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

人民教育出版社 中国地图出版社

普 通 高 中 教 科 书

信息技术

选择性必修 1

数据与数据结构

人民教育出版社课程教材研究所信息技术课程教材研究开发中心
中国地图出版社教材出版分社 编著

总主编 祝智庭 樊 磊

人民教育出版社 中国地图出版社
·北京·

总 主 编：祝智庭　樊　磊
副总主编：高淑印　郭　芳　李　锋

本册主编：刘光然　罗　梅
编写人员：张茹桂　刘丽颖　张　磊　杨海丽　兰大鹏

责任编辑：兰大鹏　黄应会
美术编辑：李　媛　徐海燕

普通高中教科书　信息技术　选择性必修 1　数据与数据结构
人民教育出版社课程教材研究所信息技术课程教材研究开发中心
中国地图出版社教材出版分社　编著

出　　版　人民教育出版社
（北京市海淀区中关村南大街 17 号院 1 号楼　邮编：100081）
中国地图出版社
（北京市西城区白纸坊西街 3 号　邮编：100054）
网　　址　http://www.pep.com.cn
http://www.ditu.cn

前言

同学们，欢迎探索信息技术这个神奇而充满魅力的世界。

在以往的学习、生活中，你们已经积累了许多信息技术方面的知识与技能，例如：在网上查阅资料，用手机与亲朋好友保持联系，使用移动终端、自动柜员机等设备……你们知道这些应用中都包含哪些关键技术，涉及哪些领域吗？怎样有效地利用这些技术帮助我们培养信息意识，提升计算思维，进而通过数字化学习与创新，承担起信息社会责任呢？即将开始的这门课程，会帮助你们对信息技术有更多的认识和思考，获得更丰富的体验和感受。

为了很好地掌握信息技术，希望同学们按以下三个要求去努力。

1. 认真阅读教科书，理解基本概念和原理。信息技术发展非常迅猛，各类信息系统不断涌现，但信息系统的基础和运行体系相对稳定，离不开算法的设计及对数据的利用。只有夯实基础，才能学好本领，跟上时代发展的步伐。

2. 敢于动手，勤于实践。信息技术是一门实践性较强的课程。实践能帮助同学们熟练操作技能，进一步掌握知识。因此，要认真阅读理解每章的主题学习项目，并逐步完成“实践活动”“思考活动”“技术支持”“阅读拓展”等栏目的学习内容，在实践中获取知识和经验。

3. 要有积极探究、锲而不舍的精神。掌握信息技术的知识与技能需要一个过程，不可能一蹴而就。信息技术学科内容非常丰富，各知识点之间联系密切，但名词术语多，有可能令人感到繁杂，甚至产生畏难情绪。学习新知识，首先要知其然，接着通过不断学习，积极动手操作，大胆请教，加深对知识的理解，然后才能知其所以然，在不断的探索过程中取得进步。

本书中涉及的配套资源，可在教科书配套教学资源平台的信息技术栏目中获得。让我们开始一段信息技术新旅程，成长为信息社会中合格的中国公民！

目录

第 1 章
走进数据世界

举世闻名的都江堰水利工程，为战国时期蜀郡郡守李冰率领蜀地各族人民修建而成。此后，成都平原水旱从人，不知饥馑，被誉为“天府之国”。都江堰水利工程的成功在于，李冰经过长期观察和资料积累，充分掌握了地理、地势、河流及其泥沙等规律和情况，即对诸多因素产生的“数据”进行了采集、分析和处理。由此可见，掌握数据并有效利用数据对形势进行判断，是能够做出正确决策的前提条件，这也是适应未来社会的必备能力。

近年来，互联网、物联网、云计算等技术的快速发展使得全球数据量出现爆炸式增长。对数据价值的深度挖掘及应用，引发了以数据为核心的“大数据”时代的深刻变革，数据驱动发展已成为大势所趋。党的十九大报告提出：加快建设制造强国，加快发展先进制造业，推动互联网、大数据、人工智能和实体经济深度融合。大数据已被视为 21 世纪的石油和金矿。数据成为最宝贵的生产要素之一，既是商家争相研究的秘密宝藏，更是提升国家竞争力的关键资源。

本章将通过主题学习项目“析说身边数据”走进数据世界，了解数据的作用，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。通过剖析实例，认识数据作为新的原材料、生产资料和基础设施的价值与意义，更深入地体会大数据对国家、社会和个人的影响。

1 主题学习项目：析说身边数据

项目目标

本章围绕主题“析说身边数据”开展项目学习，从比较感兴趣的身边事例入手，自拟主题，并结合主题有目的地收集、整理和分析数据，认识数据的价值与意义，感受数据对生活的影响，并以多媒体作品的方式进行班内交流。

1. 围绕项目主题，收集并整理数据，利用思维导图来设计项目方案，分析数据与社会相关领域的关系。

2. 通过分析项目涉及的实际问题，认识并理解数据作为新的原材料、生产资料和基础设施在其中的价值和意义。

项目准备

为完成项目，需做如下准备。

- 全班同学按“组间同质、组内异质”的方式分成若干组，每组4 ~ 5人，明确项目研究主题和人员分工。
- 商讨项目主题及其涵盖内容，采取适当方式收集相关数据和资料。
- 收集并整理素材，选用一款多媒体编辑工具，为制作多媒体作品做准备。

项目过程

在学习本章内容的同时开展项目活动。为了保证本项目的顺利完成，要在以下各阶段检查项目的进度。

制订方案，获取数据 ①

完成分组并开展小组讨论。围绕生活中的实际问题，探讨其中蕴含的数据和信息，确定项目主题，收集项目所需的数据资料。P8

处理数据，完成作品 ②

探究数据的价值，通过整理与分析数据，认识数据对生活的影响，筛选与主题相关的数据资料，制作、检查多媒体作品并在全班交流。P15

项目总结

完成本章学习，形成一份项目学习过程记录表，并提交本组的项目成果（包括可视化的数据图表和多媒体作品）。全班各小组之间进行交流展示。

1.1
深入认识数据

学习目标 ▶▶▶

- 理解数据的概念，能够分析数据与社会各领域的重要关系。
- 了解数据安全的重要性，能够自觉遵守信息社会的伦理道德和法律法规。

体验探索

体验共享数据

如今，人们的出行经常需要使用导航定位服务，共享位置服务在获取用户实时位置信息的基础上，利用移动互联网等新技术手段实现数据资源的实时传输和优化显示，为人们的出行带来更多的便利。

