高等院校计算机应用系列教材

数据结构

(C 语言，慕课版)

主　编　殷　超　李庆印

副主编　肖爱梅　郑明文　麻云轩

梁志睿　王红霞

清華大學出版社

北　京

内 容 简 介

“数据结构”是计算机、信息技术等相关专业的一门重要的专业基础课程、核心课程。本书内容适应MOOC+SPOC 线上线下混合式教学模式，贴近当前普通高等院校“数据结构”课程的现状和发展趋势；符合研究生考试大纲要求，难度适中，通俗易懂；书中案例典型、丰富，结构清晰，重难点突出。本书内容共分 13 章，主要包括数据结构概述，算法分析基础，线性表，栈，队列，串，数组，广义表，树，二叉树，图，查找与排序等。每章均提供了线上资源，读者可通过扫描本书提供的二维码，使用配套课程MOOC，进行线上学习，参加小节弹题测试、章节测试、章讨论、课程测试，并获得定期在线答疑服务。

本书可作为普通高等院校计算机专业、信息与计算科学专业等相关专业“数据结构”课程的教材，也可供准备参加计算机专业研究生考试人员，以及从事计算机软件开发和应用的工程技术人员阅读和参考。

图书在版编目(CIP)数据

数据结构：C 语言：慕课版 / 殷超, 李庆印主编. —北京：清华大学出版社，2024.7
高等院校计算机应用系列教材
ISBN 978-7-302-65700-2

Ⅰ. ①数… Ⅱ. ①殷… ②李… 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.①TP311.12②TP312.8

中国国家版本馆 CIP 数据核字(2024)第 051123 号

责任编辑：王 定
版式设计：思创景点
封面设计：周晓亮
责任校对：马遥遥
责任印制：曹婉颖

出版发行：清华大学出版社
网 址：https://www.tup.com.cn，https://www.wqxuetang.com
地 址：北京清华大学学研大厦 A 座 邮 编：100084
社 总 机：010-83470000 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
印 装 者：三河市东方印刷有限公司
经 销：全国新华书店
开 本：185mm×260mm **印 张**：19 **字 数**：474 千字
版 次：2024 年 9 月第 1 版 **印 次**：2024 年 9 月第 1 次印刷
定 价：69.80 元

产品编号：098296-01

前　言

党的二十大报告对“实施科教兴国战略，强化现代化建设人才支撑”作出专章部署，为新时代教育工作和科技创新工作指明了前进方向，为加快建设教育强国提供了根本遵循和行动指南。党的二十大报告强调：“教育、科技、人才是全面建设社会主义现代化国家的基础性、战略性支撑。必须坚持科技是第一生产力、人才是第一资源、创新是第一动力，深入实施科教兴国战略、人才强国战略、创新驱动发展战略，开辟发展新领域新赛道，不断塑造发展新动能新优势。”

数据结构作为计算机科学的核心基础学科，对于推动计算机技术的进步和科技创新具有至关重要的作用。高等院校作为人才培养和科技创新的重要基地，应加强对数据结构等基础科学的研究和教育，为培养高素质的科技人才和推动科技创新提供有力支持。本书遵循“两性一度”标准，贯彻“立德树人”的教育本质，全面融入课程思政，旨在培养读者的科学精神和工程设计能力，突出专业课程的价值引领功能作用。

本书内容涵盖“数据结构”课程所有知识点，紧贴研究生入学考试“数据结构”课程大纲要求，内容围绕常见的数据结构和基本数据操作，共分 13 章，主要包括数据结构概述，算法分析基础，线性表，栈，队列，串，数组，广义表，树，二叉树，图，查找与排序等。本书采用类 C 语言作为数据结构和算法的描述语言，在对数据的存储和算法描述时，充分考虑 C 语言的特色，同时兼顾数据结构和算法的可读性。读者在实际上机操作时，可以很容易地将本书中的数据结构和算法转换成 C 语言程序或其他程序设计语言程序。

本书对理论知识的阐述由浅入深、语言通俗易懂，既着眼于数据结构基础，又突出课程重难点。采用提出问题、分析问题、解决问题的问题求解过程，以及问题分析、得出算法思想、算法描述的三级递进模式讲解算法，降低了理解算法的复杂性，帮助读者提高认知效率。

本书内容适应 MOOC+SPOC 线上线下混合式教学模式，贴近当前高等院校“数据结构”课程的现状和发展趋势，书中案例典型、丰富，结构清晰，线上线下资源非常丰富。课程 MOOC 已上线国家高等教育智慧教育平台、山东省课程联盟、智慧树在线教育平台。读者可通过扫描下方二维码，注册后使用配套课程 MOOC，进行线上学习；在线上可参加

小节弹题测试、章节测试、章讨论、课程测试。另外，本书还提供配套MOOC课程、MOOC视频、教学课件、教学大纲、习题参考答案等教学资源，读者可通过扫描相应二维码获取。


