用易学的 Python 语言
描述复杂的数据结构

DATA STRUCTURE
IN PYTHON

数据结构
——Python语言描述

张光河　主编

不仅给出基于 **Python** 实现的**算法代码**，而且提供
与之对应并可**独立运行**的 Python 程序

提供**基础实验**、**综合实验**和**习题**，帮助读者
检测是否真正**掌握**知识点

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

语言

描述复杂的数据结构

DATA STRUCTURE
IN PYTHON

数据结构
——Python语言描述

张光河　主编

人民邮电出版社
北京

图书在版编目（CIP）数据

数据结构 : Python语言描述 / 张光河主编. -- 北京 : 人民邮电出版社, 2018.8
ISBN 978-7-115-48577-9

Ⅰ. ①数… Ⅱ. ①张… Ⅲ. ①数据结构②软件工具－程序设计 Ⅳ. ①TP311.12②TP311.561

中国版本图书馆CIP数据核字(2018)第117581号

内 容 提 要

Python 是目前流行的程序设计语言，国内高校已陆续使用。本书根据普通高等院校计算机专业本科生的教学需求，并按照数据结构课程教学大纲的规定，同时在参考兄弟院校使用的经典教材和教案的基础上，基于 Python 作为描述语言编写而成。

本书介绍了线性表、栈、队列、串、树和图等基本数据结构，以及这些数据结构的相关应用，还介绍了查找和排序的常用算法。本书在介绍内容时，理论和实践并重，而且配有一定数量的上机实验和习题，帮助读者加深对相关知识点的理解。

本书内容重点突出，语言精练易懂，可作为普通高等院校计算机及相关专业数据结构课程的教材，也可供计算机及相关专业的教学人员、科研人员、数据结构或算法的爱好者使用。高职高专类学校选用本书时可以根据学校和学生的实际情况略去某些章节。

◆ 主　　编　张光河
责任编辑　刘　博
责任印制　沈　蓉　彭志环

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京圣夫亚美印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：28.75　　2018 年 8 月第 1 版
字数：781 千字　　2018 年 8 月北京第 1 次印刷

定价：69.80 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315

前言 PREFACE

数据结构在计算机及相关专业中的地位是不言而喻的，学好这门课对学生日后在 IT 行业工作大有裨益。Python 具有足够的抽象性，非常适合描述数据结构，因此，基于 Python 来讨论数据结构中的基本问题是一件值得尝试的事情，国外不少大学已经将其作为代替 C 语言的教学语言，国内部分学校也在尝试调整教学计划，以适应这一趋势。

在教学中，我注意到以下情况：学生使用基于伪语言描述的数据结构教材时，常常难于将其用自己熟悉的语言实现，从而使学习激情湮灭；而在使用基于 C 语言描述的数据结构教材时，又因为对 C 语言中指针和结构体掌握不好，无法理解教材提供的源程序，也无法调试和运行这些程序，更谈不上自己用 C 语言将其实现；在使用基于 C++和 Java 语言描述的数据结构教材时，学生感觉难度更大。

本书共 9 章：第 1 章介绍数据结构的概念和相关术语，并对算法的性能评价进行简要说明；第 2 章介绍线性表及应用；第 3 章重点介绍栈和队列，并介绍如何利用栈来将递归算法转换为非递归形式；第 4 章重点介绍串，并简要介绍了数组和广义表的存储；第 5 章首先介绍树，然后重点介绍二叉树的定义及相关操作，最后再引出森林的概念；第 6 章介绍图及其典型应用，如最短路径和关键路径；第 7 章介绍查找，包括基于静态查找表的查找和基于动态查找表的查找；第 8 章和第 9 章分别介绍了内排序和外排序，对一些常用的排序算法进行了详细的描述，并进行了效率分析。本书的特点如下。

（1）每章除了相应的知识内容之外，还包括基础实验、综合实验和习题。现在市面上有些教材偏重于理论的讲解，对于实验教学这一方面，通常由教师自行解决。本书为每一章提供了一定数量的基础实验和综合实验，用于解决在实验教学时教师和学生“无米下锅”的尴尬，同样，适量的习题也有助于学生检测自己是否真正掌握了某一知识点。