小明和小华约定分别从各自位置出发去参观某新建的博物馆，但是小明对路线并不熟悉，于是，他和小华利用共享位置服务，如图1.1.1所示，最终顺利到达目的地。小明和小华在共享位置的过程中，既获取了数据，同时也产生了数据。

图1.1.1　共享位置示意图

思考：

1. 共享位置服务是如何获取小明和小华的位置数据信息的？
2. 数据在共享服务中的作用是什么？

1.1.1 数据

如果有人问“珠穆朗玛峰有多高？”若回答“很高”“非常高”“最高”，则只能得到一个模糊的答案，因为每个人对“很”“非常”有不同的理解，“最”也是相对的；若回答“海拔高度8 844.43 m”，就可以通过数据比较，确认珠穆朗玛峰是世界最高峰。此处的数据起到了对事物进行量化、精确化的作用。数据可以用于描述事物以及人类的生产活动，特别是以交换为目的的商品生产活动，如货币、度量衡等都离不开数据。数据不仅可以描述世界，其广泛应用还极大地推动了人类文明的进步。

思考活动

竺可桢与气象数据研究

竺可桢（1890—1974），中国气象学家、地理学家、科学史家和教育家，也是中国近代地理学和气象学的奠基者（图1.1.2）。

图1.1.2　工作中的竺可桢

竺可桢在历史气候变迁领域的研究蜚声国际。他重视物候和天气的观察记录。自留学回国的第二天至1974年逝世的前一天，他坚持每天观察并记录物候和天气，同时广泛收集历史物候资料，曾在国内建立了拥有四十多个气象站和一百多个雨量测量站的中国气象观测网。基于这些数据资料，竺可桢著有《东南季风与中国之雨量》《中国近五千年来气候变迁的初步研究》等学术论著，为建立和发展中国现代气象事业做出了重大贡献。

思考：

1. 气候研究需要哪些气象数据？
2. 如果要研究当地的气候，你会用什么方式收集数据？
3. 竺可桢五十余年如一日记录数据的行为对我们有什么启发？

数字与数值

在印度有一个古老的传说：有个国王打算奖赏他的宰相。国王问他的宰相想要什么，宰相回答说：“陛下，请您在这张棋盘（图1.1.3）的第1个小格里，赏给我1粒麦子，在第2个小格放2粒，在第3个小格放4粒，以后每一个小格都比前一个小格多放一倍的麦粒。请您像这样摆满棋盘上所有64个小格，然后把这些麦粒都赏给您的仆人吧！”国王觉得这要求太容易满足了，就下令给他这些麦粒。很快，国王就发现：就是把全印度的麦子都拿来，也满足不了这位宰相的要求。那么，宰相要求得到的麦粒到底

有多少呢？

假设麦粒数为s，根据已学习过的数学知识，可以得到这样一个公式：

$$s=2^0+2^1+2^2+2^3+\cdots+2^{63}$$

利用Python语言中的循环结构可以解决这个数学问题，程序如下：

```
import math
s = 0
for i in range(0, 64):
    s = s + math.pow(2, i)
print(s)
```

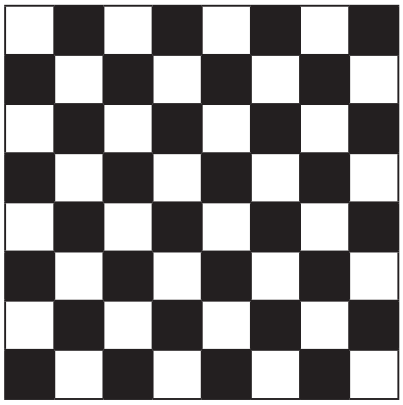


图1.1.3　棋盘示意图

程序运行结果：

```
1.8446744073709552e+19
```

可以看出，大约需要$1.844\ 674\ 407\ 370\ 955\ 2\times10^{19}$粒麦子。每千克麦子按3万粒计算，国王应该赏赐宰相多少千克麦子呢？

数字是用来记数的符号，如数字0，1，2，3，4，5，6，7，8，9。数值则是把数字写在不同的数位上，用来表示一个量的多少。很多情况下，要在数值尾部加上单位，才能表示具体的含义。例如，身高175 cm、汽油40 L等。

数据及其应用

计算机问世前，数据多指用于统计的数字或数值。计算机问世初期，数据主要用于数值计算，如科学计算等。随着信息技术的发展，数据逐步应用于数据处理，如工资、库存管理等。从此，信息技术有了数值计算和数据处理的差别。

随着计算机、互联网和移动互联网的普及与发展，计算机能够处理的数据类型更加丰富，现代意义的“数据”已不再局限于数字或数值，而是有了更多的内涵和更广的外延，字符、数字和各种数学符号、图形、图像、音频、视频和动画等可识别的记号或符号都称为数据。

由此可见，数据是对客观事物的符号化表示。在计算机科学中，数据是所有能输入到计算机中并被计算机程序处理的符号的总称。

现实社会中，人们的生活、工作和学习等各方面都蕴藏着数据。例如，购物时扫描的二维码；工厂中产品质量管控的数据记录；体育赛事的视频资料及各项成绩；健康检查时心、肺、血液等各项指标；智能电子设备记录的健康指标等。可以说，数据及其应用已融入我们的生活，成为我们生活的一部分，并随时随地伴随和影响着我们。

1.1.2　大数据

大数据指在社会生产、生活和管理服务过程中形成的，依托现代信息技术采集、传输并汇总的，超过传统数据系统处理能力的数据。通常而言，大数据具有巨量性、多样

性、迅变性和价值性的特征，通过整合共享、交叉复用、提取分析可获取新的数据，创造新的价值。

思考活动

数据解读生活

在互联网上常有一些网站以海量数据为基础，对社会某个领域或热门话题进行数据统计分析，并以可视化的形式呈现给用户。例如，图1.1.4是某网站开展的全民健身调查统计结果之一，图中数据简洁明了地展示了人们喜欢健身的主要原因。