MOOC 课程


MOOC 视频


教学课件


教学大纲


习题参考答案

本书配套的MOOC视频由编者及郑明文、王红霞、何华共同录制，多名本科生参与了算法调试与文稿校对工作，在此表示衷心感谢！

因编者水平有限，书中不足之处在所难免，恳请专家和读者不吝指正。

编　者

2024年5月

目　录

第1章 绪 论

自 1946 年第一台计算机问世以来，计算机产业的飞速发展已远远超出人们的预料。如今，计算机已深入人类社会的各个领域。计算机的应用已不再局限于科学计算，更多地应用于控制、管理及数据处理等非数值计算的领域。与此相应，计算机加工处理的对象由纯粹的数值发展到字符、表格和图像等具有一定结构的数据，这就给程序设计带来了新的问题。为了编写出一个“好”的程序，编程人员必须分析待处理的对象的特性，以及各处理对象之间存在的关系。

1.1 数据结构的发展

早期的计算机主要应用于科学计算领域。随着计算机的发展和应用范围的拓展，计算机需要处理的数据量越来越大，数据的类型越来越多，结构越来越复杂。计算机处理的对象从简单的纯数值性数据发展为非数值和具有一定结构的数据。计算机加工处理的对象逐渐受到重视和研究。人们开始研究数据的特性、数据之间存在的关系，以及如何有效地组织、管理、存储数据，从而提高计算机处理数据的效率。“数据结构”学科就是在此背景下逐渐形成和发展起来的。

最早对数据结构学科发展作出杰出贡献的是 D.E.Kunth 教授和 C.A.R.Hoare 教授。D.E.Kunth 教授的《计算机程序设计技巧》和 C.A.R.Hoare 教授的《数据结构札记》对数据结构学科的发展作出了重要贡献。随着计算机科学的飞速发展，到 20 世纪 80 年代初期，数据结构的基础研究日臻成熟，已经成为一门完整的学科。

数据结构起源于在程序设计中应如何组织待处理的数据及数据之间的关系(结构)。

目前，面向各专门领域中特殊问题的数据结构正在得到研究，如多维图形数据结构等，各种空间数据结构也在被探索中。另外，从抽象数据类型和面向对象的观点来讨论数据结构已成为一种新的趋势，且越来越被人们所重视。

数据结构随着程序设计的发展而发展。程序设计经历了 3 个阶段：无结构阶段、结构化阶段和面向对象阶段。相应地，数据结构的发展也经历了 3 个阶段，如图 1-1 所示。

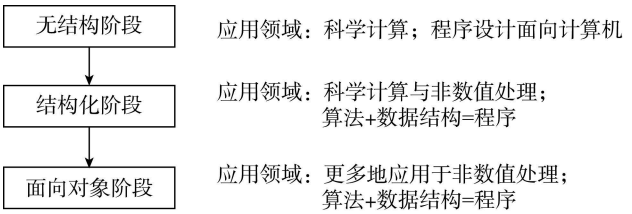


图 1-1　数据结构的 3 个发展阶段

1.2　数据结构的概念

人们在使用计算机解决实际问题时，总是希望计算机越智能越好。但是可能多数使用者没有去思考两个问题，即计算机的智能是如何得来的？如何使计算机智能提高？为了更好地回答这两个问题，先看一个简单的例子。

【例 1-1】已知集合 A={1，3，4，6，7，8，97}，B={1，3，5，7，8，10，12}，求集合 A 和集合 B 的交集。

对于这个问题的求解，如果运用人脑，是一道非常简单的题目。如果借助计算机求解，应该如何处理呢？

借助计算机解决实际问题的一般步骤如下：

(1) 定义问题。分析问题是什么，明确问题要求是什么，理解问题是做什么。

(2) 建立模型。将实际问题中的客观对象的属性及联系，抽象成逻辑数据模型。

(3) 定义数据。将数据模型的对象定义成计算机能存储处理的存储结构。

(4) 设计算法。根据存储结构，找出求解问题的策略和方法步骤。

(5) 编写程序。将算法使用程序设计语言表示出来。

(6) 调试运行。将数据和程序输入计算机，查错修改，运行得到结果。

(7) 分析结果。计算结果是否符合要求，若符合则结束，否则，返回检查修改。

在上述 7 个步骤中，从步骤(1)到步骤(5)是人工工作部分，步骤(6)也不全是计算机工作，人工要对程序进行调试，步骤(7)也是人工工作。可见，整个过程中人工工作占绝大部分。在人工工作步骤中，建立模型和设计算法是最关键且较困难的两个步骤。借助计算机解决问题的完整步骤如图 1-2 所示。

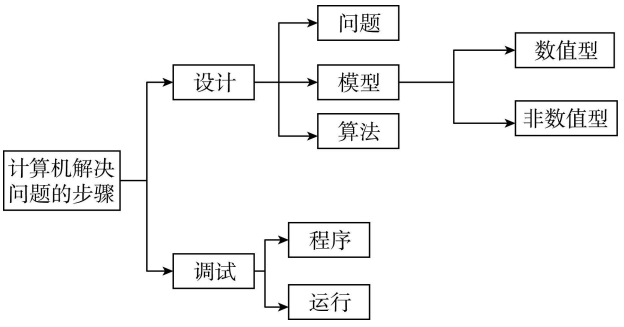


图 1-2　计算机解决问题的步骤

例如，已知圆的半径 r，求圆的周长 C，求解公式为 $C=2\pi r$ 。根据抽象出来的数学模型，在算法中，输入半径 r，由公式求出周长 C。然后编程，得出程序，将程序在计算机上调试运行，得出运行结果。最后验证这个结果是否满足原问题的要求。在整个过程中，得出问题模型是至关重要的一步，而从实际问题抽象出问题模型的过程就是构建实际问题的数据结构的过程，问题的数据结构也就是它的模型。

