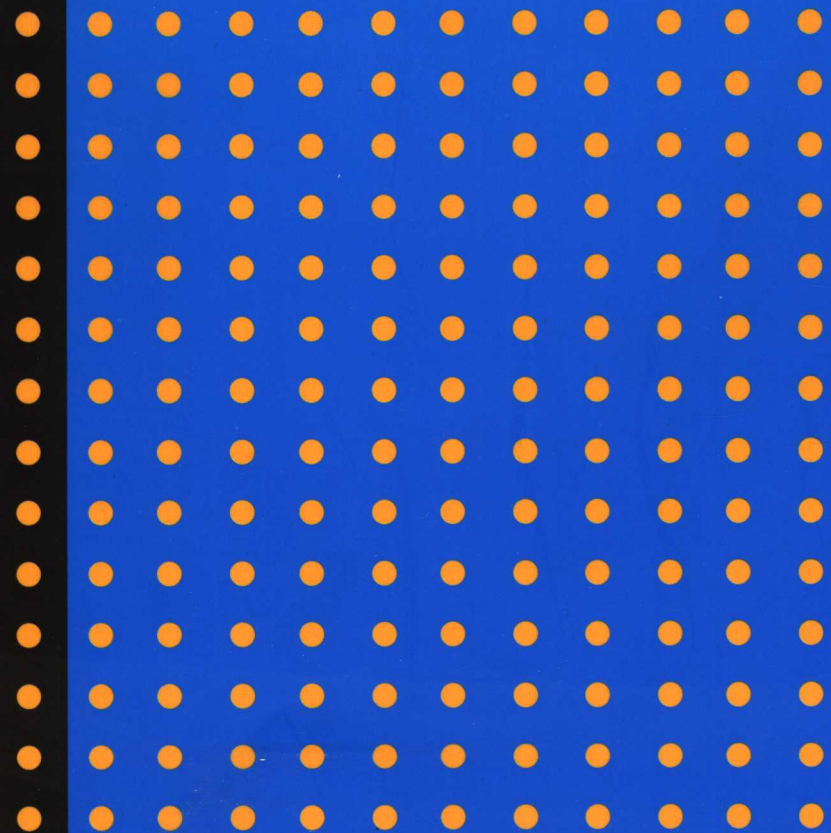
重点大学计算机专业系列教材

数据结构(C++描述)

金远平　编著
陈道蓄　审

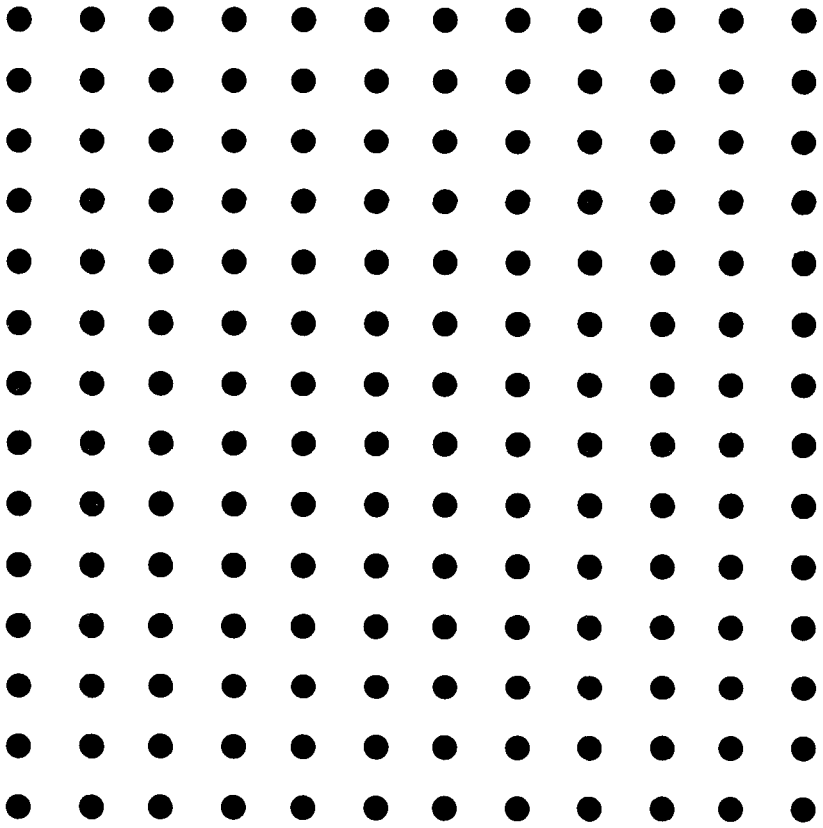


清华大学出版社

重点大学计算机专业系列教材

数据结构(C++描述)