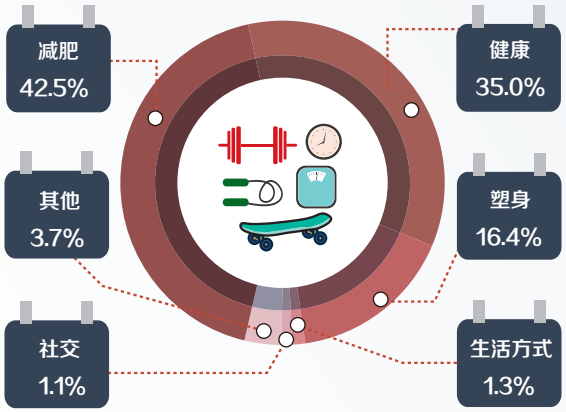


图1.1.4　健身原因调查结果

思考：

利用互联网查找类似的网站，看看它们关注了哪些社会领域，又是如何用数据解读生活的？

生活中的大数据

大数据与传统数据的区别主要有：规模不同，传统数据以MB（1 MB=1 024 KB）为基本单位，大数据常以GB（1 GB=1 024 MB）甚至是TB（1 TB=1 024 GB）、PB（1 PB=1 024 TB）为基本处理单位；数据类型不同，传统数据指结构化的数据（如二维表中的数据），大数据包括结构化、非结构化数据（如文档、图像、视频等）和半结构化数据；采集方法不同，传统数据多来源于抽样数据，可通过调查、访谈等方式获取，大数据采集多依靠电子设备自动获取，如街头的摄像头采集视频资料；处理方法不同，传统数据处理会涉及排序、统计等操作，大数据会涉及预处理、分析与挖掘技术等，如对图像和视频进行编辑操作；处理工具不同，处理传统数据仅使用一种或少数几种工具即可，处理大数据则需要使用多种不同处理工具共同应对。

如今，智能设备已成为人们日常生活的一部分，它们每时每刻都在创造着数据。例如，智能系统可以记录人们每次点击屏幕或鼠标的操作轨迹；搜索引擎会记录人们的历史搜索关键字；智能手机可以记录人们的位置和速度；智能家居设施能记录人们的生活习惯；购物平台能记录人们的购物历史……同时，各种各样的传感器随时测量和传递着

有关对象的位置、运动、震动、空气温度、湿度乃至化学物质含量变化的数据，从而产生了海量数据。此外，互联网更是一个广袤的数据海洋，它既有百科全书、电影、音乐等各种专业数据库，也有权威机构发布的门类广泛的统计数据……各种数据像涓涓细流，汇聚到一起，构成大数据。数据已经不仅是计算工具所处理的对象或信息的载体，它也成为人们获得信息、推动信息社会发展的一项动力。

阅读拓展

大数据与无人驾驶汽车

无人驾驶汽车（图1.1.5）是大数据的典型应用，它主要依靠车内以计算机系统为主的智能驾驶仪来实现无人驾驶的目标。与普通汽车不同的是，无人驾驶汽车需要借助全球卫星导航系统（如北斗卫星导航系统等）来规划路径，以及激光雷达、摄像头、红外相机和一系列传感器等感应设备。通过这些设备，无人驾驶汽车不断地收集地理位置、路面情况、与前后车辆的精确距离、车流移动速度、道路两旁出现的交通标志和前方交通信号等数据。然后与系统已有数据进行对比和分析，快速识别车辆的方位和周边环境。根据分析结果，系统在极短时间内判断出是否需要加速、减速、换道或拐弯等，并执行相应动作。

图1.1.5　无人驾驶汽车

数据安全与隐私保护

互联网技术的飞速发展使数据的获取、传输和共享更加便利，同时，数据隐私问题也越来越暴露在公众面前。2018年，国外某社交网站约5 000万名用户数据遭到泄露，引发了人们对数据安全与隐私保护的关注。

事实上，人们在互联网上的各种行为都留有痕迹，例如在线学习、网上购物和网络社交等。因此，数据分析和处理的过程中，对用户数据的隐私保护十分重要。

数据安全与隐私保护并非人们认为的那样仅是技术问题，它更是社会问题，需要社会各界共同解决。

对于国家而言，数据安全关乎国家安全，不容忽视。

2017年6月1日，《中华人民共和国网络安全法》正式施行。它不仅是打击计算机犯罪的法律依据，更是保护知识产权、数据安全、商业机密及个人隐私的有效武器。

实践活动

移动应用与数据安全

在教师的指导下，从手机中选择一款自己经常使用的应用程序，分析此程序记录了哪些隐私数据，是否存在安全隐患，应该如何保护个人的隐私？

项目实施

制订项目方案，获取数据

一、项目活动

依据项目目标从项目分工、项目内容、项目过程和项目检查等方面制订项目方案。参考表 1.1.1 的样式，绘制并填写相关表格，记录项目实施过程。

表1.1.1　项目信息表

项目主题				
人员及分工				
进度1	时间：			
进度2	时间：			
备注				

1. 围绕项目主题，分小组讨论该主题涉及哪些大数据。项目主题可参考：大数据与体育；大数据与智慧城市；基于大数据研究个人职业规划……从某学科角度出发，研究数据在其中的意义等。

2. 小组成员分工，收集并整理相关资料、制作排版和处理图像等。

3. 根据小组项目主题，选择合适的方法，有针对性地采集数据资料。在采用这些数据之前，需对数据的来源尽可能地进行多方查证，然后分门别类地整理、存档。

二、项目检查

各小组以思维导图的形式绘制完成本项目主题所需的方案，例如，项目涉及的若干分支，以及其中明确的分工和任务，并备份整理后的数据资料。

练习提升

1. 结合学习经历，谈谈数据对我们学习某一学科有哪些帮助。

2. 结合实际生活体验，试写出三种以上数据采集方法。

1.2
数据的价值

学习目标 ▶▶▶

- 能够举例说明数据作为新的原材料、生产资料和基础设施的重要性。
- 了解数据挖掘在发现数据价值过程中的作用。

体验探索

C919大型客机翱翔蓝天

2017年5月5日，承载着中国航空人数十年梦想的C919大型客机（图1.2.1）在全世界的共同见证下试飞成功，这标志着中国科技已走上加速发展的轨道。

图1.2.1　C919大型客机

国产大飞机首飞成功的背后有着大数据做出的巨大贡献。例如，科研团队利用当时全球计算速度最快的“天河二号”超级计算机，持续运算数天，共产生了十几亿个数据包，在大数据分析处理后才最终得到客机最优的空气动力参数模型。

在首飞过程中，技术团队采集的飞机各种参数的数据包多达5亿个，用以分析飞机飞行过程中发生的细微变化，如发动机温度、机翼压力、机翼扭曲率等，从而改善飞机性能，保证飞行安全。

思考：

1. 大数据在航空领域的应用还有哪些？
2. 大数据是机场运营中的基础设施吗？为什么？

1.2.1 数据作为新的原材料的价值

原材料是生产某种产品的基本原料和材料，如自然形态下的原木、矿砂、原油，以及原粮和原棉等，它们被适当加工后就会成为人们需要的产品。

蒸汽机的出现与改良引发了第一次工业革命，它把人们从繁重的体力劳动中解放出来，此时，煤炭成为重要资源；以电力的广泛应用为标志的第二次工业革命，极大地推动了社会生产力的发展，此时，电力、石油成为重要资源；以原子能、电子计算机与互联网、空间技术和生物工程的发明和应用为主要标志的第三次工业革命，很大程度上得益于计算机和互联网带来的自动化，此时，数据成为重要资源，计算成为生产力，人类真正开始进入数据时代。

