普通高等教育“十三五”规划教材
新工科建设之路·计算机类专业规划教材

算法与数据结构

（C++ 语言版）

冯广慧　吴　昊　文全刚　编著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

普通高等教育“十三五”规划教材
新工科建设之路·计算机类专业规划教材

算法与数据结构

（C++语言版）

冯广慧　吴　昊　文全刚　编著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书按照“全国硕士研究生招生考试计算机科学与技术学科联考计算机学科专业基础综合考试大纲”的要求编写，基本涵盖所有知识点，并加入部分高校及全国统一考试真题作为自测题，同时给出参考答案和题目解析。本书主要介绍各种常用的经典数据结构（如线性表、栈、队列、串、数组、树、图、集合等）和算法，并在时间复杂度和空间复杂度之间进行平衡与取舍。

本书将 C++语言作为数据结构的算法描述语言，将数据结构与面向对象技术有机结合。书中的算法讲解都有完整的 C++代码实现，并在 Visual Studio 2010 环境下编译通过。

本书既可作为普通高等院校计算机及相关专业的数据结构课程教材，也可作为考研参考书，还可作为工程技术人员的工具书。

图书在版编目（CIP）数据

算法与数据结构：C++语言版 / 冯广慧等编著. －北京：电子工业出版社，2019.1
ISBN 978-7-121-35071-9

Ⅰ. ①算… Ⅱ. ①冯… Ⅲ. ①算法分析－研究生－入学考试－自学参考资料 ②数据结构－研究生－入学考试－自学参考资料 Ⅳ. ①TP301.6 ②TP311.12

中国版本图书馆 CIP 数据核字（2018）第 217294 号

策划编辑：冉 哲 刁伟兴
责任编辑：冉 哲
印 刷：涿州市京南印刷厂
装 订：涿州市京南印刷厂
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×1 092 1/16 印张：21.5 字数：606 千字
版 次：2019 年 1 月第 1 版
印 次：2019 年 1 月第 1 次印刷
定 价：56.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：ran@phei.com.cn。

前　言

随着计算机技术的飞速发展，计算机在各个学科和领域得到广泛应用，而这些应用所面临的首要问题就是对于信息量大、种类繁多、结构复杂的数据和数据关系的处理，因此必须设计好数据结构和数据组织方式，以便有效地实现数据存储、数据传输和数据处理等操作。数据结构主要研究数据的逻辑结构，数据在计算机中的存储实现，以及处理不同结构数据的算法。我们研究数据结构的目的是编写更高效的程序，而高效、简捷的程序取决于数据结构和算法的设计。

“数据结构”是计算机程序设计的重要理论基础，是计算机专业最为核心的一门专业基础课程，也是非计算机专业的主要选修课程，同时还是一门考研课程。数据结构前承高级语言程序设计和离散数学，后接操作系统、编译原理、数据库原理等专业课程，为研制开发各种系统和应用软件奠定理论和实践基础。该课程的学习效果不仅关系到后续课程的学习，而且直接关系到软件设计水平的提高和专业素质的培养，在计算机学科教育中有非常重要的作用。

考虑到初学者普遍对算法设计问题感到比较困难且思路不明确，本书不仅注重基本概念的引入和阐述，更加注重算法的设计、分析与实现，强调实践环节的重要性。本书具有如下特点。

（1）将 C++语言作为数据结构的算法描述语言，让数据结构与面向对象技术有机结合。在设置例题时，充分考虑应用型人才培养的需求，更加侧重于算法的程序实现。书中的算法讲解都有风格优美而完整的 C++代码实现，并在 Visual Studio 2010 环境下编译通过，这将有利于读者掌握算法的程序实现及对算法进行分析与比较。

（2）按照“全国硕士研究生招生考试计算机科学与技术学科联考计算机学科专业基础综合考试大纲”的要求编写，基本涵盖该考试大纲所有的知识点，并在重要知识点之后附加部分高校及全国统一考试真题作为自测题。读者在完成相应问题的同时，既能巩固知识点，又能有选择地提高能力，还能有效地检验阶段学习效果。

（3）系统、全面地介绍各种传统的数据结构，按照“线性结构、树结构、图结构、集合结构”四大模块顺序安排内容。部分章节还设有算法设计举例，意在提高初学者的算法分析和设计的能力。

