国家教材委员会专家委员会审核通过
2019

普通高中教科书

信息技术

选择性必修 1

数据与数据结构

Shuju yu Shuju Jiegou

广东教育出版社

普 通 高 中 教 科 书

信息技术 选择性必修 1

数据与数据结构

Shuju yu Shuju Jiegou

徐福荫　主编

广东教育出版社

· 广州 ·

图书在版编目(CIP)数据

信息技术. 选择性必修1：数据与数据结构 / 徐福荫主编. —广州：广东教育出版社，2019.12（2021.1重印）
普通高中教科书
ISBN 978-7-5548-3027-7

Ⅰ. ①信… Ⅱ. ①徐… Ⅲ. ①计算机课—高中—教材
Ⅳ. ①G634.671

中国版本图书馆CIP数据核字（2019）第208284号

编写单位　广东教育出版社
主　　编　徐福荫
副 主 编　朱光明　黄国洪
本册主编　刘毅婉　严开明
核心编写人员（以姓氏笔画为序）
　　丘韶科　李苑文　周玳珈　黄淑燕

责任编辑　姜树彪　严洪超
责任技编　杨启承　陈　瑾
装帧设计　何　维

信息技术 必修1 数据与数据结构
XINXI JISHU BIXIU 1 SHUJU YU SHUJU JIEGOU
广 东 教 育 出 版 社 出 版
（广州市环市东路472号12-15楼）
邮政编码：510075
网址：http://www.gjs.cn
广东新华发行集团股份有限公司发行
广东新华印刷有限公司南海分公司印刷
（佛山市南海区盐步河东中心路）
890毫米×1240毫米　16开本　9印张　180 000字
2019年12月第1版　2021年1月第3次印刷
ISBN 978-7-5548-3027-7
定价：10.59元
批准文号：粤发改价格［2017］434号　举报电话：12315

前　言

信息技术作为当今先进生产力的代表，已经成为我国经济发展的重要支柱和网络强国的战略支撑。信息技术涵盖了获取、表示、传输、存储和加工信息在内的各种技术。自电子计算机问世以来，信息技术沿着以计算机为核心、到以互联网为核心、再到以数据为核心的发展脉络，深刻影响着社会的经济结构和生产方式，加快了全球范围内的知识更新和技术创新，推动了社会信息化、智能化的建设与发展，催生出现实空间与虚拟空间并存的信息社会，并逐步构建出智慧社会。

在数字化时代，数据对科学发现、技术进步、经济发展以及人们的日常生活有着越来越深刻的影响。理解数据的作用及价值，对同学们适应信息社会、学会数字化生存有着重要意义。数据结构是信息技术学科的核心内容之一，对培养同学们的信息意识与计算思维、深入理解及掌握信息技术学科知识与实践方法、形成学科核心素养，具有非常重要的作用。本教科书是针对数据、数据结构及其应用而设置的选择性必修模块。

通过本教科书的学习，同学们可以进一步了解数据的作用，在掌握常用数据结构的概念、特点、操作、编程实现方法等内容的基础上，能对简单的数据问题进行分析，选择恰当的数据结构，用一种程序设计语言编程实现，在问题解决过程中对数据抽象、数据结构的思想与方法有初步的认识。

本教科书按“数据与数据结构”“数据的组织和存储”“数据结构应用”三部分内容展开，围绕信息技术学科核心素养，设计了“超市数据与社会关系的调查”“超市商品的信息化管理程序设计”“超市服务自动化的模拟实验”“俄罗斯方块游戏的抽象数据类型案例分析”“超市促销商品的选择与查询程序设计”项目范例。教师围绕“情境→主题→规划→探究→实施→成果→评价”的项目范例主线开展教学活动，帮助同学们掌握本教科书的基础知识、方法与技能，增强信息意识，发展计算思维，提高数字化学习与创新能力，树立正确的信息社会价值观和责任感，从而促进同学们的信息素养提升。

本教科书要求同学们对现实世界中的真实性问题进行自主、协作、探

究学习。同学们围绕“项目选题→项目规划→方案交流→探究活动→项目实施→成果交流→活动评价”的项目学习主线开展学习活动，体验“做中学、学中创、创中乐”的项目学习理念和“从实践入手、先学后教、先练后讲”的项目学习策略，将知识建构、技能培养与思维发展融入运用数字化工具解决问题和完成任务的过程中，从而促进信息意识、计算思维、数字化学习与创新、信息社会责任的信息技术学科核心素养达成。

本教科书设置了“项目范例”“项目选题”“项目规划”“方案交流”“探究活动”“项目实施”“成果交流”“活动评价”等学习栏目，指导同学们开展项目学习活动。其中，“项目范例”是教师通过“情境”“主题”“规划”“探究”“实施”“成果”“评价”等活动，引导同学们了解开展项目学习活动的全过程；“项目选题”是同学们从真实世界选择自己感兴趣的项目主题；“项目规划”是同学们根据项目选题，制订自己的项目方案；“方案交流”是同学们展示、交流自己设计的项目方案，师生共同探究、完善其方案；“探究活动”是同学们通过“问题”“观察”“分析”“阅读”“思考”“交流”“实践”“实验”“体验”“调查”“讨论”“拓展”等活动，获取知识和技能的过程；“项目实施”是同学们运用在项目学习过程中所获得的知识和技能来完成项目方案；“成果交流”是教师组织同学们展示交流项目成果，共享创造、分享快乐；“活动评价”是教师组织同学们开展项目评价活动。

本教科书各章首页的导言，叙述了本章的学习目的与方式、学习目标与内容，让同学们对整章有个总体认识。每章设置了“本章扼要回顾”，通过知识结构图把每章的主要内容及它们之间的关系描述出来，这有助于同学们建立自己的知识结构体系。每章结尾的“本章学业评价”设计了基于学业质量水平的测试题，并通过本章的项目活动评价，让同学们综合评价自己在信息技术知识与技能、解决实际问题的过程与方法，以及相关情感态度与价值观的形成等方面，是否达到了本章的学习目标。此外，本教科书中为同学们提供了配套学习资源包，里面含有超市数据与社会关系的分析调查报告、超市商品的信息化管理程序、超市服务自动化的模拟程序及超市促销商品的选择与查询程序等，还提供了同学们开展学习探究的程序设计示例。当然，同学们还可以自己收集素材，让自己的项目学习作品更有特色。

CONTENTS

目录

第一章
认识数据和数据结构

在数字化时代，数据与大数据对科学发现、技术进步、经济发展以及人们的日常生活有着越来越深刻的影响。人们产生数据、分析数据、使用数据，使数据成为新的原材料、生产资料和基础设施。把握数据的内涵与外延，才能把握时代的脉搏。

本章将通过“数据与社会关系的调查”项目，进行自主、协作、探究学习，让同学们理解数字、数值和数据的基本含义，懂得对生活中的实际问题进行数据抽象，认识数据结构在解决问题过程中的重要作用，从而将知识建构、技能培养与思维发展融入运用数字化工具解决问题和完成任务的过程中，促进信息技术学科核心素养达成，完成项目学习目标。

- 数据及其价值
- 对实际问题的数据抽象
- 认识数据结构

项目范例　超市数据与社会关系的调查

情境

现代超市涉及大量的数据处理，对管理和技术提出了很高的要求。超市商品、客户群、资金流的管理都是数据处理的问题。如超市设置的收银机（如图1–1所示），收银的过程就包含了大量的数据处理工作；对超市客户数据的高效管理也能使经营更有针对性。超市借助计算机管理系统减少人工操作所带来的低效率、易出错等问题，精确适时地反映和处理超市各项经营活动。

图1–1　超市收银台

主题

超市数据与社会关系的调查

规划

根据项目范例的主题，在小组中组织讨论，利用思维导图工具，制订项目学习规划，如图1–2所示。

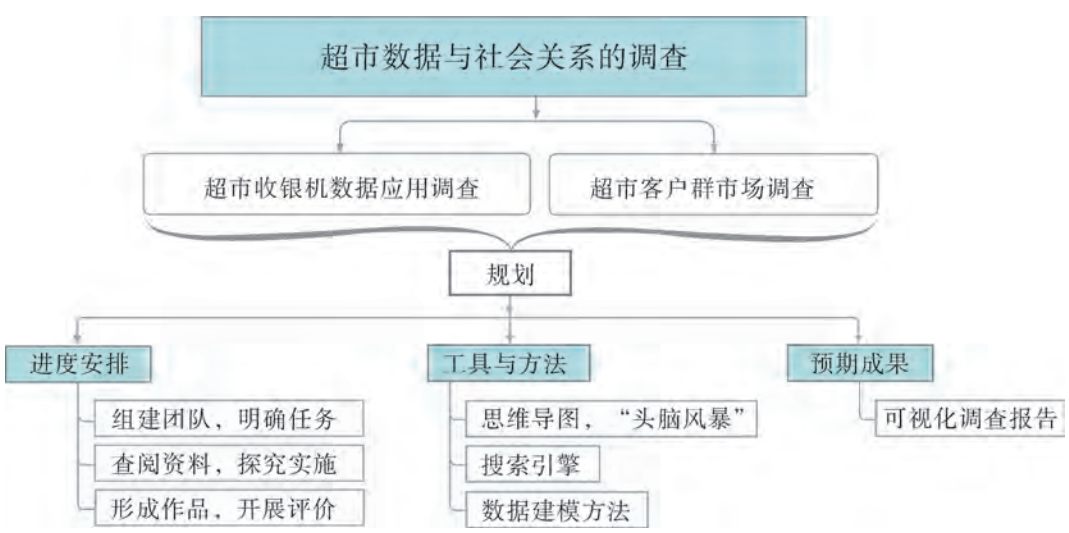


图1–2　“超市数据与社会关系的调查”项目学习规划

探究

根据项目学习规划的安排，通过调查、案例分析、文献阅读和网上资料搜索，开展“超市数据与社会关系的调查”项目学习探究活动，如表1–1所示。

表1–1　“超市数据与社会关系的调查”项目学习探究活动

探究活动	学习内容		知识技能
超市收银机数据应用调查	列举数据应用实例。	调查收银机的功能。	分析数据与社会各领域的关系。 理解数字、数值和数据的基本含义。 认识数据在社会应用中的价值和意义。
	抽取事物中的数据。	调查收银机收集的数据。	
	分析数据关系。	分析收银机使用的数据之间的关系。	
超市客户群市场调查	列举数据应用实例。	调查超市客户群体。	理解数据结构的概念。 认识数据结构在解决问题过程中的重要作用。
	抽取事物中的数据。	分析客户的属性。	
	确定数据关系。	分析客户数据关系。	
	建立数据模型。	确定客户数据模型。	

实施

实施项目学习各项探究活动，进一步理解数据对信息社会的重要性。

成果

在小组开展项目范例学习过程中，利用思维导图工具梳理小组成员在“头脑风暴”活动中的观点，建立观点结构图，运用多媒体创作工具（如演示文稿、在线编辑工具等），综合加工和表达，形成项目范例可视化学习成果，并通过各种分享平台发布，共享创造、分享快乐。例如，运用在线编辑工具制作的“超市数据与社会关系的调查”可视化报告，可以在教科书的配套学习资源包中查看，其目录截图如图1–3所示。

图1–3　“超市数据与社会关系的调查”可视化报告的目录截图

根据教科书附录2的“项目活动评价表”，对项目范例的学习过程和学习成果在小组或班级上进行交流，开展项目学习活动评价。

同学们以3~6人组成一个小组，选择下面一个参考主题，或者自拟一个感兴趣的主题，开展项目学习。

1. 图书馆数据与社会关系的调查
2. 校运会数据与社会关系的调查
3. 社团活动数据与社会关系的调查

各小组根据项目选题，参照项目范例的样式，利用思维导图工具，制订相应的项目方案。

各小组将完成的方案在全班进行展示交流，师生共同探讨、完善相应的项目方案。

1.1 数据及其价值

数据是现实世界客观事物的符号记录，是以文字、图像、声音等表现形式存在的数字资源，是信息的载体，是计算机加工的对象。

数据，已经渗透到当今每一个行业和业务职能领域，成为重要的生产因素。人们对于大数据的挖掘和运用，预示着新一波生产率增长和消费者盈余浪潮的到来。

1.1.1 数字、数值与数据

人类从很久以前就懂得采用不同的方法对生产、生活中的事项进行记录，如用绳结的多少来记录数量的多少，用符号、图形、文字来记录事件、场景。记录的结果是产生了数据。而随着技术的不断发展进步，记录的方法和工具不断改进，数据的内涵也在不断拓展和延伸。

1. 数字

数字（Number）是一种用来表示数的书写符号。数字在不同的记数系统中表示的形式各不相同，阿拉伯数字是最普遍的一种。同一个数在不同的记数系统中有不同的表示，比如，数37（十进制阿拉伯数字）可以有多种写法：十进制阿拉伯数字写作“37”，中文数字写作“三十七”，罗马数字写作“XXXVII”，二进制阿拉伯数字写作“100101”（除十进制和二进制外，还有其他进制）。

2. 数值

量是事物在同一种属性上的差别，通常用数字和单位来表示。例如，对于“质量”这种属性，一个大苹果质量约150克，一个小苹果质量约100克；对于“长度”这种属性，一张A4纸的两条边长分别为29.7厘米、21厘米……

一个量用数字表示时，这个数字叫作这个量的数值（Numerical Value）。例如，“150克”的“150”，“29.7厘米”的“29.7”。

3. 数据

数据（Data）是指对现实世界客观事物进行记录并可以鉴别的符号。数据是事实或观察的结果，是对客观事物的逻辑归纳，是用于表示客观事物的未经加工的原始素材。

例如，一个苹果，通过观察可以发现它的颜色是红色，形状近似球形；称重可以获得它的质量（如150克）；吃起来可以知道它的味道是香甜的；也可以画一幅画来描绘它，照一张相片来记录它……红色、150克、香甜、画、相片都是数据，可用于表示这个苹果——“相片中、画中描述的、近似球形的、红色的、香甜的、质量为150克的物品”。当然，数据越详细越准确，对于“苹果”这个事物的表示就越准确。

在计算机科学中，数据是对所有输入计算机并被计算机识别、存储和处理的符号的总称，是联系现实世界和计算机世界的途径。现在的计算机存储和处理的对象十分广泛，数据的含义也变得广泛了，如图像、声音等也可以经过一定的输入变换，利用计算机进行加工和处理。

数据是信息的载体，是计算机程序加工的“原材料”。例如，在数值计算中所使用的数据是整数和实数，文本编辑程序中使用的数据是字符串。计算机处理的数据既可以是“150”这样的数值数据，也可以是文字、图片等非数值数据。而随着技术的应用和发展，计算机处理的对象越来越多的是非数值数据。

在大数据时代，数据不仅是信息的载体，也是人们提取信息、做出决策的重要依据，成为人们认识和理解现实世界客观事物的重要资源。

探究活动

分析

有人说：“数字是形，数据是体，数值是魂。”请根据数字、数值和数据的含义，分析下列案例分别包含了哪些数字、数值和数据，并根据已有的样例补充完整。

➢案例一

每年的5月17日，全球电信业都要庆祝自己共同的节日——世界电信日。从2006年开始，这个日子又增添了一个特殊的含义，联合国和国际电信联盟一起向世界发出邀请，与各成员国共同庆祝首届世界信息社会日。

分析案例中包含的数字、数值和数据，将表格补充完整。

数字	5，17，2006。
数值	
数据	5月17日，2006年，世界电信日。

➢案例二

知网，是国家知识基础设施（National Knowledge Infrastructure，NKI）的概念，由世界银行于1998年提出。中国知网（CNKI）工程是以实现全社会知识资源传播共享与增值利用为目标的信息化建设项目，由清华大学发起，始建于1999年6月。在中国知网中检索“核心素养”的网页截图如图1–4所示。

图1–4　中国知网检索“核心素养”的网页截图

分析案例中包含的数字、数值和数据，将表格补充完整。

数字	
数值	7807条结果中的“7807”。
数据	《谈“核心素养”》一文被引用的次数为307次，被下载的次数为25 582次。

1.1.2 数据与社会的关系

回家过年是每一个中国人的期盼，一年一度的春运也随之成了牵动全社会的民生大事。2017年春运前后，与出行相关的政府部门、网站、媒体纷纷通过各种渠道发布了春运大数据报告，以下是其中的一些报道。请仔细阅读，并与同学讨论，体会当今社会数据、大数据与社会生产、生活的关系。

● 根据交通运输部数据统计，2016年春运期间全国运输系统共发送旅客29.1亿人次，2017年春运将有望突破30亿人次。春运期间的出行需求主要以跨市出行（离开常住城市）为主，跨市出行的客流量将直接影响整个春运期间的客流量。

● 2017年春运大潮从1月13日开始，至2月21日结束。春运前夕，新京报联合某在线票务服务公司，依据实时数据，发布了2017年春运大数据报告。自发布日起，每周滚动更新除夕前三天（1月24日—1月26日）由北京飞往全国各主要热门城市的机票价格情况，以及节后返程高峰期（2月2日—2月4日）全国主要热门城市飞回北京的机票价格情况，并给予订票提示。

● 因2017年春节相比去年提前10天，春运车票的预订大幕已经开启。为了方便用户提前安排春节出行，抢到一张炙手可热的回家车票，某网站通过对60多万条飞机航线、50余万条铁路客运线进行大数据计算，发布《2017年春运大交通数据报告》，为回家旅客提供参考。

● 某公司曾根据往年春运大数据，预测2017年春运返乡高峰会出现在腊月二十八，次高峰为腊月二十四。而实际上，次高峰预测准确，但返乡高峰却提前至腊月二十七。一方面是由于大部分企业会在该日期前后给员工提前放春节假期，另一方面则可能是此预测发布后，用户纷纷选择错峰出行。

分析以上报道，可以得出：

（1）当今社会，不管知道还是不知道、愿意还是不愿意，人们在社会生活中生成的数据正在被各种途径收集、整理、利用。

（2）大数据计算可以给人们的社会活动提供参考。

（3）大数据计算的结果，能够调节、影响人们的活动。

当今社会是一个高速发展的社会，科技发达，信息流通，人们之间的交流越来越密切，生活也越来越方便。我们每个人都在为社会这个“数据库”贡献数据：我们一出生，身份数据就被采集；从上学到就业、从工作到生活，都离不开各种数据的传送和接收。尤其在信息技术飞速发展的今天，人们利用互联网社交、娱乐、购物、理财……社交平台、教育平台、电商平台、政府公共服务平台、金融平台，互联网、物联网，庞大的数据量正在这个看不见、摸不着的信息流中急剧增加、快速流动。

1. 社会中的每个人都成为数据的提供者

人们在电商平台购买商品、在网站预定车票和酒店、到银行办理业务、通过通信软件聊天、注册成为某个商家的VIP客户……当人们在生活中刷各种各样的卡、读取各种各样的证件、在线录入各种各样的信息、线下填写各种各样的表格的时候，就是一次次主动向不同的系统提供数据的时候。

另外，人们浏览网页时，点击的网页被记录下来；使用电商平台时，查看的历史被记录下来；在街上行走时，经过的路径会被监控系统记录下来；使用手机时，位置、运动信息、通信情况也可能被记录下来……不管人们知情还是不知情，生活中的很多数据已经被记录下来。

2. 自然界中的事物也是数据的提供者

通过各种监测系统和设备，人们可以获得空气质量的数据、河流水位的数据、气候变化的数据等。而且，技术的发展和应用，使得人们不但可以记录自然界中的事物，还能记录它们的变化和关联。

3. 大数据正在影响和改变着人们的生活

手机APP查询物流情况让人们及时掌握快递的运送状态；导航软件让人们在陌生环境中也能快速而准确地到达目的地；网上购票免去人们寒风中彻夜排队的辛劳……人们日益普及的网络行为，无时无刻不在产生大量的数据。这些无法用常规软件工具进行高效捕捉、管理和处理的数据集合，被称为大数据。大数据的分析和应用为人们的生活、公司的运营甚至政府的决策提供依据，影响和改变着人们的生活，成为推动信息社会发展的重要资源。我们正面临着一个“一切都被记录，一切都被分析”的数据化时代。

讨论

以小组为单位，收集数据应用的事例，就下列问题开展讨论。

（1）中国互联网络信息中心每年都会发布《中国互联网络发展状况统计报告》，请下载最新一期的报告并进行阅读，讨论其中的数据来源。

（2）北斗卫星导航系统（BeiDou Navigation Satellite System，BDS）是我国自行研制的全球卫星导航系统，致力于向全球用户提供高质量的定位、导航和授时服务，如图1-5所示。导航系统的自动化地图和路线规划功能，使人们无论是在熟悉还是陌生的环境下，都能泰然处之，快速而准确地到达目的地。数据的应用给人们的生活带来了便捷的同时，是否也带来了其他影响？你还了解其他同类的应用事例吗？

图1-5　卫星定位系统

1.1.3 数据的价值和意义

阅读

图1-6是网购物品物流详情的手机截图。请仔细阅读这幅图，与同学一起探讨可以从中获得什么数据，得到什么信息。

图1-6 物流数据

从图中大致可以获得：

△商品预计“2016-12-08”送达

△快递单号“6671208650**”

△商品一共“4件”

△“2016-12-06 02:19:35”商品从无锡发出

△商家所在地是“苏州”

……

细心的同学可能还会获得：

△截图的时间是“20:24”

△手机使用的网络是“中国电信”

△手机的“蓝牙”状态开启

△手机设置了“闹钟”

△正在使用“4G”数据流量上网

……

通过对读图提取到的数据进行分析，人们能够掌握一个正在运输过程中的包裹的信息、截图所使用的手机的当前状态信息。

数据中蕴藏着宝藏。在数字化时代，数据和大数据对科学发现、技术进步、经济发展以及人们的日常生活有着越来越深刻的影响。随着海量数据的生成和存储，数据的价值越来越多地得到体现。数据和大数据成为新的原材料、生产资料和基础设施。理解数据和大数据的作用及价值，掌握数据获取、加工、管理、分析的方法，对我们适应信息社会、学会数字化生存有着十分重要的意义。

1. 新的原材料

在大数据时代，数据本身也扮演原材料的角色。人们生产出数据产品，提供基于数据分析的服务，都建立在“有数据可供加工”的基础之上。

例如，通过电子商务网站记录的数据，商家可以获取一个用户浏览、收藏、购买的数据，以此为材料可以加工分析出该用户的购买偏好、价格偏好、消费水平。商家可以精准地推送广告从而获得收益。

2. 生产资料

2013年被公认为世界的大数据元年。在这一年里，数据呈现井喷现象，各行各业的管

理者都在讨论大数据，庞大的数据资源渗透了社会各个领域。在未来，数据将会像土地、石油和矿产一样，成为经济运行中的根本性资源，也是重要生产力。

3. 基础设施

2009年，在甲型H1N1流感爆发的几周前，一家互联网公司的工程师们通过分析人们在网上的搜索记录成功地预测了流感的传播。这种分析和预测建立在大数据的基础上：在此过程中，他们分析了超过5000万条词条，处理了4.5亿个不同的数据模型。

基于大数据的数据挖掘和分析，将成为未来极其可观的一个产业方向，并将会越来越多地应用在交通、教育、医疗、公共服务等领域。大数据逐渐成为现代社会基础设施的一部分，就像公路、铁路、港口、水电和通信网络一样不可或缺。

以小组为单位，讨论下列案例中数据的价值和意义。

➢案例一

越来越多的政府部门、企业等开始意识到数据正在成为组织中最重要的资产，数据分析能力正在成为组织的核心竞争力。

➢案例二

联合国在2012年发布了大数据政务白皮书，指出大数据对于联合国和各国政府来说是一个历史性的机遇，人们如今可以使用极为丰富的数据资源，来对社会经济进行前所未有的实时分析，帮助政府更好地响应社会和经济运行。

➢案例三

有一份名为“大数据，是下一轮创新、竞争和生产力的前沿”的专题研究报告提出，“对于企业来说，海量数据的运用将成为未来竞争和增长的基础”。该报告在业界引起广泛反响。

当我们解决实际问题的时候，数据究竟从哪来？它们如何发生作用？

在项目范例中，通过调查获知，超市设置收银机，收银员可以很方便地对顾客购买的商品进行精确计价，提高收付款速度；收银结算的同时也获得了商品销售的数据。而为了实现精确收费，收银机联接的计算机系统还需要处理大量数据。该调查形成的报告可参考配套学习资源包中的文档“第一章\范例成品\超市数据与社会关系的调查报告.docx”。请在小组中交流你对该调查报告的感悟，与同学们共同考虑如何开展本小组的项目。

各小组根据项目选题及拟订的项目方案，结合本节所学知识，开展以下活动。

1. 收集所选定项目中的数据。
2. 分析数据与社会关系及其价值。

1.2 对实际问题的数据抽象

计算机越来越多地用于控制、管理及数据处理等非数值计算的工作，这些工作的操作对象及其关系是一些具有一定结构的数据，无法用数学方程进行描述。因此，我们必须对实际问题进行数据抽象，分析待处理对象的特性以及各处理对象之间存在的关系，建立问题的数据模型。

1.2.1 抽象问题中的数据

人们利用计算机来帮助解决现实生活中的许多问题。

用计算机解决问题，一般要经历以下过程：从问题中抽象出一个适当的数学模型，然后设计一个解此数学模型的算法，再编写程序调试，直到得到最终解答。寻求数学模型的过程实质就是分析问题、从中提取操作对象，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。例如对于鸡兔同笼的问题，其数学模型是一个二元一次方程组，设计一个算法解此方程组，并编程实现，就解决了这一问题。但是，有许多非数值计算的问题，无法用数学方程来表示操作对象。

当前，计算机已成为高效处理数据的工具。数据是计算机程序处理的对象，通过计算机程序处理数据，速度快、效率高、出错少。用数据来表示现实世界的事物及其活动，让这些事物可以在计算机中存储、计算、处理，是实现计算机解决问题的基础。

探究活动

分析

对于“超市客户”这一特定事物，可从“分析数据以便定位超市经营策略”的需求出发，抽取其中包含的数据。分析如下：

1. 明确问题解决目标。

根据“项目范例”的“主题”“规划”和“探究”的设计，明确项目要解决的问题，并逐步细化，如图1-7所示。

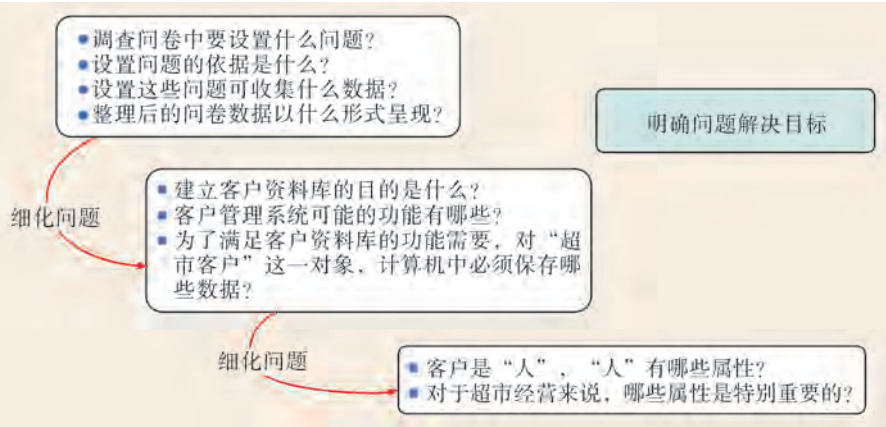


图1-7 根据目标细化问题

2. 收集资料，进行分析，探究问题的解。

研究表明，对超市经营有影响的各种因素中，人的因素起重要作用。“人”的属性众多（如图1-8所示），“超市客户”的属性则需要筛选，可以从超市客户管理需要的角度进行筛选，如表1-2所示。

图1-8　“人”的属性

表1-2　“超市客户”属性的筛选

	消费水平	跟踪联系	消费欲望	消费习惯
姓名		√		
年龄				√
性别				√
职业	√			
地址			√	
家庭人数			√	
联系方式		√		

根据上述分析，超市的客户管理系统要做到准确推送商品信息，跟踪目标客户的消费情况，确定超市的经营定位，则姓名、性别、联系方式、年龄、职业、地址等属性应列入数据表的考虑范围，如表1-3所示。

表1-3　解决问题所需要的数据

事物	属性
人	姓名、年龄、性别、家庭人数、收入、职业、教育状况、身高、体重、视力、联系方式、地址……
目的	**需抽取的数据**
1. 准确推送商品信息	姓名、联系方式、地址。
2. 确定超市的经营定位	年龄、职业、收入。

我们已经知道，数据是事实或观察的结果，用来表示客观事物的属性。抽取出来的属性中，姓名、性别、职业等属性，直接采用收集到的数据如“张达”“男”“公务员”就可以了；而对于“年龄”这一项，因为年龄会随着时间不断变化，因此可以换成同样可以区分年龄大小但值固定的“出生年月”来表示；“联系方式”属性则采用了现在最普遍使用的“手机号码”来表示。最后，得到如表1-4所示的超市客户表。

表1-4 超市客户表

姓名	性别	出生年月	职业	手机号码	地址	消费记录
张达	男	197401	公务员	13712345678	× × ×	× × ×
熊二	男	197509	工人	13523456789	× × ×	× × ×
李季	女	197808	律师	13034567891	× × ×	× × ×
⋮	⋮	⋮	⋮	⋮	⋮	⋮

现实世界的事物纷繁多样，事物本身具备多种属性。从复杂多样、表现各异的现实世界的事物到规则有序的计算机世界的数据，让现实世界的问题在计算机中存储、计算、处理，需要一个转换的过程：从解决问题的需要出发，抽取出与问题解决相关的属性，用合适的数据表示这些属性；理清数据间的关系，建立数据模型。这个过程就是数据抽象的过程。

各小组试着列出选定项目中事物及其活动的属性，再从中选择与问题解决目标相关的属性。这个过程可能一次成功，也可能要经过反复几次的调整修改。

1.2.2 分析数据之间的关系

计算机处理的数值计算问题能直接抽象出数学模型，但更多的非数值计算的问题、数据间的复杂关系并不能直接用数学模型表示。这些复杂的关系中，最基本的关系有三种，即线性关系、层次关系和网状关系。

1. 线性关系

一个班学生的学号，从1号到50号，学号之间是顺序排列的，前后有序，如图1-9所示。

图1-9 顺序排列的学号

从图1-9中可以看出，数据间的关系比较简单，每个数据仅有一个直接前驱和一个直接后继（第一个数据仅有后继，最后一个数据仅有前驱）。数据间的这种关系称为线性关系。如表1-4的“超市客户表”，表中的各行数据之间就是线性关系。线性关系是计算机中最常见的数据关系。

2. 层次关系

初读《红楼梦》的同学可能不能很好地分清其中家族成员之间的关系。但如果借助如图1–10所示的“关系树”，则可以清晰地描述这些关系。

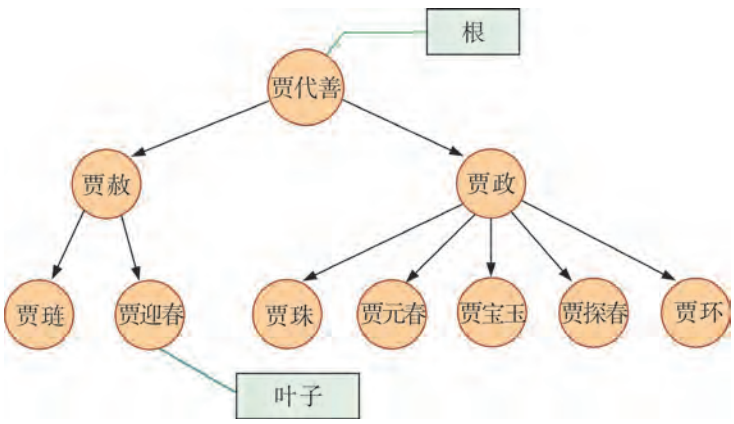


图1–10　《红楼梦》家族部分成员关系

图1–10就像一棵倒立的树。每个结点（圆圈）有且仅有一个前驱（根除外），有任意多个后继（叶子可以看作具有0个后继的结点）。这种数据间具有的一对多的联系称为层次关系，具有层次关系的数据表示为一棵倒立的树。现实生活中也有不少例子，其数据间的关系为层次关系。例如，记录博弈过程的棋盘数据之间就是层次关系；学校里年级、班级的组织架构就是层次关系；班级成员中的班长、组长、组员间也构成层次关系。

3. 网状关系

随着经济的发展，城市之间的交通发达，人们出行的选择变得多样化。城市之间的交通联系可用如图1–11所示的关系来表示。

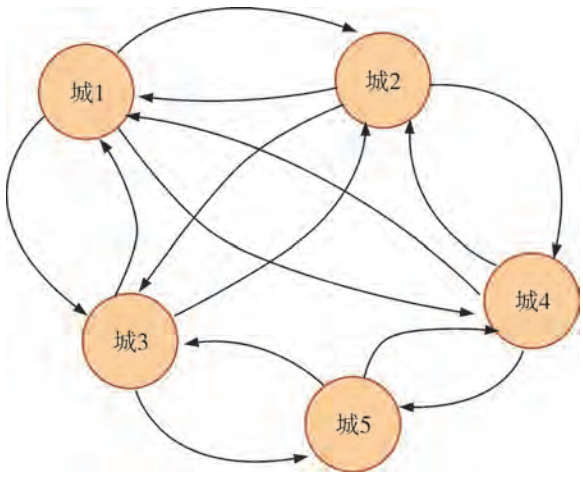


图1–11　城市间的交通联系

在图1–11中，数据间的联系是多对多的：每个数据既有多个前驱，也有多个后继。这种数据间的多对多联系称为网状关系。例如，学校教育教学活动中涉及的人员之间的关系就是网状关系：一个教师同时教多个学生，一个学生也同时有多个教师授课。

综合以上三个例子可见，描述这类非数值计算问题的数学模型不再是数学方程，而是用诸如表、树、图之类的数据模型进行描述。

1.2.3　建立数据模型

数据模型是客观事物及其关系的数据描述，是对客观事物进行数据抽象的结果。

当描述客观事物属性的数据被抽取出来，数据间的关系就能理清，建立数据模型也就水到渠成了。建立数据模型的过程如图1–12所示。

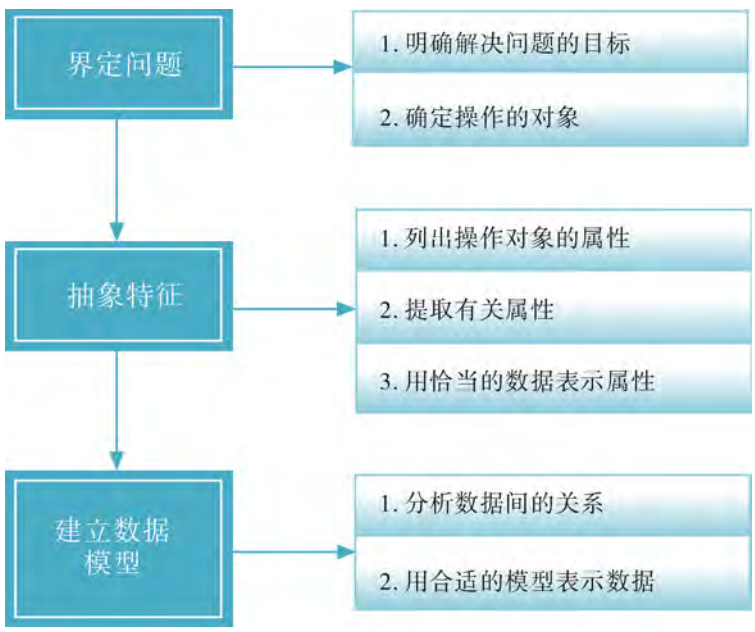


图1–12　建立数据模型的过程

具有线性关系的数据，其数据模型是表——如超市客户表、商品表、各种场景下的学生表等。线性关系是计算机中最常见的数据关系，表是计算机中最常见的数据模型。在本教科书的后续章节中将要学习到的链表、队列、栈、字符串，其数据模型都是表。