思考活动

大数据成为战略资源的缘由

信息技术的出现为数据处理提供了自动化的方法和手段，推动数据成为继物质、能源之后的又一战略资源。大数据是信息化发展到一定阶段的必然产物，这主要是由信息技术的不断低成本化，以及互联网及其延伸所带来的信息技术应用引发的。

思考：

上网搜集近几年来信息技术的发展变化的资料，谈谈对“低成本化”的理解。

现代社会中，数据与人们的生活息息相关。有人把数据比作石油和金矿。数据被合理加工后能生成新的产品，产生新的价值，因此，数据成了新的原材料。

■ 数据与机器学习

近年来，在人机对战围棋赛中战无不胜的阿尔法围棋（AlphaGo）是一个典型的基于大数据的机器学习产物，如图1.2.2所示。它的棋力来自于几十万张人类对弈棋谱以及上千万次的自我对弈。简单地说，人类把一堆棋谱提供给阿尔法围棋，它通过深度学习具备了战胜世界顶尖棋手的能力。人类不需要通过编程来教机器如何工作，而是通过给出一个学习框架，“告诉”机器如何根据自身当前的设置以及提取环境的反馈去进一步学习，从而更新参数以达到一个更好的工作表现。

图1.2.2　阿尔法围棋

大数据极大地促进了人工智能的发展。基于机器学习的人工智能可以在图像分类、语音识别和语言翻译等领域发挥重大作用，创造巨大价值，这意味着处理复杂的任务会变得更容易、更快速、更加个性化。

■ 数据与政府管理

2016年，大数据作为国家战略被写入“十三五”规划纲要，很多地区（如广东省、贵州省等）都设立了大数据发展管理机构。政府部门应用大数据管理的重要目的是预警，即政府利用大数据来进行社会治理，为当地百姓提供更好的服务。此外，在更高的层面上，政府部门的数据统筹有助于智慧城市的建设，即政府利用先进的信息和通信技术手段对城市运行的各项数据进行监测、分析和整合，从而为民生、环保、交通和工商业等领域的活动提供更加智能、快捷的服务，更好地满足人们对美好生活的需求。

1.2.2 数据作为新的生产资料的价值

生产资料是劳动者从事生产时所需要使用的工具或物质资源，包括劳动资料（如土地、厂房、工人使用的机器和农民使用的农具等）和劳动对象（如自然界原有的地下矿石或加工过的原材料）两大类。将作为原材料的数据经过一定处理转换为某一数据产品的生产资料时，数据就成了新的生产资料。

人类可利用的资源分为可再生资源和不可再生资源两类。风能作为一种安全、清洁的可再生资源，被广泛应用于电力行业，随着智能电网的普及，风力发电的前景非常喜人。与风能类似，数据资源也是可再生的，这源于其价值的多样化。通过创新数据利用方式，可以创造持续的数据价值。

■ 数据与电子商务

每天，来自世界各地的订单通过互联网汇聚到电子商务平台上，这些数据成为电商贸易的原材料。

如今，电商平台中通常都有“猜你喜欢”的功能。当用户登录电商平台时，经常会看到诸如“根据您的浏览历史记录精心为您推荐”“购买此商品的顾客同时也购买了某商品”和“浏览了该商品的顾客最终购买了某商品”等提示。这些其实是电商平台将原数据进行加工处理后再生成的数据。对于电商平台运营公司来说，这些数据成了他们用于开发新价值的生产资料。当人们从推荐商品中选择购买心仪的商品时，又给电商平台提供了新的数据价值，而这在传统商业中是难以实现的。

一般来说，电商的“猜你喜欢”都是在一定的算法基础上，搭建一套符合自身特点的规则库：该算法会同时考虑其他顾客的选择和行为（相关数据），在此基础上搭建商品相似性矩阵和用户相似性矩阵，并找出最相似的顾客或最关联的商品，从而完成商品的推荐。

对于商家来说，从庞杂的数据中挖掘、分析出用户的行为习惯和喜好，找出更符合用户“口味”的产品和服务，并结合用户需求对产品和服务进行有针对性的调整和优化，这就是大数据的价值。

一般地，通过相应的算法对大量的数据进行自动分析，揭示数据当中隐藏的规律和趋势，发现潜在的数据价值，为决策者提供参考，这就是数据挖掘。

■ 数据与天气预报

现在，人们每天的生活、学习和工作一般都离不开各类网站和移动应用程序。这些网站和移动应用程序提供的信息源于大量或海量的数据分析。例如，某款天气预报应用程序（其界面如图1.2.3所示），它每天产生的日志量大约在2 TB，其日志分析业务采用的是阿里云大数据平台，主要的日志分析场景是天气查询业务和广告业务。整个过程中数据量庞大，计算复杂，这对云平台的大数据能力、生态完整性和开放性均有很高要求。

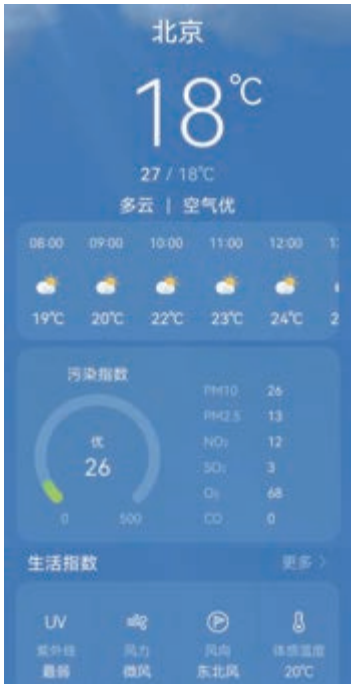


图1.2.3　某款天气预报应用程序界面

天气预报不仅为我们提供短期的气象预报，同时通过积累这些数据，作为气候预测的坚实基础，为包括农业在内的各行业发展保驾护航。

各行各业都在积累大量的数据，通过对这些数据的科学分析和不断挖掘，这些数据必将产生更大的、不可估量的价值。

1.2.3　数据作为新的基础设施的价值

提起基础设施，人们能够想到公路、铁路、桥梁、机场、港口、通信设施和水利枢纽等。它们是人们工作和生活共同的物质基础，是城市主体设施正常运行的保障。

前文提到大数据是21世纪的石油和金矿，这些石油和金矿贮藏在哪儿？它们其实贮藏在数据中心。数据中心解决了大数据汇聚的问题，它相当于一个国家的高速公路、高速铁路和机场，是新的基础设施。