本例抽象出来的数学模型是求圆的周长，属于数值型模型。

除数值型问题外，现实中还有非常多的非数值型问题(即不能用数学公式表达出来的问题)。例如，学生信息管理系统中的记录文件结构是一种典型的线性结构，不能用一个数学公式的形式表达出来，因此称为非数值型问题。家谱结构是一种典型的非数值的树形结构；交通网络结构是典型的非数值的图状结构。

简而言之，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，以及对象之间的关系和操作的学科。

1.2.1　数据结构研究的领域

在实际应用中，对数据的操作不单纯是数值计算(仅占计算机数据处理的10%)，如求函数值、求方差等，更多的是非数值计算，如检索、排序、插入、删除等操作。

数值计算问题在“数值分析”(又称“计算方法”)学科中有专门的研究。非数值计算问题是“数据结构”学科所要讨论的主要内容。建立模型和算法设计就是数据结构学科重点研究的两个领域。下面先来了解几个非数值计算的问题。

1. 学生学籍信息系统

在学生学籍信息系统中，当需要查询某位学生的档案信息时，在学籍文件中一般不会从头开始查找，因为这样费时且耗力。每位学生都有唯一的学号，所以可以按照学号分类去查找。表 1-1 是依据学生的学籍信息整理成的表格，每一串学号对应一位学生，计算机可以按照特定的学号信息查找指定的学生。与之类似的还有图书管理、人事管理、物资管理、商品管理等领域的各种系统。在这类文档管理的数学模型中，计算机处理的对象之间通常存在着一种简单的线性关系，这类数学模型可被称为线性数据结构。

表 1-1　学生学籍文件

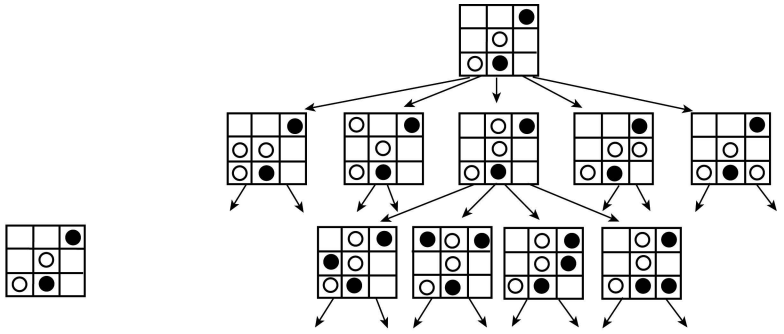
学　号	姓　名	性　别	出生日期	政治面貌
22040101	王　飒	男	2003/09/02	团员
22040102	李凤黔	男	2002/12/25	党员
22040103	吴颖霖	女	2003/03/26	团员
…	…	…	…	…

2. 计算机对弈问题

计算机之所以能和人对弈，是因为程序设计者将对弈的策略事先存入了计算机。由于对弈的过程是在一定的规则下随机进行的，为使计算机能灵活对弈，就必须对对弈过程中所有可能

发生的情况及相应的对策考虑周全。并且，一个“好”的棋手在对弈时不仅要看棋盘当时的状态，还应该能够预测棋局的发展，甚至最后的结局。

例如，如图 1.3(a)所示为井字棋的一个格局，格局之间的关系是由比赛规则决定的。通常，这个关系不是线性的，因为从一个棋盘格局可以派生出几个格局，如从图 1.3(a)所示的格局中可以派生出 5 个格局，如图 1.3(b)所示，而从每一个新的格局又可以派生出 4 个可能的格局。因此，若将从对弈开始到结束的过程中所有可能出现的格局都画在一张图上，则可得到一棵倒长的“树”。“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杈到某个叶子的过程。计算机游戏、组织机构的层次结构等许多实际问题都可以抽象成“树”状数据结构。



(a) 井字棋的一个格局　　(b) 对弈树的局部

图 1-3　对弈问题中格局之间的关系

3. 赛程安排问题

设某田径比赛共有 6 个比赛项目，规定每个选手至多可参加 3 个项目，有 5 人报名参加比赛，如表 1-2 所示。设计比赛日程表，使比赛能在尽可能短的时间内完成。分别用 A、B、C、D、E、F 这 6 个不同的代号代表跳高、跳远、标枪、铅球、100 米、200 米 6 个不同的项目；用顶点代表比赛项目；在不能同时进行比赛的顶点之间连上一条边(同一选手参加的项目之间必定有边相连)；最后给顶点涂色，任何有边相连的顶点不能涂同一种颜色，且使涂色数目尽量少。图 1-4 建立了一种图状数据结构。由涂色结果可得，仅需要 4 个时间段即可完成所有比赛赛段，如表 1-3 所示。像课程安排、工程管理等大量问题均可以抽象成图状数据结构。

表 1-2　项目报名表

姓 名	项目 1	项目 2	项目 3
丁 一	跳 高	跳 远	100 米
刘 二	标 枪	铅 球	—
张 三	标 枪	100 米	200 米
李 四	铅 球	200 米	跳 高
王 五	跳 远	200 米	—