具有层次关系的数据，其数据模型是树——如家族成员树（如图1–10所示）、班级成员树、学校机构树等。

具有网状关系的数据，其数据模型是图——如城市交通图（如图1–11所示）、学校人员图、（师生）教学图等。

表中的每一行，树、图中的每一个结点都被称为一个数据元素（Data Element）。数据元素是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。

有时候一个数据元素可由若干个数据项（Data Item）组成。例如表中每个客户信息是一个数据元素，而客户信息中的每一项（如姓名、出生年月等）为一个数据项。数据项是数据的不可分割的最小单位。

数据对象（Data Object）是性质相同的数据元素的集合，是数据的一个子集。例如，整数数据对象是集合N={0，1，-1，2，-2，3，-3，…}。

在小组中交流，列举更多生活中包含表、树、图模型的事物，画出数据模型示意图，填写表1-5。

表1-5　生活中的表、树、图

模型	生活中的事物	数据模型示意图
表		
树		
图		

针对本组的项目选题进行分析，抽取出其中的数据。可参考表1-3开展学习活动。

1.3 认识数据结构

数据以及数据的“结构”，与加工和处理这些数据的方法密切相关。为了描述和处理越来越复杂的数据关系，人们需要研究数据结构。数据结构是信息技术学科的核心内容之一，对培养信息意识与计算思维、深入理解及掌握信息技术学科知识与实践方法、形成学科核心素养，具有非常重要的作用。

1.3.1 数据结构

1. 数据的组织方式

上一节中的表1-4“超市客户表”，是对客观事物“超市客户”进行数据抽象所得的数据模型。它是现实世界中数据和数据间关系的抽象表示。在计算机世界中表示和存储数据、数据间的关系，需要用到数据结构。

超市商品种类繁多，超市的进、销、存系统经常需要对某种商品进行操作。因此，“查找”功能的效率往往直接影响整个系统的效率。那么，影响查找效率的因素有哪些呢？

假设超市商品数据是随机排列的——查找的方法最简单：从头开始逐个比较，顺序查对，直到找到目标为止。查找操作必须对所有数据进行，所以虽然方法简单却效率低下。

假设超市商品数据是有组织的——可按照商品所属类别进行分类，同类商品数据放在一起，则商品查找效率将大大提高：先找到商品所属类别，再从该类别下的首个数据开始进行逐个比较查对即可，无须对所有数据进行查找。

图1-13（a）中的数据就是随机排列的，而图1-13（b）中的数据则按洗漱用品、食品、清洁用品等分类组织。

编号	商品
001	毛巾
002	洗衣粉
003	方便面
004	食用油
⋮	⋮
102	洗发水
103	牙刷
⋮	⋮
127	牙膏
⋮	⋮

（a）随机排列的数据

种类	分类号
洗漱用品	01
食品	02
清洁用品	03

编号	商品
01001	毛巾
01002	洗发水
01003	牙刷
01004	牙膏
⋮	⋮
02001	方便面
02002	食用油
⋮	⋮
03001	洗衣粉
⋮	⋮

（b）分类组织的数据

图1-13　数据的不同组织方式

针对图1-13所列的数据组织方式，在小组中与同学就数据组织方式与数据处理效率的关系进行讨论、交流。

（1）对于商品数据，除了图中所列组织方式，还有其他方式吗？

（2）对应图中数据的不同的组织方式，如果增加新的商品数据，其操作效率如何？删除商品呢？

（3）图中的数据组织方式及其查找、新增、删除操作，除了应用于商品管理，还可以应用到哪些事物的管理？

通过以上探究活动可知，当数据组织的方式、数据之间的关系不同时，实现同一功能的数据处理的过程就不同，数据处理的效率也不同。也就是说，在用计算机程序解决问题时，数据之间的关系会影响解决问题的步骤设计和程序执行效率。为了描述和处理越来越复杂的数据关系，人们需要研究数据结构。

在计算机世界中，把数据元素以及数据元素之间的关系构成的集合称为数据结构（Data Structure）。

数据元素之间的关系包括：

（1）数据元素之间的逻辑关系，即数据的逻辑结构（Logical Structure）。

（2）数据元素及其关系在计算机存储器中的存储方式，即数据的存储结构（Storage Structure），也称为数据的物理结构。

2. 数据的逻辑结构

根据数据元素之间逻辑关系的不同，数据结构有以下四种基本结构，如图1-14所示。

（1）集合结构：数据元素除同属于一个集合之外，没有其他关系。

（2）线性结构：数据元素之间存在前后有序的一对一的关系。

（3）树形结构：数据元素之间存在一对多的关系。

（4）图形结构：数据元素之间存在多对多的关系。

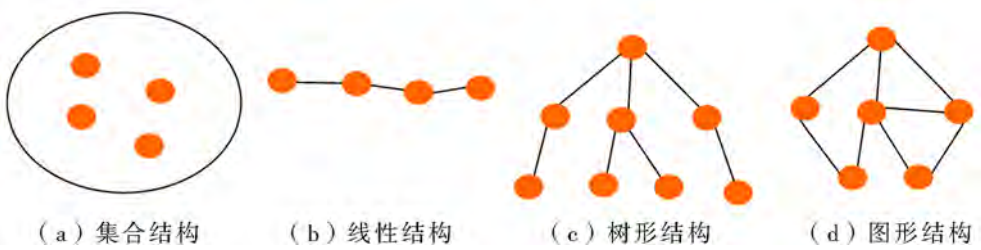


图1–14　四类基本数据结构示意图

因为数据的逻辑结构是由事物本身的逻辑关系确定的，与计算机存储器中具体如何存储这些数据无关，所以，我们所说的数据结构，一般指的就是数据的逻辑结构。

3. 数据的存储结构

数据结构在计算机存储器中的存储方式称为数据的存储结构，又称物理结构。它包括数据元素的存储和数据元素之间关系的存储。

二进制的一位是计算机存储器的最小单位，数据在计算机中的存储形式都是二进制位串。可以把这些位串看成数据元素在计算机中的存储形式。

数据元素之间的关系在计算机中有两种不同的表示方法：顺序存储和非顺序存储。因此，可以得到两种不同的存储结构：顺序存储结构和链式存储结构。

顺序存储结构是把逻辑上相邻的数据元素存储在物理位置也相邻的单元中，这是最基本的存储方法。

链式存储结构对逻辑上相邻的元素不要求其在物理位置上也相邻。这种存储结构就像链条一样一环扣一环，存入每一数据元素的同时，也存入其下一元素的存储地址。

图1–13（b）中的数据表的顺序存储结构和链式存储结构如图1–15所示。

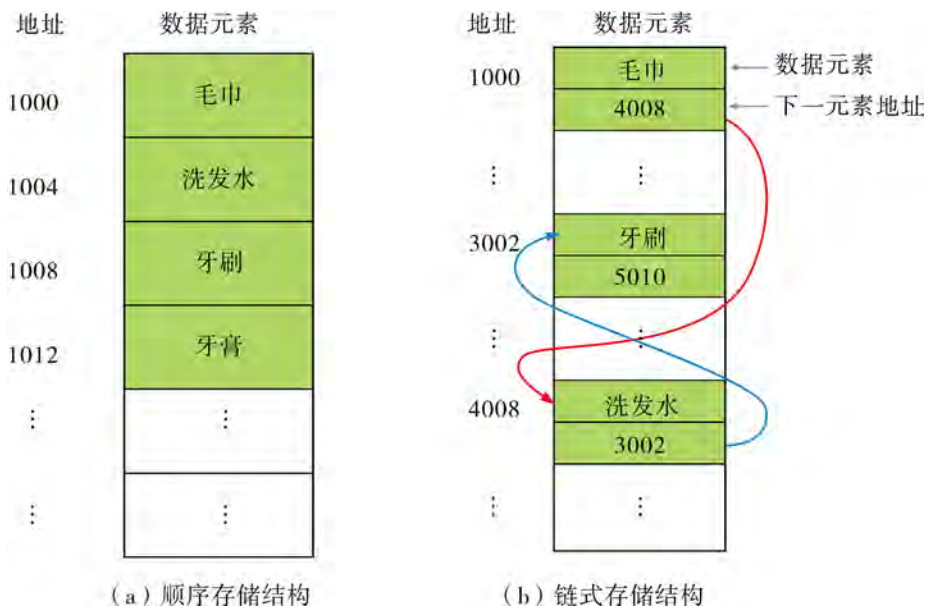


图1–15　数据表的存储结构示意图

观察

数据的存储结构不同，对数据进行同一操作的实现方法也不同。例如，对于插入操作，在顺序结构数据中的指定位置插入一个新数据，需要将插入点之后的全部数据逐个后移，空出一个位置来存放新数据，操作效率较低。而对于链式存储结构，则只需把插入点之前的地址指向新数据，把新数据的地址指向原插入点之后的数据即可。图1-16为两种不同存储结构下，在“洗发水”后增加数据元素“沐浴露”的操作过程示意图。

1. 在顺序存储结构中：

（1）查找到“洗发水”的位置。

（2）“洗发水”后的数据逐个后移，空出一个位置。

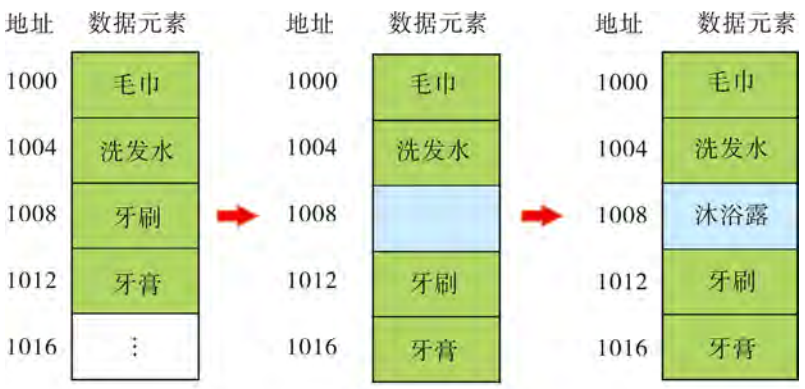
（3）“沐浴露”存放进空出的位置中。

2. 在链式存储结构中：

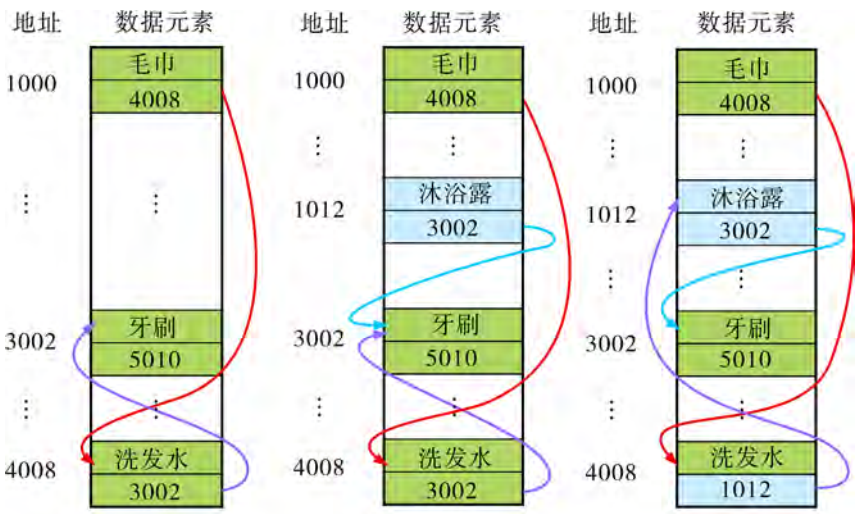
（1）找到空间存放“沐浴露”数据，记住这个位置的地址。

（2）“沐浴露”数据的下一地址指向原“洗发水”的下一地址，即指向“牙刷”数据。

（3）查找到数据“洗发水”，并使其下一地址指向“沐浴露”数据。



（a）顺序存储结构插入操作



（b）链式存储结构插入操作

图1-16　不同存储结构插入操作的过程

1.3.2 数据类型

计算机通过执行程序进行数据处理。对于不同的数据，能执行的操作不尽相同。对于大多数编写程序的人来说，只需要关心数据的取值范围、数据元素间的关系、施加在数据上的操作规则。至于某种操作在计算机中如何实现，对程序员来说并不重要。例如，对于求和操作，程序员注重的仅仅是其“数学上求和”的抽象特性，而不是其在计算机硬件上究竟如何实现。于是，封装了数据操作的数据类型就被引入程序设计语言中。

1. 数据类型

数据类型是与数据结构密切相关的一个概念，是对数据的取值范围、数据元素之间的结构以及允许施加操作的一种总体描述。

与数据结构相比，数据类型增加了对施加在数据元素上的操作的定义，即对数据的运算规则的定义。常用的运算有检索、插入、删除、更新、排序等，这些运算实质上是在抽象的数据上所施加的一系列的抽象的操作。

每种程序设计语言都会定义自己的数据类型，用于表示常用的数据结构以及在其上的操作。在用高级程序语言编写的程序中，每个变量、常量或表达式都有一个确定的数据类型。当我们定义某种类型的变量时，就是规定了这个变量的取值范围以及确定的操作（运算）规则。

如图1-17所示的为C++语言中的短整型数据类型。短整型数据类型的值集为整数的一个子集（取值范围为-32768~32767），可进行的操作有加、减、乘、除和取模等算术运算。

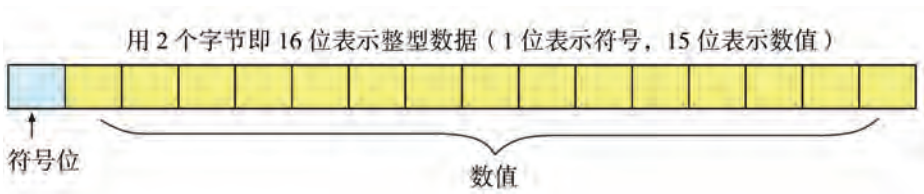


图1-17 C++语言中的短整型数据类型示意图

一种确定的数据类型有以下特点：

（1）所有数据元素都具有相同的取值范围。

（2）所有数据元素之间都具有相同的关系。

（3）一般地，数据元素经过运算、操作之后所得的结果，其值依然落在原来的取值范围内；这个结果与其他数据元素之间的关系也必须与原有数据元素之间的关系一致。

例如，整型变量之间的除法运算，结果也只能是整数。因此，整型变量的除法运算实际上是整除运算。如果整型变量的取值范围为-32767~32767，则两个整型变量的值分别

为300和500，做加法运算，结果为800，在整型变量的取值范围内，正确；但是，如果是30 000和30 000做加法，则会发生错误——结果60 000超出了整型变量的取值范围，发生溢出错误。

不同的程序设计语言和不同的计算机系统对于数据类型的定义和实现可能稍有不同。

2. 数据类型的分类

按数据类型“值”的不同特性，高级程序设计语言中的数据类型可分为两类：简单类型和结构类型。

简单类型中的每个数据都是不可再分割的整体。例如C++语言中的基本类型（整型、实型、字符型和枚举型）、指针类型和空类型。

结构类型是由简单类型按照一定的规则构造的。而且，结构类型内部还可以再包含结构类型。所以，每一种结构类型的值都是可以再分解为若干个简单类型或结构类型的值。

图1-18为简单数据类型和结构数据类型数据表。图中两个数据表均由若干行组成，（a）中每一行只有一个数据项，而（b）中每行可以看成一个两列的数据表，而每个数据表包含两个数据项，是可以分解的结构。

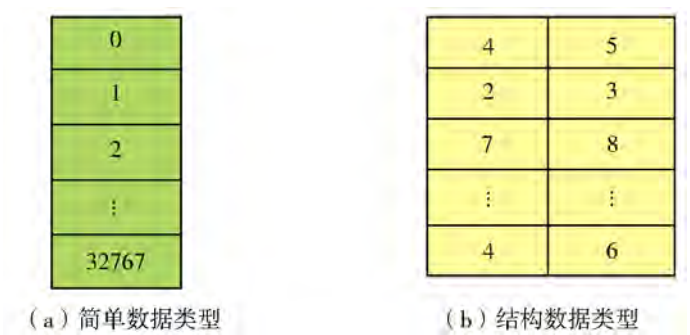


图1-18　简单数据类型和结构数据类型数据

1.3.3 数据结构的重要作用

从20世纪60年代末开始，程序设计已从技巧发展成一门科学。与程序设计联系非常密切的课程有数据结构、算法分析与设计、程序设计方法。瑞士的计算机科学家尼古拉斯·沃斯曾经提出：“算法+数据结构=程序”（Algorithms+Data Structures=Programs）。这句话简洁明了地概括了算法、数据结构、程序三者之间的关系。

算法是解决问题的有限步骤的序列，是一系列解决问题的清晰指令。

数据结构是数据元素以及数据元素之间的关系的集合。

程序是为实现特定目标或解决特定问题而用计算机语言编写的命令序列的集合。

本质上，数据是客观事物表示的一种抽象结果，而数据结构这门课程就是研究如何把客观世界待处理的信息逐层抽象成计算机可以接受的某种形式。通俗地说，数据结构就是研究数据之间的相互关系，也就是数据的组织形式的一门科学。

分析

数据结构在解决问题的过程中有重要作用。例如本节开头的商品查询问题，与此相似的还有电话查询、图书查询、人员查询等，此类问题的共同点是从大量的数据中查找目标数据。如电话查询，从表1-6的通信录中查找到对应的姓名，获得其电话号码。

表1-6 通信录

姓名	号码
张达	13712345678
熊二	13523456789
李三	13012345678
⋮	⋮

用计算机解决此问题，首先要对数据进行组织，设计数据结构。

最简单的方法就是将生活中的通信录直接转化存储，构造一张号码表，表中每个数据元素包含两个数据项：姓名和号码。将表中的数据元素顺序存储在计算机中，查找时从头开始依次查对姓名，直到找到目标或找完整张表为止。

当数据量大的时候，这种方法就不实用了。要提高查找速度，就要改进数据表的结构和存储方式。可通过以下方法改进数据结构的设计：

（1）号码表中的数据元素按姓氏排列。

（2）设计一张姓氏索引表，采用如图1-19所示的存储结构。

姓氏	号码
李	0000
⋮	⋮
熊	4206
⋮	⋮
张	9700
⋮	⋮

（a）索引表

地址	姓名	号码
0000	李三	13012345678
0006	李四	13023456789
⋮	⋮	⋮
4206	熊二	13523456789
4212	熊三	13523456789
⋮	⋮	⋮
9700	张达	13712345678
9706	张二	18923456789
⋮	⋮	⋮

（b）有序的号码表

图1-19 带索引的号码表

对应以上数据结构的查找过程是：

（1）先在索引表中查对姓氏，得到该姓氏在号码表中的起始地址。

（2）在号码表中从待查姓氏的起始地址开始进行查找。

这样查找号码表时，就无须查找其他姓氏的名字了。因此，在这种新的结构上产生的查找算法就更为有效。

程序员通过定义数据结构来描述数据以及数据之间的关系。算法的设计取决于选定的数据结构（逻辑结构），而算法的实现依赖于采用的存储结构。选择什么样的逻辑结构决定了处理数据的步骤和方法，选择何种存储结构则决定了如何编写程序语句。如果数据结构选择不好，除影响程序开发速度之外，更重要的是影响设计出来的程序的运行效率。所以，学好数据结构是学好程序设计的基础。

项目实施

各小组根据项目选题及拟订的项目方案，结合1.2节和1.3节所学知识，厘清如下问题，并参照项目范例的样式，完成所选定项目调查报告的撰写。

1. 针对收集到的数据，哪些数据适合所选定的项目？
2. 应采用哪种数据模型组织数据？
3. 进一步理解数据、数据结构的概念。

成果交流

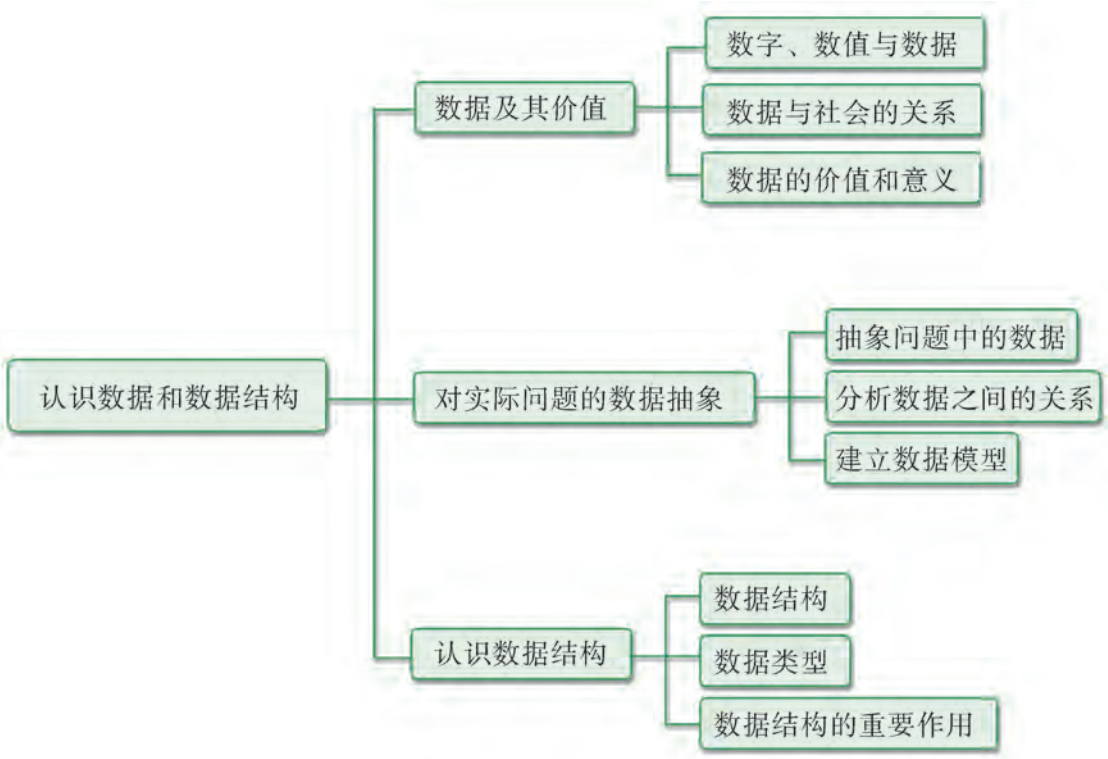
各小组运用数字化学习工具，将所完成的项目成果，在小组或班级上进行展示与交流，共享创造、分享快乐。

活动评价

各小组根据项目选题、拟订的项目方案、实施情况以及所形成的项目成果，利用教科书附录2的“项目活动评价表”，开展项目学习活动评价。

本章扼要回顾

同学们通过本章学习，根据“认识数据和数据结构”知识结构图，扼要回顾、总结、归纳学过的内容，建立自己的知识结构体系。



回顾与总结

本章学业评价

同学们完成下列测试题（更多的测试题可以在教科书的配套学习资源包中查看），并通过“本章扼要回顾”以及本章的项目活动评价，综合评价自己在信息技术知识与技能、解决实际问题的过程与方法，以及相关情感态度与价值观的形成等方面，是否达到了本章的学习目标。

1. 单选题

（1）对数字、数值、数据，以下描述正确的是（　　）。

A.“一个苹果重150克”——“150克”是数值

B.“观察一幅画获得了美的感受”——“美的感受”是数据

C.“100读作一百”——同一个数在不同记数系统中的写法不同

D.“计算机擅长计算”——计算机只能处理数值数据

（2）关于大数据的描述，下列说法错误的是（　　）。

A. 当今社会注重隐私，只要人们不愿意，人们在社会生活中生成的数据就不会被收集

B. 大数据计算可以给人们的社会活动提供参考

C. 大数据计算的结果，能够调节、影响人们的活动

D. 自然界中的事物也能提供数据

（3）数据结构在解决问题的过程中有重要作用。下列对数据结构的描述中正确的是（　　）。

A. 对同一事物，只能构造出一种数据结构

B. 选择的数据结构不同，解决问题的步骤也不同

C. 数据逻辑结构中相邻的数据，其存储位置也一定相邻

D. 对同一操作如插入、删除等，不同数据存储结构的实现方法相同

2. 思考题

将同一事物放置在不同应用场景中，其被抽取出来的数据属性是否一样？例如，在学校正常的教学活动中各种不同场景下的“学生”数据：参加运动会的学生，学业管理系统中的学生，食堂管理系统中的学生，社团管理系统中的学生。不同场景下的“学生”数据各应包含什么属性？

3. 情境题

学校举办校运会，为了鼓励各班踊跃参加各比赛项目，准备评选出校运会优秀班级。其计分规则为：每参与一个项目计1分；项目获前一、二、三名分别计5、4、3分；项目破学校纪录另加2分；全校各班计算总分，总分前三名的获优秀班级称号。

请根据以上情境，回答下列问题。

①抽取出校运会班级数据，建立班级数据模型，设计数据结构。

②简单描述使用上述数据结构计算选出优秀班级的过程。

第二章
数据的存储方式

为解决不同的实际问题，数据的存储方式各不相同。对于数据元素之间具有相邻顺序关系的，通常采用线性结构进行存储。有的数据存储在一组地址连续的存储单元中，属于顺序结构；有的则存储在一组任意地址的存储单元中，这组存储单元可以是连续的，也可以是不连续的，属于链式结构。因为存储方式不同，我们对其中数据元素的访问和操作的形式也不同。事实上，我们可以利用数组和链表这两种基本的数据结构来实现以上不同方式的数据存储与组织，这些结构都属于线性结构。

本章将通过“信息化管理程序设计”项目，进行自主、协作、探究学习，让同学们开展案例分析、编程实践，认识数据的一般存储结构，理解数组和链表的概念及其基本操作，掌握合理选用数据结构组织、存储数据的方法，从而将知识建构、技能培养与思维发展融入运用数字化工具解决问题和完成任务的过程中，促进信息技术学科核心素养达成，完成项目学习目标。

- 数据存储的顺序结构与链式结构
- 数据的顺序存储与组织
- 数据的链式存储与组织
- 数组与链表及其应用

项目范例　超市商品的信息化管理程序设计

情境

商品的进销存是超市管理的重中之重，利用计算机建立合理的管理系统，对包含商品相关信息的数据进行存储与组织，可以帮助超市工作人员及时了解超市的进货、销售、库存等信息，从而达到减少盲目采购、库存积压或脱销，合理控制成本、制订价格策略以及提高市场灵敏度的目的。

主题

超市商品的信息化管理程序设计

规划

根据项目范例的主题，在小组中组织讨论，利用思维导图工具，制订项目学习规划，如图2–1所示。

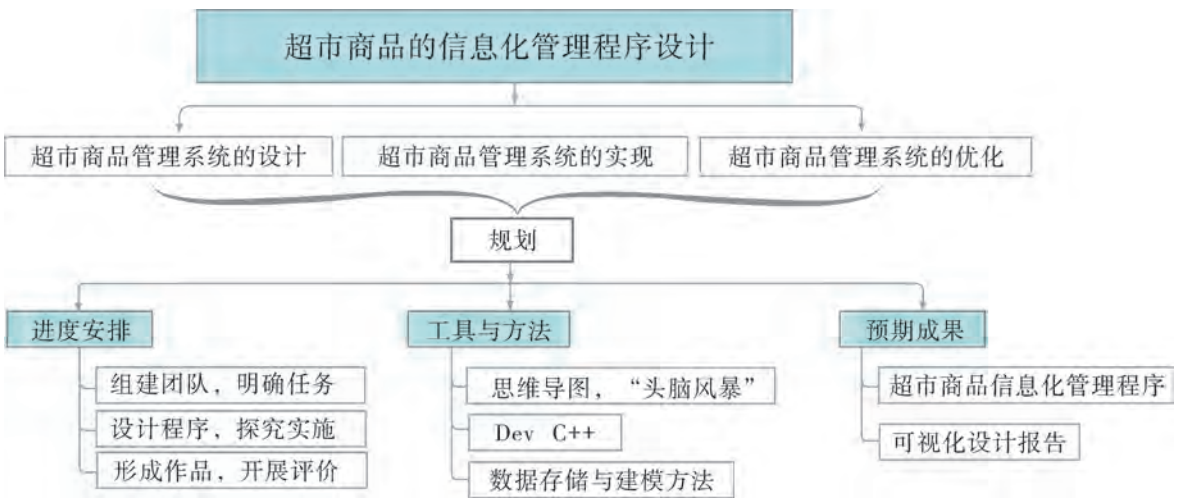


图2–1　“超市商品的信息化管理程序设计”项目学习规划

探究

根据项目学习规划的安排，通过调查、案例分析、文献阅读和网上资料搜索，开展“超市商品的信息化管理程序设计”项目学习探究活动，如表2–1所示。

表2-1 “超市商品的信息化管理程序设计”项目学习探究活动

探究活动	学习内容		知识技能
超市商品管理系统的设计	分析系统功能。	基本信息管理功能分析。 库存管理功能分析。 进货管理功能分析。 销售管理功能分析。	理解数组、链表等基本数据结构的概念。 明确数组、链表在不同类型数据存储中的应用。
	建立数据模型。	分析系统必须存储的数据。 提取重要属性。 建立商品基本信息表、进货表、销售表。	
超市商品管理系统的实现	商品基本信息管理及库存管理。	根据商品基本信息及库存数据模型选择数组存储与组织数据，解决问题。	能编程实现数组、链表等基本数据结构的相关操作。
	商品进货管理及销售管理。	根据商品进货及销售数据模型选择链表存储与组织数据，解决问题。	
超市商品管理系统的优化	选择最佳策略。 实现系统完整功能。	根据商品管理中的各种数据的操作特点，选择最优的数据结构进行存储与组织数据。	比较数组、链表的区别。

实施项目学习各项探究活动，进一步认识数组及链表。

在小组开展项目范例学习过程中，利用思维导图工具梳理小组成员在“头脑风暴”活动中的观点，建立观点结构图，运用多媒体创作工具（如演示文稿、在线编辑工具等），综合加工和表达，形成项目范例可视化学习成果，并通过各种分享平台发布，共享创造、分享快乐。例如，运用在线编辑工具制作的“超市商品的信息化管理程序设计”可视化报告，可以在教科书的配套学习资源包中查看，其目录截图如图2-2所示。

图2-2 “超市商品的信息化管理程序设计”可视化报告的目录截图

根据教科书附录2的“项目活动评价表”，对项目范例的学习过程和学习成果在小组或班级上进行交流，开展项目学习活动评价。

同学们以3~6人组成一个小组，选择下面一个参考主题，或者自拟一个感兴趣的主题，开展项目学习。

1. 学校图书馆图书信息化管理程序设计
2. 校园文具店货品信息化管理程序设计
3. 校园艺术节作品信息化管理程序设计

各小组根据项目选题，参照项目范例的样式，利用思维导图工具，制订相应的项目方案。

各小组将完成的方案在全班进行展示交流，师生共同探讨、完善相应的项目方案。

2.1 数据存储的顺序结构与链式结构

用计算机解决现实世界的问题时，我们通常用数据来表示现实事物及其活动，使其可以在计算机中存储与处理。那么数据在计算机中是如何被存储的呢？接下来，我们来学习两种最基本的数据存储结构。不同的存储结构直接影响数据的运算。

2.1.1 数据存储的顺序结构

数据元素之间有一种常见的逻辑结构——线性结构，其数据元素中除第一个和最后一个数据元素之外，其他数据元素都是首尾相接的。就如生活中的一个队伍一样，队伍中

的每个人可以看作一个数据元素，除了第一个人和最后一个人之外，其他人都是跟在某一个人之后，且其后面又跟着另外一个人。如某数据结构中，数据元素的集合K和K上二元关系的集合R如下：

$K = \{ a_1, a_2, \cdots, a_{i-1}, a_i, \cdots, a_{n-1}, a_n \}$

$R = \{< a_1, a_2 >, < a_2, a_3 >, \cdots, < a_{i-1}, a_i >, \cdots, < a_{n-1}, a_n >\}$

其中，i为整数且$1 < i \leqslant n$。

在以上数据结构中，数据元素之间具有最简单的相邻关系，其中第一个元素仅有一个后继元素，最后一个元素仅有一个前驱元素，其他的元素都有且仅有一个前驱和一个后继元素。我们说a_{i-1}是a_i的前驱，a_i是a_{i-1}的后继。这种数据元素之间的线性关系指的是逻辑结构上的，而在物理存储上则不一定。

在物理存储上，其中一种结构是顺序结构，它是指各数据元素依次存储在计算机中一组地址连续的存储单元中。采用这种存储方式，在逻辑上前后相邻的两个元素在计算机内存中也是相邻的。

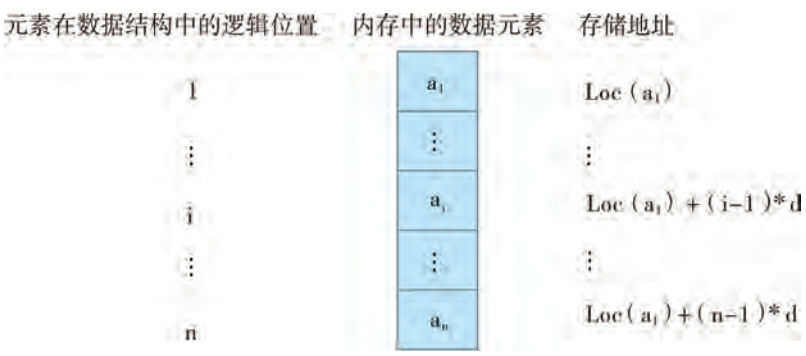


图2-3 顺序结构示意图

假设集合K中每个数据元素占用d个字节，且第一个数据元素a_1的计算机存储地址为$Loc(a_1)$，那么K的顺序结构如图2-3所示。

因为同一个数据结构中所有数据元素的类型相同，所以每个元素在存储器中占用的空间大小相同，因此要在以顺序结构存储的数据中查找某个元素是非常方便的。

探究活动

思考

为随时掌握奶粉的销售情况，超市在管理过程中，对某年的每月奶粉销量进行记录，得到如表2-2所示的数据表。请思考：若要在计算机中将数据以顺序结构进行存储，可以怎样表示？请画出示意图。

表2-2 超市某年的每月奶粉销量记录

月份	1	2	3	4	5	6	7	8	9	10	11	12
销量 / 罐	550	560	602	650	663	645	658	670	676	681	688	705