存储系统是数据中心的核心基础架构，是数据中心数据访问的最终承载体。数据中心（图1.2.4）利用强大的计算和存储能力，为各行各业提供多样化的数据服务。


图1.2.4　数据中心

■ 数据与智能交通

图 1.2.5 是某市某时刻的交通实况图。图中红色路线表示拥堵，黄色为通行缓慢，绿色为畅通。司机通过这些提示，可以合理选择路线，节省时间。交通部门利用交通大数据，一方面能够实现即时信号灯调度，提高已有线路的通行能力（科学地安排信号灯是一个复杂的系统工程，必须利用大数据计算平台才能计算出较为合理的方案）；另一方面可以利用大数据传感器的数据来了解车辆通行密度，合理规划道路。由此可见，建设智慧城市需要深入、持续地做好大数据的基础设施建设。

图 1.2.5　某市某时刻路况图

■ 数据与中国农业

中国地大物博，农业数据涉及的领域广、环节多，是跨行业、跨专业的数据集合，农业数据具有复杂性、分散性、获取难的特征。粮食统计是一件关系到国计民生的大事。我国的粮食统计最早是由省、市、县、乡、村逐级统计的。近年来，利用遥感卫星采集图像，通过图像识别将中国所有的耕地标识、计算出来，然后将这些耕地网格化，并对每个网格的耕地抽样进行跟踪、调查和统计，最后按照统计学的原理，计算（或估算）出全国的粮食数据。这种做法就是以大数据为资源，采用大数据建模的方法，打破传统的流程和组织，直接获得结果。

阅读拓展

超级计算机与大数据应用

2017 年 11 月，在全球超级计算大会上，由中国国家并行计算机工程技术研究中心研制、国家超级计算无锡中心运营，基于国产众核处理器的“神威·太湖之光”超级计算机（图 1.2.6）以每秒 12.5 亿亿次的峰值计算能力和每秒 9.3 亿亿次的持续计算能力，再次斩获世界超级计算机排名榜单第一名。本次夺冠实现了“神威·太湖之光”超级计算机的四连冠，同时这也是国产超级计算机系统在世界超级计算机比拼中取得的十连冠。

图 1.2.6　“神威·太湖之光”超级计算机

在这次大会上，基于“神威·太湖之光”系统的两项应用“全球气候模式的高性能模拟”“非线性大地震模拟”入围“戈登·贝尔”奖提名。其中“非线性大地震模拟”一举拿下了“戈登·贝尔”奖。这是继2016年同样基于“神威·太湖之光”系统的项目“千万核可扩展全球大气动力学全隐式模拟”摘得“戈登·贝尔”奖之后，我国蝉联该项大奖！

设立于1987年的“戈登·贝尔”奖，被誉为“超级计算应用领域的诺贝尔奖”。2016年之前，从未有过中国应用团队入围并获奖。

“神威·太湖之光”具有强大的数据存储、检索、计算和分析能力，有着大数据应用的天然优势，在大数据应用方面大有可为。目前，它已在天气气候、航空航天、先进制造、生物医药、新材料和新能源等多领域得到广泛应用。

实践活动

智慧医疗

智慧医疗指在诊断、治疗、康复、支付和卫生管理等各环节，基于物联网、云计算等高科技技术，建设医疗信息完整、跨服务部门、以病人为中心的医疗信息管理和服务体系，实现医疗信息互联、共享协作、临床创新和科学诊断等功能，致力于构建一个“以病人为中心”的医疗服务体系，提供高质量的个人医疗服务体验，如可挂号、充值、查询和打印的自助终端，以及智能导诊，如图1.2.7所示。

图1.2.7　某医院的智能导诊机器人

1. 通过互联网查找更多关于智慧医疗的资料，了解智慧医疗发展方向，以及我国智慧医疗行业的总体现状。

2. 在班级内交流各自生活中曾体验过、见到或听到过的关于智慧医疗的事例。

项目实施

处理项目数据，完成多媒体作品

一、项目活动

1. 运用以往所学的数据分析方法，研究、利用某种适当的数据可视化方法呈现相关数据。

2. 制作并完善项目报告。

3. 展示与评价。小组代表展示本组主题作品和项目报告，阐述项目内涵；总结项目完成过程中个人对小组的贡献和小组协作情况等；完成组间互评。

二、项目检查

如实填写表1.2.1，客观评价主题作品和项目完成情况。

表1.2.1　评价量表（参考）

评价标准	超出期望的	符合期望的	有进步空间的
主题突出，数据充分			
风格统一，画面美观			
演讲精彩，表达完美			
小组合作情况			
其他（如引用资料来源可靠等）			
作品最大亮点			

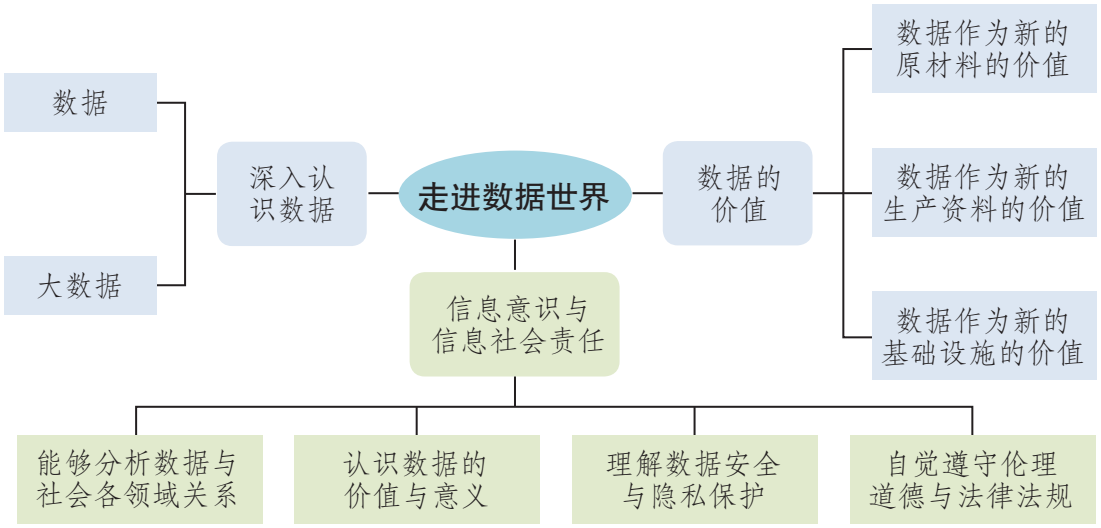
练习提升

1. 选择一款移动应用程序，分别从生产者和消费者的角度，分析数据的价值与意义。

2. 查阅大数据与人工智能的相关资料，思考两者的结合将对未来哪些领域带来挑战，试说明理由。

总结 评价

1. 下图展示了本章的核心概念与关键能力，请同学们对照图中的内容进行总结。



2. 根据自己的掌握情况填写下表。

学习内容	掌握程度
数字、数值和数据的基本含义	□不了解 □了解 □理解
大数据的价值与意义	□不了解 □了解 □理解
数据安全的重要性	□不了解 □了解 □理解
数据在社会某几个领域的作用	□不了解 □了解 □理解