金远平　编著
陈道蓄　审



清华大学出版社
北京

内 容 简 介

本书系统、全面地论述数据结构的重要内容，包括基本概念和方法、线性表、链表、树、堆结构、图、排序和搜索结构。在充分继承国内外经典教材的合理体系结构和优秀内容的基础上，结合国内实际教学情况编写，内容系统、精炼，且经过优化整合，在深度和广度上有明显增强；突出重点、难点，强调分析问题和解决问题的方法，以及产生这些方法的背景。

书中内容都经过编者深入研究，且在教学实践中反复验证，因而较易理解。本书注重启发创新思维，培养能力；概念准确，逻辑性强；自然引用面向对象设计思想，用C++语言描述算法。

本书适于作为计算机科学与技术、软件工程以及相关专业的教材，也可供从事相关工作的科技与工程人员参考。

图书在版编目(CIP)数据

数据结构（C++描述）/金远平编著. —北京：清华大学出版社，2005.7
（重点大学计算机专业系列教材）
ISBN 7-302-10798-X

Ⅰ. 数… Ⅱ. 金… 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

中国版本图书馆CIP数据核字（2005）第031451号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969
责任编辑：付弘宇
封面设计：刘艳芝
印 刷 者：北京密云胶印厂
装 订 者：三河市李旗庄少明装订厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：22　**字数**：518千字
版　　次：2005年7月第1版　2005年7月第1次印刷
书　　号：ISBN 7-302-10798-X/TP·7179
印　　数：1～4000
定　　价：29.00元

FOREWORD

前言

设计解决实际问题的计算机软件系统，首先需要建立被处理对象的数据模型。数据和世上万物一样，都是具有结构的。因此，人们很自然地用数据结构表示应用领域的被处理对象。为了模拟实际问题的求解过程和现实对象的行为，还必须提供对数据结构的相应操作。数据结构的实现是由下一层数据结构表示上一层数据结构，直至由程序设计语言提供的基本数据类型表示的过程。评价数据结构表示优劣的标准主要是其能否方便且有效地实现需要的操作，而实现操作的算法设计及其效率高低也依赖于数据结构表示。因此，数据结构的定义、表示以及操作的实现相互关联，都是数据结构研究的重要内容。

计算机软件系统可看成是通过不同层次的数据结构及其操作实现的。通过多层表示，完成计算机对应用领域问题的求解过程。在此，中间层数据结构起着核心作用。数据结构的研究产生了一批通用性强、具有很高实用价值的中间层数据结构，如数组、字符串、集合、线性表、栈、队列、链表、树、图、符号表等。这些结构不仅为我们提供了设计软件系统的有用工具，而且向我们展示了在广泛的应用领域表示与解决问题的精巧思路和技术。系统地学习和掌握数据结构知识和方法，对于提高设计与开发软件系统尤其是复杂软件系统的能力，无疑是十分重要的。因此，数据结构早已成为计算机科学与技术和软件工程等专业的核心课程。

数据结构课程内容丰富，涵盖了计算机科学与技术的许多重要的成果，分析问题和解决问题的思路和方法新颖，创新点多，技巧性强，对学生专业素质的培养作用明显，但同时也是一门较难学习的课程。我校计算机科学与工程系开设的“数据结构”课程一直采用美国南加州大学教授 E. Horowitz 等编著的《数据结构基础》作为教材。该书注重培养学生分析问题、解决问题的能力，在数据结构和算法设计以及时空复杂性分析的深度和广度方面特色明显。但在教学中也感到该书内容的表达形

式学生较难理解，方法和技术的论述还不够简明扼要，有的内容不够精炼，部分章节存在一些小错误，教学效果较多依赖于教师的讲解。为此，编者在充分继承该书体系结构和内容优点的基础上，吸收其他教材长处，进行优化整合，并结合自身多年的教学改革与实践经验，编写了这本教材，力图使其系统全面，内容深刻，表达简洁，易于理解，以适应计算机科学与技术及相关专业的教学需要。