（2）本书给出了基于 Python 实现的算法代码，还有与之对应并配套的、可独立运行的 Python 程序，这是本书的一大特色。以在单链表中插入一个数据元素为例，在书中有使用 Python 实现的在单链表的首端插入一个数据元素和在尾端插入一个数据元素的源代码，而配套的源程序中有单链表结点的类定义，单链表的类定义，创建一个单链表源代码，插入结点前和插入结点后遍历该单链表的源程序。

通过这样的设计，教师在课堂教学时可专注于讲解算法的重点和难点，学生即使完全不会编写 Python 程序，也可以通过阅读这些源代码在课后自行学习，这对培养学生的自学能力十

分有益。尤其是当学生的水平参差不齐时，由于课堂教学时间有限，教师可以更好地安排学生课余学习。

计算机及相关专业属于工科，其中绝大部分课程都需要学生动手编写大量程序并上机调试成功后，才能理解其对应的理论知识，数据结构也不例外。Python 以简洁、优美和容易使用著称，网上也有大量可用的资源，可以帮助学生更好地理解本书。

感谢在本书编写过程中给予过支持和帮助的侯静、惠敏、蔡云戈、李鑫勇和胡妍等同学！感谢在成书过程中家人所给予的支持和帮助！

作者在编写本书的过程中，参阅了大量相关教材和专著，也在网上查找了很多资料，在此向各位原著者致敬和致谢！

由于作者水平有限，加上时间仓促，书中难免存在不妥或错误，恳请读者批评指正！

作者邮箱：guanghezhang@163.com。

作者

2018 年 1 月

目 录 CONTENTS

01 第1章 绪论

随着计算机的广泛应用，无论是网上购物，还是网上订餐，或是网上购票，均需要使用计算机程序，人们的衣、食、住、行均与其密不可分。计算机由早期主要处理科学计算中的数值型数据发展到现在处理各种非数值型数据，如何为这些数据选择合理高效的数据结构，是程序设计人员无法回避的问题。

“数据结构”是计算机及相关专业最为重要的基础课程之一，学习并掌握这一课程中涉及的知识是非常有必要的，对后续学习和理解计算机专业其他课程也有所帮助。

1.1 数据结构概述

数据结构是指所有数据及这些数据之间的关系的集合。对于计算机而言，数据是指能被输入到计算机中并能被其处理的符号的集合。在使用计算机解决科学计算问题时，通常按以下步骤进行。

（1）分析问题，确定数学模型。

（2）根据模型设计相应的算法。

（3）选择合适的编程语言实现算法。

（4）调试程序，直到正确解决问题。

但对于设计类似网上订餐和网上购物的程序，其中有很多问题是很难找到与之对应的数学模型的。这时，第一步需分析程序中所要处理的数据，第二步需根据实际应用判断这些数据之间存在的逻辑关系，第三步需结合在实际应用中操作这些数据的频度来确定其整体的组织结构。通过以上 3 步便能够确定数据的逻辑结构。最后通过采取一定地策略将数据的逻辑结构在计算机中表示出来，从而对数据进行一系列的操作。

1.1.1 什么是数据结构

数据结构的概念最早由 C.A.R.Hoare 和 N.Wirth 在 1966 年提出，大量关于程序设计理论的研究表明：想要对大型复杂程序的构造进行系统而科学的研究，必须首先对这些程序中所包含的数据结构进行深入的研究。

本小节将对数据结构中的一些基本概念和术语加以定义和解释，以便于读者在后续章节中更好地学习。

数据通常用于描述客观事物，例如，在日常生活中使用的各种文字、数字和特定符号都是数据。而在计算机中，数据是指所有能够输入到计算机中存储并被计算机程序处理的符号的集合，因此对计算机科学而言，数据的含义极为广泛，如声音、图像和视频等被编码后都属于数据的范畴。