第 2 章
初识数据结构

数据作为描述事物的符号记录，它不仅可以是数字，还可以是文字、字符、图形、图像、音频和视频等。中国汉字文化博大精深，一个字可能有多个含义，几个字的排列顺序不同，就可能会组成含义不同的词句。例如，用“读”“书”“好”三个字就可以组成“读书好”“读好书”“书好读”等。从数据结构角度来看，汉字“读”“书”“好”都是数据，其排列顺序就是结构。

计算机科学是一门研究信息表示和处理的科学，而信息的表示和组织直接关系到信息处理的效率。数据结构研究的是信息在计算机中的组织和存储方式，程序设计依赖于一定的数据结构。因此，对数据及其结构的研究十分必要。

本章将通过主题学习项目“管理个人书目”来理解数据结构和抽象数据类型的基本概念，认识数据结构在解决问题过程中的重要作用，以及抽象数据类型对数据处理的重要性。结合生活实际，通过问题分析与程序实现，理解数组、链表等概念，并能够根据需求选择合适的存储方式。

02 主题学习项目：管理个人书目

项目目标

本章围绕主题“管理个人书目”开展项目学习，以小组为单位，在阅读活动中发现并体验数据结构实例，并完成项目程序“管理个人书目”。

1. 感受数据结构中所蕴含的变与不变的思想，认识数据结构在解决问题过程中的重要作用。

2. 理解数组、链表等基本数据结构的概念。

3. 提升使用Python语言编程的能力，为后面的学习打下扎实基础。

项目准备

为完成项目，需做如下准备。

- 全班同学按“组间同质、组内异质”的方式分成若干组，每组3 ~ 5人，明确项目目标和人员分工。
- 根据教学资源平台上的学习材料进行数据结构类型特征的初步学习，了解数据结构常见的基本类型。
- 搭建项目所需的Python语言环境，复习Python基础知识。

项目过程

在学习本章内容的同时开展项目活动。为了保证本项目的顺利完成，要在以下各阶段检查项目的进度。

发现阅读活动中的数据结构 1

围绕阅读活动中存在的数据结构实例，如图书的分类等，探究它们和数据结构类型的对应关系。 P28

编写程序管理读书单 2

小组合作完成“管理个人书目”程序设计，整理程序及文档，形成项目报告，小组之间进行展示、交流。 P47

项目总结

完成本章学习，提交本组的项目成果，包括阅读活动中数据结构实例的思维导图、“管理个人书目”的程序设计和项目学习过程记录表。全班各小组之间进行交流展示。

2.1
数据结构与数据类型

学习目标 ▶▶▶

- 理解数据结构的概念，能够用数据结构表达数据的逻辑关系，认识数据结构在解决问题过程中的重要作用。
- 理解抽象数据类型的概念，能够对特定生活情境中的关系进行抽象，认识抽象数据类型对数据处理的重要性。

体验探索

田忌赛马与数据结构

田忌赛马的典故出自《史记·孙子吴起列传》。

忌数与齐诸公子驰逐重射。孙子见其马足不甚相远，马有上、中、下辈。于是孙子谓田忌曰："君弟重射，臣能令君胜。"田忌信然之，与王及诸公子逐射千金。及临质，孙子曰："今以君之下驷与彼上驷，取君上驷与彼中驷，取君中驷与彼下驷。"既驰三辈毕，而田忌一不胜而再胜，卒得王千金。

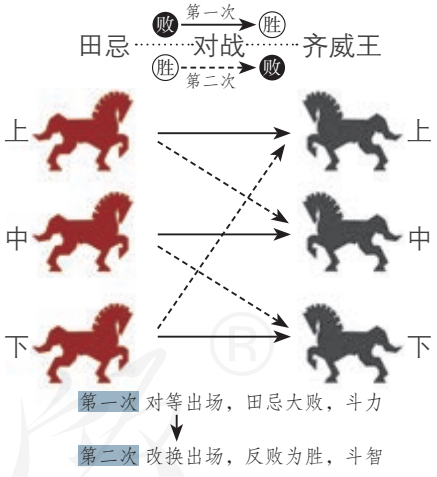


图2.1.1　田忌赛马示意图

田忌采用孙膑的计策，没有更换参赛马匹，只调整了马匹的参赛顺序，就转败为胜，如图2.1.1所示。如果把马看成"数据"，把三匹马的出场顺序看成数据之间的"关系"，那么，相同的数据不同的关系，可得到不同的"数据结构"，而不同的"数据结构"对问题解决会产生重要影响。

思考：

从上述典故可以看出，使用相同的数据，采取不同的结构，却产生了不同的结果。生活中还有哪些类似的实例？

2.1.1 数据结构

随着计算机应用的日益广泛，计算机程序的规模越来越大，程序结构越来越复杂。计算机在处理数据时不是只处理少数几个无内在联系的数据，而是处理多个数据，并且这些数据之间是有一定内在联系的。此时，定义数据及数据之间的关系就显得十分重要。

思考活动

奇妙的文字组合

中华文化源远流长，其中有非常多的成语典故闪耀着先贤的智慧。例如，“事倍功半”与“事半功倍”这两个成语，其包含的文字完全相同，但由于文字所处的相对位置不同，即文字与文字之间的关系发生了改变，故而产生了截然不同的含义。

思考：

你能说出更多因文字位置改变而影响语义的类似词语或语句吗？

基本概念和术语

学生在入学的时候，通常需要填写如表2.1.1所示的学生基本信息表。下面利用此表来解释数据结构的一些基本概念。

表2.1.1　学生基本信息表

学号	姓名	性别	出生日期
0001	陆　宇	男	2002/09/02
0002	王逸铭	男	2002/03/20
0003	兰若曦	女	2003/05/25
0004	汤晓影	女	2003/03/26
⋮			