2.1.2　数据存储的链式结构

数据的链式结构与顺序结构不同，它的特点是存储各个数据元素的计算机存储单元的地址不一定是连续的。因此，为了表示每个数据元素a_i与其后继数据元素a_{i+1}之间的逻辑关系，对于数据元素a_i来说，除了存储其本身的信息，还需要存储其后继数据元素的存储位置信息。比如，人们到银行、医院办理业务时，一般都是在叫号系统取号之后就在大厅中静坐等候。此时的人们看似无序，但叫号系统为每个人分配的号码无形中把等候的人们串成了一条链子。

当以链式结构存储数据时，一种最简单也最常用的方法是分别用两个域存储数据元素的两部分信息：数据域存储数据元素自身信息，指针域存储后继数据元素的存储位置。以表2–2所示的数据表为例，若将其进行链式存储，则计算机将分配若干不连续的存储单元，其结构如图2–4所示。

在如图2–4所示的链式结构中，超市某年1月的奶粉销售信息存储在地址为0x401400的存储单元中，其数据域为550，指针域指向2月的奶粉销售信息所在的存储位置0x401408，而存储12月奶粉销售信息的存储单元，其指针域为NIL，表示空。

还有一些更复杂的链式结构，如将最后一个元素的指针域指向第一个数据元素，或者每个数据元素不仅包含数据元素自身信息、后继数据元素的存储位置，还包含其前驱数据元素的存储位置等。

图2–4　链式结构示意图

在数据的链式存储结构中，由于每个数据元素的存储位置是保存在它的前驱元素的指针域中的，只有当访问到其前驱元素后才能沿着前驱元素的指针域访问到该数据元素，因此在查找和访问上没有顺序存储结构方便。但是在添加和删除数据元素方面，链式存储结构只需要修改指针域的指向，便可以实现插入或删除，这一点则比顺序存储结构更加灵活。在后面学习用计算机程序实现对数据的顺序及链式存储和组织时，我们还会有更具体详细的介绍。

为实现超市商品的信息化管理，可以建立管理系统，在管理系统初始设计阶段，首先需要解决以下两个问题：

1. 超市商品管理系统应具有哪些功能？

为了让超市工作人员可以在进货、销售及仓库管理等环节，使用超市商品管理系统对商品相关信息进行查询、添加、删除和修改等操作，管理系统必须具备如表2-3所示的功能。

表2-3 超市商品管理系统功能分析

管理目的	管理系统功能
基本信息管理	建立、初始化及维护商品基本信息
库存管理	根据盘点情况查询、修改商品的当前库存量信息
进货管理	根据进货情况添加、删除、修改、查询商品进货信息
销售管理	根据销售情况添加、删除、修改、查询商品销售信息

2. 要实现管理系统的各个功能，需要计算机存储哪些数据？提取商品的重要属性并建立数据模型。

根据管理系统的功能分析，需要计算机存储包含商品基本信息、库存信息、进货信息及销售信息的数据，因此需对商品提取有价值的关键属性，如商品的编号、商品名称、规格、保质期、价格、当前库存量、进货时间、进货数量、进货价格、售出时间、售出数量、售出价格等，并分别建立商品基本信息（含库存信息）表、进货表、销售表。如表2-4至表2-6所示。

表2-4 超市商品基本信息表

编号	商品名称	规格 / 克	保质期 / 天	价格 / 元	当前库存量 / 件
001	Good1	50	120	50	100
⋮	⋮	⋮	⋮	⋮	⋮

表2-5　超市商品进货表

编号	进货时间	进货数量/件	进货价格/元
001	20170505	100	40
⋮	⋮	⋮	⋮

表2-6　超市商品销售表

编号	售出时间	售出数量/件	售出价格/元
001	20170520	20	50
⋮	⋮	⋮	⋮

各小组根据项目选题及拟订的项目方案，结合本节所学知识，可参考表2-4至表2-6，对所选定项目做如下分析：

1. 所设计的信息化管理程序需要实现什么功能？
2. 根据功能需求，分析计算机需要存储哪些数据。
3. 尝试建立相应的数据模型。

2.2 数据的顺序存储与组织

数据在计算机中能够以顺序结构进行存储。用计算机程序来实现对数据元素的顺序存储与组织，可以使用数组这种数据结构。

2.2.1 数组

1. 一维数组

在计算机程序设计中，可以通过数组（Array）这种数据结构来实现对具有相同数据类型且按一定逻辑顺序排列的数据元素的存储与组织，定义了一个数组就是定义了一块可使用的连续存储空间，数组的基本类型就是数据元素的类型，数组的长度就是数组元素的最大个数。在C++语言中，数组的一般定义方式为：

```
类型说明符 数组名[数组长度];
```

其中，“类型说明符”是任一种数据类型，“数组名”是用户定义的数组标识符，“数组长度”必须为常量表达式。例如：

```
int a[10];

#define maxsize 100
float f[maxsize];
```

我们可以通过数组的下标（Index）来直接存储或访问数组元素。数组的下标从0开始，即第1个元素被存储在数组下标为0的位置上，第2个元素被存储在下标为1的位置上，以此类推，第n个元素被存储在下标为n-1的位置上，下标使用中括号[ ]进行标识。如已定义数组：

```
int a[10];
```

则a为整数型数组，其长度为10，a[0]为该数组的第一个元素，a[i-1]（1≤i≤10）为该数组的第i个元素。

探究活动

观察

观察下面所定义的不同数组，分别说出它们的数组名、数组元素类型和数组长度。

```
int num[100];
char name[20];
float a[10],b[20];
```

实践

前面我们分析了对超市商品进行信息化管理需要存储的数据类别，并相应地建立了数据模型，接下来依据数据模型，利用计算机程序实现对数据的存储。

以婴儿食品的基本信息管理及库存管理为例，需要存储包含商品基本信息（包括库存信息）的数据，步骤如下：

1. 将数据模型表2-4进行具体化，得到如表2-7所示的婴儿食品基本信息表。

表2-7 婴儿食品基本信息表

编号	商品名称	规格 / 克	保质期 / 月	价格 / 元	当前库存量 / 件
B-F-001	MILK初生乐（0~6个月）有机奶粉	400	18	100	0
B-F-002	MILK成长乐（6~12个月）有机奶粉	800	18	160	0

（续表）

编号	商品名称	规格 / 克	保质期 / 月	价格 / 元	当前库存量 / 件
B-F-003	RICE婴幼儿米粉普通型	350	18	32	0
B-F-004	RICE婴幼儿米粉加钙加锌	350	18	45	0
⋮	⋮	⋮	⋮	⋮	⋮

2. 定义数组，用于存储包含婴儿食品基本信息的数据。

```
#define maxsize 1000
//商品基本信息结构定义
struct WareInfo{
   string wareno;  //商品编号
   ……              //定义其余数据
};
WareInfo BabyFood[maxsize] ; //商品信息数组
int TotalBF=0;     //记录已有的商品数量
```

在现实问题中，我们常常需要处理包含了若干不同类型信息的数据。显然，不能使用数组来直接存储这类数据，这时候可以借助struct来根据用户需要定义一个包含多个“成员”的结构体，如以上WareInfo包含商品编号、名称、规格、保质期、价格和当前库存量等信息，这些信息组合在一起就是一个结构。定义一个结构的一般形式为：

```
struct 结构名
{
  类型说明符 成员名1;
  类型说明符 成员名2;
  ……
};
```

结构定义之后，即可进行变量定义，如BabyFood就是一个元素类型为结构类型WareInfo的数组。

结构体具有多个成员，访问结构体变量的成员使用“.”（点）运算符，例如要访问婴儿食品第1个元素的编号，其表达式为：

```
BabyFood[0].wareno
```

3. 通过访问数组元素来存储或显示某个商品的相关信息，如BabyFood[i]表示数组BabyFood中下标为i（0≤i<maxsize）的元素。当我们编写以下程序命令时，可以对婴儿食品的所有信息进行存储或显示。

（1）输入婴儿食品的基本信息：

```
cout<<"请输入商品编号：";
cin>>wareno; //输入编号
BabyFood[TotalBF].wareno=wareno;
……              //其他数据的输入与存储操作
TotalBF++;    //商品总量加一
```

（2）输出显示婴儿食品的基本信息：

```
//输出商品信息，其中变量i表示商品数组的下标
cout<<"商品编号："<<BabyFood[i].wareno<<endl;
……
```

以上程序的完整代码可参考配套学习资源包“第二章\课本素材\婴儿食品的基本信息管理及库存管理.docx”。

按数组下标的个数，可以把数组分为一维数组、二维数组、多维数组等。前面我们所定义的数组都是一维数组。

一维数组是在解决现实数据存储问题时被经常采用的数据结构，只要存储的一组数据都具有相同的数据类型，且数据之间为线性关系，大多可以利用一维数组进行存储。如用数组来存储一元多项式$a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$，可以定义一个长度大于等于n+1的一维数组，每个数组元素存储多项式中某一项的系数，该数组元素的下标表示所存储的这一项的指数。以一元多项式$7x^4+x^3+12x+25$为例，利用一维数组存储该多项式并通过访问数组再次输出显示，其代码如下：

```
#include <iostream>
using namespace std;
#define maxsize 10
float a[maxsize];
int main()
{
   int i;
   for(i=0; i<maxsize; i++)a[i]=0;    //初始化数组所有元素为0
   //存储多项式7x^4+x^3+12x+25的系数，数组的下标与指数对应
   a[0]=25;          //0次方项的系数
   a[1]=12;          //1次方项的系数
   a[3]=1;           //3次方项的系数
   a[4]=7;           //4次方项的系数
   //输出此多项式
   for(i=maxsize-1; i>=0; i--)
```

```
    {
      if(a[i]!=0)       //只输出系数不为0的项
      {
        cout<<a[i]<<"x^"<<i<<" ";
        if(i>0) cout<<"+";    //如果不是常数项则输出加号
      }
    }
}
```

2. 二维数组

数组的下标可以是一个，也可以是多个。当数组有两个下标时，就称为二维数组。二维数组可以看成一维数组的嵌套，即首先把它看作一个一维数组，这个数组的每个元素又是一个一维数组。如一个二维数组b，可看成为某个一维数组共有n个元素，分别是b[0]，b[1]，…，b[n−1]，其中每个元素b[i]（0≤i＜n）又是一个有m个元素的一维数组，分别是b[i][0]，b[i][1]，…，b[i][m−1]。从逻辑结构上，我们也可以把这个二维数组看成一个n行m列的矩阵，并把第一个下标称为行下标，第二个下标称为列下标。

在C++语言中，二维数组的一般定义方式为：

```
类型说明符 数组名[常量表达式1][常量表达式2];
```

其中，“类型说明符”是任一种数据类型，“数组名”是用户定义的数组标识符，“常量表达式1”表示第一维的长度，“常量表达式2”表示第二维的长度。例如：

```
int b[4][3];

#define maxrows 10
#define maxcolumns 20
float f2[maxrows][maxcolumns];
```

如已定义一个二维数组b：

```
int b[4][3];
```

则b为4行3列的整数型二维数组，b的所有数组元素如下：

```
b[0][0], b[0][1], b[0][2]
b[1][0], b[1][1], b[1][2]
b[2][0], b[2][1], b[2][2]
b[3][0], b[3][1], b[3][2]
```

虽然二维数组在逻辑结构上具有行与列两个方向，但它作为一种顺序结构，所有元素在计算机内存空间中的物理存储地址仍是连续的。如在n行m列的二维数组b中，每个数据元素占用d个字节，假设b的第一个数据元素的计算机存储地址为Loc(b[0][0])，那么b的存储结构如图2–5所示。

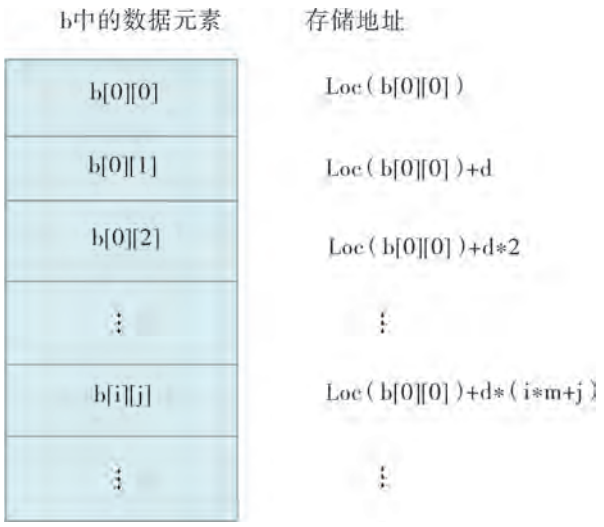


图2-5 二维数组存储结构示意图

利用二维数组，我们可以存储具有矩阵特点的事物信息，如井字棋、五子棋等棋盘游戏。一个棋盘通常由n×m个格子组成，并且由代表对弈双方的两种不同棋子完成棋盘的布局。可以用二维数组模拟棋盘，像对于井字棋棋盘可以定义3行3列的二维数组，对于围棋棋盘可以定义19行19列的二维数组……以井字棋棋盘为例，我们可以利用以下程序描述如图2-6所示的棋盘状态：

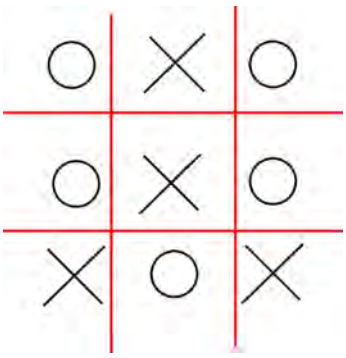


图2-6 井字棋棋盘

```
char chessboard[3][3];  //井字棋数组
//第1行
chessboard[0][0]='o';
chessboard[0][1]='x';
chessboard[0][2]='o';
//第2行
chessboard[1][0]='o';
chessboard[1][1]='x';
chessboard[1][2]='o';
//第3行
chessboard[2][0]='x';
chessboard[2][1]='o';
chessboard[2][2]='x';
```

交流

在生活中，我们常常可以看到事物以矩阵的形式进行摆放，如超市商品的货架摆放（如图2-7所示）。假设水果区的货架有n层，每一层有m列，处在某层某列的商品可能是编号为A001的苹果，也可能是编号为P005的鸭梨。在小组中展开交流，如果要用程序来实现对货架上不同位置的商品的编号进行存储，可以借助什么数据结构？又如何利用计算机程序定义相应的数据结构呢？

图2-7　超市商品的货架摆放

2.2.2　数组的基本操作

对于一维数组，通常有遍历（用于数组元素的赋值或查找）、插入元素、删除元素这几种操作。通过对数组的操作，我们可以实现以数组为数据结构的数据的各种管理功能。

一维数组的几种基本操作可用表2-8所示的程序代码实现。

表2-8　一维数组的基本操作实现代码

功能	程序段

遍历并赋值（输入）。	`#include <iostream>`
	`#define maxsize 10`
	`using namespace std;`
	
	`int a[maxsize];`
	
	`int main()`
	`{`
	`    for (int i=0; i<maxsize; i++)`
	`    {`
	`        cin>>a[i];    //依次输入每个元素`
	`    }`
	`}`

在数组a中查找元素x，如果找到，返回元素的下标；如果找不到，则返回-1。	`int array_find(int a[],int n,int x)`
	`{`
	`    for(int i=0; i<n; i++)`
	`        if(a[i]==x) return i;    //若找到，返回位置i`
	`    return -1;    //找不到，返回-1`
	`}`

（续表）

功能	程序段

在数组a中第i（i＞0）个位置插入元素x，原来第i到第n个元素向后移（n为数组中已有元素个数）。	void array_insert(int a[],int n,int x,int i)
	{
	　　//第i到第n个元素向后移
	　　for(int j=n; j>=i; j--) a[j]=a[j-1];
	　　a[i-1]=x;
	}

从数组a中删除第i（i＞0）个元素。	void array_delete(int a[],int n,int i)
	{
	　　for(int j=i; j<n; j++) a[j-1]=a[j];
	}

一维数组的几种基本操作示意图如图2-8所示。

通过对数组实施操作，可以对数组中的元素进行不同的组织与管理。例如，要在一组有序数据中插入一个新的数据，并使插入后的这组数据依然有序，可以借助以下操作：

（1）遍历数组并查找适合新数据插入的数组位置i。

（2）将数组下标为i到n-1位置的数据向后移动。

（3）将新数据保存到数组下标为i的位置中。

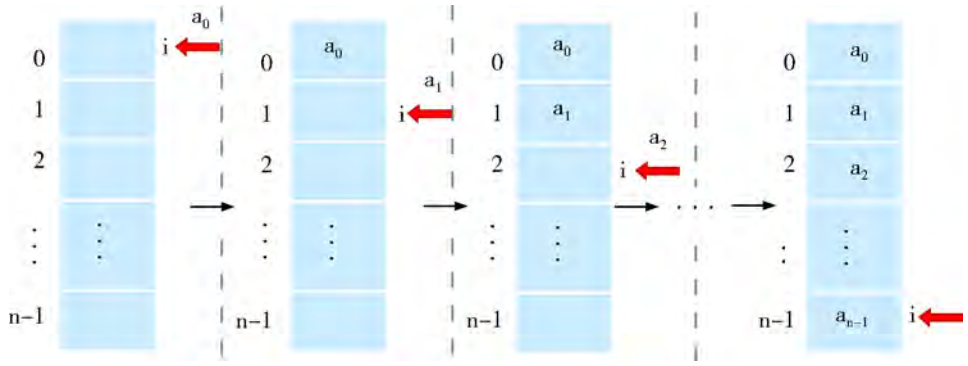
程序代码可参考配套学习资源包中的文档“第二章\课本素材\有序数组的元素插入.docx”。

对于二维数组，最常见的操作是遍历，与一维数组的遍历不同的是，我们把二维数组看成一个n行m列的矩阵，因此在程序实现时需要使用两重for循环，代码如表2-9所示。

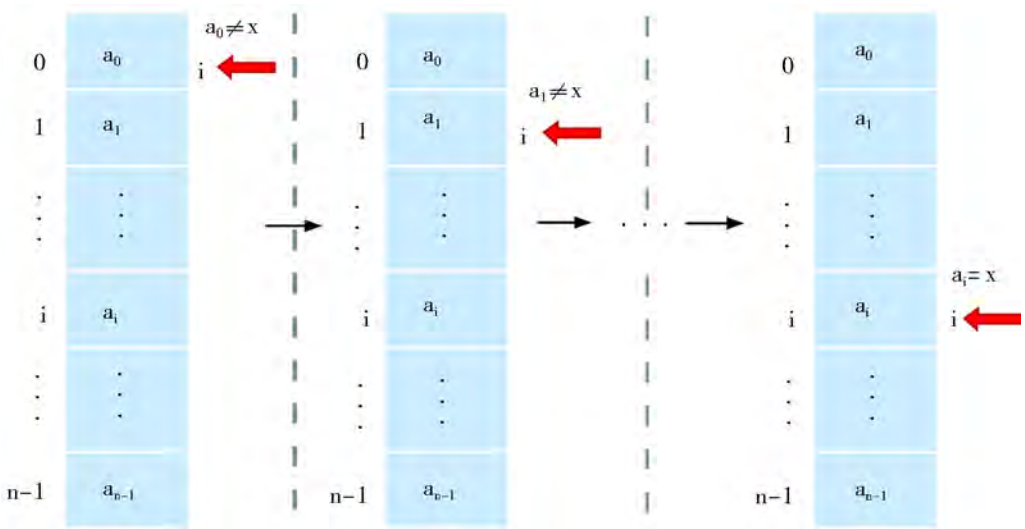
表2-9　二维数组遍历的实现代码

功能	程序段

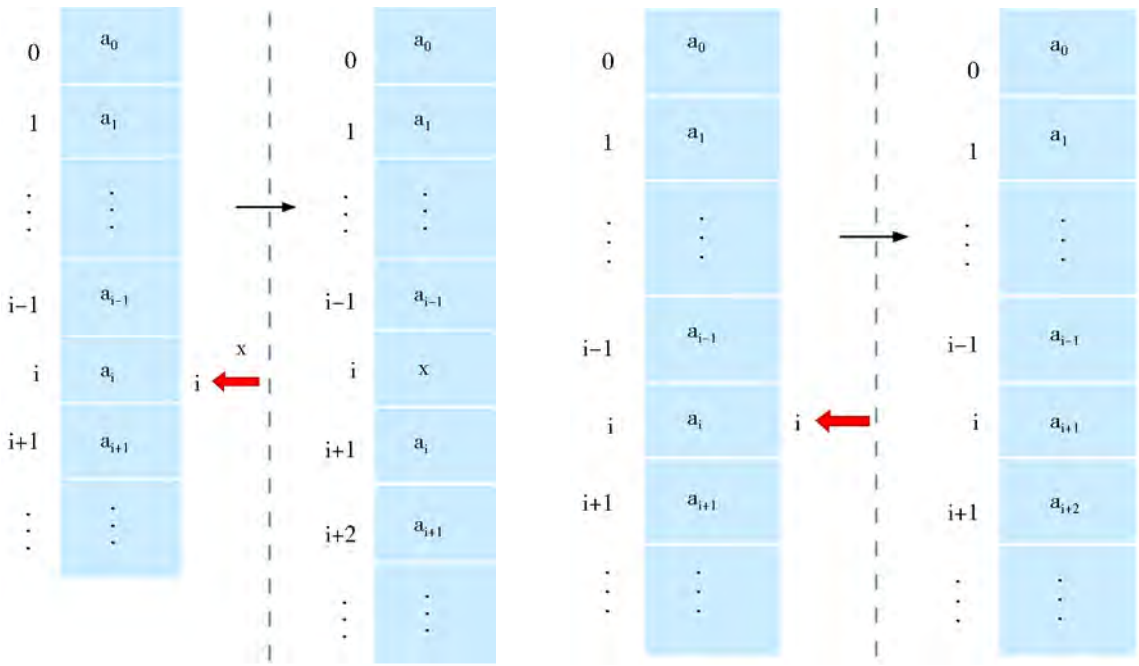
二维数组的遍历（遍历并赋值）。	#include <iostream>
	#define maxrow 5
	#define maxcolumn 10
	using namespace std;
	
	int a[maxrow][maxcolumn];
	
	int main()
	{
	　　int i,j;
	　　for(i=0; i<maxrow; i++)
	　　　　for(j=0; j<maxcolumn; j++)
	　　　　{
	　　　　　　cin>>a[i][j];　　//依次输入第(i,j)个元素
	　　　　}
	　　//按行列输出所有的元素
	　　for(i=0; i<maxrow; i++)
	　　{
	　　　　for(j=0; j<maxcolumn; j++)
	　　　　　　cout<<a[i][j]<<" ";
	　　　　cout<<endl;
	　　}
	}



（a）遍历数组并赋值



（b）遍历数组并查找x所在数组位置（以找到为例）



（c）插入元素x到数组下标为i的位置中　　（d）将数组下标为i的位置中的元素删除

图2-8　一维数组的基本操作示意图

当为超市各类商品的基本信息表定义数组后，接下来就可以通过对数组的不同操作来实现商品管理系统对商品的基本信息管理及库存管理的功能。以婴儿食品管理为例，根据功能描述，编写程序代码，可参考配套学习资源包中的文档“第二章\课本素材\婴儿食品管理的数组操作.docx”。

程序实现功能如下：

（1）初始化婴儿食品基本信息，即根据表2-7所列信息，对婴儿食品数组进行遍历和赋值操作。

（2）根据商品名称，查询该商品是否存在，即对婴儿食品数组进行查找操作。

（3）管理新上架的商品，即对婴儿食品数组进行插入操作（新商品插到数组最后）。

（4）管理刚下架的商品，即对婴儿食品数组进行删除操作（根据商品名称删除相应的数组元素）。

（5）根据商品名称，查询该商品的当前库存量，根据需要修改当前库存量。

各小组在前面的项目活动中，已分析本组需要存储哪些数据，请进一步思考并解决下列问题。

（1）在利用计算机程序解决问题时，哪些数据可以借助数组这一数据结构来实现存储？尝试用程序代码为其定义数组。

（2）在所选项目中需要对数据进行哪些管理？为实现这些管理，分别可以通过对已定义的数组进行哪些操作来实现这些管理，尝试编写相应的程序。

各小组完成程序编写后，对程序进行调试运行，验证是否能解决小组在2.1节项目实施中列出的相应问题，小组内进一步交流数组在现实数据存储与管理中的应用。

2.3 数据的链式存储与组织

数组通过连续的存储空间保存数据，通过数组下标可以对数组元素进行方便的访问。但是它也存在一个问题：数组在定义时必须预先确定长度，即数组的存储空间大小是在编写程序时就确定的，这段空间不能在程序运行时根据数据的增减动态而变化。当数据增加时，可能会超出所定义的数组大小；当数据减少时，未使用的存储空间也不会释放，造成内存浪费。若要解决这个问题，可以使用链式结构存储数据，这就涉及另一种数据结构——链表。

2.3.1　指针与指针变量

采用链式结构存储数据的特点是：每个数据元素除了要存储数据本身的信息外，还需要存储一个指示其后继数据元素的信息（即后继数据元素的存储位置）。在C++语言中，通常使用指针（Pointer）来实现这种存储位置的指示。

所谓指针，就是指某个内存单元的地址。在C++语言中，指针也是一种数据类型。与普通数据类型不同的是，定义指针变量（Pointer Variable）没有专门的关键字，而是利用“*”（星号）来进行定义。C++语言中定义指针变量的一般形式为：

```
类型说明符 *变量名;
```

例如，以下代码分别定义了一个整型指针变量pi、浮点型指针变量pf、字符串指针变量ps和结构类型指针变量pware：

```
//指针变量的定义形式：类型说明符 *变量名;
int *pi; //定义一个整型指针
float *pf; //定义一个浮点型指针
string *ps; //定义一个字符串指针
struct WareInfo *pware; //定义一个结构类型指针
```

为弄清指针的本质含义，首先要理解变量是什么。定义一个变量，其实就代表分配计算机内存中的一个存储单元用于存储数据，变量名通常就代表所存储的数据。但是，如果我们在变量名前面使用“&”运算符（取地址运算符），则表示获取变量对应存储空间的地址。例如：

```
int a=42;
int *pi=&a;
```

上面的两行代码表示：定义一个变量a用于存储整数42，定义一个整型指针变量pi，其初始值指向变量a，也就是说，变量pi保存的是变量a的内存地址，如图2-9所示。

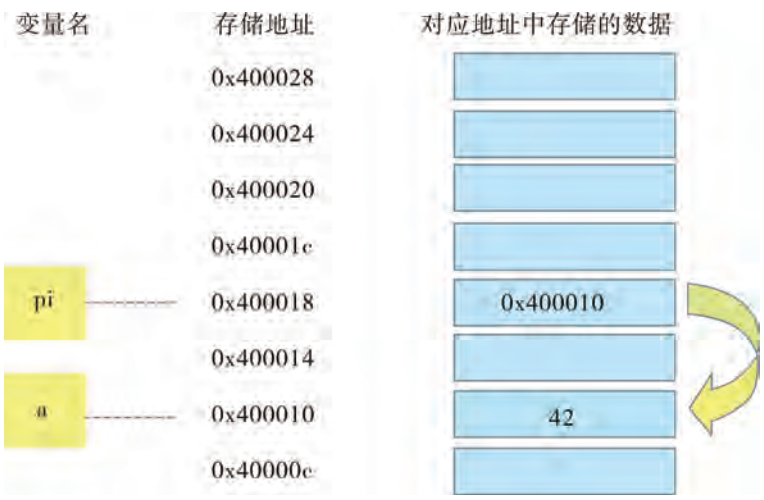


图2-9　指针变量的本质

在前面的定义中，使用了&运算符，这个运算符是一个取地址运算符，因此“&a”表示取变量a的存储地址。对于用取地址运算符&将一个变量的地址赋值给一个指针变量，我们一般也称为将该指针变量指向某个变量，例如“pi=&a;”表示将pi指向a。

对于指针变量，如果直接给它赋值，是改变指针所指向的存储单元；如果想修改它所指向的存储单元中的数据，则应该在变量名前加个“*”，如下所示：

```
int a,b;
int *pi;

pi=&a;   //pi指向变量a
*pi=88;  //*pi代表它所指向的存储单元的值，即变量a的值，
         //该语句实际是修改变量a的值为88
pi=&b;   //pi指向变量b
*pi=126; //*pi代表它所指向的存储单元的值，即变量b的值，
         //该语句实际是修改变量b的值为126
```

注意：如果将指针变量赋值为一个具体的数值，通常会导致程序出现错误。假如将上面的语句“*pi=126”改为“pi=126”，则在后续访问指针变量pi时可能会引起内存访问错误。所以我们需要分清楚是想改变指针变量所指向的存储地址，还是想修改指针变量所指向的存储单元的值。

分析以下代码的输出结果。

```
#include <iostream>
using namespace std;
int main()
{
  int a=22,b=33;
  int *pi;

  pi=&a;
  *pi=*pi+35;
  pi=&b;
  *pi=0;
  //输出变量a和pi所指内存单元的值
  cout<<"a="<<a<<",*pi="<<*pi;
}
```

2.3.2　链表

在计算机程序中，我们通过链表（Linked Lists）这种数据结构来实现链式结构的数据存储与组织，链表由一系列结点组成，结点可以动态生成。每个结点包括两个部分：一个是存储数据的数据域，另一个是存储下一个结点物理地址的指针域。数据元素的逻辑顺序是通过链表中的指针链接次序实现的。

链表有多种不同的类型，如单向链表、单向循环链表、双向链表、双向循环链表等，本节我们学习的是单向链表。

因为链表是由若干个结点链接而成的，每个结点都具有相同的结构，因此我们定义链表的结构时只需要定义链表结点的结构即可。以下是链表结点的类型定义：

```
//定义链表结点的结构
struct LinkNode{
  string data;     //结点数据
  LinkNode *next;  //下一结点指针
};
typedef struct LinkNode *LinkList;
```

在上面的定义中，结构LinkNode代表链表的一个结点，一个结点包括两个内容：data存储结点需要存储的数据，其数据类型根据实际需要定义；next是一个LinkNode类型的指针，它指向下一个链表结点。

定义最后还利用typedef关键字定义了一个名为LinkList的数据类型，它只是“struct LinkNode *”的别名。也就是说，LinkList其实是一个链表结点指针类型的别名。Typedef关键字用于为指定的类型声明一个新的名字，该名字可以像其他数据类型名称一样用于变量的定义。Typedef的一般用法为：

```
typedef 类型声明 类型别名;
```

因为链表就像铁链一样，一环扣一环，只要记住第一环，那整个链表就被记住了。所以，只需要定义一个LinkList类型的变量L（即LinkNode的指针变量），就可以记录和使用整个链表，L就称为链表的头指针，其定义如下：

```
LinkList L=NULL;  //定义链表头指针
```

在定义链表头指针L时，可将其初始值设为NULL。这是因为，对于指针变量，如果没有指向任何存储单元，我们通常会把它赋值为NULL，NULL是一个常量值，表示指针不指向任何一个存储单元。这样的指针称为空指针。

为了链表操作的方便，一般会在链表的第一个结点（保存第一个数据元素的结点）前面增加一个结点，这个结点的data域不保存数据（也可以保存链表长度或别的信息），我们将这个特殊的结点称为头结点。使用头结点的单向链表如图2-10所示，图中的符号“∧”表示指针域为空指针（赋值为NULL）。

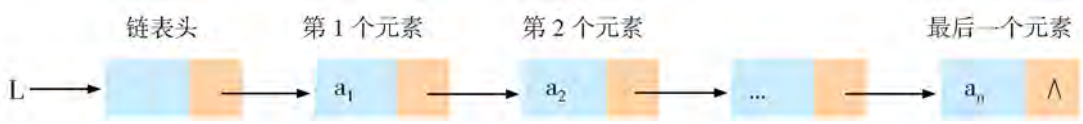


图2-10　单向链表

2.3.3 链表的基本操作

对于链表，通常有创建空链表、插入结点、删除结点、取得结点这几种操作。通过对链表的操作，可以实现以链表为存储结构的数据的各种组织和管理功能。

链表的结点可以动态增加或删除，结点的存储空间也是动态申请和释放的，在C++语言中，可以使用new关键字动态申请内存，使用delete关键字释放之前动态申请的空间。动态申请内存的基本形式如下所示：

```
指针变量=new 类型说明符;
```

释放之前动态申请的空间的基本形式如下所示：

```
delete 指针变量;
```

申请和删除链表结点的代码如下所示：

```
LinkNode *p;        //链表结点指针
p=new LinkNode;     //申请一个链表结点空间
delete p;           //释放结点所占用的空间
p=NULL;             //释放空间后将变量置为空指针
```

这里需要注意的是，使用delete释放的内存空间必须是使用new关键字申请的内存空间，否则将可能引起内存访问错误。

单向链表创建空链表、插入结点、删除结点和取结点元素的基本操作可用如表2-10所示的程序代码实现。

表2-10 单向链表的基本操作实现代码

功能	程序段

创建一个空链表。	//初始化空链表，空链表仅包含一个头结点
	void CreateEmptyList(LinkList &L)
	{
	LinkNode *p;
	p=new LinkNode; //新建头结点
	p->next=NULL; //下一结点为空
	L=p; //头指针指向该头结点
	}

在链表第i（i>0）个位置插入新的结点。	//在链表中的第i(i>0)个位置插入元素e
	//成功返回true，失败返回false
	bool ListInsert(LinkList &L,int i,string e)
	{
	LinkNode *p,*q;
	int j;
	
	//查找第i-1个结点
	p=L;
	j=0;
	while(p&&j<i-1)
	{

（续表）

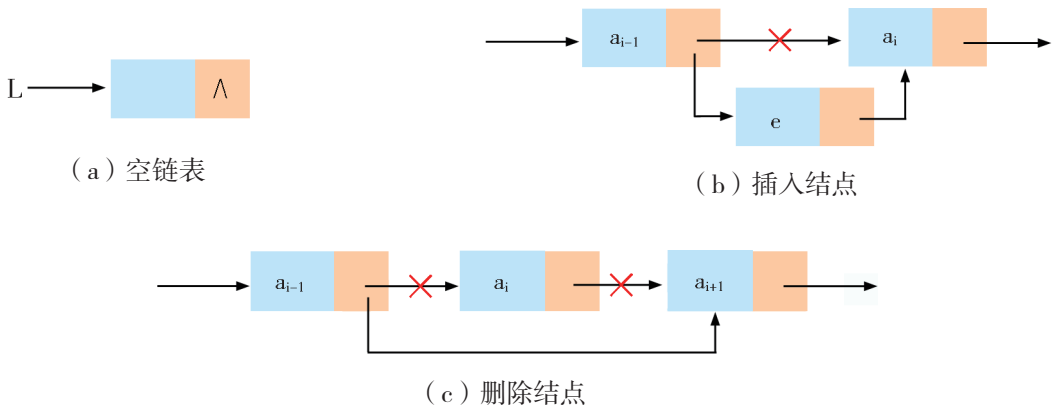
功能	程序段

在链表第i（i>0）个位置前插入新的结点。	p=p->next;
	j++;
	}
	if(!p) return false;　　//没有找到插入位置，返回false
	q=new LinkNode;　　//创建待插入结点
	q->data=e;　　//将数据存入结点
	q->next=p->next;　　//修改指针域，插入结点
	p->next=q;
	return true;
	}

