用户在使用该数据类型时自行定义。

(3) 基本操作的算法都用以下形式的函数描述：

```
函数类型　函数名(函数参数表){
    //算法说明
    语句序列
} //函数名
```

除了函数的参数需要说明类型外，算法中使用的辅助变量可以不作变量说明，必要时对其作用给予注释。当函数返回值为函数结果状态码时，函数定义为 Status 类型。为了便于描述算法，除了值调用方式外，还增加了 C++语言引用调用的参数传递方式，在形参表中，以&开头的参数即为引用参数。

(4) 赋值语句：

```
简单赋值　变量名=表达式;
串联赋值　变量名 1=变量名　2=…=变量名　k=表达式;
成组赋值　(变量名 1,…,变量名 k)=(表达式 1,…,表达式 k)
结构名=结构名;
结构名=(值 1,…,值 k) ;
变量名[ ]=表达式;
```

```
变量名[起始下标…终止下标]=变量名[起始下标…终止下标];
交换赋值   变量名↔变量名;
条件赋值   变量名=条件表达式? 表达式 T: 表达式 F;
```

(5) 选择语句:

```
条件语句 1   if(表达式)语句;
条件语句 2   if(表达式)语句;
             else 语句;
开关语句 1   switch(表达式){
               case 值 1: 语句序列 1; break;
                    ⋮
               case 值 n: 语句序列 n; break;
               default: 语句序列 n+1;
           }
开关语句 2   switch{
               case 条件 1: 语句序列 1; break;
                    ⋮
               case 条件 n: 语句序列 n: break;
               default: 语句序列 n+1;
           }
```

(6) 循环语句:

```
for 语句          for(赋初值表达式序列;条件;修改表达式序列)语句;
while 语句        while(条件)语句;
do-while 语句     do{
                     语句序列;
                  }while(条件);
```

(7) 结束语句:

```
函数结束语句        return 表达式;
                    return;
case 结束语句       break;
异常结束语句        exit(异常代码);
```

(8) 输入和输出语句:

```
输入语句  scanf([格式串],变量 1,…,变量 n);
输出语句  printf([格式串],表达式 1,…,表达式 n);
```

(9) 注释:

```
单行注释   // 文字序列
```

(10) 基本函数:

```
求最大值        max(表达式 1,…,表达式 n)
求最小值        min(表达式 1,…,表达式 n)
求绝对值        abs(表达式)
判定文件结束    eof (文件变量)或 eof
判定行结束      eoln(文件变量)或 eoln
```