■ 数据项

数据项是数据中的最小单位，通常用来描述一个属性。例如，表2.1.1中的一个学生姓名就是一个数据项，如①所示。同样，学号、性别、出生日期的值也均为数据项。

■ 数据元素

数据元素是数据的基本单位，在计算机中通常作为一个整体来处理。一般来说，一个数据元素由多个数据项组成，具体包含哪些数据项，则取决于用户的需求。例如，表2.1.1中，一个学生的基本信息就是一个数据元素，由学号、姓名、性别和出生日期这四个数据项组成，如②所示。数据元素还可以是一本书或一条借阅记录等。

■ 数据对象

数据对象是具有相同属性的数据元素的集合，是数据的一个子集。例如，表2.1.1中

每个学生的基本信息均可被视作一个数据元素，它们都有学号、姓名、性别和出生日期四个数据项，若干学生的基本信息就构成一个数据对象。在实际应用中，被处理的数据元素通常具有相同的属性，在不产生混淆的情况下，常将数据对象简称为数据。

■ 数据结构

简单地说，数据结构是相互之间存在一种或多种特定关系的数据元素的集合。在任何问题中，数据对象中的数据元素都不是孤立存在的，而是相互之间存在着某种关系，这些关系称为结构。良好的数据结构可以给程序带来更高的运行或存储效率。

数据结构可采用如图2.1.2所示的方式来定义。

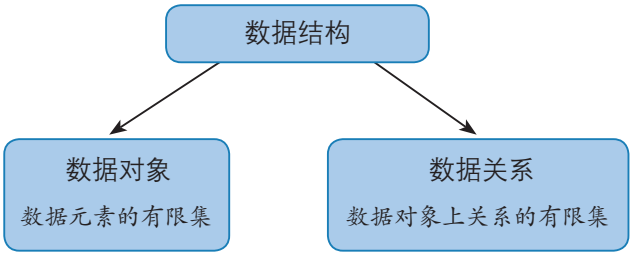


图2.1.2　数据结构的定义示意图

数据对象上的关系指属于同一对象的数据元素之间的关系。例如，可以用<*a*, *b*>来表示数据元素*a*和*b*间的前驱、后继关系，其中，*a*是*b*的前驱，*b*是*a*的后继。显然，<*a*, *b*>与<*b*, *a*>是不同的关系。

例如，用“书”“读”“好”三个字可以组成如图2.1.3和图2.1.4所示的两种不同的数据结构。

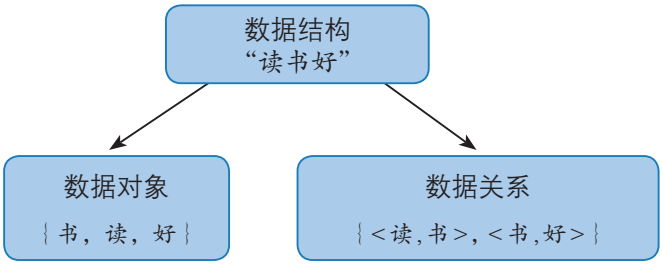


图2.1.3　数据结构“读书好”的定义示意图

“读书好”的数据对象是{书，读，好}，数据关系是{<读,书>,<书,好>}，“读”字在“书”字之前，“读”为“书”的前驱，“书”为“读”的后继。“读”无前驱，“好”无后继。

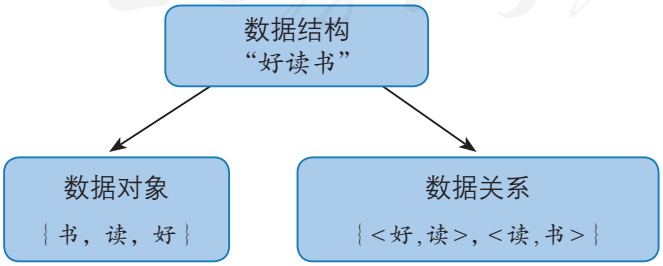


图2.1.4　数据结构“好读书”的定义示意图

“好读书”的数据对象是{书，读，好}，数据关系是{<好,读>,<读,书>}，“好”字在“读”字之前，“好”为“读”的前驱，“读”为“好”的后继。“好”无前驱，“书”

无后继。

以上两种数据结构的数据对象相同，但数据关系不同，因此是两种不同的数据结构。

实践活动　构造数据结构

用“事”“半”“功”“倍”四个字构造至少两种以上不同的数据结构，参考图2.1.3和图2.1.4的方式来定义。

根据不同的研究视角，数据结构可以分为逻辑结构和存储（物理）结构。本节详细介绍逻辑结构，存储结构将在下一节进行讲解。

逻辑结构

逻辑结构指从具体问题抽象出来的数学模型。它研究的是数据元素之间的逻辑关系，与具体的计算机无关。通常所说的数据结构一般指逻辑结构。根据数据元素之间的关系的不同特点，通常将逻辑结构分为四种基本类型：集合结构、线性结构、树形结构和图状结构。

下面利用表2.1.2来说明四种不同的数据结构，并给出相应数据结构的图形表示法。

表2.1.2　学生会部分成员基本信息表

编号	姓名	性别	所在班级	所在部门	职务
001	李元哲	男	高一1班	宣传部	干事
002	黄玉婉	女	高一2班	体育部	干事
003	刘亮亮	男	高二3班	宣传部	部长
004	孟　丹	男	高一8班	宣传部	干事
005	孙亚菲	女	高一5班	宣传部	干事
006	卢　迪	女	高二2班	宣传部	干事
007	王逸铭	男	高二6班	体育部	部长
008	兰若曦	女	高二7班		主席
009	黄佳怡	女	高二1班	生活部	部长
010	王晓敏	女	高二4班	生活部	干事

在数据结构的图形表示法中，通常用中间标有元素值的圆圈表示一个数据元素，一般也可称之为数据节点，简称节点（又称结点）。数据元素之间的关系则用圆圈之间的连线来表示。在表2.1.2中，一名学生的基本信息即可被看作一个数据元素，以下示意图均以“编号”来标识每一名学生的基本信息。

a. 学生会所有成员同属于学生会，在不研究他们之间的其他关系的情况下，所有成员构成一个集合结构，如图2.1.5所示。数据元素除了具有相同的属性外，别无其他关系，这种结构称为集合结构。