全书分为 12 章。

第 1 章介绍基础知识，讨论什么是数据结构，给出数据结构和算法的相关概念与描述方法，介绍算法分析的基本方法。

线性结构包括第 2～5 章。第 2 章介绍线性表的有关概念及其基本操作，是后续章节的基础。第 3 章在第 2 章的基础上讨论操作受限的线性表——栈与队列。第 4 章讨论数据元素为字符的特殊的线性表——串。第 5 章介绍程序设计中常用的数据类型——数组。

树结构包括第 6、7 章。第 6 章介绍树和二叉树的有关概念及基本操作。第 7 章讨论树和二叉树的应用。

图结构包括第 8、9 章。第 8 章介绍图的基本概念、存储结构及遍历运算。第 9 章讨论图的应用。

集合结构包括第 10～12 章。第 10 章以集合作为数据模型，讨论查找的方法和技术，包括静态查找表和动态查找表。第 11 章介绍一种专用于集合的存储和检索的数据结构——散列表。第 12 章介绍一些常用的排序算法，包括内部排序和外部排序。

本书由教学一线的教师主笔，结合作者多年的教学经验和教学素材，针对数据结构这门课程的

特点撰写而成，目的是使学生在扎实的编程能力基础上，掌握如何合理地组织数据、有效地存储和处理数据，并学会在时间复杂度和空间复杂度之间进行平衡与取舍。本书满足多样化的人才培养模式的需求，既可作为普通高等院校计算机及相关专业的数据结构课程教材，也可作为考研参考书，还可作为工程技术人员的工具书。在本书目录中加“*”的章节可以酌情处理。

在本书的编写和出版过程中得到电子工业出版社冉哲编辑和《算法与数据结构考研试题精析》的编者陈守孔教授的诸多帮助，得到吉林大学珠海学院领导的大力支持，在此深表感谢。

由于作者水平所限，加上计算机学科的发展十分迅速，书中难免有不妥之处，恳请读者批评指正。

作　者

目　录

第1章 概 论

算法与数据结构是计算机科学与技术、软件开发与应用、信息管理、电子商务、网络安全等专业的一门专业基础课。算法与数据结构课程的内容不仅是程序设计（特别是非数值计算的程序设计）的基础，而且是设计和实现编译程序、操作系统、数据库系统及其他系统程序的重要基础。无论读者从业于计算机行业，还是希望在计算机相关方面继续深造，该课程的学习都是必需的。

算法与数据结构研究解决非数值计算的现实问题中的数据在计算机中如何表示、快速存取和处理的方法。这里所说的数据是广义的概念，它不仅包括整型、实型、逻辑型等基本类型的数据，还包括带有一定结构的各种复杂的数据，如串、记录、向量、矩阵等，也包括各种表格、图形、音频和视频等非数值型的数据。当用计算机存储数据时不仅要存储这些数据的值，还要存储这些数据之间的相互关系。因此存储数据和这些数据之间的关系就出现了各种不同的存储方法。

学习算法与数据结构的目的是编写高质量的程序，主要包括以下三个方面：

① 能够分析研究计算机加工的对象的特性，选择合适的逻辑结构、存储结构并设计相应的算法；

② 提高处理复杂程序设计问题的能力，要求编写的程序结构正确、清晰、易读；

③ 掌握算法的时间复杂度和空间复杂度的分析技术。

本章学习目标：

- 理解数据结构的基本概念和术语；
- 掌握算法分析技术，学会计算算法的时间复杂度和空间复杂度。

1.1 什么是数据结构

美国计算机界最初出现信息结构这一名称是在20世纪60年代。1968年，D. E. Knuth教授开创了数据结构的最初体系，他所著的“*The Art of Computer Programming*”第一卷“*Fundamental Algorithms*”是第一本较系统地阐述数据的逻辑结构和存储结构及其操作的著作。20世纪70年代初，数据结构作为一门独立的课程开始进入大学课堂。