数据元素是数据的基本单位，在计算机程序中通常将其作为一个整体进行考虑和处理。在某些情况下，我们也将数据元素称为元素、结点或记录等。例如，如果我们以学号、性别和姓名来标识某个学生，那么由学号、性别和姓名组成的记录将构成一个数据元素；而从另一方面来看，某一学生的学号、性别或姓名也可以被认为是一个数据元素。

数据项是构成数据元素的不可分割的最小单位，也被称为字段、域或属性，例如，对于上述学生记录中的学号、性别和姓名而言，其中任意一项都可以被称为数据项。

数据对象是性质相同的数据元素的集合，是数据的一个子集，例如，整数的数据对象是集合 $N=\{0, \pm1, \pm2, \cdots\}$，英文字母的数据对象是集合 $C=\{$'a', 'A', 'b', 'B', $\cdots\}$。

数据结构是相互之间存在一种或多种特定关系的数据元素的集合，通常这些数据元素都不是孤立存在的，而是通过某种关系将所有数据元素联系起来，我们将这种关系称为结构。数据结构通常包括数据的逻辑结构和存储结构两个层次，后面将对其进行详细介绍。

1.1.2 数据的逻辑结构

数据的逻辑结构是从数据元素的逻辑关系上抽象描述数据，通常是从求解问题中提炼出来的。数据的逻辑结构与数据的存储无关，是独立于计算机的，因此数据的逻辑结构可以被看作是从具体问题中抽象出来的数学模型。

数据元素之间的逻辑结构是多种多样的，根据数据元素之间的不同关系特性，通常可将数据逻辑结构分为4类，即集合、线性结构、树形结构和图状（或网状）结构，具体如图1-1所示。

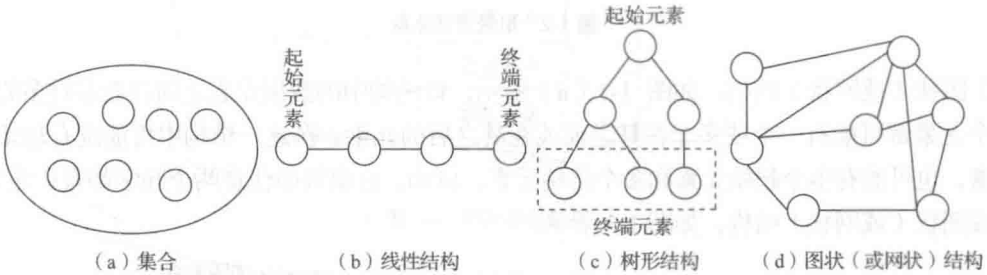


图1-1 四类基本逻辑结构关系

下面每种逻辑结构中的示例均以某校高三年级1班的学生作为数据对象，并以该班学生的某些信息作为数据元素。其中，部分学生的学号、姓名、性别和班级编号如表1-1所示。

表1-1 部分学生信息

学号	姓名	性别	班级编号
1501	王小阳	男	01
1504	宋小莹	女	01
1503	李小伟	男	01
1502	张晓晨	女	01
1506	赵晓达	男	01
1505	刘晓宏	男	01
1507	庄小谦	女	01

（1）集合：如图1-1（a）所示，该结构中的数据元素除了属于同一集合以外，两两之间并无其他关系。例如，确定某个学生是否属于本班级，此时需要将班级看作一个集合。

（2）线性结构：如图1-1（b）所示，该结构中的数据元素之间存在一对一的逻辑关系，并且起始元素和终端元素都是唯一的，除了这两个元素外，剩余的每一个元素都有且仅有一个在其之前和在其之后的元素。例如，将表1-1中的学号按照从小到大顺序进行排列，可以得到一个线性结构的序列{1501, 1502, 1503, 1504, 1505, 1506, 1507}。

（3）树形结构：如图1-1（c）所示，该结构中的数据元素之间存在一对多的逻辑关系，并且除了起始元素外，其余每一个元素都有且仅有一个在其之前的元素，除了终端元素外，其余每一个元素都有一个或多个在其之后的元素。例如，在班级中只有一个班长，班长对各个组长进行管理，而各个组长则管理自己组内的成员（即组员），由此形成一个树形结构，如图1-2所示。