从链表中删除第i（i>0）个结点。	//删除链表的第i(i>0)个结点
	//成功删除返回true，失败返回false
	bool ListDelete(LinkList &L,int i)
	{
	LinkNode *p,*q;
	int j;
	//查找第i-1个结点
	p=L;j=0;
	while (p&&j<i-1)
	{
	p=p->next;
	j++;
	}
	if(!p) return false;　　//没有找到删除位置，返回false
	q=p->next;
	if(!q) return false;　　//第i个结点不存在，返回false
	p->next=q->next;　　//修改指针域，使待删除结点脱离链表
	delete q;　　//释放结点内存空间
	return true;
	}

从链表中取得第i（i>0）个元素。	//从链表中取第i(i>0)个元素
	string GetElement(LinkList &L,int i)
	{
	LinkNode *p;
	int j;
	//查找第i个结点
	p=L;j=0;
	while(p&&j<i)
	{
	p=p->next;
	j++;
	}
	if(p) return p->data;　　//结点存在，返回结点数据
	else return NULL;　　//不存在，返回NULL
	}

如图2-11所示是单向链表的几种基本操作示意图。



（a）空链表　　（b）插入结点

（c）删除结点

图2-11　单向链表的基本操作示意图

由于链表可以实现灵活的内存动态管理，因此在数据的组织上有许多应用，如操作系统软件的实现中，大量使用各种链表存储系统信息，包括设备列表、内存分配信息、进程信息等。在很多应用软件中也会使用链表存储和处理数据，例如地图导航软件、网络数据包的记录等。

利用一维数组存储一元多项式时，数组的下标与项的指数对应，数组元素保存该项的系数，这样存储简单明了。但是对于类似这样的多项式：$3x^{100}+5x^3-2x+10$，最少需要具有101个元素的数组来保存，其中绝大部分的元素为零，不仅浪费空间，而且在进行多项式运算时，其执行效率也不高。如果采用链表来存储这样的多项式，则方便多了。用链表存储多项式，链表的每个结点对应一个项，只需要存储系数不为0的项，但如果采用这种方式，就不仅要存储项的系数，还要存储项的指数。以多项式$3x^{100}+5x^3-2x+10$为例，下面的代码定义了多项式的链表结构：

```
//存储多项式项的结点定义
struct term{
    float exponent;            //项的指数
    float coefficient;         //项的系数
    term *next;                //下一结点
};
typedef struct term *polynomial;   //定义多项式类型名称
```

基于上面的结构类型定义，我们可以定义一个向多项式添加项的函数，其代码如下：

```
//该函数向多项式中增加项（添加到最后）
void add_term(polynomial p,float exponent,float coefficient)
{
    term *q,*t;
```

```
    //新建保存项的结点
    q=new term;
    q->exponent=exponent;
    q->coefficient=coefficient;
    q->next=NULL;
    //找到链表的最后一个结点
    t=p;
    while(t->next!=NULL) t=t->next;
    //将结点q添加到末端
    t->next=q;
}
```

通过简单调用add_term函数，就可以实现多项式的存储。下面是调用代码，其中变量a为polynomial类型（即代表多项式链表的头指针）。

```
//存储多项式3x^100+5x^3-2x+10
add_term(a,100,3);
add_term(a,3,5);
add_term(a,1,-2);
add_term(a,0,10);
```

我们已经利用数组实现对超市商品的基本信息及库存的管理，下面我们尝试使用链表作为数据结构来存储超市各类商品的进货及销售信息，并通过对链表的不同操作来实现商品管理系统对商品的进货及销售管理功能。依据之前建立的数据模型表2-5、表2-6，以婴儿食品销售的几种常见管理为例，根据功能描述，编写程序代码，可参考配套学习资源包中的文档“第二章\课本素材\婴儿食品销售的链表定义和操作.docx”。

程序实现功能如下：

（1）在定义商品销售链表结构的基础上完成商品销售信息初始化，即建立商品销售空链表。

（2）当有商品售出时，根据商品编号对商品销售链表进行插入结点操作。

（3）当有误操作时，根据商品编号对商品销售链表进行删除结点操作。

（4）根据商品编号，查询某商品的销售情况，即对商品销售链表进行取结点操作。

各小组根据项目选题及拟订的项目方案，结合2.2节和2.3节所学知识，讨论以下问题，并编程实现所选定项目的信息化管理功能。

1. 针对信息化管理中的各个不同功能，分别适合使用数组还是链表来存储与组织数据？
2. 如何通过定义数组和链表，以及实施相应操作，实现信息化管理中的各个功能？
3. 进一步对比分析使用数组和使用链表在现实数据的存储与组织上的异同。

2.4 数组与链表及其应用

从超市商品管理系统的设计与实现中，我们看到，数组和链表都可以实现对数据的存储和组织，进而都可以实现对商品信息的增、删、查、改等操作。那么，这两种数据结构在实际应用时有什么区别？针对什么样的情况应该采用什么样的数据结构才能实现更有效、更科学的信息管理？下面让我们一起来分析。

2.4.1 数组与链表

在逻辑上具有简单的线性关系的数据，其数据元素中除了第一个和最后一个，其他数据元素都是首尾相接的。然而逻辑上线性排列的数据，在物理存储上却可以有两种不同的形式，即采用顺序的存储结构或链式的存储结构。根据存储结构的不同，对数据组织与管理的方式也不同，分别可以通过数组和链表来实现。

在数组中，元素之间的逻辑关系是通过数组的下标来表示的，如某个元素在数组中下标为i的位置，则该元素的前驱在数组中下标为i−1的位置，后继在数组中下标为i+1的位置。由于每个元素在物理存储器中是连续存放的，且占用内存空间大小相同，因此可以通过下标快速、简便地访问数组中的任何元素。然而数组元素的访问虽然很方便，但若要在数组中插入一个元素，则需要将待插入位置的元素及其后所有元素往后移动；同样地，如果想删除一个元素，也需要将待删除元素之后的所有元素往前移动。因此，当数组元素较多且需要频繁进行插入、删除操作时，显得相当不便。另一方面，定义一个数组，其实就是在内存中开辟一段大小固定且连续的存储空间，因此需要事先知道所需数组元素的个数。为防止定义的数组长度不足导致无法存储过多的数据，在不确定数据量多少的情况下，一般会定义长度足够大的数组，这在遇到实际存储的数据较少时无疑会造成内存空间的浪费。

在链表中，结点之间是通过结点中的指针联系到一起的，指针将多块不连续的内存空间连接，在逻辑上形成一片连续的数据存储空间。由于每个结点的物理存储位置保存在它前驱的指针域中，只有当访问到其前驱后才能通过前驱的指针访问到该结点。因此，要访

问链表中某一个结点，就得从第一个结点开始，一直找到该结点位置，显然降低了效率。但是当需要插入或删除某个结点时，则只要修改相应位置的结点的指针即可。因此，当需要频繁进行插入和删除数据元素操作时，使用链表作为存储结构则相当方便。另外，由于链表中的每个结点采用动态分配内存的形式生成，需要时可以分配内存空间，不需要时则将已分配的空间释放，不会造成内存空间的浪费。

探究活动

分析

小组内开展交流，分析数组和链表的特点及区别，填写表2-11，并说明这两种数据结构分别适用于哪些情况。

表2-11　数组与链表的对比分析

		数组	链表
存储结构的特点			
定义的方式			
操作的方式	查找元素		
	添加元素		
	删除元素		
适用的情况			

2.4.2　数组与链表的应用

数组和链表都适合存储与组织逻辑上具有简单线性关系的数据，但是它们具有各自的优缺点，我们应该根据具体情况选择最适合的方式来存储数据。

一般来说，如果数据的最大数量确定且不需要频繁进行增加、删除操作的，使用数组能达到快速访问的效果；反之，如果数据的最大数量不确定，需要频繁进行增加、删除操作的，则使用链表存储比较合适。

以超市商品信息管理系统为例：对于商品信息的存储与组织，我们既可以使用数组实现，也可以使用链表实现。但为了降低复杂性、提高效率，我们对商品种类通常相对固定、以查询操作居多的商品基本信息管理采用数组来实现，而对于每日商品频繁进货与售出，因而需频繁进行商品进出信息添加（也可能因误操作需要删除）的进货与销售管理，则采用链表来实现。

我们已经先后通过一维数组和链表实现了对一元多项式的存储，在此基础上，还可以实现多项式的加法运算。打开配套学习资源包中的文档“第二章\课本素材\多项式的加法运算.docx”，观察其中程序，程序中分别用一维数组和链表存储两个多项式，并完成这两个多项式的加法运算，请同学们分析一下它们各有什么特点。

待处理的多项式为：

多项式1：$72x^7+33x^4-18x^3+1$

多项式2：$36x^6-48x^4+22x-8$

使用一维数组实现一元多项式加法运算：

假设多项式的最高指数为n，则一维数组会存储指数从0到n的每一个项的系数（如果对应项没有，则系数为0），这样在做加法时就非常简单了，只要把相同下标的数组元素相加就能得到对应的多项式加法结果。

使用链表实现一元多项式加法运算：

用链表存储一元多项式与一维数组不同，只需要存储系数不为0的项，如果我们保存多项式时能够按照指数从高到低保存各个项，则加法运算可以描述为：

（1）取两个多项式p1、p2的第一项进行比较。

（2）如果两者的指数相同，则将系数相加作为结果项的系数，两者均取下一项继续比较。

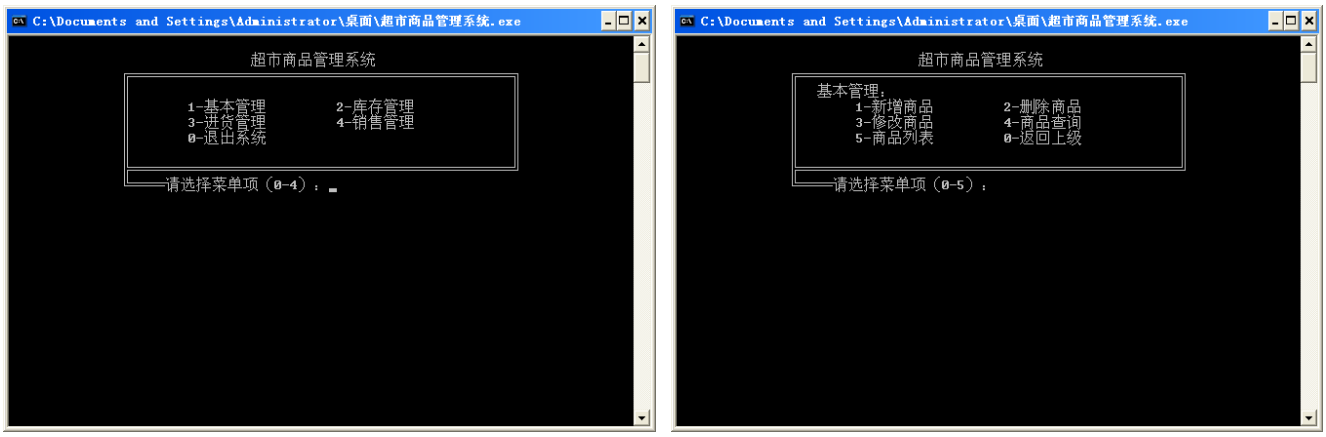
（3）如果p1的当前项指数>p2的当前项指数，则结果包含p1当前项，p1取下一项与p2的当前项继续比较。

（4）如果p1的当前项指数<p2的当前项指数，则结果包含p2当前项，p2取下一项与p1的当前项继续比较。

（5）重复以上步骤直至有一个链表处理完。

（6）将未处理完链表的剩余项包含到结果中。

根据表2-3的“超市商品管理系统功能分析”，结合本节所学知识，对管理系统中使用的数据结构再次进行整体分析，对比、选择最佳策略，实现完整的系统功能。最后形成的超市商品管理系统运行界面如图2-12所示。



（a）一级菜单界面　　（b）二级菜单界面

图2-12　超市商品管理系统运行界面

超市商品管理系统的相关实现程序等资源，可以在教科书的配套学习资源包中查看。

项目实施

各小组根据项目选题及拟订的项目方案，结合本节所学知识，对所选定的信息化管理程序中使用的数据结构再次进行整体分析，对比、选择最佳策略进行优化，最终实现完整的项目成果，并参照项目范例的样式，撰写相应的项目成果报告。

成果交流

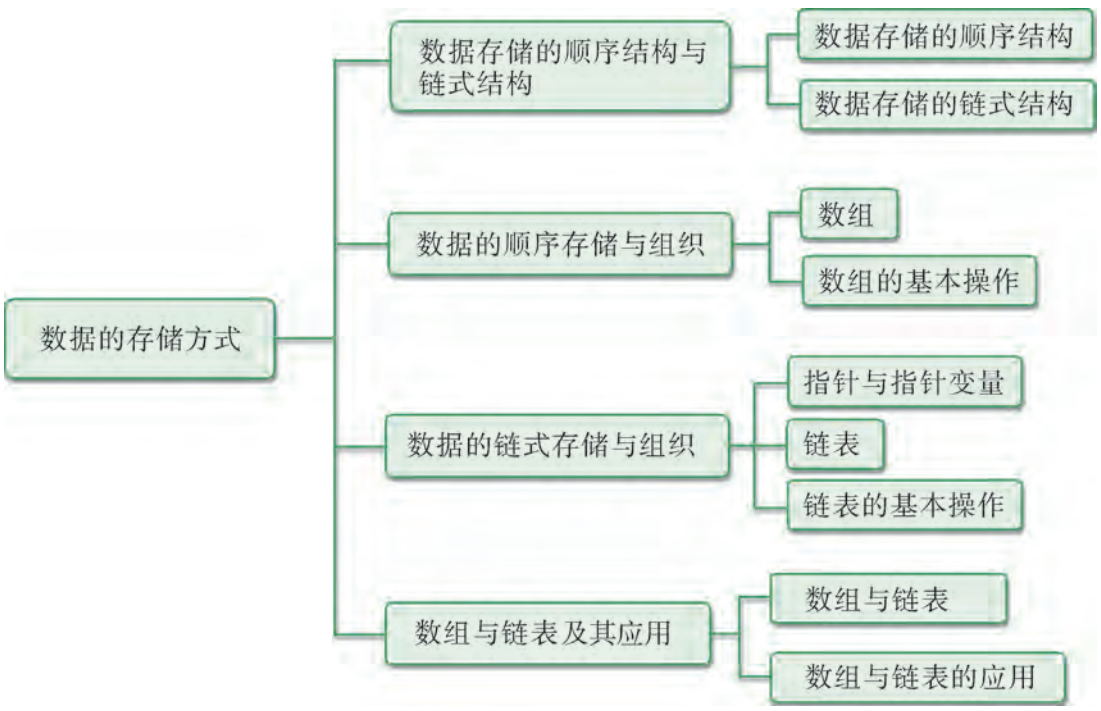
各小组运用数字化学习工具，将所完成的项目成果，在小组或班级上进行展示与交流，共享创造、分享快乐。

活动评价

各小组根据项目选题、拟订的项目方案、实施情况以及所形成的项目成果，利用教科书附录2的“项目活动评价表”，开展项目学习活动评价。

本章扼要回顾

同学们通过本章学习，根据“数据的存储方式”知识结构图，扼要回顾、总结、归纳学过的内容，建立自己的知识结构体系。



回顾与总结

本章学业评价

同学们完成下列测试题（更多的测试题可以在教科书的配套学习资源包中查看），并通过“本章扼要回顾”以及本章的项目活动评价，综合评价自己在信息技术知识与技能、解决实际问题的过程与方法，以及相关情感态度与价值观的形成等方面，是否达到了本章的学习目标。

1. 单选题

（1）有一组数1，1，2，3，5，8，…，55，保存在数组a中，这组数有这样一个特点：a[i+2]=a[i+1]+a[i]，则a[7]等于（　　）。

A. 11　　B. 13　　C. 21　　D. 34

（2）以下的程序片段是实现从具有n个元素的数组a中删除第t个元素的功能，请问横线处缺少的代码应该是（　　）。

```
for (i=t; i<n; i++)
{
_____________________________
}
```

A. a[t]=a[i];　　B. a[i]=a[i+1];

C. a[i]=a[i−1];　　D. a[i−1]=a[i];

（3）单向链表与数组都属于线性表，它们都是用于存储具有相同属性的数据，下列说法不正确的一项是（　　）。

A. 数组适合用于最大元素个数容易确定的情况

B. 存储相同的元素，单向链表比数组占用的存储空间要多

C. 查找特定元素，使用单向链表比使用数组方便

D. 对于需要频繁添加删除元素的情况，使用单向链表比使用数组合适

2. 思考题

要在一组数中找出其中的最大数：

① 假设这组数已经有序排列；

② 假设这组数是无序排列的。

对以上两种假设，分别使用数组和链表找出最大数，哪种查找效率更高？请思考并描述相应的查找过程。

3. 情境题

现实生活中，我们经常需要将两个有序的序列合并为一个新的有序序列，这在计算机中通常实现为将两个有序数组合并为一个新的有序数组。例如，图书馆有一批杂志需要按出版时间先后顺序进行整理，为了提高工作效率，安排两位同学同时整理，他们的做法是，先将杂志分为两堆，每位同学整理一堆，整理好后再合并为一堆。合并的方法是：先分别从两堆书中各取一本，对比出版年份（这里假设出版时间只有年份，没有月日）。如

果两个一样，则将两本书一起放入新的书架中，然后又取下一本继续比较；如果不一样，则将年份小的放到新的书架上，年份较大的那本留下来继续比较。接下来从刚才年份小的杂志所在的那一堆书中取下一本，继续与留下来的那本比较，方法与前面相同。依次类推，很快就可以将杂志整理好了。

假设数组a保存有第一位同学整理好的杂志的年份，数组b保存有第二位同学整理好的杂志的年份，如下所示：

```
int a[10]={1965,1966,1968,1979,1983,1984,1987,1988,1999,2005};
int b[10]={1963,1967,1968,1972,1973,1985,1997,1998,1999,2000};
```

请编程实现上面的合并工作过程。

第三章
线性数据的组织和存储

在现实生活中，具体问题抽取出来的数据模型多种多样，其中有些模型所包含的数据元素之间的关系是线性的、有序的。对这些数据的操作，有的是像排队那样先进先出，有的数据则像一叠盘子一样，只能从顶部取盘子，洗好的盘子也只能放在顶端。

本章将通过“服务自动化的模拟实验”项目，进行自主、协作、探究学习，让同学们理解线性表（包括字符串、队列、栈）的概念及其基本操作，并编程实现，从而将知识建构、技能培养与思维发展融入运用数字化工具解决问题和完成任务的过程中，促进信息技术学科核心素养达成，完成项目学习目标。

- 线性表
- 用字符串存储数据
- 用队列组织先进先出数据
- 用栈组织后进先出数据

项目范例 超市服务自动化的模拟实验

情境

超市与传统零售商在市场争夺中越来越占据优势，得益于超市的自动化管理和自动化服务。超市的自动化系统是根据超市经营规律与特点，利用计算机技术、网络与数据库技术、条码技术等来实现的。

随着人们对“自主”和“自助”关注度的提高，超市的服务自动化被提到了更高的高度，如自助询价、自助付款等。同时，超市还从多个角度探究管理自动化的策略，如商品需求管理、购物车管理等。

鲜肉菜区：行色匆匆的黄女士驻足在超市鱼肉档前，面对品种多样的新鲜食材，犹豫不决，因为她不知道如何才能把这些食材做成美味的菜。

服务区：超市服务台上排起了长队，有办卡的，有开发票的，有办理送货上门手续的，有兑换礼品的……左手推着婴儿车、右手推着购物车的“奶爸”李先生，满头大汗地寻找着办理送货上门手续的队伍。

停车场：超市服务员小丘穿梭在停车场里的购物车停放点，一趟一趟地回收顾客留下的购物车。

主题

超市服务自动化的模拟实验

规划

根据项目范例的主题，在小组中组织讨论，利用思维导图工具，制订项目学习规划，如图3–1所示。

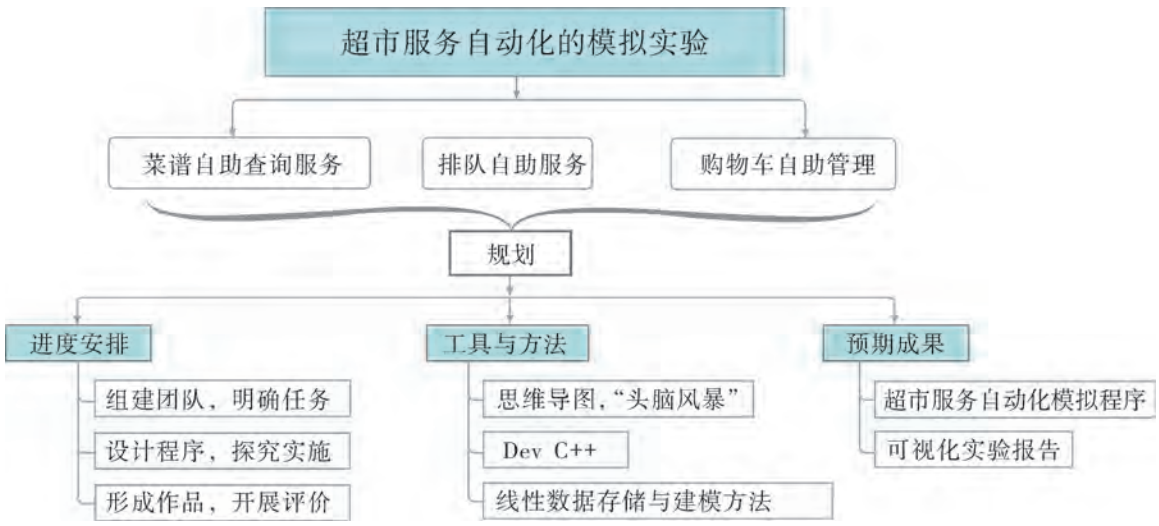


图3–1 “超市服务自动化的模拟实验”项目学习规划

根据项目学习规划的安排，通过调查、案例分析、文献阅读和网上资料搜索，开展“超市服务自动化的模拟实验”项目学习探究活动，如表3-1所示。

表3-1　“超市服务自动化的模拟实验”项目学习探究活动

探究活动	学习内容		知识技能
菜谱自助查询服务	抽取菜谱数据。	列出菜谱的属性。 抽取与问题相关的属性。 用数据表示属性。	理解字符串的概念及其基本操作。 编程实现字符串的基本操作。
	确定数据关系。	分析菜谱中各个属性的数据之间的关系。	
	建立数据模型。	理解菜谱数据结构的提炼方法。	
	选择数据结构。	根据菜谱数据模型的逻辑关系选择字符串表达菜谱数据。	
	编制模拟程序。	采用字符串的相关操作解决问题。	
排队自助服务	抽取队伍数据。	列出队伍的属性。 抽取与问题相关的属性。 用数据表示属性。	理解队列的概念及其基本操作。 编程实现队列的基本操作。
	确定数据关系。	分析队伍中各个属性的数据之间的关系。	
	建立数据模型。	理解队伍数据结构的提炼方法。	
	选择数据结构。	根据队伍数据模型的逻辑关系选择队列表达队伍数据。	
	编制模拟程序。	采用队列的相关操作解决问题。	
购物车自助管理	抽取购物车停放点数据。	列出购物车停放点的属性。 抽取与问题相关的属性。 用数据表示属性。	理解栈的概念及其基本操作。 编程实现栈的基本操作。
	确定数据关系。	分析购物车停放点中各个属性的数据之间的关系。	
	建立数据模型。	理解购物车停放点数据结构的提炼方法。	
	选择数据结构。	根据购物车停放点数据模型的逻辑关系选择栈表达购物车停放点数据。	
	编制模拟程序。	采用栈的相关操作解决问题。	

实施

实施项目学习各项探究活动，进一步理解线性表及其基本操作。

成果

在小组开展项目范例学习过程中，利用思维导图工具梳理小组成员在“头脑风暴”活动中的观点，建立观点结构图，运用多媒体创作工具（如演示文稿、在线编辑工具等），综合加工和表达，形成项目范例可视化学习成果，并通过各种分享平台发布，共享创造、分享快乐。例如，运用在线编辑工具制作的“超市服务自动化的模拟实验”可视化报告，可以在教科书的配套学习资源包中查看，其目录截图如图3–2所示。

图3–2　“超市服务自动化的模拟实验”可视化报告的目录截图

评价

根据教科书附录2的“项目活动评价表”，对项目范例的学习过程和学习成果在小组或班级上进行交流，开展项目学习活动评价。

项目选题

同学们以3~6人组成一个小组，选择下面一个参考主题，或者自拟一个感兴趣的主题，开展项目学习。

1. 图书馆借书服务自动化的模拟实验
2. 校运会运动员检录的模拟实验
3. 网络团购服务的模拟实验

各小组根据项目选题，参照项目范例的样式，利用思维导图工具，制订相应的项目方案。

各小组将完成的方案在全班进行展示交流，师生共同探讨、完善相应的项目方案。

3.1 线性表

信息化管理系统在生活中的应用很广泛，如超市商品信息化管理系统、学生档案管理系统等。而在各种信息化管理中，线性表是最基本的一种数据结构。

3.1.1 线性表及其运算

线性表（Linear List）是最简单且最常用的一种数据结构，是由若干个具有相同属性的数据元素组成的有限序列。比如，英文字母表（A，B，C，…，Z）就是一个长度为26的线性表，表中的每一个英文字母为一个数据元素；又如第一章的表1-4“超市客户表”，它的每一行就是一个数据元素，反映一位客户的相关情况，由姓名、性别、出生年月、职业、手机号码、地址等组成。

线性表具有如下的结构特点：

1. 均匀性

虽然不同数据表的数据元素可以是各种各样的，但同一线性表的各数据元素必定具有相同的数据类型和长度。

2. 有序性

各数据元素在线性表中的位置只取决于它们的序号，数据元素之间的相对位置是线性的，即存在唯一的“第一个”和“最后一个”数据元素，除了第一个和最后一个，其他元素前面均只有一个数据元素（直接前趋），后面也只有一个数据元素（直接后继）。

线性表：由n（n≥0）个数据元素a_1，a_2，…，a_n组成的有限序列。该序列中所有数据元素均具有相同的数据类型。其中，数据元素的个数n称为线性表的长度。

当n=0时，称为空表。

当n>0时，将非空的线性表记作：L=(a_1，a_2，…，a_{i-1}，a_i，a_{i+1}，…，a_n)。

L为线性表名称；

a_1为线性表的第一个数据元素，称为表头，a_n为线性表的最后一个数据元素，称为表尾；

a_1，a_2，…，a_{i-1}都是a_i（2≤i≤n）的前趋，其中a_{i-1}是a_i的直接前趋；

a_{i+1}，a_{i+2}，…，a_n都是a_i（1≤i≤n−1）的后继，其中a_{i+1}是a_i的直接后继。

对于线性表，常见的基本运算有：

（1）置空表setnull(L)，这是一个过程，其结果是将线性表L置为空表。

（2）求长度length(L)，这是一个函数，返回值为线性表L的长度。

（3）取得表中第i个元素get(L,i)，这是一个函数，当1≤i≤length(L)时，返回第i个数据元素。

（4）取直接前趋prior(L,a_i)，这是一个函数，当2≤i≤length(L)时，返回a_i的直接前趋a_{i-1}。

（5）取直接后继next(L,a_i)，这是一个函数，当1≤i≤length(L)−1时，返回a_i的直接后继a_{i+1}。

（6）根据特定值查找线性表locate(L,x)，这是一个函数，若线性表中存在值为x的数据元素a_i时，则返回i值，即a_i在线性表中的序号；若存在多个值为x的数据元素a_i时，则i为其序号最小的值；若不存在值为x的数据元素，则返回值零。

（7）插入数据元素insert(L,x,i)，这是一个过程，是将新数据元素x插入到数据元素a_i之前，因此当1≤i≤length(L)+1时有意义，运算的结果使长度为n的线性表变为长度为n+1的线性表。

（8）删除操作delete(L,i)，这是一个过程，当1≤i≤length(L)时，将线性表L中第i个数据元素删除，使长度为n的线性表变为长度为n−1的线性表。

以上这些数据的运算是定义在逻辑结构上的抽象运算，至于如何实现要待确定了数据的存储结构之后再考虑。

3.1.2 线性表的应用

对事物进行信息化管理时，通常需要先根据管理目的或需求提取事物的相关属性，将该属性下的信息借助表格进行存储、组织，以便后期使用。如第二章的表2−2呈现的超市某年每月奶粉销量记录就是一个线性表，超市基于记录和查询奶粉每阶段销售情况的目

的，提取每月奶粉销量这一属性，并将相关信息使用线性表进行表示：

```
Sales=(550,560,602,650,663,645,658,670,676,681,688,705)
```

Sales是一个长度为12的线性表，其中每个数据元素为一个整数。

还有一些更复杂的线性表，每个数据元素又可以由若干数据项组成。如第二章表2-7的婴儿食品基本信息表，也是一个线性表，表中每一个奶粉品种包含六个数据项：编号、商品名称、规格、保质期、价格和当前库存量。

3.2 用字符串存储数据

随着计算机技术的发展和应用的普及，非数值计算的应用越来越广泛。计算机中处理的非数值的数据也相应增加，如事务处理系统中的名称、序列号、产品规格说明等，这些数据一般被当作字符串数据进行处理。

3.2.1 字符串及其存储

字符串（String）一般简称为串，可以将它看作一种特殊的线性表，它的每个数据元素仅由一个字符组成。与一般线性表相比，字符串有以下特点：

（1）字符串的数据元素的类型限定为字符型。

（2）字符串操作的对象，多数情况下是字符串整体或其中一部分，当然也可以是单个的数据元素。

随着计算机技术的发展，字符串成为非数值计算问题中的主要操作对象，如信息检索、文本编辑、问答系统、音乐分析程序等。

生活中随处可见字符串的应用。

当我们注册入学的时候，我们会得到一串由数字组成的学号。每名同学的学号都是唯一的，如图3-3所示。用学号就能区分学校里的每一名同学。

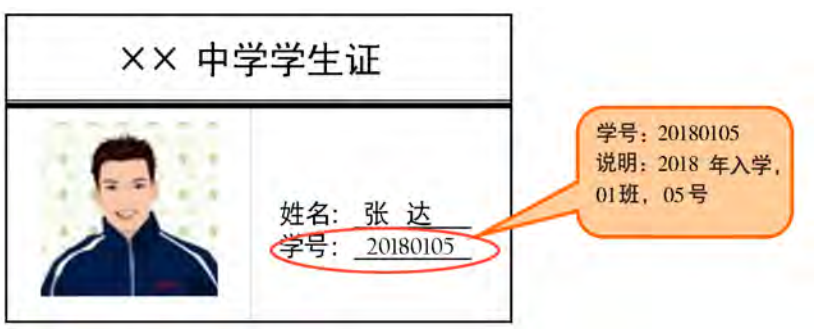


图3-3　学号

当我们出生后登记入户时，会得到一串代码，这串代码会跟随我们一辈子，成为识别我们身份的标识，它就是身份证号码，如图3-4所示。

图3-4 身份证号码

以上两个例子中，字符串被切分成小串，每个小串都被赋予了一定的意义。反过来说，在定义字符串的时候，根据解决问题的需要，将不同含义的字符序列进行有序组合。这样，在使用计算机处理问题的时候，我们就可以通过对字符串数据的操作来提高对事物的管理效率。

1. 字符串的描述

字符串是由零个或有限个字符构成的有序序列。一般记作：

$s="c_0c_1c_2\ldots c_{n-1}"\ (n\geqslant 0)$

其中：

s为串名，用双引号引起来的字符序列称为串值；c_i（$0\leqslant i\leqslant n-1$）可以是字母、数字或其他字符。下标i表示字符$c_i$在串中的位置。

双引号是串值的定界符，不是串的一部分。

字符串中字符的个数n称为串的长度。长度为0的字符串称为空串，此时串中没有任何字符。

注意：空格在字符串中也算一个字符；长度为1的字符串与单个字符的意义及可执行的操作是不同的。

例如，字符串“20180105”，长度为8，其中字符“8”的位置是3。

为了支持字符串的处理，在高级语言中引入了串的数据类型。并且，字符串变量与其他变量（如整型、实型等）一样，可以进行各种运算，字符串运算的基本函数和过程也可以同时建立。在C++语言中，字符串被定义为结构数据类型，可以直接用string来定义字符串变量：

```
string s;
```

一个字符串中任意个连续的字符组成的子序列称为该串的子串，包含这个子串的字符串就称为主串。一个子串在主串中的位置是用这个子串的第一个字符在主串中的位置来表示的。例如，s1="20180105"，s2="01"，则称s2是s1的子串，子串s2在s1中的位置是1。

探究活动

观察

运行配套学习资源包“第三章\课本素材\菜谱查询.exe”，输入“鸡腿”可得到如图3–5的运行结果。观察运行结果，结合实际需求，分析为菜谱数据建模的方法。

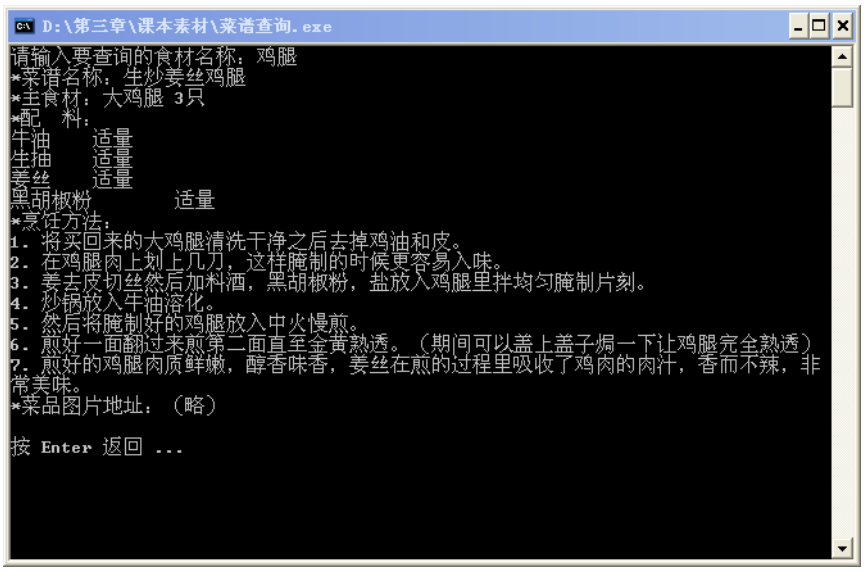


图3–5　菜谱查询程序运行情况

分析顾客对菜谱查询的需求可知，顾客通过食材的名称或编号查询到该食材的菜谱；顾客希望从菜谱中看到菜谱名称、主食材、配料、烹饪方法等内容，最好还能呈现成品菜的图片。菜谱名称、主食材、配料、烹饪方法都是一些文字描述，而通过记录存放图片文件的目录地址就可以找到并呈现图片。

菜谱是一个线性结构。每种菜谱包含菜谱名称、主食材、配料、烹饪方法、菜品图片地址等数据项。其中，菜谱名称、主食材、配料、烹饪方法等数据都是字符串，菜品图片文件的目录地址也是一串有序的字符，也属于字符串数据。在C++语言中，定义菜谱的数据结构如下：

```
struct caipu{
    string mingcheng;    //菜谱名称
    string shicai;       //主食材
    string peiliao;      //配料
    string tupian;       //菜品图片地址
    string fangfa;       //烹饪方法
};
```