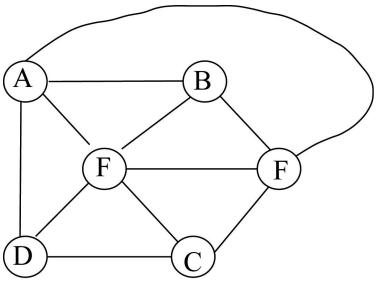


图 1-4　赛程安排结点示意图

表 1-3　赛程时间表

比赛时间	比赛项目
1	A, C
2	B, D
3	E
4	F

1.2.2　数据结构研究的内容

根据数据结构的研究对象和研究领域，数据结构研究的主要内容概括如下。

(1) 数据的逻辑结构，即数据间的联系。指设计者按数据间的内在逻辑关系和适当结构对数据的描述和组织。

(2) 数据的存储结构。要借助计算机解决问题，就要首先将待处理的数据存储在计算机内，所以数据的存储结构是从系统实现角度，将逻辑结构映射成存储结构并在计算机内表示的形式。

(3) 数据的操作算法。借助计算机解决问题一定是对所存储的数据进行的一系列操作，因此，数据的操作算法就是在某种存储结构下给出算法的具体实现。

(4) 算法的效率分析。同一个问题的解决方法不一定是唯一的，而不同的算法有各自的优劣，算法的效率分析就是分析算法的时间和空间效率，以此来评价算法的优劣。

(5) 数据结构的应用。这部分是上述 4 个部分的综合，利用数据结构内容解决实际问题。

1.2.3　数据

数据是对客观事物的符号表示。在计算机科学中，数据是指所有能够输入计算机并能被计算机程序处理的符号集合，包括数值、文字、图像、音频、视频等形式。

数据项是数据中具有独立含义的、不可再分割的最小数据单位，是数据的一个子集，是客观实体一种特征的数据表示。

数据元素是数据的基本单位，一般作为一个整体来处理。数据元素是由多个相关数据项组成的集合，是一个客观实体多种特征的数据描述，是计算机程序处理的基本单位。数据元素按其组成可分为原子型数据元素和组合型数据元素。原子型数据元素由一个数据项组成。组合型数据元素由多个数据项组成，通常携带着一个实体的多方面信息。

例如，在大写字母表中，有 26 个字符，每个字符既是数据元素，又是数据项。

【例 1-2】在学生信息管理系统中，每一条记录是一个数据元素，在每条记录中又包含姓名、学号、专业等数据项。如表 1-4 所示为学生信息管理系统中的基本信息。整张表就是数据，也称为数据对象，每一行称为一个记录，就是一个数据元素，一般在操作过程中作为一个整体被进行处理；行和列交叉的地方称为数据项，也称为域或字段。

表 1-4　学生信息管理系统中的基本信息

序号	姓名	学号	专业	电话
0001	张伟	XK385190106	信科	135********
0002	刘丽	RJ426180201	软件	150********
0003	王一	TX417190124	通信	152********
0004	赵鹏	SX332200310	数学	198********
0005	李娜	JT7111801035	交通	135********
0006	孙明	XK385190107	信科	138********
…	…	…	…	…

1.2.4　数据结构

数据结构，就是相互之间存在一种或多种特定关系的数据元素的集合。数据结构中的结构是指关系，可以简单表示为数据结构=数据+关系。数据结构也可以理解为带结构的数据集合。同一数据元素集合，所定义的关系不同，构成的数据结构也不同。那么，数据之间都有哪些关系呢？

数据结构包括逻辑结构和存储结构两个方面。

1. 数据的逻辑结构

数据的逻辑结构是对数据之间的逻辑关系的一种描述。它和数据的存储无关，是独立于计算机的。因此数据的逻辑结构可以看作是从具体问题中抽象出来的数据模型。换句话说，就是从实际非数字计算应用问题的现象中提炼出来的本质结论，这是从唯物辩证法的基本范畴之现象与本质的基本概念和相互之间的辩证关系角度得到的启发，由此可以加深对这一类理论知识的理解与应用，增强利用辩证思想思考和解决实际生活中的具体问题的能力。

数据的逻辑结构可以用一个数据元素的集合和定义在这个集合上的若干关系表示，并且可以用二元组来描述，即 $D\_S=(D,S)$，D 是数据元素的集合，S 是关系的集合。

【例 1-3】在学生考勤管理系统中，由班长对组长考勤，由组长对组内的成员考勤。假设某班有 1 个班长、4 个组长、每组有 8 位组员，则可以如下定义数据结构。

$$\text{Group}=(D,S)$$

其中，$D=\{M,G_1,\cdots,G_4,N_{11},\cdots,N_{mn}$，$1\leqslant m\leqslant 4$，$1\leqslant n\leqslant 8\}$；

$S=\{R_1,R_2\}$；

$R_1=\{\langle T,G_i\rangle \mid 1\leqslant i\leqslant 4\}$；

$R_2=\{\langle G_i, N_{mn}\rangle \mid 1\leqslant i\leqslant 4, 1\leqslant m\leqslant 4, 1\leqslant n\leqslant 8\}$ 。

数据元素的集合 D 包含班长 M，N_{11}～N_{18} 是第 1 组，组长是 G_1；再加上第 2 组 N_{21}～N_{28}，组长是 G_2；第 3 组是 N_{31}～N_{38}，组长是 G_3；第 4 组是 N_{41}～N_{48}，组长是 G_4。