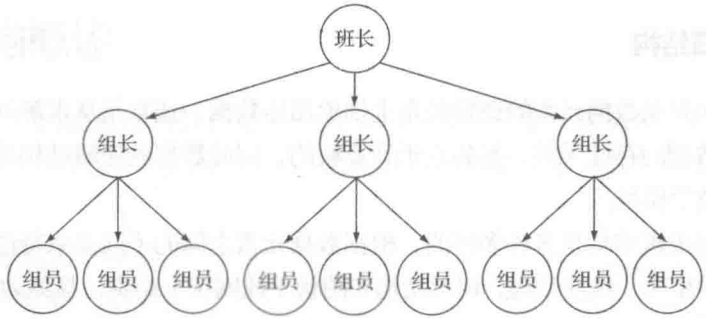


图 1-2　班级管理体系

（4）图状（或网状）结构：如图 1-1（d）所示，该结构中的数据元素之间存在多对多的逻辑关系，每个元素都可能有一个或多个在其之前或在其之后的元素。在这一结构中可能没有起始元素和终端元素，也可能有多个起始元素和多个终端元素。例如，在班级中任意两个同学都有可能是朋友，从而形成图状（或网状）结构，如图 1-3 所示。

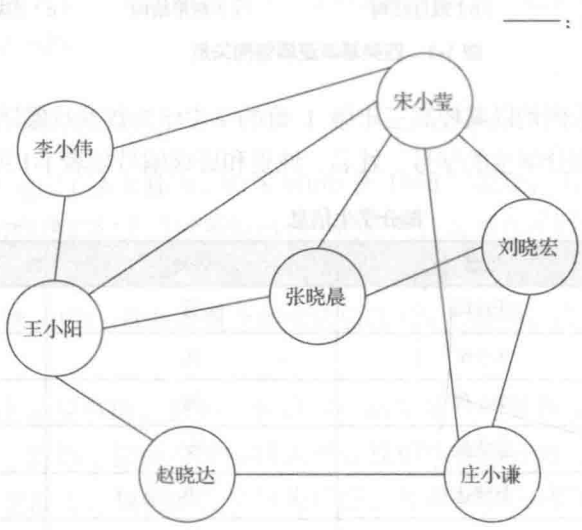


图 1-3　朋友关系

集合、树形结构和图状（或网状）结构都不属于线性结构，通常我们将其称为非线性结构。

1.1.3　数据的存储结构

数据的存储结构是指数据在计算机中的表示（又称为映像）方法，是数据的逻辑结构在计算机中的存储实现，因此在存储时应包含两方面的内容——数据元素本身及数据元素之间的关系。在实际应用中，数据有各种各样的存储方法，对其进行总结后，可大致划分为以下 4 类。

1. 顺序存储结构

顺序存储结构是指采用一组物理上连续的存储单元来依次存放所有的数据元素，如图 1-4 所示。在这一存储结构中，逻辑上相邻地两个数据元素的存储地址也相邻，因此我们只需要存储数据元素，而不需要存储这些数据元素之间的关系，因为它们的关系可以由存储单元地址间的关系来间接表示。

图 1-4 顺序存储结构

顺序存储结构将数据元素的逻辑结构直接映射到存储结构上，这十分有利于实现对数据元素的随机存取，但由于该结构要求存储单元在物理上是连续的，因此在进行数据元素的插入及删除等操作时，可能需要移动一系列的数据元素。

2. 链式存储结构

在链式存储结构中，每一数据元素均使用一个结点来存储，并且每个结点的存储空间是单独分配的，因此存储这些结点的空间不一定是连续的。

在链式存储结构中，我们不仅需要存储数据元素本身，还需要存储数据元素之间的逻辑关系，即将结点分为两部分，一部分是存储数据元素本身的，我们称其为数据域；另一部分是存储下一个结点的地址（即存储逻辑关系）的，我们称其为指针域。通过将每一个结点的指针域链接起来，从而形成链式存储结构。