2. 字符串的存储结构

字符串是一种特殊的线性表。因此，字符串的存储结构也有两种：静态的顺序存储结构，动态的链式存储结构或堆存储结构。

（1）串的顺序存储结构。

串的顺序存储结构与一维数组的类似，用一组地址连续的存储单元存储串值的字符序列，如图3-6所示。字符串的字符依次存放在这些存储单元中。因此，串可以用数组表示。

串s	2	0	1	8	0	1	0	5				
位置	0	1	2	3	4	5	6	7				

图3-6　串的顺序存储结构

字符串的顺序存储结构，使得字符的查询、访问变得简单。只要知道字符在字符串中的位置，就可以像访问数组元素一样通过位置访问这个字符。

但是，字符串的顺序存储结构给字符串的插入、删除造成不便。在字符串中插入子串时，必须将插入点后的元素全部依次后移；删除字符串的一个子串时，必须将删除后的部分全部前移。

此外，存储串的连续的存储单元一般要求先定义好最大长度，这也使得串的操作受限。两个字符串连接的结果，很可能超出这个最大长度。

（2）串的动态存储结构。

用顺序存储结构来存储字符串，因为其规模在定义的时候就已定下，容易造成存储空间的浪费；同时，线性表的插入和删除操作效率很低。因此，有些时候也采用动态存储方式。动态存储方式包括链式存储结构和堆存储结构。

在串的链式存储结构中，每个结点包含字符域和结点链接指针域，字符域存放字符，指针域存放指向下一结点的指针，如图3-7所示。因此，串可用单链表表示。

图3-7　串的链式存储结构

字符串的各种运算和串的存储结构有很大的关系，存储结构不同，其运算和操作也会有所不同。

在各种事务管理系统中，可以通过字符串的操作来实现对事物的自动化管理。例如，对于超市中商品信息的查询，可以通过商品条形码来获得商品的详细信息，也可以通过提取商品条形码中的某个子串来实现同类商品的比价功能。

为了优化系统，提高查询效率，拟升级菜谱自助查询系统的功能：除了通过食材名称获取菜谱，还可以直接选择某一类别食材的全部菜谱清单供顾客选择。优化升级的主要目的在于提供对“类别”的查询响应。因此，对数据结构进行以下改造：

方案一：改造“主食材”数据项为“食材编码”，在食材名前加入类别代码。

例如，经过改造后，主食材“鸡肉”升级为“103鸡肉”。其中，第一位是菜谱类别，为“1”时表示“肉菜”，第二、三位“03”对应为“鸡肉”。当要求查询所有“肉菜”时，则可以通过提取编码字符串中的“类别”项的信息，给出相关类别的全部菜谱。

方案二：直接增加“分类”数据项。

请根据上述的分析完成表3-2。

表3-2　系统升级优化改造方案

方案	改造数据项	增加数据项
菜谱数据结构		
查询功能的实现过程	1. 2. 3. ⋮	1. 2. 3. ⋮
数据存储空间占用、操作效率评价	优点： 缺点：	优点： 缺点：

3.2.2　字符串的基本操作

字符串的基本操作有赋值、连接、求串长、求子串、插入子串、删除子串、查找子串、判断两个串是否相等。目前，字符串在很多程序设计语言中被定义为结构数据类型，有关字符串的操作也被设计成系统函数，可以直接引用。

以C++语言为例，通常有以下几种基本操作：

（1）字符串赋值：直接赋值s="20180105"。

（2）字符串连接s1.append(s2)：把字符串s2接在s1的后面，返回连接后的新串。

（3）求字符串长度s.length( )：返回字符串s中当前所含字符个数。

（4）求子串操作s1.substr(pos1,len1)：从字符串s1中复制指定位置pos1开始、指定长度len1的子串。

（5）插入操作s1.insert(pos,s2)：将一个子串s2插入到s1的指定位置pos，返回这个新的主串。

（6）删除操作s.erase(pos,len)：删除位置pos开始的长度为len的一个子串。

（7）查找子串操作s1.find(s2)：找出主串s1中是否包含子串s2，包含则返回该子串位置，不包含则返回空值。

（8）判断两个字符串是否相等s1.compare(s2)：比较s1、s2两个字符串是否相等，相等返回true，否则返回false。

字符串相等，是指两个字符串长度相等且对应位置的元素一一相等。例如，字符串s1="693450213"，s2="693450213"，s3="693550213"，其中串s2与s1相等，s3与s1不等。

顾客若要求查询编号为“693450213”的商品销量，则需将此字符串与销售记录中的商品编号逐一比较，找到该字符串第一次出现的记录。

项目实施

各小组根据项目选题及拟订的项目方案，结合3.1节和3.2节所学知识，对所选定的模拟实验进行系统分析和数据建模。

1. 分析实验过程中，哪些活动或问题可以利用字符串这一数据结构来解决。
2. 尝试编程实现。

3.3 用队列组织先进先出数据

我们在生活中到处都能看见排队的现象，排队购物、排队就诊、排队取号等。在这些排队现象中，事物的进出顺序都有共同的特征，那就是先进先出。当我们要用计算机程序来解决生活中排队的问题，实现排队事物的先进先出时，可以借助“队列”这种数据结构。

3.3.1 队列

队列（Queue）是一种特殊的线性表，它只允许在表的一端进行插入，在表的另一端进行删除。在队列中，可以插入的一端称为队尾，可以删除的一端称为队头。把一个数据元素插入队列中的操作叫作进队，从队列中删除一个数据元素的操作叫作出队。队列中没有元素时，称为空队列。

在日常生活中，售票窗外或服务台前，顾客按到达的先后次序排成一队。排在队头的首先得到服务，然后离队。所有顾客一律平等，严格遵守秩序，不允许插队现象。也就是说，队列中总是排在最前面的对象首先离队。

因此，队列符合这个规律：先放入队列中的数据元素首先取出。故队列又被称为先进先出（FIFO：First In First Out）线性表。

探究活动

体验

运行配套学习资源包的程序“第三章\课本素材\多业务分类服务队列.exe”，如图3-8所示。按照取号、排号、叫号的流程，尝试自主操作“多业务分类服务队列”程序的“取号”“叫号”，观察“服务队列”和“出队元素”的变化情况，体验队列先进先出的特点。

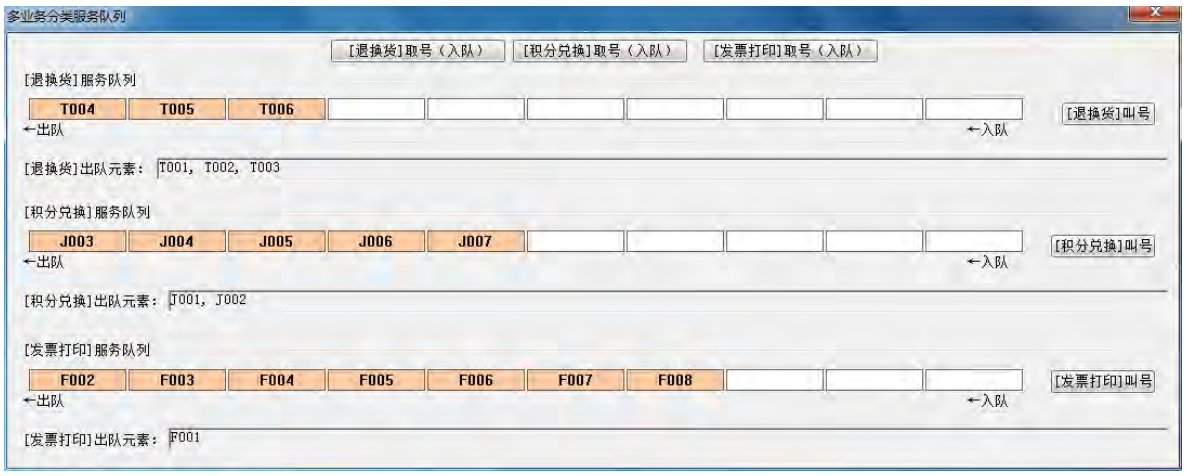


图3-8　模拟排队程序界面

观察

运行配套学习资源包的程序“第三章\课本素材\排队自助服务系统.exe”，观察其运行情况，思考该系统是如何设计的。运行情况如图3-9所示。

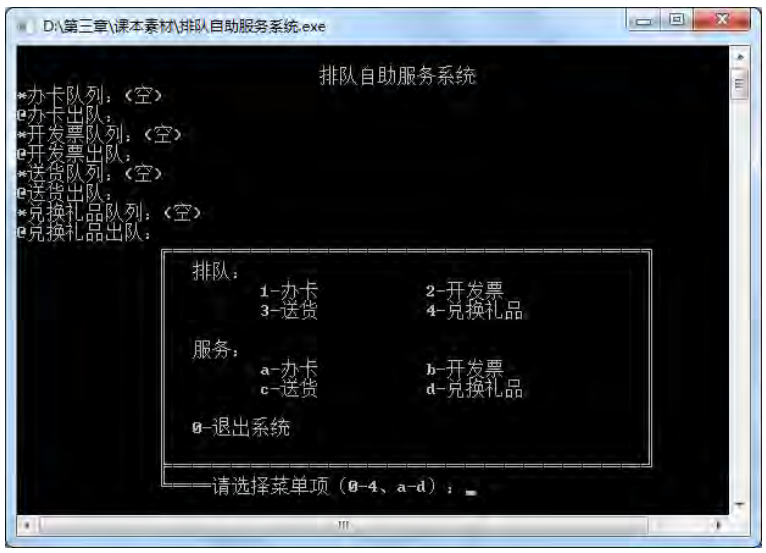


图3-9　排队自助服务系统运行情况

（1）分析数据关系。

超市为了使顾客的售后服务请求更有序，要求必须排队，排在前面的顾客先服务，排在后面的顾客后服务。如图3-10所示，顾客以a_1，a_2，…，a_n的顺序进入队列，退出排队也同样按照这个次序，即只有在顾客a_1，a_2，…，a_{n-1}都离队后，顾客a_n才能退出排队。若要设计自助服务系统来实现对排队的管理，采用队列这种数据结构最为合适。

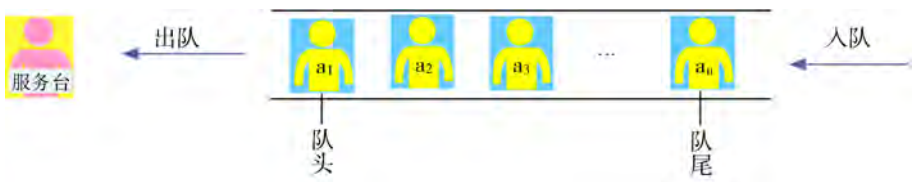


图3-10 超市售后服务请求队列

（2）为服务请求队列建立数据模型。

```
队列
{
    队列元素(一定数量的顾客编号);
    队头(即将出队的顾客的位置);
    队尾(即将入队的顾客的位置);
}
队列的基本操作;
```

（3）设计自助服务系统的操作。

自助服务系统与顾客、服务台之间互相配合，共同完成服务请求和应答，应该提供客户取号、服务系统排号以及服务台叫号的基本功能。其中：

取号→生成一个排队编号并将该编号插入到队列的队尾中；

排号→配合取号、叫号，相应改变队列的队头、队尾；

叫号→返回队列中处在队头的编号。

根据队列的概念，以C++语言为例，在程序中我们可以这样定义队列：

```
typedef struct //数据描述
{
    datatype items[maxsize];
    // datatype为队列元素的类型，maxsize为队列的最大长度
    int front; //队头标志
    int rear; // 队尾标志
}Queue;
operations; // 操作声明
```

在定义队列时，队列元素的类型datatype可以根据现实数据的情况而确定，如对于超市服务请求队列的元素类型datatype可以是代表编号的字符串；另外，采用数组作为队列元素的存储结构，数组容量的大小maxsize也应根据现实情况进行估计，太大则浪费空间，太小则可能出现元素入队列时队列容量不足，发生溢出。

运行配套学习资源包中的程序“第三章\课本素材\排队自助服务系统.exe”，分析都有哪些售后服务请求，为各种请求队列编程定义相应的数据结构，如表3-3所示。

表3-3　定义各种队列的数据结构

功能	程序段

定义队列数据结构。	typedef struct
	{
	　string items[10];
	　//队列元素为string，队列最大长度为10
	　int front; //队头标志
	　int rear; //队尾标志
	}Queue;

定义各种服务队列的变量。	Queue card; //办卡服务队列
	Queue invoice; //开发票服务队列
	Queue delivery; //送货服务队列
	Queue gifts; //兑换礼品服务队列

各小组就以下问题展开讨论：

（1）小组选择的项目中有哪些是需要排队的事物或现象？

（2）项目要解决哪种或哪些排队问题（如提高有序性、提高效率、智能化等）？

在问题提出的基础上合作完成以下任务：

（1）列出项目中具备队列特征的事物或现象，并用画图的方式描述出这种事物或现象的工作过程（画出队列，标识队头及队尾）。

（2）为以上事物或现象建立队列模型。

（3）根据建立的队列模型，为项目中的所有队列编写程序，定义相应的数据结构。

3.3.2　队列的基本操作

对于队列，通常有以下几种基本操作：

（1）初始化队列：构造一个空队列，初始化队头、队尾标志。

（2）元素入队：若队列非满，插入一个元素到队列的队尾标志指向的位置，该元素成为新的队尾元素，队尾标志向后移动一位。

（3）元素出队：删除队头标志指向的队头元素，队头标志向后移动一位，若此时队列非空，则队头标志指向的元素成为新的队头元素。

（4）求队列长度：返回队列当前所含元素个数。

（5）队空判断：若队列为空，则返回“真”，否则返回“假”。

（6）队满判断：若队列为满，则返回“真”，否则返回“假”。

假设已定义队列q、队头front、队尾rear，则初始化队列为空、入队和出队几种情况如图3-11所示。

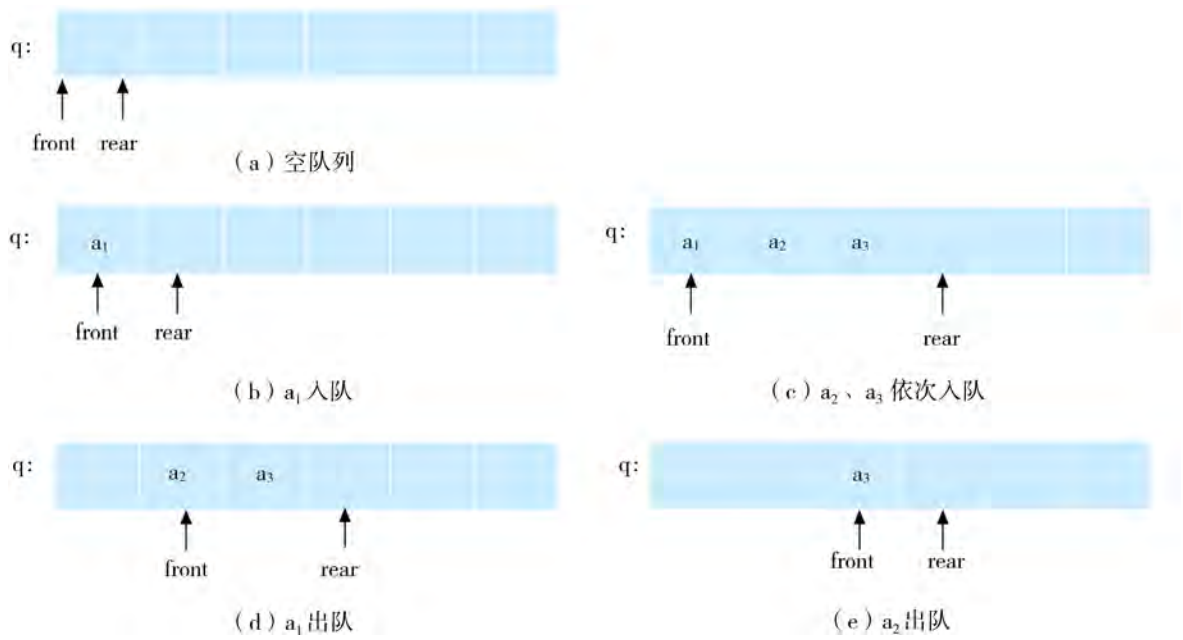


图3-11　空队列、入队和出队示意图

（q为队列，front为队头标志，rear为队尾标志）

小组内分析如何实现超市排队自助服务系统的各种功能，需要完成哪些工作。

（1）超市排队自助服务系统具有顾客自助取号、系统自动排号、服务台自动叫号等服务请求与应答功能，设计时首先为每一项服务类型的排队定义队列，然后通过对队列的操作来实现自助服务系统的各种功能。

（2）对应服务台和顾客的不同行为，自助服务系统提供相应的不同功能，并通过不同的队列操作来实现这些功能，列表分析如表3-4所示。

表3-4　超市排队自助服务请求与应答分析

服务台/顾客行为分析	自助服务系统功能分析	队列操作
营业开始，服务台做准备。	初始化所有服务队列为空。	初始化队列。
顾客选择某种服务并取号。	判断某服务队列是否非满，是则生成一个号码并插入队列的队尾，将队列的队尾向后移动一位。同时显示前面还有多少人在等待。	队满判断，元素入队，求队列长度。
服务台对某种服务叫号。	判断某服务队列是否非空，是则返回队列中处在队头的号码，并将队列的队头向后移动一位。	队空判断，元素出队。

3.3.3　队列的实现

1. 顺序队列

在队列的程序定义中，可以利用数组这种具有一定容量的顺序存储结构来存储队列元素，并用队头标志front指示即将出队的元素的位置，用队尾标志rear指示即将入队的元素的位置，它们的初始值在队列初始化时均为0。同时我们约定：当入队或出队时，队尾标志rear或队头标志front只能往后移动一位，且其最大值为队列的最大长度maxsize（数组的容量），我们把这种队列称为顺序队列。假设已定义队列q，队头front，队尾rear，数组容量为6，其顺序队列存储方式如图3-12所示。

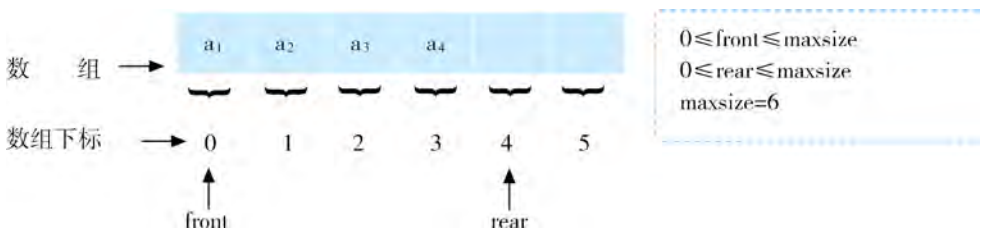


图3-12　顺序队列

基于以上约定，在编程实现队列操作时，队头标志和队尾标志会有几种不同的变化，还可以利用队头标志和队尾标志来求得队列的当前长度：

（1）初始化队列：front = rear = 0

（2）入队操作：rear = rear+1

（3）出队操作：front = front+1

（4）队空判断：front == rear

（5）队满判断：rear == maxsize

（6）求队列长度：rear – front

顺序队列的操作可用如表3-5所示的程序代码实现。

表3-5 顺序队列的操作代码

功能	程序段

初始化队列：
队头标志和队尾标志均为0，这时队列为空，没有元素。
	`void Init_Queue(Queue &q)`
	`{`
	`  q.front=q.rear=0;`
	`}`

判断队列是否为空：
当front=rear时，队列为空，返回true，否则返回false。
	`bool IsEmpty_Queue(Queue &q)`
	`{`
	`  if(q.front==q.rear) return true;`
	`  else return false;`
	`}`

判断队列是否为满：
当rear=maxsize时，队列为满，返回true，否则返回false。
	`bool IsFull_Queue(Queue &q)`
	`{`
	`  if(q.rear==maxsize) return true;`
	`  else return false;`
	`}`

入队操作：
如果队列非满，则将元素x放入队尾标志rear所指位置，然后rear指向下一个位置。
	`void In_Queue(Queue &q,string x)`
	`{`
	`  if(!IsFull_Queue(q))`
	`  {`
	`  q.items[q.rear]=x;`
	`  q.rear=q.rear+1;`
	`  }`
	`}`

出队操作：
如果队列为空，返回空元素（NULL），否则返回队头标志front所指的元素，然后front指向下一个位置。
	`string Out_Queue(Queue &q)`
	`{`
	`  if(IsEmpty_Queue(q))`
	`  {`
	`  return NULL;`
	`  }`
	`  else`
	`  {`
	`  string retvalue;`
	`  retvalue=q.items[q.front];`
	`  q.front=q.front+1;`
	`  return retvalue;`
	`  }`
	`}`

取当前队列长度。	`int Size_Queue(Queue &q)`
	`{`
	`  return q.rear-q.front;`
	`}`

讨论

小组内展开讨论：随着顺序队列不断有元素出队和入队，当队尾标志达到队列的最大长度时，整个队列中还有剩余空间吗？还能继续入队吗？评价顺序队列在对存储空间的利用率上存在什么问题，想想是否有改进的办法。

2. 循环队列

当我们对顺序队列不断执行入队操作时，队列就有可能出现溢出现象。如已定义顺序队列q，队头front，队尾rear，数组容量为6，当有元素入队时，有下面两种情况：

（1）真溢出。

当队列中的实际元素个数达到队列最大长度maxsize时，队列满，这时做入队操作将产生空间溢出的现象，如图3-13所示。真溢出是一种出错状态，应设法避免。

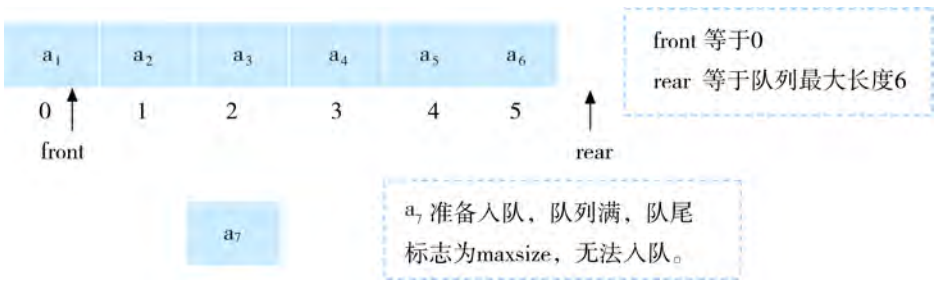


图3-13　真溢出

（2）假溢出。

由于同时有入队和出队两种操作，队头标志front、队尾标志rear的值只增加不减少，致使已出队元素的空间永远无法重新利用。当队列中的实际元素个数小于队列最大长度（即数组容量）maxsize时，也可能由于队尾标志已达到maxsize而不能做入队操作，这时就出现了假溢出。如图3-14所示。

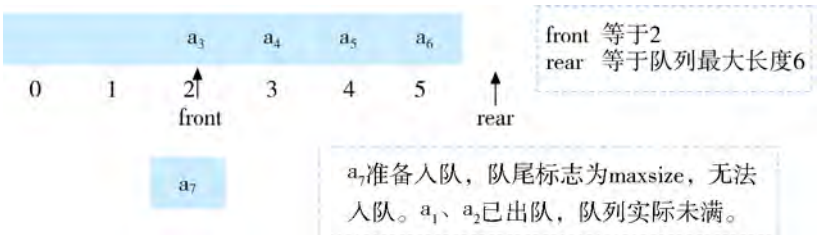


图3-14　假溢出

为充分利用队列的存储空间，克服“假溢出”现象，可以将数组存储区想象为一个首尾相接的圆环，并称存储在其中的队列为循环队列。同时，为了区别队满和队空，我们约定：当数组中只剩下一个元素为空时，即若队尾标志继续后移一位将与队头标志重叠，就判为队满，此时不能再做入队操作。

如使用循环队列存储前面的队列q，则空队列、元素入队和元素出队的几种情况如图3–15所示。

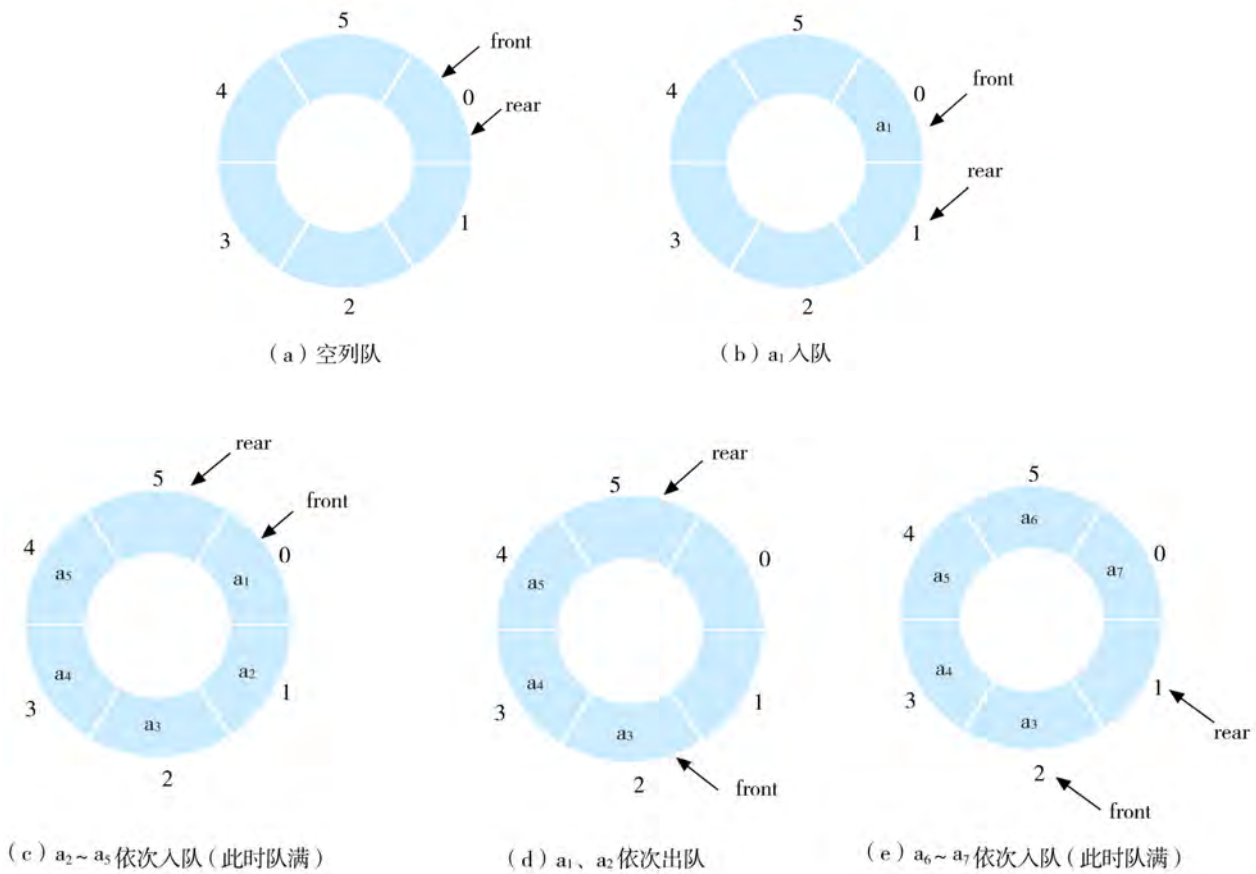


图3–15　循环队列的空队列、入队和出队示意图

不难发现，循环队列中的元素被删除后，其原来的空间仍然可以使用，因而循环队列能实现对空间更大限度的利用。

队列是在程序设计中被广泛使用的线性数据结构，除了用于解决排队问题，还可以应用于其他符合按“先进先出”规则进行操作的问题。

小组内交流并完成对比分析：顺序队列与循环队列采用的存储方式有何异同？将交流结果列表呈现，并说说它们分别适用于哪种场景。

基于循环队列的特征，在编程实现队列的几种操作时，要配合取模（余数）运算，同时队头标志和队尾标志会有以下几种变化：

（1）初始化队列：front = rear = 0

（2）入队操作：rear = (rear+1) % maxsize

（3）出队操作：front = (front+1) % maxsize

（4）队空条件：front == rear

（5）队满判断：(rear+1) % maxsize == front

与顺序队列不同，循环队列的队满判断、元素入队、元素出队及求队列长度的操作可描述为：

（1）判断队列是否为满：当rear的下一个位置等于front时，队列为满，返回true，否则返回false。

（2）入队操作：如果队列非满，则将元素x放入队尾标志rear所指位置，然后rear指向下一个位置。

（3）出队操作：如果队列为空，返回空元素（NULL），否则返回队头标志front所指的元素，然后front指向下一个位置。

（4）取当前队列长度：如果队尾标志rear大于或等于队头标志front，返回rear与front之差，否则返回循环队列最大长度与（front–rear）之差。

程序代码可参考配套学习资源包中的文档“第三章\课本素材\循环队列的操作代码.docx”。

根据表3–4“超市排队自助服务请求与应答分析”，思考如何采用顺序队列或循环队列完成其队列操作，对比优劣并编写程序实现超市排队自助服务功能。可参考配套学习资源包“第三章\课本素材\排队自助服务系统.cpp”。

项目实施

各小组根据项目选题及拟订的项目方案，结合本节所学知识，分工开展探究活动，根据前面所分析的在项目活动中待解决的排队问题，完成以下任务：

1. 基于已定义的队列和已分析的需要实施的队列操作，编写完整的程序，解决排队问题。同时对程序进行运行测试，检验结果的正确性。

2. 形成本项目活动的成果，形式可以是研究报告、问题解决方案说明书等，可附上相应程序。

3.4 用栈组织后进先出数据

队列对应了生活中的排队现象，但还有另一种现象，如对一叠碗的取放：每次把洗净的碗放好时总是放在这叠碗的最上面，而每次取用的时候也总是取最上面的。在这种现象中，事物的进出顺序都有共同的特征，那就是后进先出。

3.4.1 栈

栈（Stack）是限制只能在一端进行插入和删除的特殊线性表。栈中能进行插入和删除的一端称为栈顶（Top），而另一固定端称为栈底（Bottom）。把一个数据元素放入栈中的操作叫作进栈或压栈（Push），从栈中取出一个数据元素的操作称为出栈或弹出（Pop）。栈中没有元素时，称为空栈。

栈的形式在日常生活中经常出现，例如一叠书、一叠盘子，如果规定取书、取盘子或放入书、放入盘子都只能在顶部进行，则它就是一个栈。

生活中有很多类似的例子，如购物车的取放（如图3-16所示）、仓库、码头物品的堆放等。在计算机程序中，我们可以用“栈”这种数据结构来组织和实现这种工作方式。

栈符合这个规律：后放入栈中的数据元素首先取出。故栈又被称为后进先出（LIFO：Last In First Out）线性表。

图3-16　超市手推购物车停放

探究活动

体验

运行配套学习资源包中的程序“第三章\课本素材\模拟栈操作.exe”，如图3-17所示。尝试自主操作“模拟栈操作”程序的“入栈”“出栈”，观察“栈”和“出栈序列”的变化情况，体验栈后进先出的特点。

图3-17　模拟栈操作

观察

运行配套学习资源包中的程序“第三章\课本素材\购物车管理程序.exe”，观察运行情况。购物车模拟管理程序运行情况如图3-18所示。

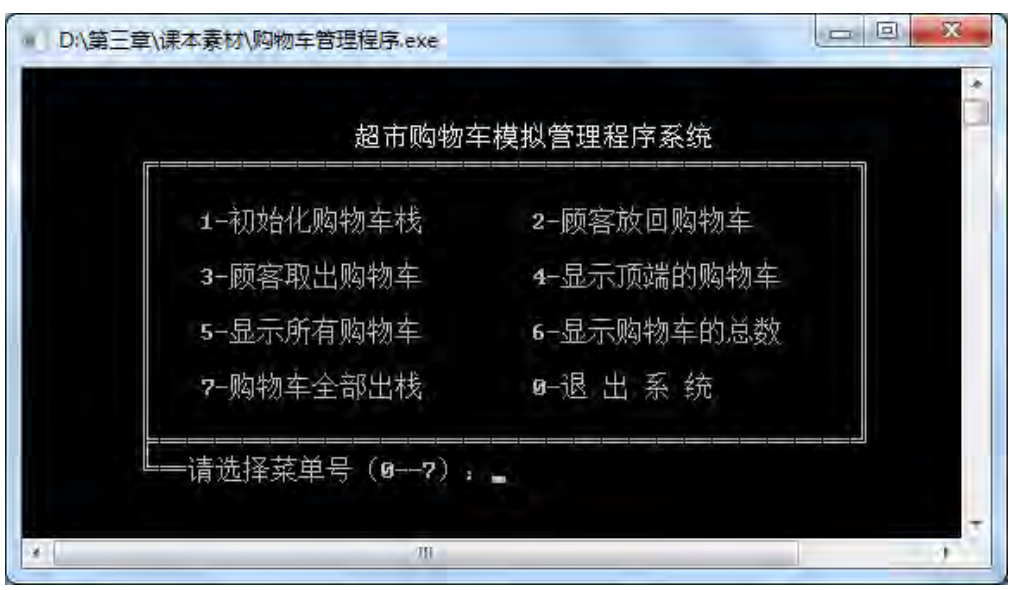


图3-18　购物车模拟管理程序运行情况

（1）分析数据关系。

超市购物车的取放其实就是购物车不断取出和推回的过程，符合后进先出的工作方式，所以购物车模拟管理程序可采用“栈”这种数据结构来组织管理。把所有购物车编号，顾客取出购物车前必须先在管理程序中登记（即购物车的出栈操作）才能取出，顾客把购物车放回停放点时也要在管理程序中登记（即购物车的进栈操作）。

如图3-19所示，购物车以Car_1，Car_2，…，Car_N的顺序进入栈中，而从栈中取出购物车的次序是Car_N，Car_{N-1}，…，Car_1。

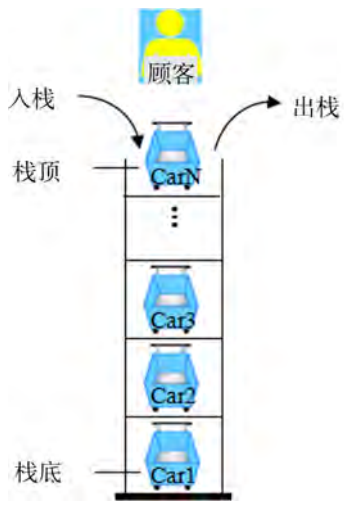


图3-19　购物车进出栈

（2）建立数据模型。

```
栈
{
    栈元素(一定数量的购物车编号);
    栈顶(即将出栈的购物车的位置);
    栈底(即堆在最底的购物车的位置);
}
栈的基本操作;
```

（3）设计自助服务系统的操作。

顾客取出购物车前必须先在管理程序中登记才能取出，顾客把购物车放回停放点时同样要到管理程序中登记，购物车的使用过程才全部结束。程序还要提供显示停放点最前面购物车和所有购物车的编号、购物车总数等数据的功能。