数据关系的集合 S 由两种关系组成：R_1 和 R_2。关系 R_1 是指班长对 4 位组长考勤，尖括号括起来的字符，表示序偶关系，它是有顺序的，M 和 G_1 之间的序偶表示班长 M 对组长 G_1 考勤。同理 M 和 G_2、G_3、G_4 之间也有序偶关系。关系 R_2 是指组长对组员考勤，G_1 是组长，要对 N_{11}～N_{18} 考勤，所以 G_1 和 N_{mn}(m=1，1≤n≤8)之间存在序偶关系。同理 G_2 对 2 组成员之间，……，G_4 对 4 组成员之间也存在序偶关系。

逻辑结构二元组中的关系的集合 S，是数据结构分类的主要依据，根据数据元素之间关系的不同，数据的逻辑结构分为以下 4 种。

(1) 集合结构：数据元素之间未定义任何关系的松散集合。数据关系的集合 S 是空集，数据元素之间的关系是离散的，它们除同属于一个集合外，不存在其他关系。集合结构如图 1-5 所示。

(2) 线性结构：数据元素之间定义了次序关系的集合(全序集合)，描述的是一对一关系，也就是线性关系。例如，春、夏、秋、冬 4 个季节是一种线性关系。线性结构如图 1-6 所示。

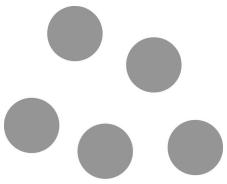


图 1-5　集合结构

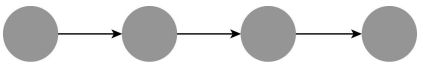


图 1-6　线性结构

(3) 树形结构：数据元素之间定义了层次关系的集合(偏序集合)，描述的是一对多关系。例如，传统的家谱是一种典型的树形结构。树形结构如图 1-7 所示。

(4) 图状结构：数据元素之间定义了网状关系的集合，描述的是多对多关系。例如，求最短路径的交通网络，它的逻辑结构就是多对多的图状结构。图状结构如图 1-8 所示。

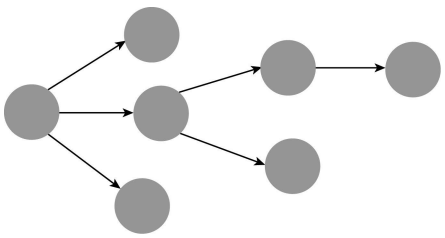


图 1-7　树形结构

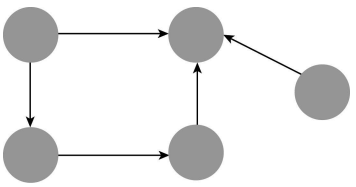


图 1-8　图状结构

2. 数据的存储结构

数据的存储结构(亦称物理结构)是数据(逻辑)结构在计算机存储器中的具体实现形式。存储结构与孤立的数据元素表示形式不同，数据结构中的数据元素不但要表示其本身的实际内容，还要表示数据元素之间的逻辑结构。

存储结构主要有两种：一种是顺序结构，另一种是非顺序结构。顺序结构需要有连续的存储空间，数据元素在逻辑与物理上是都相邻的，一般用数组描述；非顺序结构不需要连续的存

储空间，在逻辑上相邻的数据元素，在物理上不一定相邻，一般用指针类型描述。

常见的存储结构有以下几种。

(1) 顺序存储结构。其特点是借助数据元素的相对存储位置来表示数据元素之间的逻辑结构。在顺序结构中，通过数据元素物理位置的相邻来描述数据元素逻辑上的前驱、后继关系。

(2) 链式存储结构。其特点是借助指示数据元素地址的指针来表示数据元素之间的逻辑结构。在链式存储结构中，通过指针来指示数据元素逻辑上的前驱、后继关系。

(3) 散列存储结构。其特点是可以通过结点的关键字直接计算出该结点的存储地址。通过把关键字映射到表中的指定位置来直接访问记录，以加快访问速度。这个映射表也称为散列函数或哈希函数，存放记录的数组称为哈希表或散列表。

(4) 索引存储结构。其特点是采用附加的索引表的方式来存储结点信息。索引表由若干索引项组成。索引存储方式中索引项的一般形式为(关键字，地址)。其中，关键字是能够唯一标识一个结点的数据项，地址指示一组结点的起始存储位置。

1.2.5 数据类型

数据类型是一个数据值的集合和定义在这个值的集合上的一组操作的总称，即数据类型=数据值的集合+一组操作的集合。

例如，对于整型变量 x，若采用 16 位来存储整型，则数据值的集合 x 的取值范围为−32768～32767；整型的操作集合有加、减、乘、除、取模等。这就是数据类型的实例——整型。

抽象数据类型(abstract data type，ADT)是一个数学模型及定义在该模型上的一组操作，即抽象数据类型=数学模型+一组操作的集合。

抽象数据类型是从问题抽象出来的逻辑结构和运算，抽象数据类型不考虑具体的存储结构和操作实现。

可以使用三元组(D, S, P)来表示抽象数据类型，其中 D 表示数据对象的集合，S 表示数据关系的集合，P 表示基本操作的集合；集合 D 和 S 是数据结构逻辑层面所包含的二元组：数据元素的集合和数据关系的集合。加上基本操作集合 P，三者便构成了抽象数据类型 ADT。