本书共分为8章。第1章论述数据结构的基本概念和方法，包括数据结构与软件系统，数据抽象与封装，算法，递归，性能分析、性能测量，以及效率与权衡等。特别引入了代价分摊分析方法，将相关的操作序列联系起来分析，从而得到更接近实际代价的结果。此外，还从软件重用的角度描述了C++的模板机制。

第2章介绍线性表的概念及其顺序表示方法，讨论了通过线性表表示的多项式、稀疏矩阵和字符串等结构。在描述著名的字符串模式匹配算法KMP时，采用了简明易懂的图示方法。还通过两个字符串的最长公共子序列问题的求解，展示了利用动态规划改进算法效率的方法。由于栈和队列是受限的线性表，因此也被整合到这一章。此章不仅给出了通用栈和队列的实现方法，还分别通过求解迷宫问题、表达式计算以及机场模拟问题描述了栈和队列的应用。

第3章论述以链表形式实现线性表的方法和技术，讨论了遍历通用模板类容器对象的游标技术。还介绍了广义表的功能及其实现方法，并结合C++的动态类型，讨论了异构表的实现方法。

第4章介绍最基本的非线性结构——树，包括树和森林的概念、二叉树、二叉树的遍历及应用、线索二叉树、胜者树、败者树、森林的二叉树表示及遍历、树在并查集问题中的应用和二叉树计数等。特别给出了有一定难度的中序线索二叉树的后序遍历算法，还给出了生成所有可能的二叉树的算法。

第5章介绍支持各种优先队列的堆结构，包括最大堆、最大最小堆、双堆、左偏树、二项式堆和斐波纳契堆。在分析二项式堆和斐波纳契堆的性能时，采用了代价分摊方法。

第6章介绍更普遍的非线性结构——图，包括图的定义和表示方法、图的遍历、图的连通性、最小代价生成树、最短路径和传递闭包以及活动网络等。特别讨论了应用斐波纳契堆改进最短路径算法性能的技术。

在数据结构中，数据元素之间的次序是一种重要的关系。按照数据元素的特定属性对其进行排序是最频繁的计算任务之一。第7章介绍各种典型的排序方法，包括插入排序、希尔排序、快速排序、归并排序、堆排序、基数排序、基于链表和映射表排序结果的顺序化以及外排序。

第8章介绍符号表概念以及实现符号表的各种结构，包括二叉查找树、AVL树、2-3树、Splay树、B树、B+树、Trie、静态散列和动态散列等，特别分析了二叉查找树的平均性能。在分析Splay树的性能时，采用了代价分摊方法。此外，还论述了上述结构的演变关系，帮助读者理解其设计思想。

本书可供各种层次的读者选用，既适用于教学，也可供从事相关工作的科技与工程人员参考。可以按“数据结构基础”（64学时，必修）和“高级数据结构”（24～32学

时，选修）两门课组织教学。本书的 2.4、2.9、3.10、4.5、4.8、4.9、5.2～5.6、6.4、7.8、8.4、8.5、8.7 和 8.9 节可作为“高级数据结构”课程的内容，其余作为“数据结构基础”课程的内容。

本书引用了数据结构研究的大量先进成果，在此，作者谨向这些成果的原创者表示崇高的敬意和衷心的感谢。同时，对本书所引用的参考文献的作者也表示衷心的感谢。

在本书的写作过程中，作者与徐宝文、孙志挥、王茜、徐冬梅、王树梅、吉根林和张丽晖老师开展了卓有成效的讨论，并由此得到很多启发。南京大学计算机科学与技术系陈道蓄教授认真审读了全部书稿，并提出了十分宝贵的修改意见，在此对他们表示最诚挚的谢意。感谢清华大学出版社的鼓励与支持。感谢东南大学教学改革基金的资助。还要感谢作者的众多学生，他们在数据结构课程学习过程中表现出的热情与执着给了作者很大的鼓励，与他们的讨论和交流使作者对教学内容和教学方法的改进有了更深刻的认识。

限于作者水平，书中难免有错误或不足之处，恳请广大读者批评指正。来信请发至 ypjin@seu.edu.cn。

作　者

2004 年 12 月于东南大学

CONTENTS

目录

CHAPTER 1

第1章 基本概念和方法

本质上，数据结构的原理和方法是独立于具体的描述语言的。本书采用目前普遍使用的C++语言描述各种数据结构，假设读者具有坚实的程序设计基础，了解C++的基本结构和语法。为了突出数据结构的主题，本书没有用专门篇幅介绍C++知识，而只在必要时对所用的C++结构作辅助说明，以帮助理解主题内容。本章介绍学习和研究数据结构所必需的并且将在本书中反复出现的基本概念和方法。在以后的章节中，还将适时结合范例介绍新的概念和方法。