显示停放点最前面购物车和所有购物车的编号、购物车总数等数据的功能可用显示栈顶和栈内元素、栈的长度等基本操作实现。

1. 根据所选择的项目，列出其中具备栈特征的事物或现象，并用画图的方式描述出这种事物或现象的工作过程（画出栈，标识栈顶、栈底）。

2. 尝试为以上事物或现象建立栈模型。

根据栈的概念，在程序中我们可以这样定义栈：

```
typedef struct     //数据描述
{
  datatype items[maxsize];
  //datatype为栈元素的类型，maxsize为栈的最大高度
  int top;         //栈顶标志
  int bottom;      //栈底标志
}Stack;
operations;        //操作声明
```

与队列一样，在定义栈时，栈元素的类型datatype可以根据现实数据的情况而确定；另外，若采用数组来作为栈元素的存储结构，数组容量的大小maxsize也需根据现实情况进行估计。

3.4.2　栈的基本操作

栈的常用基本操作有以下几种：

（1）初始化栈：构造一个空栈，初始化栈顶标志。

（2）元素入栈：若栈非满，栈顶标志上移一位，插入一个元素到栈顶标志指向的位置，该元素成为新的栈顶元素。

（3）元素出栈：删除栈顶标志指向的栈顶元素，栈顶标志下移一位，若此时栈非空，则栈顶标志指向的元素成为新的栈顶元素。

（4）栈空判断：判断栈是否为空。

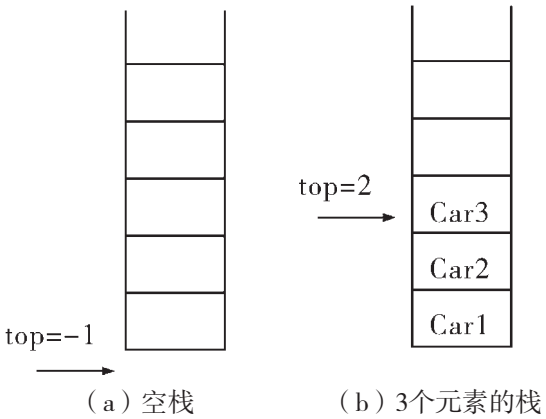
（5）栈满判断：判断栈是否为满。

（6）栈的长度：求栈的元素个数。

3.4.3　顺序栈的实现

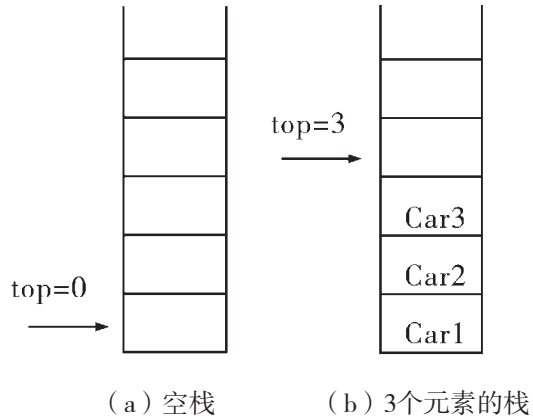
栈的存储也可采用顺序存储结构的方法来实现，采用顺序存储结构的栈称为顺序栈，是指利用一组连续的存储单元依次存放自栈底到栈顶的数据元素，同时设置指针top来动态地指示栈顶元素的当前位置。

可以用一维数组来实现用于存储顺序栈中数据元素的存储区域，首先定义一个足够大（Maxsize）的一维数组，将数组下标为0的一端设为栈底，top指针指向栈顶元素的具体位置有两种方法：一种是让top指针始终指向栈顶元素所在的位置，当top= -1时表示空栈（如图3-20所示）；另一种是让top指针指向栈顶元素的下一个位置，当top= 0时表示空栈（如图3-21所示）。



（a）空栈　　（b）3个元素的栈

图3-20　Top指针指向栈顶元素



（a）空栈　　（b）3个元素的栈

图3-21　Top指针指向栈顶元素的下一个位置

由于栈是一个动态结构，而数组是一个静态结构，所以在顺序栈的操作过程中会有“溢出”情况的出现。当栈满时，如果再有元素入栈，将会产生“上溢（Overflow）”；当栈空时，如果再执行出栈操作，将会产生“下溢（Underflow）”。为了避免发生栈溢出情况，在进行入栈和出栈操作前应先检测栈是否已满或已空。

顺序栈的程序定义如下：

```
typedef struct
{
  string items[maxsize];
  int top;
}CarStack;
```

其中，maxsize为栈中允许存放元素个数的最大值，top是栈顶标志，指示存放数组的下标，不是真正的指针，当top= -1时表示空栈，当top= maxsize -1时表示满栈。

顺序栈的操作可用如表3-6所示的程序代码实现。

表3-6　顺序栈的操作代码

功能	程序段

初始化栈。	void InitStack(CarStack &st)
	{
	  st.top=-1; //把栈顶指针置为-1
	}

（续表）

功能	程序段

入栈操作：
将元素x进栈，元素进栈成功返回true，否则返回false。
	bool PushStack(CarStack &st,string x)
	{
	  if(st.top==maxsize-1)
	  return false; //栈满不能插入元素，返回false
	  else
	  {
	  st.top++;
	  st.items[st.top]=x;
	  return true; //成功将元素入栈，返回true
	  }
	}

出栈操作：
栈st的栈顶元素出栈，出栈元素存放在x中,出栈成功返回true，否则返回false。
	bool PopStack(CarStack &st,string &x)
	{
	  if(st.top==-1)
	  return false; //栈空不能出栈，返回false
	  else
	  {
	  x=st.items[st.top];
	  st.top--;
	  return true; //成功将元素出栈，返回true
	  }
	}

栈空判断：
若栈st为空栈，则返回true，否则返回false。
	bool EmptyStack(CarStack &st)
	{
	  if(st.top==-1)
	  return true;
	  else
	  return false;
	}

栈满判断：
若栈st为满栈，则返回true，否则返回false。
	bool FullStack(CarStack &st)
	{
	  if(st.top==maxsize-1)
	  return true;
	  else
	  return false;
	}

栈的长度：
返回栈中当前元素的个数。
	int StackLength(CarStack &st)
	{
	  return st.top+1;
	}

体验

参照上述实现栈操作的程序段，查看配套学习资源包“第三章\课本素材\购物车模拟管理程序.cpp”，为所选项目中涉及栈的操作设计程序，补充完善表3-7。

表3-7　顺序栈的操作代码

功能	程序段
初始化： 栈置空	
服务请求： 入栈	
服务应答： 出栈	

项目实施

各小组根据项目选题及拟订的项目方案，结合本节所学知识，分工开展探究活动，打开在前面探究活动中完成的程序文件，修改完善之前编写的有关栈操作的程序代码，运行测试，检验结果的正确性。参照项目范例的样式，撰写相应的项目成果报告。

成果交流

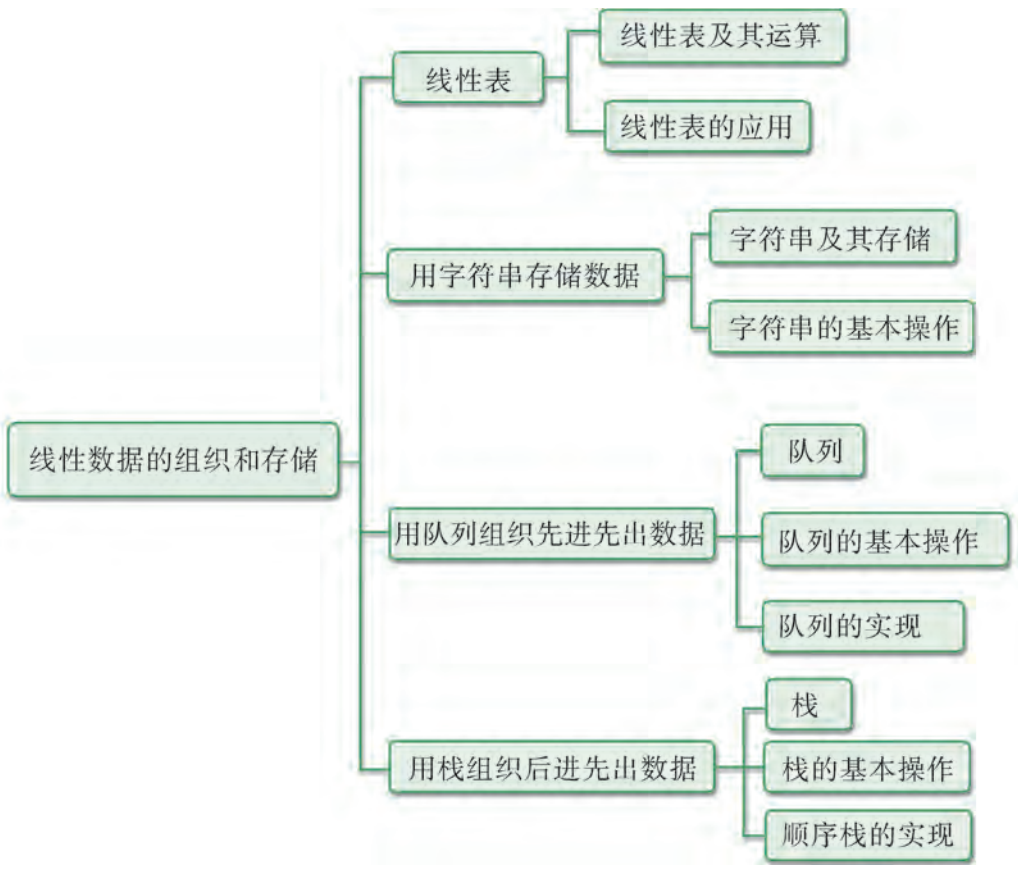
各小组运用数字化学习工具，将所完成的项目成果，在小组或班级上进行展示与交流，共享创造、分享快乐。

活动评价

各小组根据项目选题、拟订的项目方案、实施情况以及所形成的项目成果，利用教科书附录2的“项目活动评价表”，开展项目学习活动评价。

本章扼要回顾

同学们通过本章学习，根据“线性数据的组织和存储”知识结构图，扼要回顾、总结、归纳学过的内容，建立自己的知识结构体系。



回顾与总结

本章学业评价

同学们完成下列测试题（更多的测试题可以在教科书的配套学习资源包中查看），并通过“本章扼要回顾”以及本章的项目活动评价，综合评价自己在信息技术知识与技能、解决实际问题的过程与方法，以及相关情感态度与价值观的形成等方面，是否达到了本章的学习目标。

1. 单选题

（1）在一个长度为n的线性表中，在第i个元素（1≤i≤n+1）之前插入一个新的数据元素，需要向后移动元素的个数是（　　）个。

A. n−i　　B. n−i+1　　C. n−i−1　　D. i

（2）一个栈经过以下栈运算：InitStack(s)，Push(s,a)，Push(s,b)，Pop(s)，GetTop(s,x)，最后x的值是（　　）。

A. a　　B. b　　C. 1　　D. 0

（3）设某个栈的输入序列为A、B、C、D，则借助这个栈所得到的输出序列不可能是（　　）。

A. A、B、C、D　　B. D、C、B、A

C. A、C、D、B　　D. D、A、B、C

2. 思考题

在各种文本编辑程序中，如果把用户输入的内容看成一个字符串，那么尽管各种文本编辑程序的功能有所不同，但其基本操作是相同的：一般包括串的输入、修改（插入、删除）、查找、输出等。请思考：如何运用字符串操作实现上述文本编辑的功能？

3. 情境题

舞伴配对：在周末的舞会上，男士们和女士们进入舞厅后，分别编号坐在舞池两边的椅子上。跳舞开始时，依次从男士和女士中各出一人配成舞伴。若两队初始人数不相同，则较长的那一队中未配对者等待下一轮舞曲。编写一个程序模拟上述舞伴配对过程。

第四章
抽象数据类型

生活中的问题想要用计算机编程求解，不是用简单的数据类型就能全部实现，还需要用抽象数据类型来描述问题的数据模型和基本操作。除了如线性表这种线性关系，还有很多数据之间的关系是层次或网状的，即以非线性形式存在的。

本章将通过“抽象数据类型案例分析”项目，进行自主、协作、探究学习，让同学们认识抽象数据类型对数据处理的重要性，理解抽象数据类型的概念，了解二叉树的概念及其基本操作方法，从而将知识建构、技能培养与思维发展融入运用数字化工具解决问题和完成任务的过程中，促进信息技术学科核心素养达成，完成项目学习目标。

- 认识抽象数据类型
- 用抽象数据类型表示队列和栈
- 用抽象数据类型表示二叉树

项目范例 俄罗斯方块游戏的抽象数据类型案例分析

情境

超市的商品包罗万象，能满足不同年龄段人群的需求，让他们都能在超市找到适合自己的商品。由于对所在区域的消费人群及消费兴趣估算不当，超市管理者发现超市的“俄罗斯方块”游戏机进货量过多，出现积压滞销的情况，而长期积压滞销商品会使超市亏本。超市管理者经过研究，组织了“俄罗斯方块”电子竞技活动，参与者都可以打折购买该款游戏机。很多人都会玩“俄罗斯方块”游戏，但知道该游戏程序实现基本原理的却很少，下面就让我们一起来探究一下游戏的编写原理吧！

主题

俄罗斯方块游戏的抽象数据类型案例分析

规划

根据项目范例的主题，在小组中组织讨论，利用思维导图工具，制订项目学习规划，如图4–1所示。

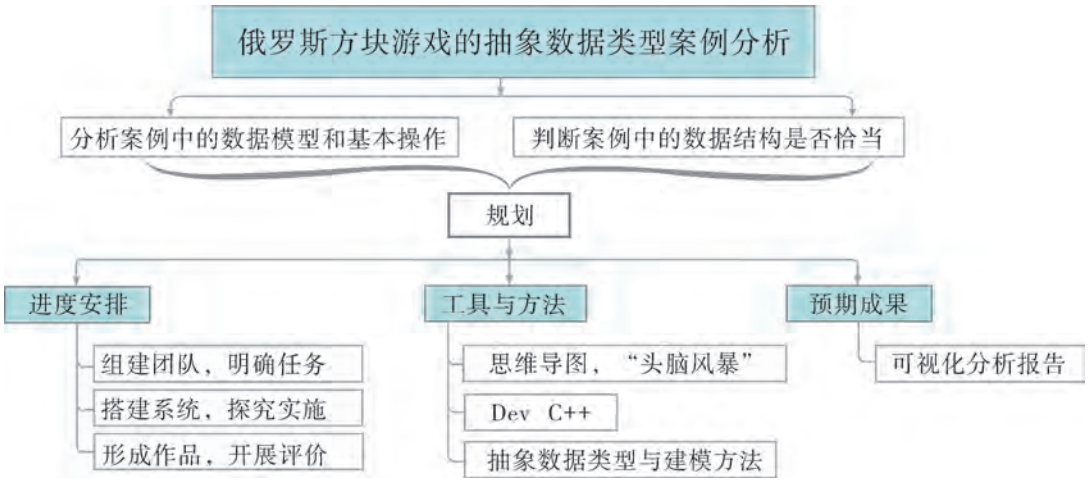


图4–1 “俄罗斯方块游戏的抽象数据类型案例分析”项目学习规划

探究

根据项目学习规划的安排，通过调查、案例分析、文献阅读和网上搜索资料，开展“俄罗斯方块游戏的抽象数据类型案例分析”项目学习探究活动，如表4–1所示。

表4-1　“俄罗斯方块游戏的抽象数据类型案例分析”项目学习探究活动

探究活动	学习内容		知识技能
分析案例中的数据模型和基本操作	列举抽象数据类型应用实例。	列举案例。	认识抽象数据类型对数据处理的重要性。 理解抽象数据类型的概念。
	分析数据模型及其基本操作。	分析游戏方块的数据模型及各种操作。	
判断案例中的数据结构是否恰当	了解队列、栈和二叉树的抽象数据类型表示方法。	分析判断游戏方块的基本操作有哪些，是否需要用到队列、栈和二叉树等数据结构。	理解队列、栈和二叉树的抽象数据类型表示方法。 了解二叉树的概念及基本操作方法。

实施项目学习各项探究活动，进一步理解抽象数据类型对数据处理的重要性。

在小组开展项目范例学习过程中，利用思维导图工具梳理小组成员在“头脑风暴”活动中的观点，建立观点结构图，运用多媒体创作工具（如演示文稿、在线编辑工具等），综合加工和表达，形成项目范例可视化学习成果，并通过各种分享平台发布，共享创造、分享快乐。例如，运用在线编辑工具制作的“俄罗斯方块游戏的抽象数据类型案例分析”可视化报告，可以在教科书的配套学习资料包中查看，其目录截图如图4-2所示。

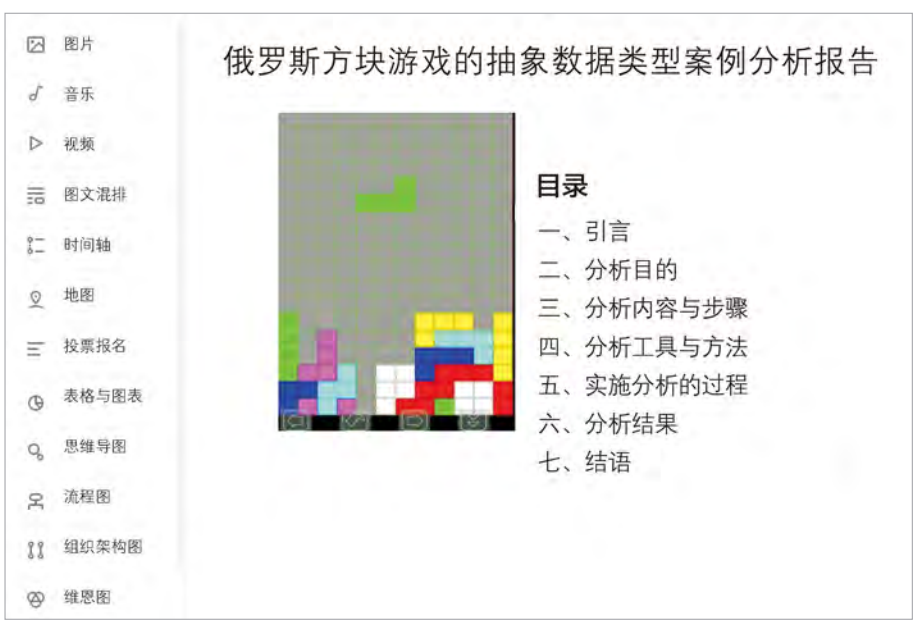


图4-2　“俄罗斯方块游戏的抽象数据类型案例分析”可视化报告的目录截图

根据教科书附录2的“项目活动评价表”，对项目范例的学习过程和学习成果在小组或班级上进行交流，开展项目学习活动评价。

同学们以3~6人组成一个小组，选择下面一个参考主题，或者自拟一个感兴趣的主题，开展项目学习。

1. 图书馆中的抽象数据类型案例分析
2. 校运会中的抽象数据类型案例分析
3. 社团活动中的抽象数据类型案例分析

各小组根据项目选题，参照项目范例的样式，利用思维导图工具，制订相应的项目方案。

各小组将完成的方案在全班进行展示交流，师生共同探讨、完善相应的项目方案。

4.1　认识抽象数据类型

高级语言中常用的数据类型有整型、实型、字符型和字符串类型等，这些数据类型分别存储不同种类的数据，同时我们能对这些数据进行特定的操作。但随着计算机解决的问题、处理的数据越来越复杂，需要定义更多复杂的数据模型，同时还需要对这些数据模型进行特定的操作，这就产生了抽象数据类型。

4.1.1　抽象数据类型

在用计算机解决实际问题的过程中，对于要解决的问题，不管用什么语言编写程序，其解决的方法和思路是相同的。为了便于描述问题和抽象问题，在一般数据类型的基础上

提出了抽象数据类型，它可以有效地帮助我们思考、描述问题的数据模型和基本操作。

很多高级语言都有“整型”这种数据类型，其实整型也是一种抽象数据类型，因为不同的计算机对整型变量的存储是不一样的。而我们用到整型变量时，并不关心它是如何存储的，只需要了解其取值范围（值域）和能够进行的操作运算（加、减、乘、除、取模等），就可以正确地使用整型变量了。这正体现了抽象数据类型的本质：忽略不同机器、不同语言的实现细节，在更普遍、更高的层次抽象出问题的数据模型，并定义数据模型的相关操作，从而实现对问题的解决。

抽象数据类型（Abstract Data Type，简称ADT），是由一种数据结构和在其上的一组操作（运算）所组成。抽象数据类型包含一般数据类型的概念，但含义比一般数据类型更广、更抽象。一般数据类型通常由具体语言系统内部定义，直接提供给用户使用；而抽象数据类型还包括用户在编程处理数据、设计软件系统时自己定义的数据类型，通常由用户自行定义，包括定义其所包含的数据（数据结构）和在这些数据上所进行的操作。在定义抽象数据类型时，就是定义其数据的逻辑结构和操作说明，不考虑数据的存储结构和操作的具体实现，这样具有较好的通用性和可移植性，便于用任何一种语言实现。

本书中抽象数据类型采用以下格式定义：

```
ADT 抽象数据类型名
{
  数据：<数据描述>
  操作：<基本操作的定义>
}ADT 抽象数据类型名
```

为了便于理解和表达的简洁，对抽象数据类型定义中的数据（数据结构）及基本操作，我们采用中文说明和类C++语言的形式进行描述（借用C++语言的一些语法元素，但是不严格遵循C++语言）。例如，我们可以定义“复数”这种抽象数据类型来表示数学上的复数：

```
ADT 复数
{
  数据：
    实部;
    虚部;
  操作：
    初始化复数;
    获取实部;
    获取虚部;
    获取模;
    获取辐角;
    复数加法;
```

```
        复数减法;
        复数乘法;
        复数除法;
    }ADT 复数
```

这里关于复数的数据和基本操作，是基于数学中复数的概念和运算进行抽象的，其抽象的前提是可以通过计算机实现。显然，复数的抽象数据类型定义用C++语言实现起来并不困难。

4.1.2　抽象数据类型的应用

利用抽象数据类型定义复数，并通过编程实现，就可以使用计算机处理复数了。除了数学运算外，在解决现实问题和实际编写计算机应用软件时，因为要解决的问题更复杂，所以更需要大量应用抽象数据类型来描述问题和设计解决方案。

如项目范例中的俄罗斯方块游戏，方块的颜色、形状都有多种，每一个方块都有左旋、右旋、左移、右移、下落等多种操作，变化较多。游戏程序实现的关键是如何操控这些方块，可以定义一种抽象数据类型“方块”，其中包含方块的形状、颜色、中心点等，以及对它的左旋、右旋、左移、右移、下落等多种操作。类似的，我们还可以将俄罗斯方块游戏的区域也定义为一种抽象数据类型，从而完成游戏的设计。

下面我们再举两个例子。

（1）象棋游戏里（如图4–3所示），每一个棋子上面的文字都不一样，有的是“马”、有的是“兵”等，对阵双方的棋子颜色一般为黑色和红色，每个棋子必须响应鼠标或键盘的控制动作，实现棋子的拿起、移动、放下等操作。这里可以将棋子定义为一种抽象数据类型，其数据模型包括棋子的颜色、棋子的文字、棋子在棋盘上的坐标等，基本操作包括拿起、移动、放下等。

图4–3　中国象棋

（2）电子地图的测距、导航等问题（如图4–4所示）。测距只需要计算地图上两点的距离即可，但导航问题比较复杂，需要从地图上寻找两个确定地点的最佳路径，这需要建立复杂的数据模型，如地点的经纬度、周边的路况等，并寻找合适算法。而要利用计算机处理该数据模型，就需要把其中涉及的数据模型合理存储并对这个数据模型进行操作。

图4–4　电子地图测距、导航

4.1.3　抽象数据类型的实现

为帮助大家更好地理解，接下来以定义抽象数据类型“长方形”为例，呈现抽象数据类型的定义过程和程序实现过程。

假定用rectangle来表示“长方形”的抽象数据类型名，其数据部分长、宽用a、b表示，类型为实数；其基本操作包括初始化、求长方形的周长和求长方形的面积，求周长的函数名用perimeter表示，求面积的函数名用area表示，则长方形的ADT描述如下：

```
ADT rectangle{
  数据：
    float a,b;      //长和宽
  操作：
    void init(float a1,float b1);    //长、宽初始化
    float perimeter(); //求周长函数
    float area();      //求面积函数
}ADT rectangle
```

我们用C++语言编写上述“长方形”的完整程序如下：

```
#include <iostream>
using namespace std;
struct rectangle
{
    float a,b;
    void init(float a1,float b1) //长、宽初始化
    {
      a=a1;
      b=b1;
```

```
    }
    float perimeter() //求周长操作
    {
      return (a+b)*2;
    }
    float area() //求面积操作
    {
      return a*b;
    }
}

int main()
{
    float a,b;
    float c,s;
    rectangle rect;          //创建长方形rect
    cout<<"请输入一个长方形的长和宽："<<endl;
    cin>>a>>b;               //读入长、宽的值
    rect.init(a,b);          //长方形rect长、宽初始化
    c=rect.perimeter();      //求周长
    s=rect.area();           //求面积
    cout<<endl;
    cout<<"长方形的周长为："<<c<<endl;
    cout<<"长方形的面积为："<<s<<endl;
    return 0;
}
```

探究活动

思考

现实中还有哪些问题可以定义为抽象数据类型？

实践

尝试定义一个抽象数据类型（例如三角形、梯形等），定义其数据及基本操作，并编写程序实现。

项目实施

各小组根据项目选题及拟订的项目方案，结合本节所学知识，开展以下活动。

1. 列举选题中的抽象数据类型案例，分析其数据模型及基本操作。

2. 初步选取本组研究的抽象数据类型案例。

4.2 用抽象数据类型表示队列和栈

队列和栈是两种典型的抽象数据类型，因为在计算机解决实际问题和很多软件的实现中，都会用到队列和栈。通过抽象数据类型来表示队列和栈，能够更加清楚地认识和理解这两种数据类型的特征和操作。

4.2.1 用抽象数据类型表示队列

队列是线性表的一种，队列的元素之间具有线性关系。另外，队列具有队头、队尾，元素从队尾入队、从队头出队，因此队列具有先进先出（FIFO）的特点。针对队列的操作有：初始化队列、元素入队、元素出队、求队列长度、队列判满、队列判空。我们可以用下面的抽象数据类型表示队列：

```
ADT 队列
{
  数据：
    队列元素；
    队头；
    队尾；
  操作：
    初始化队列；
    元素入队；
    元素出队；
    求队列长度；
    队列判满；
    队列判空；
}ADT 队列
```

因为队列属于线性表，队列元素的组织和存储可以使用数组，也可以使用链表，在第三章中我们已经分别用数组和链表实现了队列的存储和操作。虽然两种实现方法具体细节各有不同，但是都体现出了元素的线性关系，并且在相关操作中也遵循先进先出的原则，这就是队列的特点。

4.2.2 用抽象数据类型表示栈

栈和队列一样，属于线性表的一种，其元素之间也具有线性关系。与队列不同的是，栈元素的入栈、出栈都是从同一头进行的，所以栈具有后进先出（LIFO）的特点。从抽象数据类型的角度来看，栈的数据部分包括线性关系的数据元素和栈顶、栈底。关于栈的操作有：初始化栈、元素入栈、元素出栈、栈空判断、栈满判断、求栈的长度。类似的，我们可以用下面的抽象数据类型表示栈：

```
ADT 栈
{
  数据：
    栈元素；
    栈顶；
    栈底；
  操作：
    初始化栈；
    元素入栈；
    元素出栈；
    栈空判断；
    栈满判断；
    求栈的长度；
}ADT 栈
```

与队列类似，栈的数据元素可以使用数组存储，也可以使用链表存储，不同的存储方式，其实现细节会有一些区别。但是不管实现方式有何不同，其基本的数据元素关系是相同的，栈的基本操作也是相同的，具有后进先出的特点。有关栈的实现，可以参考第三章的相关内容。

探究活动

对比队列和栈的抽象数据类型表示，结合前面关于队列和栈的定义与实现，总结队列和栈的共同点和区别。

4.3 用抽象数据类型表示二叉树

4.3.1 树

阅读

超市管理团队达成了“出售的不仅是商品，还有服务”的理念后，超市经理构思了一系列贴心的服务。经过细心观察，他发现有些顾客来到超市，总是不能很快地找到自己所需要的商品所在的货架，需要询问导购员。怎样才能让顾客更方便地挑选商品?

经过思考，他认为商品货架的分类摆放很重要，于是他把超市的商品按照“衣食住行”分成了四大类，每大类下面又细分了小类，形成了如图4–5所示的树形结构图。

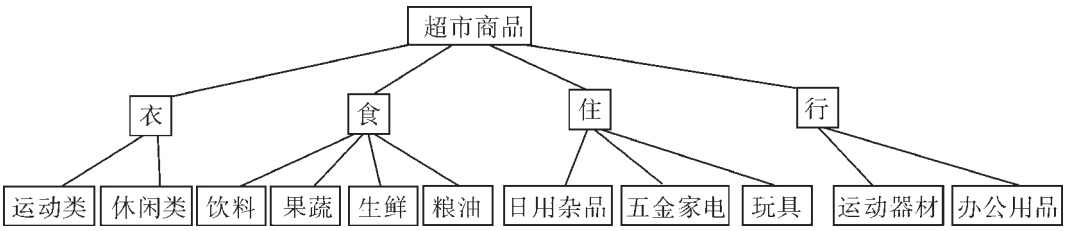


图4–5　超市商品分类

然后根据超市的环境，进行了货架的布局调整，并制作了相应的指示牌。经过调整后发现，顾客能更快更准地找到自己需要的商品了。

在我们日常生活中，树形结构广泛存在。例如，第一章图1–10中的家族成员关系树。又如，王贵有两个儿子：王永前、王永胜，王永前有三个孩子：王家栋、王家梁、王家辉，王永胜有一个孩子：王家莉，我们可以把他们的关系画成一棵树，如图4–6所示。

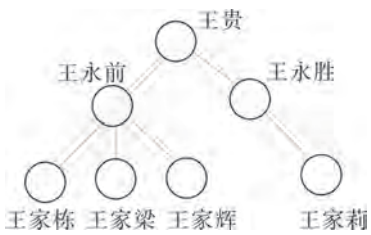


图4–6　家族成员关系树

在第一章图1–10和本章图4–6中，数据元素之间是一对多的关系，属于非线性关系。现实中，除了一对多的关系外，还有多对多的关系，例如第一章图1–11城市间的交通关系图，以及地图上不同地点之间的道路连接，都属于多对多的关系。具有一对多或多对多特点的数据结构称为非线性结构。

1. 树

树是n（n≥0）个结点的有限集。在任意一棵非空树中：①有且仅有一个特定的称为根的结点；②当n>1时，其余结点分为m（m>0）棵互不相交的有限子树，每棵子树又是一棵树。

如图4–7所示，树T由根结点A及两棵子树T1、T2组成。同样，T1中，B是根结点，其下又可以细分出三棵子树（D、EHI及F）；T2中，C是根结点，其下又可以细分出1棵子树（GJ）。

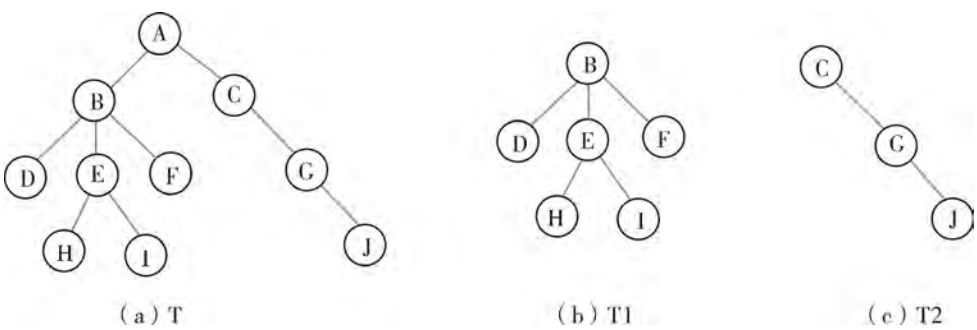


图4–7　树的结构

2. 树的基本概念

（1）结点的度。

每个结点具有的子树数称作结点的度。例如图4–7（a）树T中， B结点的度为3，I结点的度为0。

（2）分支结点和叶子结点。

在一棵树中，度等于0的结点称作叶子结点。度大于0的结点称为分支结点或非终端结点。树T中，D、H、I、F、J为叶子结点，A、B、C、E、G为分支结点。

（3）孩子结点、双亲结点、兄弟结点。

在一棵树中，每个结点的子树的根，称为该结点的孩子结点，相应地，该结点被称为孩子结点的双亲、父亲或父母。具有同一双亲的孩子互称兄弟。在图4–7（a）树T中，结点B是结点A的孩子结点，结点A是结点B的双亲结点，结点B和C互为兄弟。

（4）树的深度。

从树根开始，根结点为第一层，它的孩子结点为第二层，如此类推。树中所有结点的最大层数称为树的深度。图4–7（a）树T的深度为4。

交流

同学们列举生活、学习、工作中具备树结构特征的事物或现象，以树的结构表示出这种事物或现象，并尝试用树的概念描述。

4.3.2 二叉树

如图4-8所示的是自然界中的二叉树，它每一个枝都只有两个分枝，树枝的尽头才是叶子，是不是很有趣呢?