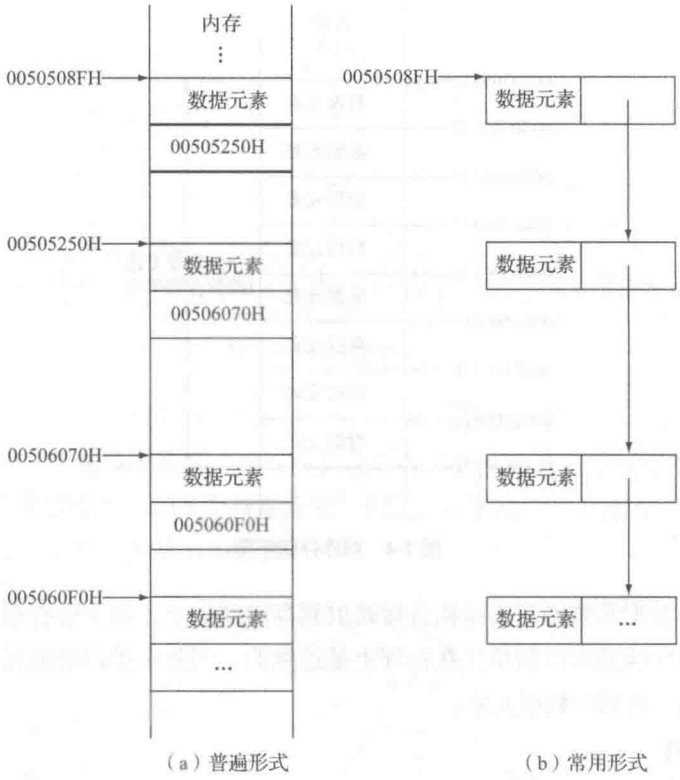
在链式存储结构中插入或删除数据元素时，可以直接通过修改指针域中的地址来实现，而不必移动大量结点。因为链式存储结构需要使用指针表明数据元素之间的逻辑关系，所以存储空间的利用率较低，并且由于结点的物理地址不一定是相邻的，因此只能通过结点的指针域找到存储的数据元素，而不能对数据元素进行随机存取。

图 1-5（a）所示为采用链式存储结构存储数据元素的普遍形式，为了让读者更好地理解链式存储结构，在后续章节中描述链式存储结构时，我们均采用图 1-5（b）所示的常用形式。

3. 索引存储结构

在索引存储结构中，不仅需要存储所有数据元素（称之为主数据表），还需要建立附加的索引表。在存储时，每个数据元素都由一个唯一的关键字来标识，由该关键字和对应的数据元素的地址构成一个索引项，并将其存入索引表中。通常索引表中的所有索引项是按关键字有序排列的，在查找数据元素时，首先由关键字的有序性，在索引表中查找出关键字所在的索引项，并取出该索引项中的地址，再依据此地址在主数据表中找到对应的数据元素。

由于借助索引表可以通过关键字快速地定位到数据元素，因此索引存储结构的查找效率很高，但索引表需要额外的空间进行存储，导致存储空间的利用率较低。



（a）普遍形式　　（b）常用形式

图 1-5　链式存储结构

4. 哈希（或散列）存储结构

哈希（或散列）存储结构是指依据数据元素的关键字，通过事先设计好的哈希（或散列）函数计算出一个值，再将其作为该数据元素的存储地址，因此使用哈希（或散列）存储结构也可以实现快速地查找，并且在采用哈希（或散列）存储结构时，只需要存储数据元素，而不需要存储数据元素之间的关系。

上述 4 种存储结构既可以单独使用也可以组合使用，确定了逻辑结构后，采用何种存储结构要视具体问题而定，通常需要考虑的是操作的方便性、效率，以及对时间和空间的要求。

1.2　数据类型概述

1.2.1　数据类型

类型是指一组值的集合，而数据类型则是指一组值的集合及定义在这组值上的一组操作的总称。例如，Python 中字符串类型的集合是由单引号或双引号标识的一连串字符，定义在其上的操作有字符串连接、重复输出字符串、截取字符串中的一部分和通过索引获取字符串中的一部分等。