抽象数据类型的描述格式如下。

```
ADT  <抽象数据类型名称 class>
{
    数据元素定义：给出数据元素的特性确切描述；
    数据关系定义：给出数据元素之间关系的确切描述；
    数据操作定义：给出施加在数据元素上的各种操作的确切描述；
} <抽象数据类 class>
```

数据操作的形式定义如下。

```
函数返回值类型   函数名(参数表)

{
    操作过程表示；
}
```

【例 1-4】现有部分通讯录如表 1-5 所示。通讯录即一个数据结构，其中，一行表示一条记录，用结点表示。每条记录由姓名、区号和电话号码 3 个数据项组成。

表 1-5　通讯录表

姓名	区号	电话号码
赵一	010	53644587
钱二	020	89634159
孙三	021	45976528
李四	024	63427544

分析表 1-5 的逻辑结构可知，表的第一行“赵一”所在的结点是首结点，它没有直接前驱结点；最后一行“李四”所在的结点是尾结点，它没有直接后继结点；中间两行是内部结点，它们各有一个直接前驱结点和一个直接后继结点。显然，通讯录表的逻辑结构是线性结构。

若通讯录表(简称通讯录)主人结识新友，要把新友信息(包括姓名、区号和电话号码)添加到通讯录中，则须对该表进行插入操作。思考新结点可能的插入位置有哪些？若通讯录中有联系人更新了电话号码，则须对该表进行修改操作；若要查找某联系人的电话号码，则须对该表进行查找操作。

【例 1-5】如图 1-9 所示为××大学专业的设置情况。在图 1-9 中，把大学名称看作树根，把下设的若干学院名看作树枝中间结点，把系别看作树叶，这就形成了一个树形结构。树形结构通常用来表示结点的分层组织，结点之间是一对多的关系。对于树形结构的主要操作有遍历、查找、插入、删除等。

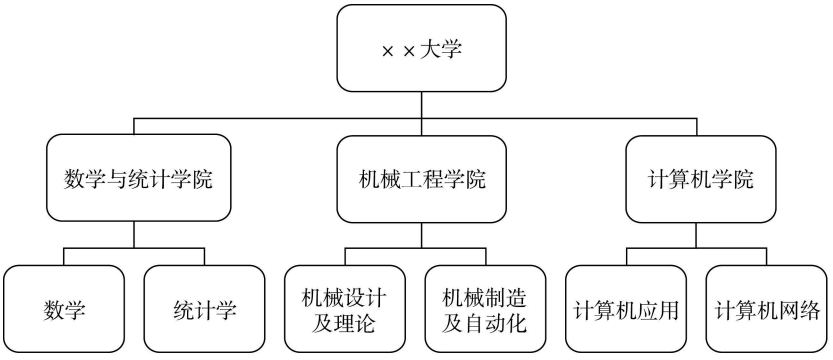


图 1-9　××大学专业的设置情况

1.3　算法和算法分析

算法是指问题解决方案的准确而完整的描述，是一系列解决问题的清晰指令。算法代表着用系统的方法描述解决问题的策略机制，即能够对一定规范的输入，在有限时间内获得所要求的输出。如果一个算法有缺陷，或不适合于解决某个问题，则执行这个算法将不会解决这个问

题。不同的算法可能使用不同的时间、空间或效率来完成同样的任务。一个算法的优劣通常用空间复杂度与时间复杂度来衡量。

1.3.1 算法的概念

1. 算法和算法的特征

算法是针对特定问题求解步骤的一种描述，它是指令的有限序列。算法须具有以下5 个特征。

(1) 输入：算法有零个或多个输入。

(2) 输出：算法至少产生一个输出。

(3) 确定性：算法的每一条指令都有确切的定义，没有二义性。

(4) 可行性：算法的每一条指令都足够基本，可以通过执行有限次已经实现的基本运算来实现。

(5) 有穷性：算法总能在执行有限步骤后终止。

描述一个算法的方法有多种。算法可以用自然语言、流程图或程序设计语言等来描述。当一个算法直接使用计算机程序设计语言描述时，该算法便成为程序。算法必须在执行有限步骤后终止，但计算机程序没有这一限制，如操作系统是一个程序，但不是一个算法。

2. 衡量算法性能的标准

衡量一个算法的性能，主要有以下几个标准。

(1) 正确性：算法的执行结果应当满足预先规定的功能和性能要求。

(2) 简明性：一个算法应当思路清晰、层次分明、简单明了、易读易懂。

(3) 健壮性：当输入不合法数据时，应能进行适当的处理，不至于引起严重后果。

(4) 效率：有效使用存储空间，并具有高时间效率。

其中，算法的正确性是指在合法的输入下，算法应实现预先规定的功能和计量精度要求。算法的健壮性是当程序遇到意外时，能按某种预定的方式进行适当的处理。正确性和健壮性是相互补充的。正确的程序并不一定是健壮的，一个可靠的程序应当能在正常情况下正确工作，而在异常的情况下也能进行适当的处理。这种处理不是输出错误信息或异常，并中止程序的运行，而是返回一个表示错误或错误性质的值，以便在更高的抽象层次上处理。算法的效率通常指算法执行的时间和所需的存储空间。