图4-8　现实中的二叉树

计算机数据结构中的二叉树是一种特殊树形结构，它的特点是每个结点至多只有两棵子树（即二叉树中不存在度大于2的结点），而且二叉树的子树有左右之分，其次序不能任意颠倒。二叉树在计算机领域有着广泛的应用。

如图4-9（a）所示的二叉树，其左子树如图4-9（b）所示，右子树如图4-9（c）所示，可见其左右子树也是二叉树。

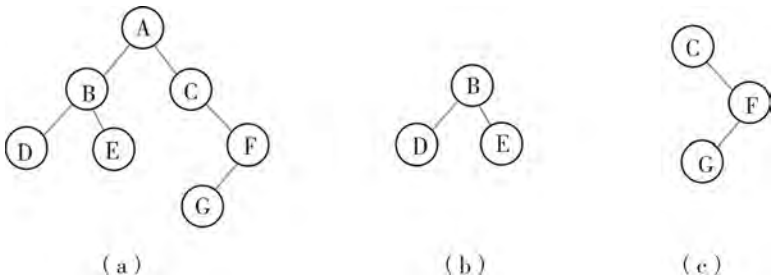


图4-9　二叉树示例

在二叉树中，每个结点的左子树的根结点被称为左孩子，右子树的根结点被称为右孩子。二叉树有五种基本形态，如图4-10所示。

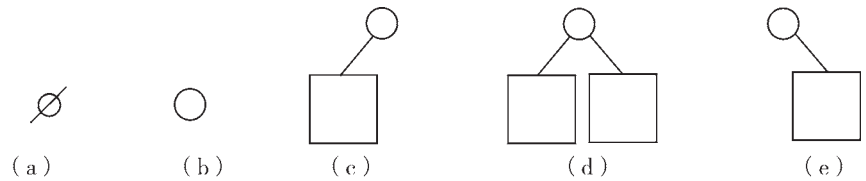
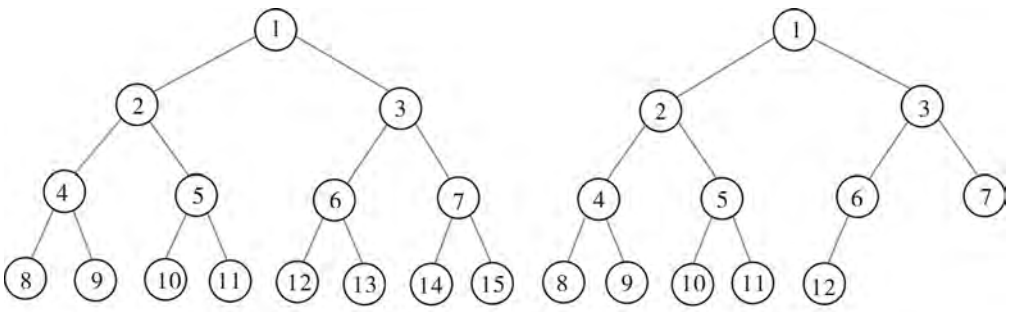


图4-10　二叉树的五种基本形态

（a）为空二叉树；（b）为仅有根结点的二叉树；（c）为右子树为空的二叉树；（d）为左右子树均非空的二叉树；（e）为左子树为空的二叉树。

一棵深度为k且有2^k-1个结点的二叉树称为满二叉树，如图4-11（a）所示。在一棵二叉树中，除最后一层外，若其余各层都是满的，并且最后一层或者是满的，或者是在右边缺少连续若干个结点，则称此树为完全二叉树，如图4-11（b）所示。满二叉树是完全二叉树的特例。



（a）满二叉树　　（b）完全二叉树

图4-11　满二叉树和完全二叉树

4.3.3　二叉树的抽象数据类型

二叉树的抽象数据类型的数据部分为一棵用任一种方式表示的二叉树，操作部分包括初始化二叉树、建立二叉树、遍历二叉树、查找二叉树、输出二叉树和清除二叉树等常用操作。下面给出二叉树的抽象数据类型的参考定义：

```
ADT 二叉树
{
  数据：采用任一种方式存储的二叉树，其存储类型用BinTreeType标识符表
  示，该类型的一个二叉树用BT标识符表示。二叉树结点所保存的数据元素类
```

```
    型用ElemType表示。
      操作：
      void InitBTree(BinTreeType &BT);
        //初始化二叉树，把它置为一棵空树
      void CreateBTree(BinTreeType &BT);
        //建立对应的存储结构
      bool EmptyBTree(BinTreeType &BT);
        //判断一棵二叉树是否为空，若是则返回true，否则返回false
      void TraverseBTree(BinTreeType &BT);
        //按照一定的次序遍历一棵二叉树，使得每个结点的值均被访问一次
      bool FindBTree(BinTreeType &BT, ElemType &node);
        //从二叉树中查找值为node的结点，若存在，返回true，否则返回false
      int BTreeDepth(BinTreeType &BT);
        //求出一棵二叉树的深度
      void PrintBTree(BinTreeType &BT);
        //输出一棵二叉树
      void ClearBTree(BinTreeType &BT);
        //清除二叉树中的所有结点，使之变为一棵空树
}ADT 二叉树
```

4.3.4　二叉树的基本操作方法

二叉树的基本操作较多，下面以遍历为例，了解二叉树的基本操作方法。

遍历是二叉树的一种基本操作，是指按一定的次序访问树中所有结点，并且每个结点的值仅被访问一次的过程。由于二叉树是非线性结构，因此二叉树的遍历实质上是将二叉树的所有结点转换成一个线性序列的过程。

根据二叉树的递归定义，一棵非空二叉树由根结点、左子树和右子树所组成。因此，遍历一棵非空二叉树的问题可分解为三个子问题：访问根结点、遍历左子树、遍历右子树。若用L、D、R分别表示遍历左子树、访问根结点和遍历右子树，则对一棵二叉树的遍历有DLR、LDR、LRD、DRL、RDL、RLD六种情况，若限定先左后右，则只有前三种情况，分别称为前序遍历（或先根遍历）、中序遍历（或中根遍历）、后序遍历（或后根遍历）。

（1）前序遍历的操作步骤为：

```
若二叉树为空，则结束操作；
否则：①访问根结点；
```

②前序遍历左子树；
③前序遍历右子树。

（2）中序遍历的操作步骤为：

若二叉树为空，则结束操作；
否则：①中序遍历左子树；
②访问根结点；
③中序遍历右子树。

（3）后序遍历的操作步骤为：

若二叉树为空，则结束操作；
否则：①后序遍历左子树；
②后序遍历右子树；
③访问根结点。

图4-9二叉树（a）的前序遍历为：ABDECFG；中序遍历为：DBEACGF；后序遍历为：DEBGFCA。

1. 篮球淘汰赛赛制编排。小白是学生会的体育部部长，他正在组织校际篮球赛，比赛在本校八个班（编号1A~1H）和 ×× 学校八个班（编号2A~2H）之间进行。为了缩短赛程，小白采用淘汰赛赛制，并绘制了如图4-12所示的篮球赛冠军产生过程。这幅图是一棵二叉树吗？它具有二叉树的特征吗？

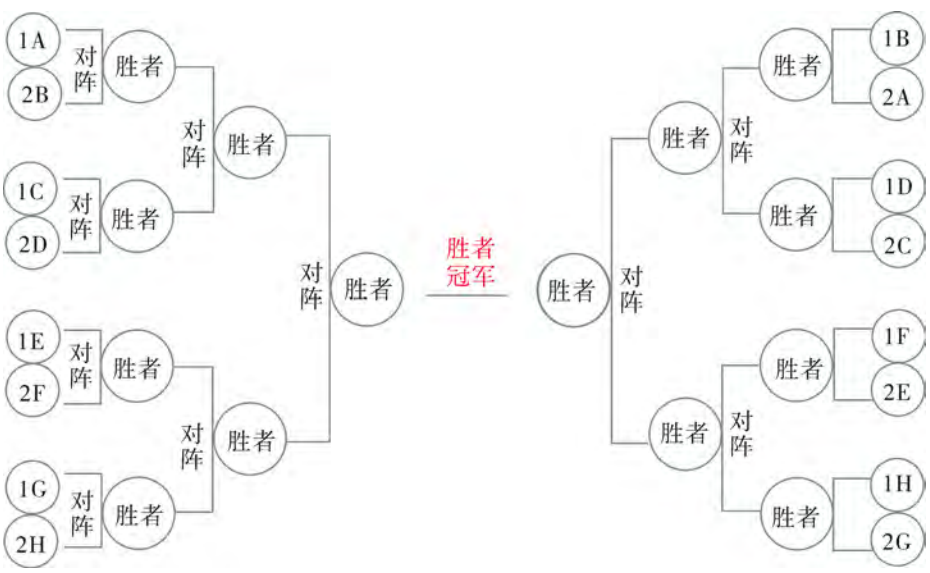


图4-12　篮球淘汰赛赛制

2. 算术表达式树。小白上数据结构课的时候，发现可以把一个算术表达式表示成一棵二叉树：运算符作为根结点，运算符的前后两个运算对象分别作为根的左、右两棵子树。如把算术表达式(a+b)*(c-d/e)/f表示成二叉树，结果如图4-13所示。

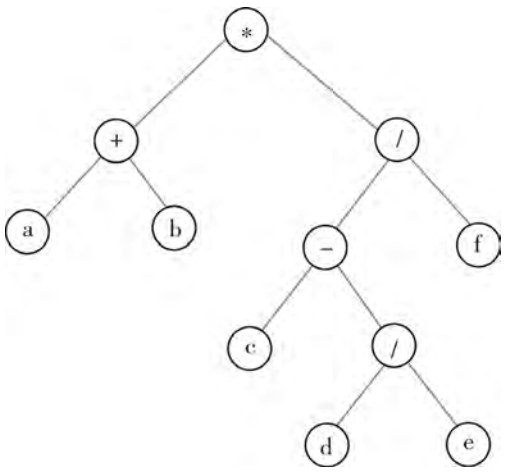


图4-13　算术表达式树

小白想知道，哪些算术表达式可以表示成这样的树？这种表达式树在表示表达式时，有什么优势？请尝试写出图4-13的算术表达式树的前序遍历、中序遍历、后序遍历。

项目实施

各小组根据项目选题及拟订的项目方案，结合4.2节和4.3节所学知识，分析判断所选案例的抽象数据类型是否符合队列、栈或树的特征，进一步完善所选择的抽象数据类型案例的数据结构、基本操作定义，判断分析其合理性。参照项目范例的样式，撰写相应的项目成果报告。

成果交流

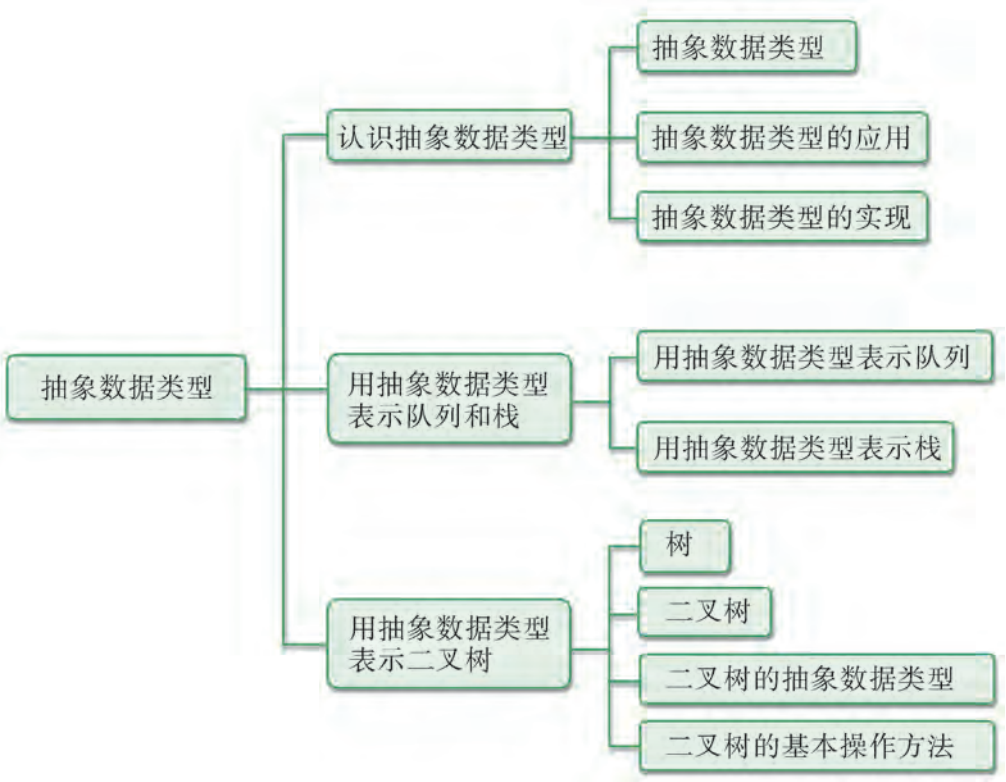
各小组运用数字化学习工具，将所完成的项目成果，在小组或班级上进行展示与交流，共享创造、分享快乐。

活动评价

各小组根据项目选题、拟订的项目方案、实施情况以及所形成的项目成果，利用教科书附录2的“项目活动评价表”，展开项目学习活动评价。

本章扼要回顾

同学们通过本章学习，根据“抽象数据类型”知识结构图，扼要回顾、总结、归纳学过的内容，建立自己的知识结构体系。



回顾与总结

本章学业评价

同学们完成下列测试题（更多的测试题可以在教科书的配套学习资源包中查看），并通过“本章扼要回顾”以及本章的项目活动评价，综合评价自己在信息技术知识与技能、解决实际问题的过程与方法，以及相关情感态度与价值观的形成等方面，是否达到了本章的学习目标。

1. 单选题

（1）如果树的根算第一层，那么一棵n层的二叉树最多有（　　）个结点。

A. 2^n-1　　B. 2^n　　C. 2^n+1　　D. 2*n+1

（2）完全二叉树共有（2*n−1）个结点，则它的叶结点数是（　　）个。

A. n−1　　B. n　　C. 2*n　　D. 2*n−1

（3）已知包含七个结点的二叉树的先根遍历是1 2 4 5 6 3 7（数字为结点的编号，下同），中根遍历是4 2 6 5 1 7 3，则该二叉树的后根遍历是（　　）。

A. 4 6 5 2 7 3 1　　B. 4 6 5 2 1 3 7

C. 4 2 3 1 5 6 7　　D. 4 6 5 3 1 7 2

2. 思考题

抽象数据类型与一般数据类型的关系是什么？请举例说明。

3. 情境题

小明和小白对二叉树的遍历产生了浓厚的兴趣，他们提出了以下问题：

（1）已知前序遍历序列为GDAFEMHZ，中序遍历序列为ADEFGHMZ，请画出这棵二叉树。

（2）已知一棵二叉树的前序和中序遍历序列，能求出其后序遍历序列吗？其方案唯一吗？

第五章
数据结构的应用

查找和排序，与我们的学习、生活息息相关。如从字典中查找汉字，从电话号码本中查找电话，在图书馆中查找图书，高考查询成绩等；教师按身高来安排学生的座位，大学按照高考成绩高低顺序录取新生，网上商城按照销量高低排序来推荐商品等。在信息时代，数据的增长速度越来越快，导致信息量呈几何级的增长。在庞大的数据里进行人工查找和排序是非常困难的，甚至是无法办到的，所以必须依靠计算机才能快速、准确地对数据进行查找和排序。

本章将通过“数据结构应用程序设计”项目，进行自主、协作、探究学习，让同学们通过实现数据的排序和查找，体验迭代和递归的方法，理解算法与数据结构的关系，从而将知识建构、技能培养与思维发展融入运用数字化工具解决问题和完成任务的过程中，促进信息技术学科核心素养达成，完成项目学习目标。

- 迭代与递归
- 查找
- 排序
- 算法与数据结构的联系与区别

项目范例　超市促销商品的选择与查询程序设计

情境

超市开业快两周年了，为了回馈老客户、吸引新客户，提高超市的销售业绩和效益，超市负责人决定利用开业两周年庆典来进行各种促销活动，而且促销的力度要大，效益要高。策划小组经过讨论，决定采用多买多赠和组合销售等方式进行促销活动：

优惠方式一：对库存多、滞销的商品可采取多买多赠的方式促销，赠送的可以是价格较低的物品或购物券，老用户还可以赠送不同的积分。

优惠方式二：销售量高的商品与销售量低的商品一起组合销售，利润大的商品与利润小的商品一起组合销售。

主题

超市促销商品的选择与查询程序设计

规划

根据项目范例的主题，在小组中组织讨论，利用思维导图工具，制订项目学习规划，如图5–1所示。

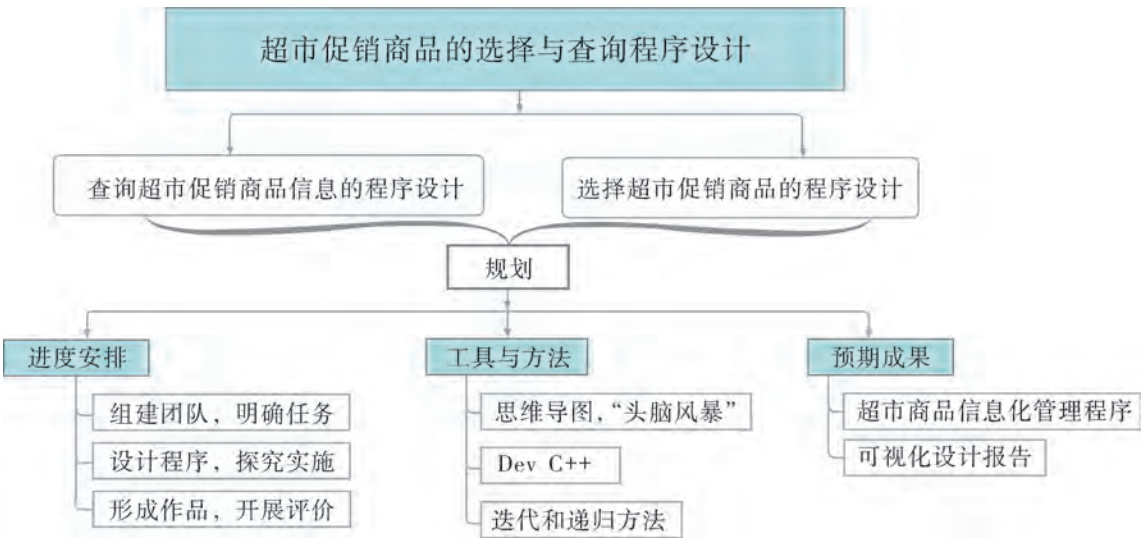


图5–1　“超市促销商品的选择与查询程序设计”项目学习规划

探究

根据项目学习规划的安排，通过调查、案例分析、文献阅读和网上搜索资料，开展“超市促销商品的选择与查询程序设计”项目学习探究活动，如表5–1所示。

表5-1　“超市促销商品的选择与查询程序设计”项目学习探究活动

探究活动	学习内容		知识技能
查询超市促销商品信息的程序设计	数据建模。	抽取促销商品的属性，用数据表示属性。 分析促销商品各个属性之间的关系，确定数据关系。 理解促销商品数据结构的提炼方法，建立数据模型。	理解算法与数据结构的关系。
	组织、存储数据。	根据促销商品数据模型的逻辑关系选择数组组织、存储促销商品的数据。	
	设计算法。	根据促销商品的数据量，选择合适的查找算法。	体验迭代和递归的方法。
	编写程序。	采用数组组织、存储数据，选择合适的算法编写程序，解决问题。	编写程序实现数据的查找。
选择超市促销商品的程序设计	数据建模。	抽取促销商品的属性，用数据表示属性。 分析促销商品各个属性之间的关系，确定数据关系。 理解促销商品数据结构的提炼方法，建立数据模型。	理解算法与数据结构的关系。
	组织、存储数据。	根据促销商品数据模型的逻辑关系选择数组组织、存储促销商品的数据。	
	设计算法。	根据促销商品的数据量，选择合适的排序算法。	体验迭代和递归的方法。
	编写程序。	采用数组组织、存储数据，选择合适的算法编写程序，解决问题。	编写程序实现数据的排序。

实施项目学习各项探究活动，进一步熟悉查找和排序算法，理解算法与数据结构的关系。

在小组开展项目范例学习过程中，利用思维导图工具梳理小组成员在“头脑风暴”活动中的观点，建立观点结构图，运用多媒体创作工具（如演示文稿、在线编辑工具等），综合加工和表达，形成项目范例可视化学习成果，并通过各种分享平台发布，共享创造、分享快乐。例如，运用在线编辑工具制作的“超市促销商品的选择与查询程序设计”可视化报告，可以在教科书的配套学习资源包中查看，其目录截图如图5-2所示。

图5-2　“超市促销商品的选择与查询程序设计”可视化报告的目录截图

评价

根据教科书附录2的“项目活动评价表”，对项目范例的学习过程和学习成果在小组或班级上进行交流，开展项目学习活动评价。

项目选题

同学们以3~6人组成一个小组，选择下面一个参考主题，或者自拟一个感兴趣的主题，开展项目学习。

1. 学生成绩排序与查询的应用程序设计
2. 校运会竞赛成绩排序与查询的应用程序设计
3. 校园歌手大赛成绩排序与查询的应用程序设计

项目规划

各小组根据项目选题，参照项目范例的样式，利用思维导图工具，制订相应的项目方案。

方案交流

各小组将完成的方案在全班进行展示交流，师生共同探讨、完善相应的项目方案。

5.1 迭代与递归

在利用计算机解决实际问题中，迭代和递归都是非常实用的算法，很多难解的问题都是通过迭代或递归算法解出来的。

5.1.1 迭代

1. 迭代法

迭代法是用计算机解决问题的一种基本方法。它利用计算机运算速度快、适合做重复性操作的特点，让计算机重复执行一组指令（或一定步骤），在每次执行这组指令（或这些步骤）时，都从变量的原值推出它的一个新值。

例1：从键盘输入n，求s=1+2+…+n 。

算法分析：可以用变量sum记录总和，利用迭代的方法把1，2，…，n一步一步地加到sum中。

迭代过程中，sum变量的变化如表5-2所示。

表5-2 迭代过程变量分析

i值	迭代变量sum的迭代过程	sum式子的变化
	sum初始化为0	sum=0
1	sum=sum+i =0+1=1	sum=1
2	sum=sum+i =1+2=3	sum=1+2
3	sum=sum+i =3+3=6	sum=1+2+3
⋮	⋮	⋮
n	sum=sum+n	sum=1+2+…+n

根据以上分析，可以用循环累加的方法实现，核心代码如下：

```
int n,sum=0;  //sum初始化为0
for(int i=1; i<=n; i++)  //用循环实现迭代
{
    sum=sum+i; //迭代过程
}
```

2. 用迭代法解决问题

用迭代法解决问题，需考虑三个方面的内容。

（1）确定迭代变量。

在可以用迭代法解决的问题中，至少存在一个直接或间接地不断由旧值递推出新值的变量，这个变量就是迭代变量。在例1中，sum就是迭代变量。

（2）建立关系式。

迭代关系式是指如何从变量的前一个值推出其下一个值的公式（或关系）。迭代关系式的建立是解决迭代问题的关键，通常可以使用顺推或倒推的方法来完成。例1中，“sum=sum+i”就是迭代关系式，通过此式可以进行迭代求和。

（3）过程控制。

编写迭代程序必须考虑在什么时候结束迭代过程，不能让迭代过程无休止地重复执行下去。迭代过程的控制通常可分为两种情况：一种是所需的迭代次数是个确定的值，可以计算出来；另一种是所需的迭代次数无法预先确定。对于前一种情况，可以构建一个固定次数的循环来实现对迭代过程的控制；对于后一种情况，需要进一步分析确定迭代过程结束的条件。在例1中，用了for循环语句进行过程控制。

例2：一对刚出生的小兔子，一个月后就能成长为成年兔，再过一个月后（即第三个月起）就每月生一对兔子。新生的兔子也按这个规律繁殖。现在仅有一对刚出生的小兔子，问在没有兔子死亡的情况下，一年后总共繁殖成多少对兔子。

算法分析：设f(n)表示第n个月的兔子对数，先看前几个月的情况：第一个月有一对刚出生的兔子，即f(1)=1；第二个月，这对兔子长成成年兔，兔子对数还是只有1对，即f(2)=1；第三个月，这对成年兔生出一对小兔，共有两对兔子，即f(3)=2；第四个月，成年兔又生出一对小兔，第三个月出生的兔子长成成年兔，共有三对兔子，即f(4)=3；第五个月，原成年兔又生出一对小兔，新成年兔也生出一对小兔，共有五对兔子，即f(5)=5……以此类推，每个月兔子数为1，1，2，3，5，8，13，21，…

转化为数学模型，设f(n)为第n个月的兔子对数，则有：

$$\begin{cases} f(1)=1, \\ f(2)=1, \\ f(n)=f(n-1)+f(n-2)\ (n>=3) \end{cases}$$

下面是实现求一年后繁殖的兔子总数的核心代码，其中f1、f2、fn这三个迭代变量分别代表前两个月、前一个月、当前月的兔子数，用迭代关系式“fn = f1 + f2;”计算当月的兔子数，再用“f1 = f2; f2 = fn;”为下一个月的迭代计算做好准备。

```
int f1=1,f2=1,fn=0;       //迭代变量
for(int i=3; i<=12; i++) //用i的值来控制迭代的次数
{
  fn=f1+f2;    //计算当月数量
  f1=f2;       //f1、f2迭代计算，为下个月迭代赋初值
  f2=fn;
}
```

每个月的兔子对数组成数列：1，1，2，3，5，8，13，21，34，55，89，144，…此数列也称为斐波那契数列。

探究活动

讨论

分组讨论，日常学习、生活中有哪些活动、工作流程能体现迭代的思想？理解迭代算法思想及其编程实现原理。

5.1.2 递归

1. 递归的基本概念

若一个对象用自己来定义自己或部分地包含自己，则称这个对象是递归的；若一个过程直接地或间接地调用自己，则称这个过程是递归过程。在程序设计中，函数直接调用函数本身，称为直接递归调用；在函数中调用其他函数，其他函数又调用原函数，构成了函数自身的间接调用，则称为间接递归调用。

2. 递归在数学中的应用

在数学中，有这样一种数列，很难求出它的通项公式，但数列中各项间关系却很简单，于是人们想出另一种办法来描述这种数列，即通过初值及与前面临近几项之间的关系来描述。要使用这样的描述方式，至少要提供两个信息：一是最前面几项的数值，二是数列间各项的关系。

例3：阶乘数列是由这样一组数组成：1，2，6，24，120，720，…，阶乘公式可以表示为$n!=n\times(n-1)\times(n-2)\times\cdots\times1(n>0)$，数学上用这样的方式来描述阶乘数列：

$$n!=\begin{cases}1 & (n=1)\\ n\times(n-1)! & (n>1)\end{cases}$$

下面的递归函数f可以实现求n!的值：

```
int f(int n)  //定义递归函数f
{
    if(n==1) return 1;  //当n=1时返回1，结束递归
    return n*f(n-1);    //当n>1时返回n*f(n-1)，调用函数f来计算(n-1)!
}
```

这是递归函数的最简单形式，从中可以明显看出递归函数一般的写法特点：先处理一些特殊情况（递归终结条件），再处理递归关系。

递归函数的执行过程总是先通过递归关系不断地缩小问题的规模，直到简单到可以作为特殊情况处理而得出直接的结果，再通过递归关系逐层返回到原来的数据规模，最终得

出问题的解。

下面以求阶乘数列的函数f(n)为例来描述递归的执行过程。如求f(3)的值，由于3不是特殊值，因此需要计算3×f(2)；而f(2)又是对它自己的调用，于是再计算f(2)，2也不是特殊值，需要计算2×f(1)；于是再计算f(1)的值，1是特殊值，于是直接得出f(1)=1，返回上一步，得出f(2)= 2×f(1)=2，再返回上一步，得出f(3)= 3×f(2)=6，进而得到最终解。求阶乘函数f(3)的执行过程如图5–3所示。

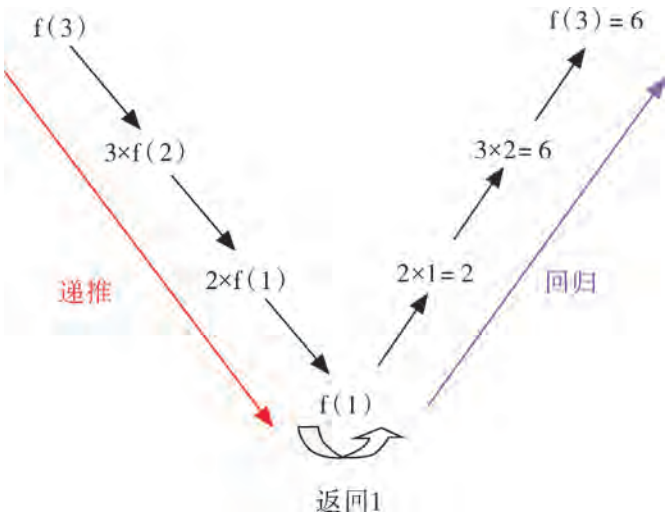


图5–3　求阶乘函数f(3)的执行过程图

3. 递归算法的思想与应用

（1）递归算法的思想。

为求解一个不能或不好直接求解的“大问题”，设法将它分解成规模较小但解法相同的“小问题”，并且这些“小问题”也能采用同样的分解方法再分解成规模更小的“小问题”，当规模最小的一个或几个值能直接得出解时，再利用这些“小问题”的解构造出“大问题”的解。

（2）递归算法解决问题的特点。

递归算法有四个特性：

①必须有明确的递归终结条件，否则程序将陷入无穷循环而造成栈溢出。

②子问题在规模上比原问题小，或更接近终止条件。

③子问题可通过再次递归调用求解或因满足终止条件而直接求解。

④子问题的解应能组合为整个问题的解。

（3）递归算法一般用于解决的三类问题。

①数据的定义是按递归定义的。如数学上常用的阶乘数列、斐波那契数列、幂函数等，它们的定义都是递归的。

②问题方便用递归算法实现。有些问题用递归方法实现更方便，如求阶乘函数的值。

③数据的结构形式是按递归定义的。如链表、树的遍历、图的搜索等。

例4：斐波那契数列（Fibonacci），又称黄金分割数列，指的是由这样一组数组成的数列：1，1，2，3，5，8，13，21，34，55，89，144，233，…

斐波那契数列的第一项是1，第二项也是1，从第三项开始，每一项的值都等于前两项之和。在前面例2中，我们用迭代算法实现了对兔子繁殖数量组成的斐波那契数列的求解。同样，斐波那契数列也可以用递归算法实现求解，其核心代码如下：

```
int fib(int n)  //定义递归函数fib，参数n是指数列的第几项
{
    if(n==1||n==2) return 1; //当n=1或n=2时，返回1，结束递归
    return fib(n-1)+fib(n-2);
    //当n>2时，本项的值等于前两项之和，调用函数fib来计算前两项fib(n-1)
    //和fib(n-2)
}
```

分组交流，在日常学习、生活和工作中，有哪些活动或流程能体现递归的思想？

5.2 查找

在日常生活和学习中，我们经常要进行查找工作，例如从字典中查找汉字、单词，从电话号码本中查找电话，在图书馆中查找图书，高考查询成绩等。其实，“电话号码本”“字典”等都可以看作一张表，查找则是在一个含有众多数据元素（或记录）的表中找出某个特定的数据元素（或记录）。在信息时代，由于信息量巨大，人工查找是非常困难的，甚至是无法办到的，所以必须依靠计算机才能快速、准确地查找信息。

5.2.1 顺序查找

顺序表是指采用顺序存储的方式存储的集合或线性表。在本章，我们主要探讨一维数组这种顺序表的顺序查找和二分查找方法。

顺序查找也称为线性查找，它的基本思想是：从顺序表的一端开始，将每个元素的关键字与给定值k进行比较，如果相同，则表明查找成功，返回该元素的下标；如果在所有元素都进行了比较后，仍找不到关键字为k的元素，则表明查找失败，返回特定值-1。

在超市中，要实现查询某商品在哪个货架，我们可以用数组存储商品的编号、存放的货架等信息。程序运行时，输入需查询商品的编号，然后在数组中依次查找编号，就可查

找出该商品的信息，方便顾客查找商品。下面是顺序查找算法的核心代码：

```
int search(ShangPinType a[],int k,int n)
//在有n个元素的数组a中查找k
{
    for(int i=0; i<n; i++)          //顺序依次查找
    if(k==a[i].BianHao) return i;  //查找成功，则返回数组下标
    return -1;                      //查找失败，返回-1
}
```

查询促销商品信息的完整程序请参阅教科书配套学习资源包，程序运行界面如图5-4和图5-5所示。

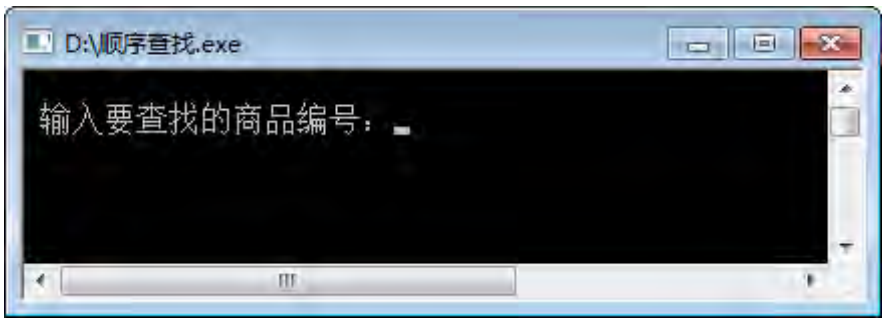


图5-4　输入查找商品的编号

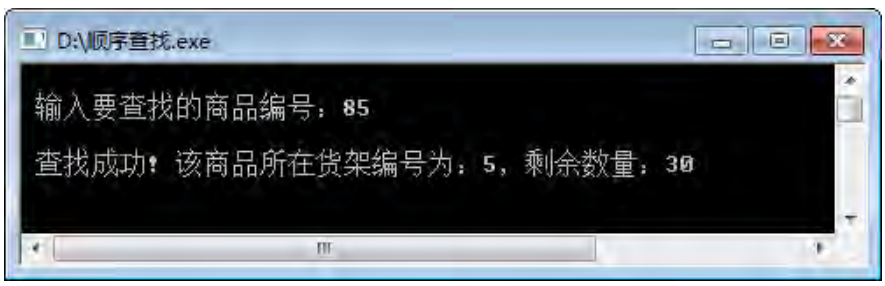


图5-5　显示查找结果

5.2.2　二分查找

顺序查找虽然能帮助顾客查找到所需信息，但由于超市销售的商品种类繁多，数据量比较大，而顺序查找每次都要从头到尾去查找，需要花费不少时间，很多顾客没有耐心长时间等待查询结果。所以查找的速度必须要快，可以考虑用查找速度更快的二分查找来实现。

二分查找又称折半查找，是针对顺序存储的有序表（有序表是指各元素按关键字的值以升序或降序存放的表）进行的查找，它是一种较常用的查找方法。在按升序存储的顺序表a中查找k，其二分查找算法流程如下：

（1）选取查找范围a[0]~a[n-1]。

（2）选定查找范围的中点元素a[mid]，与k值比较。

（3）若相等，则查找成功，返回该元素的下标。
（4）若k< a[mid]，则将范围缩小到左子表a[0]~a[mid−1]。
（5）若k> a[mid]，则将范围缩小到右子表a[mid+1]~a[n−1]。
（6）迭代以上过程，直到找到k或当前查找区间为空（即查找失败）。
二分查找算法的过程如图5–6所示。

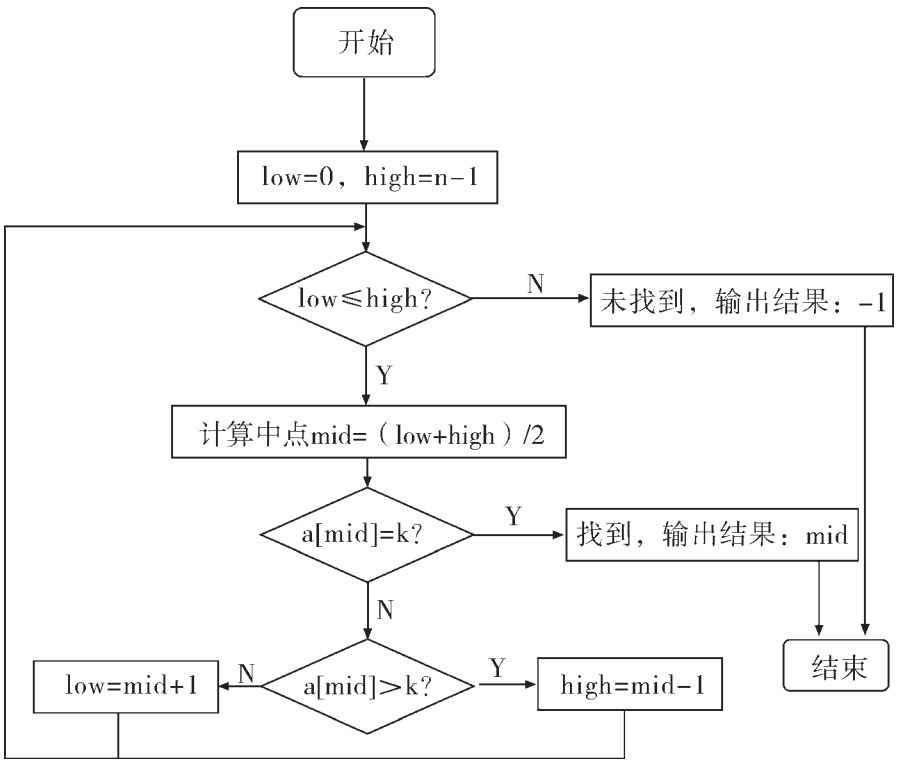


图5–6 二分查找流程图

二分查找算法的实现代码如下：

```
int binsearch(ShangPinType a[],int k,int n)
//在n个元素的数组a中二分查找k
{
  int low=0,high=n-1; //待查区间下界、上界
  while(low<=high)
  {
    int mid=(low+high)/2; //取待查区间内中间元素的下标
    if(k==a[mid].BianHao) return mid;  //中间元素即待找元素，返回mid值
    else if(k<a[mid].BianHao) high=mid-1;  //修改区间上界，在左子表
                                           //继续查找
    else low=mid+1;  //修改区间下界，在右子表继续查找
  }
  return -1;  //未找到，返回-1
}
```