数据结构起源于程序设计，它随着大型程序的出现而出现。随着计算机科学与技术的不断发展，计算机的应用领域已不再局限于科学计算，而更多地应用于控制、管理等非数值处理领域。据统计，当今处理非数值计算类问题占用了90%以上的计算机时间。与此相应，计算机处理的数据也由纯粹的数值发展到字符、表格、图形、图像、声音等具有一定结构的数据，处理的数据量也越来越大，这就给程序设计带来一个问题：应如何组织待处理的数据，表示数据之间的关系及实现数据运算？

计算机解决一个具体问题时，大致需要经过下列几个步骤：首先要从具体问题中抽象出一个合适的数学模型，然后设计一个解此数学模型的算法，之后编出程序、进行测试/调整，直至得到最终解答。

寻求数学模型的实质是分析问题，从中提取操作的对象，并找出这些操作对象之间的关系，

然后用数学的语言加以描述。当人们用计算机处理“数值计算问题”时（如求解弹道轨迹），所用的数学模型是用数学方程描述的，所涉及的运算对象一般是整型、实型和逻辑型等基本类型的数据，因此程序设计者主要关注程序的设计技巧，而不是数据的存储和组织方式。然而，计算机应用的更多领域是“非数值计算问题”，它们的数学模型无法用数学方程描述，而需要用数据结构描述，解决此类问题的关键是设计出合适的数据结构。描述非数值型问题的数学模型通常是用线性表、树、图等结构来描述的。

【例 1.1】 学生信息管理系统

对学生信息表常用的操作有：查找某个学生的有关情况，统计班级人数，按某个字段排序，增加或删除学生信息（转专业）等，由此可以建立一张按学号顺序排列的学生信息表，如表 1.1 所示。诸如此类的表结构还有图书馆书目管理系统、人事档案管理系统、仓库管理系统等。

表 1.1 学生信息表

学号	姓名	性别	专业	住址
04180101	侯亮平	男	计算机科学与技术	北京
04180102	高小琴	女	计算机科学与技术	深圳
04180103	陆亦可	女	计算机科学与技术	珠海
04180104	陈海	男	计算机科学与技术	上海
04180105	李达康	男	计算机科学与技术	杭州
04180106	高育良	男	计算机科学与技术	南京
04180107	赵东来	男	计算机科学与技术	武汉
04180108	陈岩石	男	计算机科学与技术	重庆
04180109	沙瑞金	男	计算机科学与技术	珠海
04180110	蔡成功	男	计算机科学与技术	广州
……	……	……	……	……

在这类问题中：

① 计算机处理的对象是各种表；

② 元素之间的逻辑关系是线性关系；

③ 施加于对象上的操作有遍历、查找、插入、删除等。

【例 1.2】 人机博弈

计算机之所以能和人博弈，是因为事先已经将对弈的策略和评价规则等输入了计算机中。在人机对弈问题中，计算机操作的对象是对弈过程中可能出现的称为格局的棋盘状态。如图 1.1 所示是井字棋对弈树，包括了多个对弈的格局，格局之间的关系是由比赛规则决定的。通常，这个关系是非线性的，因为从一个棋盘格局可以派生出几个格局，而从每一个新的格局又可派生出多个可能的格局。因此，如果将从对弈开始到结束的过程中所有可能出现的格局都表示出来，就可以得到一棵“树”。其“树根”是对弈开始之前的棋盘格局，而所有的“叶子”就是可能出现的结局，对弈的过程就是从树根沿树枝到每个叶子的过程。诸如此类的树状结构还有家族的族谱、计算机的文件系统、单位的组织结构图等。