1.3.2 算法的复杂性分析

1. 算法的复杂性的含义

算法的复杂性是指算法运行所需要的计算机资源的量，所需资源越多，该算法的复杂性越高；反之，所需资源越少，该算法的复杂性越低。对于计算机资源来说，最重要的是时间和空间(存储器)资源。因此，算法的复杂性通常分为时间复杂性和空间复杂性。需要时间资源的量称为时间复杂性，需要空间资源的量称为空间复杂性。

算法的复杂性取决于求解问题的规模、具体的输入数据和算法本身的设计。

为了能够较客观地反映出一个算法的效率，在度量一个算法的工作量时，不仅应该与所使

用的计算机、程序设计语言及程序编制者无关，还应该与算法实现过程中的许多细节无关。因此，可以用算法在执行过程中所需基本运算的执行次数来度量算法的工作量。基本运算反映了算法运算的主要特征，所以，使用基本运算的执行次数来度量算法工作量是客观的也是可行的，有利于比较解决同一问题的不同算法的优劣。

例如，在考虑两个矩阵相乘时，参考下面代码段，可以将两个实数之间的乘法运算 $a[i][k]*b[k][j]$ 作为基本操作。整个算法的执行时间与该操作(乘法)重复执行的次数 n^3 成正比，记作 $T(n)=O(n^3)$。

```
for (i=1; i<=n; ++i)
   for (j=1; j<=n; ++j)
   {
      c[i][j]=0;
      for (k=1; k<=n; ++k)
         c[i][j]+=a[i][k]*b[k][j];        //基本操作
    }
```

2. 算法的时间复杂度

若令 N、I 和 A 分别表示问题的规模、具体的输入和算法本身，用 C 表示复杂度，则 $C=F(N, I, A)$。若将时间和空间分开，分别用 T 和 S 表示，且 A 通常隐含在复杂度函数名中，则可得 T 和 S 简写为：$T=T(N,I)$ 和 $S=S(N,I)$。

时间复杂度 $T(N,I)$ 的计算为

$$T(N,I)=\sum_{e_i}^{t_i}(N,I)$$

其中，t_i 为执行抽象计算机的第 i 种指令一次所需要的时间，这里假定抽象计算机共有 k 种指令，$e_i(N,I)$ 为经过统计后得到的执行抽象计算机的第 i 种指令的次数，即

算法的执行时间 = ∑原子操作的执行次数 × 原子操作的执行时间

简而言之，算法中基本操作重复执行的次数是问题规模 n 的某个函数 $f(n)$，算法的时间复杂度记作

$$T(n)=O(f(n))$$

它表示随着问题规模 n 的增大，算法执行时间的增长率和 $f(n)$ 的增长率相同，称为算法的渐进时间复杂度，即时间复杂度。

算法的时间复杂度又可分为最坏情况下的时间复杂度、最好情况下的时间复杂度及平均情况下的时间复杂度。表达式如下。

最坏情况下的时间复杂度

$$\max T(n)=\max\{T(I)\mid \text{size}(I)=n\};$$

最好情况下的时间复杂度

$$\min T(n)=\min\{T(n)\mid \text{size}(I)=n\};$$

平均情况下的时间复杂度

$$\text{avg}T(n)=\sum_{I=1}^{n}p(I)T(I);$$

其中 $p(I)$ 是 I 出现的概率。

【例 1-6】有嵌套的循环程序段如下。

```
x=0; y=0;
for(k=1; k<=n; k++)
    x++;
    for(i=1; i<=n; i++)
        for(j=1; j<=n; j++)
            y++;
```

该算法段的时间复杂度为 $T(n)=O(n^2)$。

当有若干个嵌套循环语句时，算法的时间复杂度通常由嵌套层数最多的循环语句中最内层语句的执行次数决定的。

【例 1-7】求数组中的最小值，程序段如下。

```
int ArrayMin(int a[ ], int n)
{
    min=a[0];
    for (i=1; i<n; i++)
       if (a[i]<min) min=a[i];
    return min;
}
```

该算法段的时间复杂度为 $T(n)=O(n)$。

如果循环变量只与问题规模 n 有关，则时间复杂度一般为 $O(n)$。

当算法是非递归算法时，在分析其时间复杂度时，可以观察算法的特点：如果是顺序语句，则将各语句的时间复杂度相加；如果是 for 循环或 while 循环，则使用循环体内时间复杂度×循环次数作为算法的时间复杂度；如果是嵌套循环算法，则使用最内层循环体内时间复杂度×所有循环次数作为算法的时间复杂度；如果含有 if-else 语句，则选用 if 语句计算时间和 else 语句计算时间的较大者作为算法的时间复杂度。

比较常见的时间复杂度有 $O(1)$、$O(\log n)$、$O(n)$、$O(n^c)$、$O(c^n)$ 和 $O(n!)$，分别称为常数阶、对数阶、线性阶、多项式阶、指数阶和阶乘阶。它们的时间复杂性从低到高，其中 n 为问题的规模，c 为常量。

3. 算法的空间复杂性

类似于算法的时间复杂度，空间复杂度指算法在运行过程中临时占用存储空间大小的量

度，记作

$$S(n)=O(f(n))$$

一个算法所占用的存储空间要从多个方面综合考虑。例如，对于递归算法来说，算法本身一般都比较短，所占用的存储空间较少；但运行时需要一个附加的工作栈，从而占用较多的临时工作单元。对于非递归算法，算法本身可能比较长，所占用的存储空间较多；但运行时可能需要较少的存储空间。