Python 中的变量不需要声明，即变量没有具体的数据类型，但变量在使用前必须被赋值，赋值后该变量才会被创建，创建后变量将有具体的数据类型。

Python 语言的数据类型分为原子类型和结构类型，前者的值是不可分解的，而后者的值则是由若干成分按某种结构组成的，因此是可以分解的，下面介绍几种基本数据类型。

1. 数字（Number）数据类型

Python 中的数字数据类型用于存储数值，如整型、浮点型和复数型，定义在其上的操作有加、减、乘和除等。

2. 字符串（String）数据类型

字符串是 Python 中最为常用的数据类型之一，通常使用单引号或双引号来创建。定义在其上的操作有字符串连接（“+”）、重复输出字符串（“*”）、通过索引获取字符串中的字符（“[]”）、截取字符串中的一部分（“[:]”）、若包含指定字符则返回 True（“in”）、若不包含指定字符则返回 True（“not in”）、原始字符串（“r/R”）和格式字符串（“%”）等。

3. 列表（List）数据类型

列表是 Python 中最常用的数据类型之一，通常使用方括号来创建。定义在其上的操作有访问列表中的值、更新列表和删除列表元素等，同时与字符串类似，列表也包括连接、重复和截取等操作。

4. 元组（Tuple）数据类型

Python 中元组与列表类似，但元组使用小括号创建，并且其中的元素不能修改。定义在元组上的操作有访问元组、修改和删除元组，同时元组也包括连接、重复和截取等操作。

5. 集合（Set）数据类型

集合是由一组无序且不重复的元素组成的序列，常使用{}或者 set()函数来创建。定义在其上的操作有进行成员关系测试和删除重复元素等。

6. 字典数据类型

Python 中字典形如{key1:value1,key2: value2,…}，其中 key1 和 key2 部分被称为键（必须是唯一的），value1 和 value2 被称为值。定义在字典上的操作有修改和删除等。

事实上，在计算机中，数据类型的概念并非局限于高级语言中，每个处理器（包括计算机硬件系统、操作系统、高级语言和数据库等）都提供了原子类型或结构类型。例如，一个计算机硬件系统通常含有“位”“字节”和“字”等原子类型，它们的操作通过计算机设计的一套指令系统直接由电路系统完成，而高级语言提供的数据类型，其操作需通过编译器或解释器，转化成汇编语言或机器语言的数据类型来实现。

从硬件的角度考虑，引入某一数据类型的目的是解释该类型数据在计算机内存中对应信息的含义，而对使用这一数据类型的用户来说，则实现了信息的隐蔽，即将一切用户不必了解的细节都封装在相应的数据类型中。

例如，用户在使用“字符串”类型时，既不需要了解“字符串”在计算机内部是如何表示的，也不需要知道其操作具体是如何实现的。

1.2.2 抽象数据类型

抽象数据类型（Abstract Data Type，ADT）是指一个数学模型及定义在该模型上的一组操作。抽象数据类型的定义仅取决于它的一组逻辑特性，与其在计算机内部如何表示和实现无关，具体包括数据对象、数据对象上关系的集合，以及对数据对象的基本操作的集合。我们用以下格式定义抽象数据类型。

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>

}

其中，数据对象是具有相同特性的数据元素的集合，数据关系是对这些数据元素之间逻辑关系的描述。基本操作的声明格式如下。

基本操作名（参数表）

初始条件：<初始条件描述>

操作目的：<操作目的描述>

操作结果：<操作结果描述>

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，若为空，则可省略；操作目的描述了执行该操作应完成的任务；操作结果描述了该操作被正确执行后，数据结构的变化状况和应返回的结果。