在这类问题中：

① 计算机处理的对象是树状结构；

② 元素间的关系是一种一对多的层次关系；

③ 施加于对象上的操作有遍历、查找、插入、删除等。

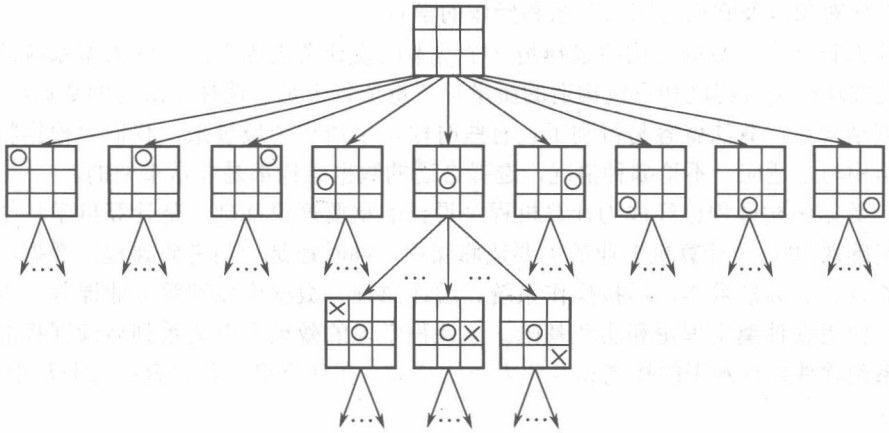


图 1.1　井字棋对弈树

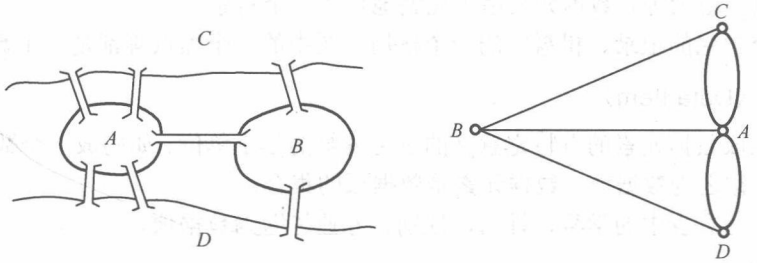
【例 1.3】　哥尼斯堡七桥问题

在哥尼斯堡的一个公园里，有 7 座桥将普雷格尔河中两个岛与两边的河岸相互连接起来，如图 1.2（a）所示。问是否可能从这 4 块陆地中任意一块出发，恰好通过每座桥一次，再回到起点？这一问题很久没有人找到答案。

1736 年，数学家欧拉把这 4 块陆地抽象为 4 个点 A、B、C、D，每座桥抽象为连接两个点的一条边，如图 1.2（b）所示。这样哥尼斯堡七桥问题被抽象为：从某一点出发，寻找经过每条边一次且仅一次，最后回到出发点的路径，这也被称为无向图的欧拉回路问题。欧拉回路存在的充分必要条件是：

① 图是连通的；

② 图中与每个顶点相连的边数（即顶点的度）必须是偶数。



（a）哥尼斯堡七桥　　（b）哥尼斯堡七桥的图表示

图 1.2　哥尼斯堡七桥问题

因为哥尼斯堡七桥问题不满足上述条件，所以它是无解的。欧拉由哥尼斯堡七桥问题所引发的对图的研究论文是图论的开篇之作，因此欧拉也被称为图论之父。从哥尼斯堡七桥问题可以看到，要利用图来解决问题，关键的第一步是找到现实问题的实体与图的点和边的对应关系。诸如此类的结构还有城市网络交通图、网络工程图等。

在这类问题中：

① 计算机处理的对象是各种图；

② 元素间的关系是复杂的图或网状关系，是一种多对多的关系；

③ 施加于对象上的操作有遍历、查找、插入、删除等。

上述三个例子表明，描述这类非数值计算问题的数学模型不再是数学方程式，而是诸如表、

树和图之类的数据结构。因此，简单地说，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的学科。

在程序的设计中，数据结构的选择是一个基本的设计考虑因素。一些大型系统的构造经验表明，系统实现的困难程度和系统构造的质量都严重依赖于是否选择了最优的数据结构。通常，确定了数据结构后，算法就容易得到了。有些时候，事情也会反过来，我们要根据特定算法来选择数据结构与之适应。不论哪种情况，选择合适的数据结构都是非常重要的。

至今，数据结构课程已经成为计算机程序设计的重要理论基础，是计算机学科中一门综合性的专业基础课,也是非计算机专业的主要选修课程，同时还是一门考研课程，数据结构前承高级语言程序设计和离散数学，后接操作系统、编译原理、数据库原理等专业课程，为研制开发各种系统与应用软件奠定理论和实践基础。该课程学习的效果不仅关系到后续课程的学习，而且直接关系到软件设计水平的提高和专业素质的培养，在计算机学科教育中有非常重要的作用。