一个算法通常情况下的空间复杂度，只考虑在运行过程中为局部变量分配的存储空间的大小。它包括为参数表中形参变量分配的存储空间和为在函数体中定义的局部变量分配的存储空间。若一个算法为递归算法，其空间复杂度为递归所使用的工作栈空间的大小，它通常等于一次调用算法所分配的临时存储空间的大小乘以被调用的次数。当一个算法所占的存储空间不随被处理数据量 n 的大小而改变时，它的空间复杂度是一个常数，可表示为 $O(1)$。当一个算法的空间复杂度与以 2 为底的 n 的对数成正比时，可表示为 $O(\log_2 n)$；当一个算法的空间复杂度与 n 成线性比例关系时，可表示为 $O(n)$。

对于一个算法，其时间复杂度和空间复杂度往往是相互影响的。当追求一个较好的时间复杂度时，可能会使空间复杂度的性能变差，即可能导致较多的存储空间被占用；反之，当追求一个较好的空间复杂度时，可能会使时间复杂度的性能变差，即可能导致它的运行需要更长的时间。另外，算法的所有性能之间都存在着或多或少的相互影响。因此，当设计一个算法(特别是大型算法)时，要综合考虑算法的各项性能、算法的使用频率、算法处理的数据量的大小、算法描述语言的特性、算法运行的机器系统环境等各种因素，才能设计出更适合实际环境的优秀算法。

1.4　习题

一、选择题

1. (　　)是数据的最小单位。

A. 数据项　　B. 表元素　　C. 信息项　　D. 数据元素

2. 算法的计算量的大小称为计算的(　　)。

A. 效率　　B. 复杂性　　C. 现实性　　D. 难度

3. 算法的时间复杂度取决于(　　)。

A. 问题的规模　　B. 待处理数据的初态　　C. A 和 B

4. 计算机算法指的是①(　　)，它必须具备②(　　)这 3 个特性。

① A. 计算方法
　 B. 排序方法
　 C. 解决问题的步骤序列
　 D. 调度方法

② A. 可执行性、可移植性、可扩充性
　 B. 可执行性、确定性、有穷性
　 C. 确定性、有穷性、稳定性
　 D. 易读性、稳定性、安全性

5. 下列关于算法的描述中，错误的是(　　)。

A. 算法最终必须由计算机程序实现

B. 为解决某问题的算法与为该问题编写的程序含义是相同的

C. 算法的可行性是指指令不能有二义性

D. 以上几个都是错误的

6. 下列说法中，错误的是(　　)。

① 算法原地工作的含义是指不需要任何额外的辅助空间。

② 在相同的规模 n 下，复杂度 $O(n)$的算法在时间上总是优于复杂度 $O(2n)$的算法。

③ 所谓时间复杂度是指最坏情况下，估算算法执行时间的一个上界。

④ 同一个算法，实现语言的级别越高，执行效率就越低。

A. ①　　B. ①②　　C. ①④　　D. ③

7. 从逻辑上可以把数据结构分为(　　)两大类。

A. 动态结构、静态结构　　B. 顺序结构、链式结构

C. 线性结构、非线性结构　　D. 初等结构、构造型结构

8. 顺序存储中，存储单元的地址(　　)。

A. 一定连续　　B. 一定不连续

C. 不一定连续　　D. 部分连续，部分不连续

9. 下列时间复杂度最好的是(　　)。

A. $O\left(\log_2 n\right)$　　B. $O(n)$　　C. $O(n^2)$　　D. $O(1)$

10. 逻辑结构是(　　)关系的整体。

A. 存储结构之间　　B. 数据元素之间逻辑

C. 数据类型之间　　D. 数据项之间逻辑

11. 数据结构有(　　)种基本逻辑结构。

A. 2　　B. 3　　C. 1　　D. 4

12. 下列 4 种基本的逻辑结构中，数据元素之间关系最弱的是(　　)。

A. 线性结构　　B. 树状结构　　C. 集合　　D. 图状结构

13. 一个算法的时间复杂度为 $(n^3+n^2\log_2 n+14n)/n^2$，其数量级表示为(　　)。

A. $O(n^4)$　　B. $O(n^3)$　　C. $O(n^2)$　　D. $O(n)$

二、填空题

1. 数据的物理结构包括_____的表示和_____的表示。

2. 对于给定的 n 个元素，可以构造出的逻辑结构有_____、_____、_____、_____ 4 种。

3. 数据的逻辑结构是指_____。

4. 一个数据结构在计算机中的_____称为存储结构。

5. 数据结构中，评价算法的两个重要指标是_____、_____。

6. 数据结构研究的是数据的_____和_____，以及它们之间的相互关系，并对这种结构定义相应的_____，设计出相应的_____。

7. 一个算法具有 5 个特性：_____、_____、_____、有零个或多个输入、有一个或多个输出。

8. 计算机执行下列语句段时，语句 *s* 的执行次数为_____。

```
for(i=1; i<n-1; i++)
    for(j=n; j>=i; j--)
        s;
```

9. 下列程序段的时间复杂度为_____。(n>1)

```
sum=1;
for (i=0;sum<n;i++) sum+=1;
```