1.1 数据结构与软件系统

设计解决实际问题的计算机软件系统，首先需要建立被处理对象的数据模型。数据和世上万物一样，都是具有结构的。因此，人们很自然地用数据结构表示应用领域的被处理对象。例如，可以用树表示现实中的层次关系。树对应的数据对象由一个结点数据集合组成，结点之间存在父子关系，这些结点以及结点之间的关系构成了树这一数据结构。又如，可以用图表示数学中的图论问题。图对应的数据对象由一个结点数据集合和一个边数据集合组成，边与结点之间存在邻接关系。更进一步，边可与权重数据相联系，每个结点可包含若干个属性数据。这些不同类型的数据元素以及元素之间的关系构成了图这一数据结构。可见，**数据结构**由一个数据对象以及该对象中的所有数据元素之间的关系组成。由于数据元素本身可以是数据结构，因此，能够构造非常复杂的数据结构。

为了模拟实际问题的求解过程和现实对象的行为，还必须提供对数据结构的相应操作。例如，对于图，需要有增加或删除边、求两点之间的最短路径、遍历所有结点等操作。数据结构的实现是自顶向下，以下一层数据结构表示上一层数据结构，直至以程序设计语言提供的基本数据类型来表示的过

程。实际上，通过编译软件转化，程序设计语言的基本数据类型及其操作也是由计算机提供的基本数据结构（如字节和机器字线性序列）表示的。评价数据结构表示能力的标准主要是它能否方便且有效地实现需要的操作，而实现操作的算法的设计及其效率高低也依赖于数据结构表示。因此，数据结构的定义、表示及其操作的实现相互关联，都是数据结构研究的重要内容。

计算机软件系统可看作是通过不同层次的数据结构及其操作实现的。下面看一个民航订票系统的例子。图 1.1 表示实现该系统需要的一些数据结构。在应用层，系统必须处理用户直接关注的数据对象（如时刻表、航班、日期和预定）。这些应用层的数据对象可用文件、表、链表、记录以及字符串等中间层数据结构表示。例如，所有航班的总时刻表可用某种表结构表示，对今后某一航班的预定可以作为与该航班相关联的文件中的一组记录保存。这些记录可包含表示预定旅客的姓名、电话号码和其他信息的字符串和数字数据。中间层数据结构最终由字节和数字数据这些计算机基础数据结构表示。这样，通过多层表示，完成对应用领域问题的求解过程。

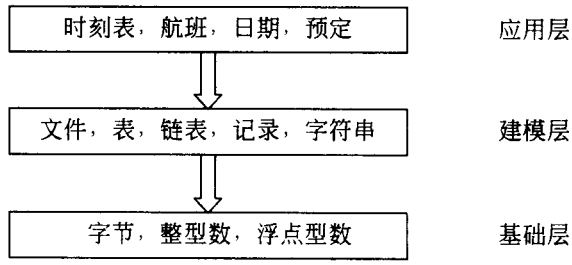


图 1.1　民航订票系统使用的数据结构

在此，中间层数据结构起着核心作用，称之为建模层。对数据结构的研究产生了一批通用性强、具有很高实用价值的中间层数据结构，如数组、字符串、集合、线性表、栈、队列、链表、树、图、符号表等。这些结构不仅提供了设计软件系统的有效工具，而且展示了在广泛的应用领域中表示与解决问题的精巧思路和技术。系统地学习进而掌握数据结构的知识和方法，对于提高设计与开发软件系统尤其是复杂软件系统的能力，无疑是十分重要的。

1.2　数据抽象与封装

抽象和封装的概念在日常生活中是普遍存在的。例如，观察人们常用的手机，可以发现：手机与用户的所有交互都是通过其界面上的按钮进行的，而手机内部的结构对用户是屏蔽的，用户不能直接与之交互，这体现了封装的原理；手机的使用手册说明各按钮的功能，而并不解释这些功能在其内部的软硬件中是如何实现的，这体现了抽象的原理。

这些原理同样适用于在软件设计中对数据的组织，从而产生了数据封装和数据抽象

的概念。通过**数据封装**，将一个数据对象的内部结构和实现细节对外屏蔽。通过**数据抽象**，将一个数据对象的规格说明与其实现分离，对外提供简洁、清晰的接口。上一节讨论的数据结构多层表示的过程，反过来也就是从基础数据结构到应用领域数据结构的不断抽象与封装的过程。