1.2 基本概念和术语

1. 数据（Data）

数据是信息的载体，是描述客观事物的数字、字符，以及所有能输入计算机中的、被计算机程序识别和处理的符号的集合。

例如，数学计算中所用到的整数和实数、文本编辑所用到的字符串等都是数据。随着计算机软、硬件技术的发展，计算机能够处理的对象范围也在扩大，相应地，数据的含义也被拓宽了。文字、图像、图形、声音、视频等非数值型数据也都是计算机可以处理的数据。

2. 数据元素（Data Element）

数据元素是数据中的一个“个体”，是数据的基本单位。在有些情况下，数据元素也称为元素、结点、顶点、记录等。数据元素用于完整地描述一个对象。

例如，一个学生的记录、棋盘中的一个格局、图中的一个顶点等都是一个数据元素。

3. 数据项（Data Item）

数据项是组成数据元素的有特定意义的不可分割的最小单位。如构成一个数据元素的字段、域、属性等都可称之为数据项。数据元素是数据项的集合。

例如，学生信息表中的学号、姓名、性别、专业等即为数据项。

4. 数据对象（Data Object）

数据对象是具有相同性质的数据元素的集合，是数据的一个子集。

例如，非负整数数据对象是自然数集合 N={0,1,2,…}，英文小写字母数据对象是字符集合 $C=\{'a','b',\cdots,'z'\}$ 等。

5. 数据结构（Data Structure）

数据结构通过抽象的方法研究一组有特定关系的数据的存储与处理。数据结构主要研究三个方面的内容，如图 1.3 所示。

① 数据之间的逻辑关系，即数据的逻辑结构；

② 数据及其逻辑关系如何在计算机中存储与实现，即数据的存储结构；

③ 在某种存储模式下，对数据施加的操作是如何实现的，即运算实现。

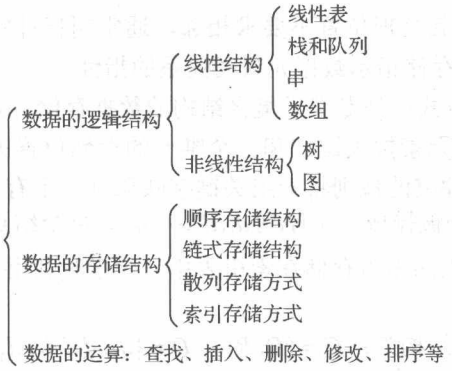


图 1.3　数据结构的三要素

6．数据的逻辑结构

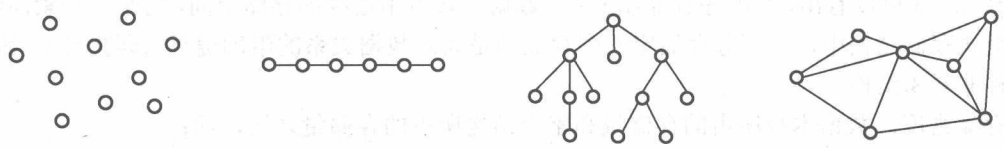
数据的逻辑结构讨论的是数据元素间的逻辑关系，与存储实现无关，是独立于计算机的。常见的逻辑结构如下。

① 集合结构：数据元素间的次序是任意的。数据元素之间除“属于同一集合”的联系外，没有其他的逻辑关系，如图 1.4（a）所示。

② 线性结构：数据元素为有序序列。数据元素之间存在着一种一对一的关系。这种结构的特征是，若结构是非空集，那么，除第一个和最后一个数据元素外，其余数据元素都有且只有一个直接前驱（元素）和一个直接后继（元素），如图 1.4（b）所示。

③ 树结构：数据元素之间存在着一种一对多的关系。在这种结构中，除一个特殊的结点（根结点）外，其他所有结点都有且仅有一个前驱（结点）和零至多个后继（结点），如图 1.4（c）所示。

④ 图结构：数据元素之间存在着一种多对多的关系。在这种结构中，所有结点均可以有多个前驱（结点）和多个后继（结点），如图 1.4（d）所示。图结构也称为网状结构。



（a）集合结构　（b）线性结构　（c）树结构　（d）图结构

图 1.4　数据的逻辑结构示意图

数据（逻辑）结构可形式化定义为：Data _ Structure＝(D, R)。在数据结构的二元组中，D 是数据元素的有限集合，R 是 D 上的关系的有限集合。其中，每个关系都是从 D 到 D 的关系。在表示每个关系时，用尖括号表示有向关系，如$<a,b>$表示存在结点 a 到结点 b 之间的关系；用圆括号表示无向关系，如(a,b)表示既存在结点 a 到结点 b 之间的关系，又存在结点 b 到结点 a 之间的关系。