二分查找过程分析：假设一维数组a中有10个元素：1，2，6，8，12，13，36，45，49，85，在其中查找k=45的过程如下（用“[ ]”表示low和high，即当前查找范围的下界和上界，mid指示查找范围的中间位置）：

```
                下标  0  1  2  3  4  5  6  7  8  9
第一次查找：元素 [ 1  2  6  8  12 13 36 45 49 85 ]
                                  ↑
                                 mid
mid=(0+9)/2=4，取a[4]元素12，因为12<45，所以low=mid+1=5。

第二次查找：元素  1  2  6  8  12 [ 13 36 45 49 85 ]
                                        ↑
                                       mid
mid=(5+9)/2=7，a[7]元素为45，查找成功，返回mid=7。
```

在其中查找k=48的过程如下：

```
                下标  0  1  2  3  4  5  6  7  8  9
第一次查找：元素 [ 1  2  6  8  12 13 36 45 49 85 ]
                                  ↑
                                 mid
mid=(0+9)/2=4，取a[4]元素12，因为12<48，所以low=mid+1=5。
第二次查找：元素  1  2  6  8  12 [ 13 36 45 49 85 ]
                                        ↑
                                       mid
mid=(5+9)/2=7，取a[7]元素45，因为45<48，所以low=mid+1=8。
第三次查找：元素  1  2  6  8  12 13 36 45 [ 49 85 ]
                                           ↑
                                          mid
mid=(8+9)/2=8，取a[8]元素49，因为49>48，所以high=mid-1=7。
第四次查找：元素  1  2  6  8  12 13 36 45][49 85
                                        ↑↑
                                     high low

此时，high<low，循环结束，说明查找失败，返回-1。
```

解决同一个问题，我们可以用不同的算法编写程序，不同的算法对数据结构的要求可能是不同的。对比顺序查找与二分查找算法，当数据量较大时，二分查找会比顺序查找快很多，这是它的主要优点，但它要求查找表是有序的，所以在进行二分查找之前，必须先对查找表进行排序。另外，二分查找要求表是顺序存储的，为保持表的有序性，在进行插入、删除操作时，都必须移动大量记录。因此，二分查找的高查找效率是以排序为代价的，所以它特别适合一经建立就很少移动而且经常需要查找的线性表。

了解不同算法的特点，可以帮助我们在解决实际问题时，从不同的算法中选择出较为合适的一种，或对现有算法进行改进或创新，从而设计出更好的算法。

理解顺序查找和二分查找的思想和算法实现过程，分析二分查找中的迭代过程是如何进行的。体验顺序查找、二分查找两种实现方法在不同数据量上运行速度的差异。数据量要多大，才有明显差异呢？

在编写“超市促销商品的选择与查询程序设计”程序之前，首先要抽取促销商品的相关数据，列出促销商品共有的属性，如编号、名称、数量、价格、存放位置、销量、销售额等，根据本次促销活动的原则，选择与问题相关的属性，并用数据来表示这些属性；然后分析提取出来的各属性之间的关系，确定它们之间的数据关系，进而建立数据模型。

根据促销商品数据模型的逻辑关系，选择一维数组来组织、存储促销商品的数据，并定义如下的数据结构：

```
struct ShangPinType  //定义查找所用到的数据结构
{
  int BianHao  //数据项商品编号的类型为整型
  int Where    //数据项商品存放货架的类型为整型
  int Count    //数据项剩余商品数量的类型为整型
};
```

根据上面的数据结构，设计查询促销商品信息的算法：输入需查询商品的编号，然后在数组中查找该编号，若查找到此编号就输出该商品的信息。数据量小时可以用顺序查找算法；数据量比较大时，顺序查找花费时间较多，应该选用查找速度更快的二分查找。完整程序请参阅教科书配套学习资源包。

项目实施

各小组根据项目选题及拟订的项目方案，结合5.1节和5.2节所学知识，参照项目范例中用顺序查找和二分查找算法实现的查询超市促销商品信息的完整程序，完成下列任务。

1. 定义选定项目所需的数据结构，选择合适的数据结构组织、存储数据。
2. 分别用顺序查找和二分查找算法编写完整程序，实现数据查找的功能。
3. 体验迭代方法，理解算法与数据结构的关系。

5.3 排序

排序与我们的日常生活息息相关。例如，教师按身高来安排学生的座位，试卷和答题卡按从小号到大号的顺序来整理，各类比赛按成绩的高低来排名，查询火车票时会按照出发的先后来显示，到网上购物会参考销量高低来排序购买等。排序是数据处理和分析中最常用的运算之一，它往往可以提高数据处理的效率；排序也是最基本的算法之一，其他很多算法都是以排序算法为基础，所以研究和掌握排序算法是非常重要的。在信息时代，面对庞大的信息量，想要靠人工进行排序，会耗费大量时间和精力，甚至无法完成。所以，依靠计算机快速、准确地对数据进行排序，是很有必要的。

5.3.1 认识排序

1. 排序

排序是指把一个任意序列的数据元素重新排列成按照某关键字递增或递减序列的过程。作为排序依据的数据项称为排序关键字，简称关键字（Key）。排序时选取哪一个数据项作为关键字，应根据具体情况而定。例如，表5-3为超市某天商品销售的部分数据表，表中每个商品的信息包括条形码、商品名称、单价、销售数量、单位以及销售额。

表5-3 超市某天商品销售的部分数据表

条形码	商品名称	单价 / 元	销售数量	单位	销售额 / 元
6947503702526	签字笔	2.3	25	支	57.5
6903388620003	牙刷	6.5	10	支	65
6911989541832	日记本	4.8	17	本	81.6
6924743910447	薯片	8.8	13	包	114.4
4891338010528	牙膏	17.5	10	盒	175
6923450605288	口香糖	1.5	37	盒	55.5
6902088702347	洗衣粉	12	6	袋	72

以表5-3中的“条形码”作为关键字排序，可以快速查找到商品的销售数据，因为每个商品的条形码是唯一的。若想以“销售数量”来排列名次，就应把“销售数量”作为关键字进行降序排列。待排序的元素可以是任意的数据类型，其关键字可以是整型、实型、字符型等基本数据类型，通过排序可以构造一种新的按关键字有序排列的序列。

2. 排序的分类

假定在待排序的序列中，存在多个具有相同键值的数据元素，若经过排序，这些数据元素的相对次序仍然保持不变，则称这种排序是稳定的排序；否则，称这种排序是不稳定的排序。例如以表5-3中的“销售数量”来排列名次：

待排序的序列：25 10 17 13 10 37 6 //给其中一个“10”加底色以区分

稳定的排序：6 10 10 13 17 25 37 //因为10还是在10的前面

不稳定的排序：6 10 10 13 17 25 37 //因为10已在10的后面

在排序过程中，可以使用内存储器，也可以使用外存储器。如果参加排序的数据元素的数量不多，能够全部调入内存中进行排序，则称这种排序为内部排序；若参加排序的数据元素的数量较大，需要分批调入内存中进行排序，排序后再分批存回外存储器，则称这种排序为外部排序。

内部排序的方法很多，按照排序过程中所用策略的不同，通常分为五大类：插入排序、选择排序、交换排序、归并排序和基数排序。其中，交换排序又包含冒泡排序和快速排序，都是常用的排序算法。冒泡排序是一种简单、常见的排序算法，适合初学者学习；快速排序是对冒泡排序的改进，它是目前内部排序中平均速度最快的排序算法，也是20世纪十大算法之一。本教科书主要讨论冒泡排序、快速排序的基本思想和算法实现。

3. 排序的存储结构

排序的方法很多，通常都要进行两种基本操作：

（1）比较两个元素关键字的大小。

（2）根据比较结果，将元素从一个位置移到另一个位置。

其中，第（1）种操作对大多数排序方法来说都是必要的，第（2）种操作可通过改变元素的存储方式避免，比如采用链表存储结构存储的数据元素。

从操作角度看，排序是线性结构的一种操作，待排序元素可以用顺序存储结构和链式存储结构来存储。在顺序存储结构中，待排序元素按自然顺序存放在连续的一块内存空间中，元素之间的次序关系由其存储位置决定，所以排序过程中一定要移动元素；在链式存储结构中，待排序元素按原始次序链接起来，排序时只需要修改链表指针，不用移动元素。

本教科书除特殊说明外，排序所采用的存储结构均为顺序结构，即用数组存储。

5.3.2　冒泡排序

1. 冒泡排序的基本思想

冒泡排序（Bubble Sort）是一种简单的交换排序算法，它是通过交换相邻的两个数据元素，逐步将待排序列变成有序序列。它的基本思想如下：

（1）假设待排序元素有n个，从第一个元素开始，依次交换相邻的两个逆序元素，直到最后一个元素为止，当第一趟排序结束，就会将最大（小）的元素移动到序列的末尾。

（2）然后按照以上方法进行第二趟排序，次大（小）的元素将会被移动到序列的倒数第二个位置。

（3）依次类推，经过n-1趟排序后，整个元素序列就成了一个有序（升序或降序）的序列。

每趟排序过程中，值小（大）的元素向前移动，值大（小）的元素向后移动，就像气泡一样向上升，因此将这种排序方法称为冒泡排序。

例5：假设有五个元素的待排序列为25、10、17、37、13，将此序列按升序排序的冒泡排序过程如图5-7所示。

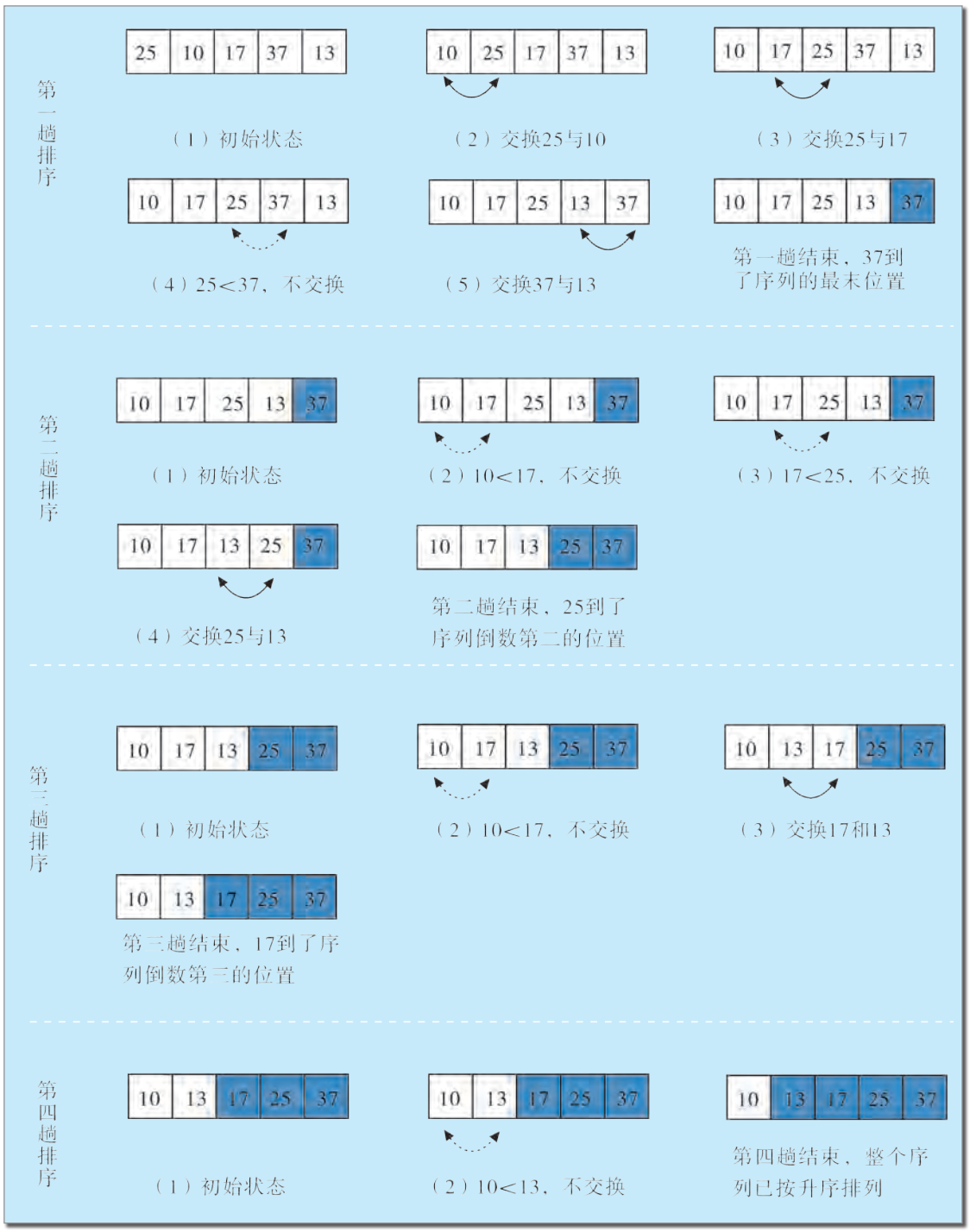


图5-7　冒泡排序的过程

2. 冒泡排序的算法实现

可以用以下程序实现冒泡排序算法：

```
void BubbleSort(ShangPinType r[],int n)
//对顺序表r[0..n]按关键字key作冒泡排序
{
  ShangPinType t;  //t用作交换操作的暂存单元
```

```
  for(int i=0;i<n;i++)  //需要进行n-1趟排序
  {
    for(int j=0;j<n-i;j++)  //每一趟排序需要比较n-i次
      if(r[j].key>r[j+1].key)
      {
        t=r[j];r[j]=r[j+1];r[j+1]=t;  //交换r[j]和r[j+1]
      }
  }
}
```

5.3.3　快速排序

快速排序是平均排序次数最少的排序算法之一，也是目前平均速度最快的排序，故称快速排序（Quick Sort），特别适合于数据量较多的排序。

1. 快速排序的基本思想

快速排序采用分治法的策略：将原问题划分成规模更小但与原问题相似的小问题，然后用递归法解决这些小问题，最后将它们组合形成原问题的解。

快速排序的基本思想是：从待排序元素序列中选取一个元素（通常是第一个元素）作为基准元素，其关键字设为key，然后将其余关键字小于key的元素移至key前面，而将关键字大于key的元素移至key后面，实现将待排序元素序列分为两个子表，最后将关键字key的元素插入其分界线的位置上，这个过程称为一趟快速排序；通过这一趟划分后，以关键字key的元素为界，待排序列被分为两个子表，且前面子表中所有元素的关键字均不大于key，而后面子表中所有元素的关键字均大于key。继续对分割后的子表按上述原则进行划分，直到所有子表的表长不超过1为止，此时的元素序列就成了一个有序序列。

2. 快速排序算法的步骤

以关键字升序排列为例，设待排序列存放在数组a[1..n]中，n为元素个数， i、j分别是待排序列首尾关键字的下标，初值分别为1和n，令a[1]作为基准元素赋给pivotkey：

（1）j从右往左扫描，若i<j且a[j].key >= pivotkey，j=j−1，否则将a[j]与a[i]交换。

（2）i从左往右扫描，若i<j且a[i].key < pivotkey，i=i+1，否则将a[i] 与a[j] 交换。

（3）重复执行（1）和（2），直到出现i≥j，则将基准元素pivotkey移动到a[i]中。此时整个元素序列在位置i被划分成两个部分，前一部分的元素关键字都小于或等于a[i].key，后一部分元素的关键字都等于或大于a[i].key，即完成了一趟快速排序。

（4）继续对分割后的子表进行上述划分，直至所有子表的表长不超过1为止，此时的元素序列就成了一个有序序列。

例6：假设待排序列为39，26，54，83，91，47，18，32，用快速排序算法对该序列进行升序排序，第一趟快速排序的过程如图5-8所示，快速排序的全部过程如图5-9所示。

```
        序号   1   2   3   4   5   6   7   8
        初始  39  26  54  83  91  47  18  32
              ↑i                        ↑j往左扫描（32<39，交换32和39）

第一次交换后  32  26  54  83  91  47  18  39
              ↑i往右扫描                ↑j
              32  26  54  83  91  47  18  39
                  ↑i                    ↑j          （54>39，交换54和39）
第二次交换后  32  26  39  83  91  47  18  54
                  ↑i                    ↑j往左扫描
              32  26  39  83  91  47  18  54
                  ↑i                ↑j          （18<39，交换18和39）
第三次交换后  32  26  18  83  91  47  39  54
                  ↑i往右扫描        ↑j
              32  26  18  83  91  47  39  54
                      ↑i            ↑j          （83>39，交换83和39）
第四次交换后  32  26  18  39  91  47  83  54
                      ↑i            ↑j往左扫描
              32  26  18  39  91  47  83  54
                      i↑↑j                      （i=j，本趟结束）
```

图5-8 一趟快速排序的过程

序号	1	2	3	4	5	6	7	8
初始状态	39	26	54	83	91	47	18	32
第一趟排序结果	{32	26	18}	39	{91	47	83	54}
第二趟排序结果	{18	26}	32	39	{54	47	83}	91
第三趟排序结果	18	{26}	32	39	{47}	54	{83}	91
快速排序最终排序结果	18	26	32	39	47	54	83	91

图5-9 快速排序的全部过程

3. 快速排序的算法实现

一趟快速排序（即将元素序列进行一次划分）的算法实现如下：

```
int Partition(ShangPinType r[],int i,int j)
//对顺序表r[i..j]的元素进行一趟排序，使基准元素前面元素的关键字小于基准
//元素的关键字，基准元素后面元素的关键字大于等于基准元素的关键字，并返回
//基准元素位置
{
  ShangPinType t;
  int pivotkey=r[i].key;  //用顺序表的第一个元素作为基准元素
  while(i<j)
  {
    while(i<j&&r[j].key>pivotkey)  //从右向左扫描
      --j;
    t=r[i];r[i]=r[j];r[j]=t;  //将比基准元素小的元素交换到左端
    while(i<j&&r[i].key<=pivotkey) //从左向右扫描
      ++i;
    t=r[i];r[i]=r[j];r[j]=t;  //将比基准元素大的元素交换到右端
  }
  return i; //返回基准元素所在位置
}
```

通过多次递归调用一趟快速排序算法，就可完成整个快速排序，其算法实现如下：

```
void QuickSort(ShangPinType r[],int i,int j)
//对顺序表r[i..j]作快速排序
{
  int pivotloc; //pivotloc是基准元素所在位置
  if(i<j)       //顺序表的元素个数大于1
  {
    pivotloc=Partition(r,i,j);   //将r[i..j]一分为二
    QuickSort(r,i,pivotloc-1);  //对左子表递归排序
    QuickSort(r,pivotloc+1,j);  //对右子表递归排序
  }
}
```

探究活动

交流

理解冒泡排序和快速排序的基本思想和算法、步骤，了解快速排序中的递归是如何进行的。

实践

通过上机实验，设计、对比不同的数据量，体验冒泡排序和快速排序两种算法在运行速度上的差异。图5-10和图5-11分别是项目范例中冒泡排序和快速排序的运行结果，完整程序请参阅教科书配套学习资源包。

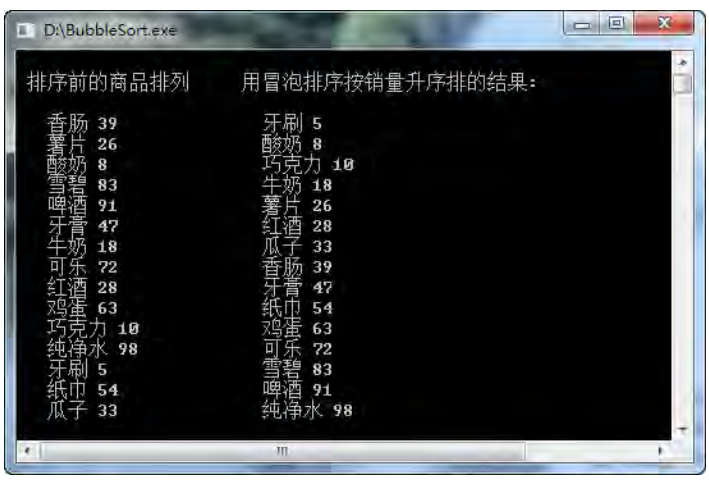


图5-10　项目范例中冒泡排序的运行结果

图5-11　项目范例中快速排序的运行结果

5.4 算法与数据结构的联系与区别

算法与数据结构都是计算机程序设计的重要理论和技术基础，在计算机学科中处于核心地位。运用计算机求解实际问题，往往是对实际问题的模型进行求解，把模型映射到存储器并将算法转换为程序。要开发出高效可行的计算机应用程序，就必须处理好算法与数据结构的关系。

5.4.1 算法与数据结构的联系

科学家提出一著名的公式：算法+数据结构=程序，它描述了计算机程序是由组织、存储信息的数据结构和处理信息的算法组成，也揭示了算法和数据结构都是计算机科学领域的重要支柱，两者是相辅相成、不可分割的。

程序设计的实质是为需要解决的实际问题设计好数据结构，再设计相应算法去实现。因此，既不能离开数据结构去抽象地分析求解问题的算法，也不能脱离算法去孤立地研究程序的数据结构，下面来看一个具体实例。

例7：编写程序实现超市某月各商品销售总额的查询（数据以1~6月的10种商品为例）。

分析：把超市1~6月10种商品的销售总额存放在一个数据表中，数据表记录了商品的编号和各个月的销售总额，输入商品的编号和月份，然后从数据表中查询此商品在该月的销售总额。

我们可以看出解决此问题的算法主要是在数据表中进行查询操作，并不复杂，如何进行查询则完全依赖于数据表中的数据是如何组织和存储的。数据组织和存储的方案有多种，下面给出其中的两种方案。

方案一：全部数据放在一个表中，数据表包含商品编号、月份和销售总额三个数据项，一个月的数据对应一条记录，每个商品1~6月的数据就对应有六条记录，用一维数组存储所有记录，数据结构如下所示：

```
struct ShangPin_1
{
  int BianHao;     //商品编号
  int Month;       //月份
  int Total;       //本月销量总额
};
```

其算法实现如下所示（完整程序可参阅教科书配套学习资源包）：

```
int Query_1(ShangPin_1 a[],int c,int BH,int Yue)
//在有c个元素的顺序表a中查找商品编号为BH的Yue月的销售总额
{
  for(int i=0;i<c;i++)
  {
    if(a[i].BianHao==BH)
      if(a[i].Month==Yue) return a[i].Total;
  }
  return -1;
}
```

方案二：全部数据放在一个表中，数据表包含商品编号和一个数组Total（Total[0..5]分别存放对应1~6月的销售总额），共两个数据项，一个商品的数据对应一条记录，每个商品1~6月的数据就只对应一条记录，用一维数组存储所有记录，数据结构如下所示：

```
struct ShangPin_2
{
  int BianHao;     //商品编号
  int Total[6];    //1~6月的销量总额
};
```

其算法实现如下所示（完整程序可参阅教科书配套学习资源包）：

```
int Query_2(ShangPin_2 a[],int c,int BH,int Yue)
//在有c个元素的顺序表a中查找商品编号为BH的Yue月的销售总额
{
  for(int i=0;i<c;i++)
  {
    if(a[i].BianHao==BH) return a[i].Total[Yue-1];
  }
  return -1;
}
```

程序运行界面如图5–12所示。

图5–12　查询某月销售总额程序界面

方案一中的数据表总共有60条记录，在查找时不但要比较编号，而且要比较月份；方案二中的数据表只有10条记录，在查找时只需比较编号就可以。显然方案二的效率要比方案一的高很多，特别是在数据量大的时候更明显。由此可见，算法与数据结构密切相关。数据结构是算法实现的基础，数据结构直接影响算法的设计和运行效率；算法的操作对象是数据，它必须依赖于具体的数据结构来实现。

由于数据的逻辑结构和存储结构有多种，用户可以根据自己的方案来自行选择和设计，所以解决同一个实际问题也会有多种不同的算法，采用不同的存储结构也将导致算法的差异很大。以第二章第四节一元多项式相加为例，分别用一维数组和链表存储时的算法完全不同，具体请参看第二章第四节的内容。而且，就算是具有相同的逻辑结构和存储结构，算法设计的思想和技巧不同，设计的算法也会大不相同，算法的运行效率也相差很大，如在本章5.3节中学习的冒泡排序算法和快速排序算法。

5.4.2　算法与数据结构的区别

算法是根据需要解决的实际问题，在数据的逻辑结构和存储结构的基础上施加的一种运算（包括数值和非数值的运算）。数据结构关注的是数据的逻辑结构和存储结构，也就是数据表示，即将数据组织起来存储在计算机中；而算法关注的是如何在数据结构的基础上解决实际问题，是对数据运算的描述，也就是数据处理，即设计解决方法和处理数据。算法是编程思想，而数据结构则是这些思想的逻辑基础。

项目实施

各小组根据项目选题及拟订的项目方案，结合5.3节和5.4节所学知识，参照项目范例中用冒泡排序和快速排序算法实现的对参与超市促销活动商品按销量升序排列的完整程序，完成下列任务。

1. 分别用冒泡排序和快速排序的方法编写完整程序，实现数据排序的功能。
2. 体验递归方法，理解算法与数据结构的关系。
3. 参照项目范例的样式，撰写相应的项目成果报告。

成果交流

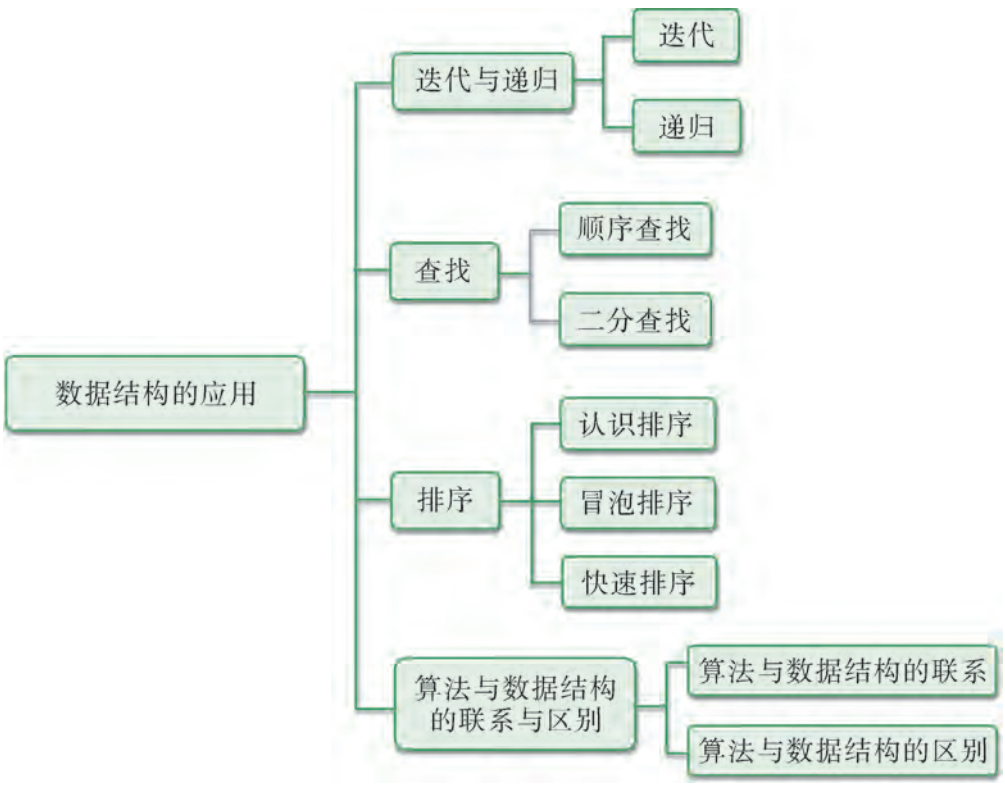
各小组运用数字化学习工具，将所完成的项目成果，在小组或班级上进行展示与交流，共享创造、分享快乐。

活动评价

各小组根据项目选题、拟订的项目方案、实施情况及所形成的项目成果，利用教科书附录2的“项目活动评价表”，开展项目学习活动评价。

本章扼要回顾

同学们通过本章学习，根据“数据结构的应用”知识结构图，扼要回顾、总结、归纳学过的内容，建立自己的知识结构体系。



回顾与总结

本章学业评价

同学们完成下列测试题（更多的测试题可以在教科书的配套学习资源包中查看），并通过“本章扼要回顾”以及本章的项目活动评价，综合评价自己在信息技术知识与技能、解决实际问题的过程与方法，以及相关情感态度与价值观的形成等方面，是否达到了本章的学习目标。

1. 单选题

（1）对有序数组（5，13，19，21，37，56，64，75，88，92，100）进行二分查找，成功查找元素19的查找长度（比较次数）是（　　）次。

A. 1　　B. 2　　C. 3　　D. 4

（2）假定对元素序列（7，3，5，9，1，12，8，15）进行快速排序（升序排序），则进行第一次划分后，得到的左区间中元素的个数为（　　）个。

A. 2　　B. 3　　C. 4　　D. 5

（3）若递归模型为f(1)=1，f(n)=f(n−1)+n (n>1)，则递归出口（递归终结条件）是（　　）。

A. f(1)　　B. f(n)　　C. f(1)=1　　D. f(n)=n

2. 思考题

大数据时代对数据的查找与排序提出了哪些新的要求？谈谈你的应对方法或经验。

3. 情境题

每次考试结束后，老师就要对学生的成绩进行统计和分析，以研究学生的学习情况并给出相应的学习建议。由于学生较多，手工进行统计和分析难以完成，于是老师找来会编写程序的你帮助他完成任务。具体要求如下：

（1）通过键盘输入n个（n>10）学生的考试成绩信息，考试成绩信息由姓名、语文成绩、数学成绩和英语成绩组成。

（2）计算出每个学生的三科总分，并按三科总分的高低输出每个学生在本次考试中的排名，分数相同的为并列名次。请用冒泡排序和快速排序算法分别编写程序实现。

（3）能够查询某个学生的三科成绩、总分和排名，并输出查询的结果。请用顺序查找和二分查找算法分别编写程序实现。

附录1　部分术语、缩略语中英文对照表

ADT（Abstract Data Type）	抽象数据类型（4）
Algorithm	算法（1）
Array	数组（2）
BDS（BeiDou Navigation Satellite System）	北斗卫星导航系统（1）
Bottom	栈底（3）
Bubble Sort	冒泡排序（5）
Data	数据（1）
Data Element	数据元素（1）
Data Item	数据项（1）
Data Structure	数据结构（1）
Fibonacci	斐波那契数列（5）
FIFO（First In First Out）	先进先出（3）
Key	关键字（5）
LIFO（Last In First Out）	后进先出（3）
Linear List	线性表（3）
Linked Lists	链表（2）
Logical Structure	逻辑结构（1）
Number	数字（1）
Numerical Value	数值（1）
Pointer	指针（2）
Pointer Variable	指针变量（2）
Pop	出栈（3）
Program	程序（1）
Push	进栈（3）
Queue	队列（3）
Quick Sort	快速排序（5）
Stack	栈（3）
Storage Structure	存储结构（1）
String	字符串（3）
Top	栈顶（3）

附录2　项目活动评价表

以培养信息素养为目标，以知识体系为载体，以项目学习活动过程与评价为途径，促进同学们的信息技术学科核心素养达成。

项目学习主题：_____

项目学习过程	学科核心素养达成	一级指标	二级指标	评价结果	支撑材料
选定项目	从现实世界中选择明确的项目主题，形成对信息的敏感度和信息价值的判断力。 分析项目目标与可行性。	项目选题	从现实世界选择项目主题的能力。 化抽象概念为现实问题的能力。 对信息的敏感度和价值的判断力。	□优秀 □良好 □中等 □仍需努力	
		项目分析	分析项目目标的能力。 分析项目可行性的能力。 从现实世界发现项目素材的能力。	□优秀 □良好 □中等 □仍需努力	
规划设计	组建团队与明确项目任务，体现正确的信息社会责任意识。 规划项目与交流方案。	项目规划	组建团队与明确项目任务的能力。 规划项目学习工具与方法的能力。 预期项目成果的能力。	□优秀 □良好 □中等 □仍需努力	
		方案交流	交流项目方案的能力。 完善项目方案的能力。 体现正确的信息社会责任意识。	□优秀 □良好 □中等 □仍需努力	
活动探究	通过团队合作，围绕项目进行自主、协作学习。 开展探究活动，提升信息获取、处理与应用、创新能力。	团队合作	自主学习能力。 分工与协作能力。 交流与沟通能力。	□优秀 □良好 □中等 □仍需努力	
		探究活动	信息获取与处理能力。 探究与联想能力。 实践与创新能力。	□优秀 □良好 □中等 □仍需努力	

（续表）

项目学习过程	学科核心素养达成	一级指标	二级指标	评价结果	支撑材料
项目实施	针对给定的任务进行分解，明确需要解决的关键问题，并采用计算机科学领域的思想方法，在形成问题解决方案的过程中产生一系列思维活动。 完成方案中预设的目标。	工具方法	采用计算机领域的思想方法能力。 使用数字化工具与资源能力。 数字化学习能力。	□优秀 □良好 □中等 □仍需努力	
		实施方案	针对给定的任务进行分解。 明确需要解决的关键问题。 完成方案中预设的目标。	□优秀 □良好 □中等 □仍需努力	
项目成果交流与评价	与团队成员共享创造与分享快乐，提升批判性思维能力与信息社会责任感。 评价项目目标与成果质量效果。	成果交流	清晰表达项目主题与过程。 与团队成员共享创造与分享快乐。 提升批判性思维能力与信息社会责任感。	□优秀 □良好 □中等 □仍需努力	
		项目评价	运用新知识与技能实现项目目标。 项目成果的可视化表达质量。 项目成果解决现实问题效果。	□优秀 □良好 □中等 □仍需努力	
综合评价	□优秀　□良好　□中等　□仍需努力				

注：1. 评价得分90～100分为优秀（A）；75～89分为良好（B）；60～74分为中等（C）；60分以下为仍需努力（D）。

2. 综合得分=互评×30%+自评×30%+教师评×40%。

绿色印刷产品

批准文号：粤发改价格［2017］434号　　举报电话：12358

ISBN 978-7-5548-3027-7

定价：10.59元