用抽象数据类型（Abstract Data Type，ADT）描述数据抽象与封装是一种自然、有效的方法。在程序设计语言中，已经有了数据类型的概念。一个**数据类型**由一个数据对象的集合和一组作用于这些数据对象的操作组成。例如，C++的基本数据类型包括 char、int、float 和 double，以及由此衍生的指针和参考类型。此外，C++还提供三种组织数据的机制，即 array、struct 和 class。任何一种程序设计语言都应该提供基本的预定义数据类型和构造新的、用户定义的数据类型的机制。一个**抽象数据类型**是一个数据类型，该数据类型的组织遵循将数据对象及对这些数据对象的操作的规格说明与这些数据对象的表示、操作的实现相分离的原则。当强调一个数据对象的结构时，使用数据结构的概念。比较数据结构和抽象数据类型的概念，抽象数据类型包含了一个数据结构的集合，还包含了对数据结构的操作。因此，抽象数据类型成为描述数据结构及其操作的有效方式。

在后面的数据结构学习中，一般先给出数据对象的 ADT 定义，使读者能掌握该数据对象的实质要素，而不受其内部表示和具体操作实现的干扰。一旦建立该数据对象的 ADT 定义的准确概念，就可进一步研究其表示和实现问题。

定义 ADT 的语言本质上不依赖于任何具体的程序设计语言，为了与本书的描述语言一致，书中全部采用 C++定义 ADT。需要注意的是，有的 C++操作（如<<）被用户定义的 ADT 重载时，不是作为相应类的成员函数、而是作为普通函数存在的。因此，虽然这些操作实际上是 ADT 的组成部分，但却在 C++类定义之外声明。

例 1.1　ADT 1.1 是用 C++描述的抽象数据类型“圆”的定义。其中，该抽象数据类型的名称为 Circle，数据对象定义为几何圆，操作包括构造函数、计算周长和面积等。注释给出了这些操作的语义。**注意**：这些定义不依赖于数据对象的具体表示，也没有给出操作的具体实现。

```
class Circle {
// 对象：几何圆
public:
   Circle(float r);              // 构造函数，创建一个半径为 r 的 Circle 对象实例
   float  Circumference( );      // 返回该实例的周长
   float  Area( );               // 返回该实例的面积
};
```

ADT 1.1　抽象数据类型 Circle

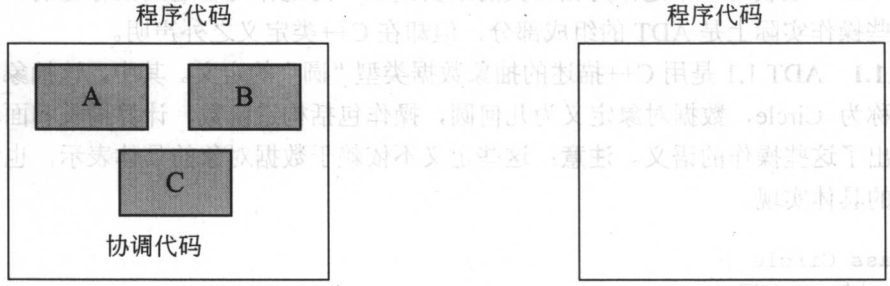
数据抽象和封装机制对于高效开发结构优良的软件程序具有以下几项重要的意义。

（1）**简化软件开发**　数据抽象有助于将开发一个软件系统的复杂任务分解为一组相对简单的子任务。假设一个问题经分析将使用 A、B、C 三个数据类型和协调代码求解，

且这些数据类型的规格说明已确定。如果有四位程序员，可由其中三位程序员各开发一个数据类型，另一位程序员实现协调代码。这样，每一位程序员都只须集中精力按照规格说明开发自己负责的代码，而无须考虑其他程序员的工作细节及其对自己的影响。如果只有一位程序员，数据抽象也可减少其在某一具体时间需要考虑的范围。这位程序员可按照规格说明逐个开发 A、B、C 三个数据类型，在开发 A 时，不必考虑 B、C 和协调代码。A、B 和 C 完成后，可根据它们的规格说明集中精力开发协调代码，而不必关心它们的实现细节。