7．数据的存储结构

讨论数据结构的目的是在计算机中存储数据并实现对它的操作，因此还需要研究数据及其逻辑关系如何在计算机中存储与实现，即数据的存储结构（也称物理结构）。常用的存储结构说明如下。

① 顺序存储结构：借助数据元素在存储器中的相对位置来表示数据元素之间的关系，通常用数组实现。

② 链式存储结构：借助表示数据元素存储地址的指针显式地指出数据元素之间的逻辑关

系，逻辑上相邻的数据元素其物理位置不要求相邻，通常用指针实现。采用链式存储结构，除存储数据元素本身之外还要存储指示数据元素间关系的指针。

③ 散列（哈希）存储方式：是专用于集合结构的数据存储方式。在散列存储方式中，用一个散列（哈希）函数将数据元素按关键字和一个唯一的存储位置关联起来。方法是，根据设定好的散列函数 f(key)和处理冲突的规则将一组关键字映象到一个有限的连续的地址集(区间)上。

④ 索引存储方式：数据被排成一个序列 $d_1, d_2, \cdots, d_n$，每个结点 d_i 在序列里都有对应的位序 $i(1 \leqslant i \leqslant n)$，位序可以作为结点的索引存储在索引表中。检索时利用结点的顺序号 i 来确定结点的存储地址。

【例 1.4】 有一个数据结构为：$G=(D,R)$，$D=\{d_1, d_2, d_3, d_4, d_5\}$，$R=\{\langle d_1, d_2\rangle, \langle d_2, d_3\rangle, \langle d_3, d_4\rangle, \langle d_4, d_5\rangle\}$，集合 D 中的数据元素具有线性关系，假定每个数据元素占 4 个存储单元，则顺序存储结构如图 1.5 所示（结点 d_1 放在 2000H 号单元中），链式存储结构如图 1.6 所示（头指针 head 指向 2040H 号单元）。

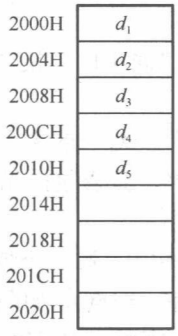


图 1.5 顺序存储结构

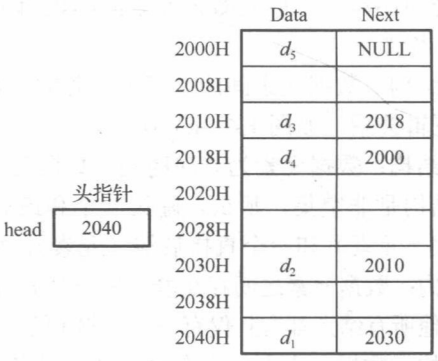


图 1.6 链式存储结构

从图 1.5 可以看出，在顺序存储结构中，数据元素占用连续的存储空间，它是一种紧凑结构。而在链式存储结构中，一部分存储空间中存放的是表示数据关系的附加信息（即指针），因此这是一种非紧凑结构。

存储密度：数据本身所占的存储量和整个结构所占的存储量之比，即：

$$d=\frac{\text{数据本身所占的存储量}}{\text{整个结构所占的存储量}}$$

由此可见，紧凑结构的存储密度可以达到 1，非紧凑结构的存储密度小于 1。存储密度越大，存储空间的利用率越高。但是，非紧凑结构中存储的附加信息会给某些运算带来极大的方便，例如，在进行插入、删除等运算时，链式存储结构比顺序存储结构就方便得多。非紧凑结构牺牲了存储空间，换取了时间。