下面以复数为例，给出其抽象数据类型的定义，具体如表 1-2 所示。

表 1-2　　复数的抽象数据类型的定义

数据对象	DataSet={e1,e2|e1,e2∈R,R 是实数集}		
数据关系	S={<e1,e2>|e1 是复数的实部，e2 是复数的虚部}		
基本操作	序号	操作名称	操作说明
	1	InitComplex(Complex)	初始条件：无。 操作目的：初始化复数。 操作结果：复数 Complex 被初始化
	2	CreateComplex(Complex,e1,e2)	初始条件：复数 Complex 已存在。 操作目的：e1 和 e2 分别被赋给复数 Complex 的实部和虚部。 操作结果：复数 Complex 被创建
	3	DestroyComplex(Complex)	初始条件：复数 Complex 已存在。 操作目的：销毁复数 Complex。 操作结果：复数 Complex 不存在
	4	GetReal(Complex,er)	初始条件：复数 Complex 已存在。 操作目的：获取复数 Complex 的实部并赋给 er。 操作结果：返回 er
	5	GetImag(Complex,ei)	初始条件：复数 Complex 已存在。 操作目的：获取复数 Complex 的虚部并赋给 ei。 操作结果：返回 ei
	6	AddComplex(Complex1,Complex2)	初始条件：复数 Complex1 及 Complex2 已存在。 操作目的：将复数 Complex1 和 Complex2 相加。 操作结果：返回相加的结果
	7	SubComplex(Complex1,Complex2)	初始条件：复数 Complex1 及 Complex2 已存在。 操作目的：将复数 Complex1 和 Complex2 相减。 操作结果：返回相减的结果

1.3 算法概述

1.3.1 什么是算法

算法是对待特定问题求解步骤的一种描述，它是指令的有限序列，其中每一条指令表示一个或多个操作。

一个算法应该具备以下5个重要特性。

（1）有穷性：一个算法对于任何合法的输入必须在执行有穷步之后结束，且每一步都可在有穷的时间内完成。

（2）确定性：算法中每一条指令都必须具有确切的含义，不能有二义性，并且，在任何条件下，算法的任意一条执行路径都是惟一的，即对于相同的输入所得的输出相同。

（3）可行性：一个算法是可行的，是指算法中描述的操作都可以通过基本运算执行有限次操作来实现。

（4）输入：一个算法有零个或多个输入，这些输入取自于某个特定对象的集合。

（5）输出：一个算法有零个或多个输出，这些输出是同输入有着某些特定关系的量。

说明：算法和程序是不同的，程序是指使用某种计算机语言对一个算法的具体实现，即程序描述了具体怎么做，而算法侧重于描述解决问题的方法。

在使用计算机求解实际中的问题时，不仅要选择合适的数据结构，还要有好的算法，那么应该如何评价一个算法的好坏呢？通常按以下指标来衡量。

（1）正确性：要求算法能够正确地执行，并满足预先设定的功能和性能要求，大致分为以下4个层次。

① 程序不含语法错误。

② 程序对于几组输入数据，能够得出满足要求的结果。

③ 程序对于精心选择的典型、苛刻而带有刁难性的几组输入数据，能够得出满足要求的结果。

④ 程序对于一切合法的输入数据，都能够得出满足要求的结果。

（2）可读性：算法主要是为了给人们阅读和交流的，其次才是在计算机上执行。一个算法的可读性好才便于人们理解，人们才有可能对程序进行调试，并从中找出错误。接下来给出几个在程序编写上提高可读性的方法。

① 注释：给程序添加注释，不仅有利于程序设计者自己阅读和查错，也为后续维护人员理解该程序带来方便。

② 变量命名：较复杂的程序通常会涉及较多的变量命名，此时应合理设计变量的名字，从而给后续使用该变量带来方便。

（3）健壮性：当输入的数据不合法或运行环境改变时，算法能恰当地做出反应或进行处理，而不是产生莫名其妙的输出结果。

（4）时间复杂度：对一个算法执行效率的度量。

（5）空间复杂度：是指一个算法在执行过程中所占用的存储空间的度量。

1.3.2 算法的时间复杂度

算法的执行时间是通过依据该算法编写的程序在计算机上执行时所需要的时间来计算的，通常