（2）**易于测试和排除错误**　在上述例子中，各数据类型可分别进行测试和排除错误，之后再测试协调代码。如果这时发现错误，则出错原因不太可能存在于数据结构 A、B、C 中，而很可能在协调代码本身，因为 A、B、C 已经测试过了。用有背景的方框表示数据结构 A、B、C，则搜索错误的范围限定在图 1.2(a)的无背景区域。如不使用数据抽象，则搜索范围是如图 1.2(b)所示的整个程序代码区域。数据抽象明显提高了测试和排除错误的效率。

（3）**有利于重用**　数据抽象和封装机制使开发人员可以将数据结构及其操作实现为可重用的软件组件。这些组件具有清晰的界面定义，很容易从一个软件系统中提取出来，应用于另一个软件系统。而如果不使用数据抽象和封装，程序代码就会错综复杂地交织在一起，很难提取重用。本书在后面研究数据结构的过程中，将注重软件重用的思想和技术。掌握了丰富的可重用数据结构及其操作的本质思想，读者设计软件、解决问题的技能将大大增强。



(a) 使用数据抽象　　(b) 不使用数据抽象

图 1.2　无背景的区域表示搜索错误的范围

（4）**便于改变数据类型的表示**　由于数据封装，外界不能直接访问数据类型的内部表示，而只能通过数据类型的操作接口处理相应的数据对象。因此，只要操作接口不变，数据类型的内部表示和实现的改变不会影响使用该数据类型的其他程序。如果不采用数据封装，则程序可直接访问数据类型的内部表示。在这种情况下，要修改数据类型的内部表示必须同时查找和修改程序中所有访问该数据类型内部表示的代码，这将耗费很大的工作量。

1.3　算法定义

数据结构的操作实际上是以算法的形式实现的，因此，算法的设计与分析是数据结构研究的重要内容，也是既充满挑战又令人兴奋的部分。设计高效算法对于开发大规模计算机系统起着关键作用。

定义 1.1　算法是一个有限的指令集合，执行这些指令可以完成某一特定任务。一个算法还应当满足以下特性：

（1）**输入**　零个或多个由外界提供的输入量。

（2）**输出**　至少产生一个输出量。

（3）**确定性**　每一条指令都有确切的语义，无歧义。

（4）**有限性**　在任何情况下都应该在执行有限步骤后结束。

（5）**有效性**　每一条指令都应能经过有限层的表示转化为计算平台的基本指令，即算法的每一条指令必须是可行的。

程序和算法不同，程序可以不满足特性（4）。例如，一个软件的总控程序在未接受新的任务之前一直处于“等待”循环中。实现数据结构操作的程序总是可结束的，即满足特性（4），因此，本书后面将不再严格区分算法和程序这两个术语。此外，必须保证指令的有效性，例如，指令“**if**（哥德巴赫猜想是真）**then** x = y;”是无效的，因为判断“哥德巴赫猜想是真”目前不可行。

描述算法可有多种方式，如自然语言、流程图以及各类程序设计语言。本书采用 C++ 语言描述大部分算法，有时，为了突出算法的本质思想，避免程序细节的干扰，也采用 C++与自然语言相结合的描述方法。下面用一个例子说明如何设计求解问题的算法。

例 1.2　二分查找：设数组 a[0], …, a[n−1]中存放着 $n \geq 1$ 个已由小到大排序的不同整数，判定整数 x 是否在数组 a 中。若是，返回 j，使得 x = a[j]；否则返回−1。

利用数组中的整数已排序的性质，可以设计高效的求解算法。令 left 和 right 分别表示当前需查找的数组部分的起点和终点下标。初始时，left = 0，right = n − 1。令 middle = ⌊(left + right) / 2⌋为中点下标。比较 a[middle]和 x，有如下三种可能：

（1）x < a[middle]　这时，若 x 存在，则 x 的下标一定在 left ~ middle − 1 范围内，因此，将 right 设置为 middle − 1。

（2）x == a[middle]　这时，返回 middle 即可。

（3）x > a[middle]　这时，若 x 存在，则 x 的下标一定在 middle + 1~ right 范围内，因此，将 left 设置为 middle + 1。

如果 x 还未找到且数组 a 中还有可供查找的整数，则可通过循环重新计算 middle，继续查找。该算法包含两个子任务：（1）判断是否还有可供查找的整数，这可转化为判断是否 left ≤ right；（2）比较 x 和 a[middle]，为了清晰描述整体算法，可用程序 1.1 给出的 compare 函数实现。