同一个逻辑结构可以采用不同的存储结构实现，如何选择合适的存储结构要根据实际的需求和具体应用而确定。

8. 数据的运算

① 创建：创建某种数据结构。

② 清除：删除数据结构。

③ 插入：在数据结构指定的位置插入一个新数据元素。

④ 删除：将数据结构中的某个数据元素删去。

⑤ 搜索：在数据结构中搜索满足特定条件的数据元素。

⑥ 更新：修改数据结构中某个数据元素的值。

⑦ 访问：访问数据结构中的某个数据元素。

⑧ 遍历：按照某种顺序访问数据结构中的每个数据元素，使每个数据元素恰好被访问一次。

自测题 1．在数据结构中，从逻辑上可以将之分为（　　）。

A．动态结构和静态结构　　B．紧凑结构和非紧凑结构

C．内部结构和外部结构　　D．线性结构和非线性结构

【2005 年中南大学】

【参考答案】D

自测题 2．已知表头元素为 c 的单链表在内存中的存储状态如下表所示。

地址	元素	链接地址
1000H	a	1010H
1004H	b	100CH
1008H	c	1000H
100CH	d	NULL
1010H	e	1004H
1014H		

现将 f 放于 1014H 处并插入单链表中，若 f 在逻辑上位于 a 和 e 之间，则 a，e，f 的链接地址依次是（　　）。

A．1010H, 1014H, 1004H　　B．1010H, 1004H, 1014H

C．1014H, 1010H, 1004H　　D．1014H, 1004H, 1010H

【2012 年全国统一考试】

【参考答案】D

1.3　算法和算法分析

Pascal 之父、结构化程序设计的先驱 Niklaus Wirth 教授于 1984 年凭借一本专著获得了图灵奖，这本专著的名字是“*Algorithms + Data Structures = Programs*”（算法+数据结构=程序），这里的数据结构指的是数据的逻辑结构和存储结构，而算法则是对数据运算的描述。由此可见，算法与数据结构关系紧密，选择的数据结构是否恰当将直接影响算法的效率，而数据结构的优劣由算法的执行来体现。程序设计的实质是，对要处理的实际问题选择一种合适的数据结构，再设计一个好的算法。

1.3.1　算法的定义及特性

算法（Algorithm）是对特定问题求解步骤的一种描述，是指令的有限序列。其中每条指令表示一个或多个操作。简单地说，算法就是解决特定问题的方法。描述一个算法可以采用文字叙述，也可以采用传统流程图、N-S 图或 PAD 图等。本书采用 C++语言描述。

算法与数据结构的关系紧密，在进行算法设计时首先要确定相应的数据结构。如果在 100 个杂乱无章的数中查找一个给定的数，则只能用顺序查找的方法，效率较低。但是，如果这 100 个数已经按照从小到大的顺序排列好，则可以采用折半查找的方法，这显然比顺序查找的效率要高得多。

一个算法具有下列重要特性。

（1）有穷性：算法只执行有限步，并且每步应该在有限的时间内完成。这里“有限”的概念不是纯数学的，而是指在实际应用中合理的和可接受的。例如，一个简单的算法程序不应该在一个月的计算时间内还不能完成。

以下算法不符合有穷性。

[代码 1.1]

```
void loopforever {
    while(1)
        cout<<"do nothing";
}
```

（2）确定性：算法中的每条指令必须有确切的含义，无二义性。在任何条件下，算法只有唯一的一条执行路径，即对于相同的输入只能得出相同的输出。

（3）可行性：算法中描述的操作都必须足够基本，也就是说，可以通过已经实现的基本运算执行有限次来实现。

例如，“把两个变量交换”和“将变量 a 的值增 1”的算法，就是足够基本的，是可行的；而“把两个变量 a 和 b 的最大公因子 s 送给变量 c”的算法就不是足够基本的，是不可行的。

（4）输入：算法具有零个或多个输入，也就是说，算法必须有加工的对象。输入取自特定的数据对象的集合。输入的形式可以是显式的，也可以是隐式的。有时，输入可能被嵌入在算法中。

（5）输出：算法具有一个或多个输出。这些输出与输入之间有某种确定的关系。这种确定关系就是算法的功能。举例如下。

[代码 1.2]

```
int getsum(int num) {
    int  sum = 0;
    for (i = 1;i <= num; i++)
        sum += i;
    return sum;
}
```

无输出的算法没有任何意义。

算法和程序十分类似，但是也有区别：

① 在执行时间上，算法所描述的步骤一定是有限的，而程序可以无限地执行下去。因此程序并不需要满足上述的第一个条件（有穷性）。例如，操作系统程序是一个在无限循环中执行的程序，因而不是一个算法。

② 在语言描述上，程序必须采用规定的程序设计语言来书写，而算法没有这种限制。

1.3.2 算法的设计要求

要设计一个好的算法通常要考虑达到以下目标。

（1）正确性（Correctness）。算法的执行结果应当满足预先规定的功能和性能要求。正确性是设计和评价一个算法的首要条件。如果一个算法不正确，则不能完成所要求的任务，其他方面也就无从谈起。一个正确的算法，是指在合理的数据输入下，能够在有限的运行时间内得出正确的结果。采用各种典型的输入数据，上机反复调试算法，使得算法中的每段代码都被测试过，若发现错误则及时纠正，最终可以验证出算法的正确性。当然，要从理论上验证一个算法的正确性，并不是一件容易的事，也不属于本课程所研究的范围，故不进行讨论。

（2）可读性（Readability）。是指算法描述的可读性。算法描述主要是为了方便人的阅读与交流，其次才是让计算机执行。因此算法描述应该思路清晰、层次分明、简捷明了、易读易懂，必要的地方加以注释。此外，晦涩难懂的算法描述易于隐藏较多的错误而难以调试和修改。可读性还要求对算法描述中出现的各种自定义变量和类型做到“见名知义”，即读者一看到每个变量（或类型名）就知道其功用。总之，算法描述的可读性不仅能让读者理解算法的设计思想，同时也可以方便算法的维护。

（3）健壮性（Robustness）。这是指一个算法对不合理（又称不正确、非法、错误等）的数据输入的反应和处理能力。一个好的算法应该能够识别错误数据并进行相应的处理。对错误数据的处理一般包括打印错误信息、调用错误处理程序、返回标识错误的特定信息、终止程序运行等。对错误输入数据进行适当处理，使得不至于引起严重后果。例如，数组都是有上下界的，当访问的下标越过该界限时，系统应该给出保护性错误提示，如直接返回一个错误指示，以避免“数组越界的错误”。

（4）高效性（Efficiency）。算法应有效地使用存储空间并且有较高的时间效率。两者都与问题的规模有关。

1.3.3 算法效率的衡量方法

解决同一个问题，可能存在多个算法，而最重要的计算资源是时间和空间，因此，评价一个算法优劣的重要依据是：程序所用算法运行时花费的时间代价和程序中使用的数据结构占有的空间代价。算法设计者需要在两者之间折中。进行算法性能分析的目的是寻找高效的算法来解决问题，提高工作效率。那么，如何评估各算法的好坏？或者据此设计出更好的算法呢？

衡量算法效率的方法主要有两大类：算法的事后统计方法（后期测试）和算法的事前分析估算方法。

（1）事后统计方法：利用计算机的时钟进行算法执行时间的统计。在算法中的某些部位插入时间函数 time ( )来测定算法完成某一功能所花费的时间。这种方法有非常明显的缺陷：首先，必须把算法转变为程序来执行；其次，时间统计依赖于硬件和软件环境，这容易掩盖算法本身的优劣。

（2）事前分析估算方法：用高级语言编写的程序，其运行时间主要取决于以下因素。

① 算法选用的策略。

② 问题的规模。随着问题涉及的数据量增大，处理起来会越来越困难、复杂。我们用问题规模来描述数据量增大程度。规模越大，消耗时间越多。例如，求 100 以内的质数和求 10000 以内的质数，其执行时间显然是不同的。

③ 编写程序的语言。对于同一个算法，实现语言的级别越高，其执行效率就越低。

④ 编译程序所产生的目标代码的质量。对于代码优化较好的编译程序，其所生成的程序质量较高。

⑤ 机器执行指令的速度。

显然，上述因素中后面三个与算法设计是无关的。也就是说，同一个算法用不同的语言实现，或者用不同的编译程序进行编译，又或者在不同的计算机上运行，其效率均不相同。这表明，使用绝对的时间单位来衡量算法的效率是不合适的。去掉这些与计算机硬件、软件有关的因素，可以认为，一个特定算法的“运行工作量”的大小，只依赖于问题的规模（通常用整数 n 来表示），或者说：它是问题规模的函数。