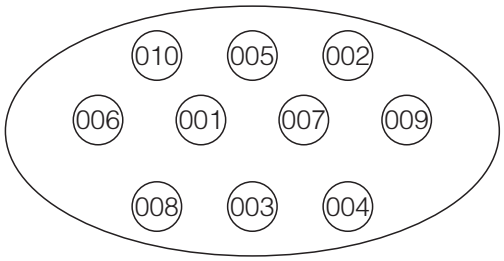


图2.1.5　集合结构示意图

b. 将表中高二学生按照班号从大到小的关系进行排列，则这些数据元素可构成一个线性结构，如图2.1.6所示。数据元素之间存在一对一的关系，这种结构称为线性结构。

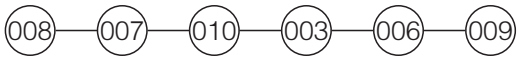


图2.1.6　线性结构示意图

c. 学生会成员间职务的级别关系是分层的，这些数据元素可构成一个树形结构。这种分层的结构与人类社会的家族关系相似，因此在树形结构中，形象地将上层节点称为双亲或父节点，将下层节点称为孩子或子节点。

可以看出，树形结构中每个数据元素至多只有一个父节点，但可以有多个子节点，其示意图就像一棵倒长的树，如图2.1.7所示。没有父节点的数据元素称为根节点，没有子节点的数据元素称为叶节点。数据元素之间存在一对多的关系，这种结构称为树形结构，可简称为树。

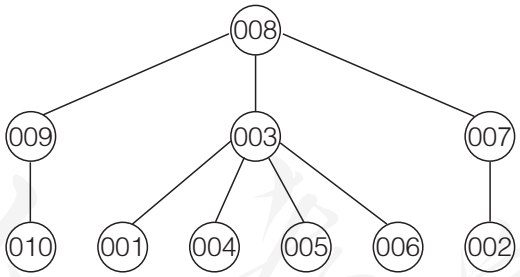


图2.1.7　树形结构示意图

d. 学生会干事之间存在的相互合作关系是多对多的，这些数据元素可构成一个图状结构，如图2.1.8所示。

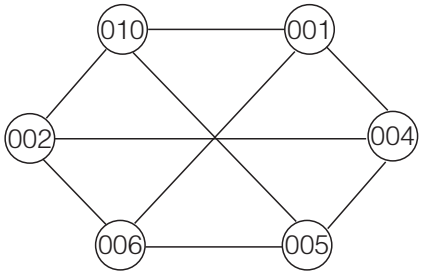


图2.1.8　图状结构示意图

可以看出，与树形结构不同，图状结构的任意两个数据元素之间都可能存在关系，因而，数据元素之间存在多对多的关系，这种结构称为图状结构，可简称为图。

简单来说，四种基本逻辑结构的特点分别为：

- 集合结构的数据元素除了同属一个数据对象外，别无其他关系；
- 线性结构的数据元素除了同属一个数据对象外，数据元素之间存在一对一的关系；
- 树形结构的数据元素除了同属一个数据对象外，数据元素之间存在一对多的关系；
- 图状结构的数据元素除了同属一个数据对象外，数据元素之间存在多对多的关系。

其中，树形结构和图状结构可统称为非线性结构。

阅读拓展

高德纳与数据结构

高德纳（Donald Ervin Knuth，图2.1.9）生于1938年，25岁毕业于美国加州理工学院数学系，获得博士学位并留校任教。30岁时，他进入斯坦福大学担任计算机系教授。

1968年，高德纳创建了数据结构的最初体系。他所著的《计算机程序设计艺术》第一卷《基本算法》是第一本较为系统地阐述数据的逻辑结构和存储结构及其操作的著作。该系列著作一经出版就震惊了世界。1974年，年仅36岁的高德纳获得计算机科学界的最高荣誉——图灵奖。

图2.1.9　高德纳

2.1.2　数据类型

数据类型是按照值的不同进行划分的。例如，在用高级程序设计语言编写的程序中，每个变量、常量或表达式都有各自的取值范围，数据类型就是用来说明变量或表达式的取值范围和所能进行的操作的。

思考活动

Python中的数据类型

在前面Python语言的程序设计学习过程中，曾经接触过Python的数据类型，如数值类型、字符串类型等。

思考：

除了上述的数据类型外，Python中还有哪些常用的数据类型？

数据类型的概念

数据类型是一组性质相同的值的集合以及定义在这个集合上的一组操作的总称。

数据类型定义了两个问题，即该类型的取值范围，以及该类型允许使用的一组运算。例如，16位整型数据的取值范围是–32 768～+32 767，可用的运算有加、减、乘、除等。高级程序设计语言中的数据类型就是已经实现的数据类型的实例，如Python中的整型、浮点型等。

抽象数据类型

抽象是用于问题表示的重要思维工具。例如，可以将“原来有6个香蕉，吃掉2个后还剩几个？”抽象表示成“6 – 2= ？”，这里抽取了问题中的数量特性，而忽略了香蕉的颜色等不相关特性。一般意义上的抽象，就是忽略研究对象的具体或无关的特性，只抽取其一般或相关的特性。

计算机科学中的抽象是将现实世界中的各种数量关系、空间关系、逻辑关系和处理过程等表示为计算机世界中的数据结构和控制结构（顺序、分支和循环等），或者说建立实际问题的计算模型。可以说，抽象是简化复杂现实问题的有效途径。例如，地图就是实际地形、地貌、地物等的抽象表示。

抽象数据类型一般可以由数据对象、数据对象上的关系以及对数据对象的一组基本操作三个要素来定义。

与数据结构定义形式相对应，抽象数据类型可用如图2.1.10所示的方式来定义。

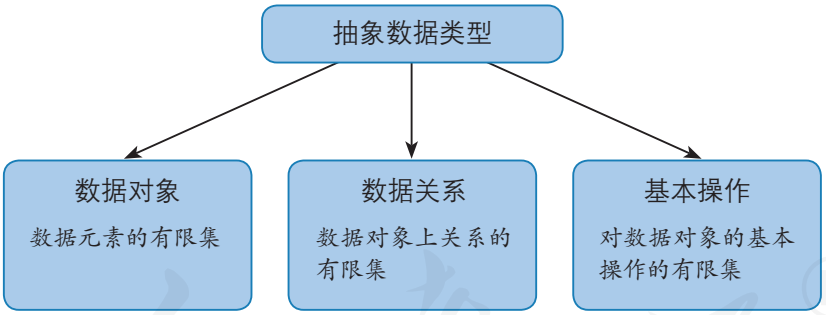


图2.1.10　抽象数据类型的定义示意图

图2.1.11所示是一个抽象数据类型的定义示例。

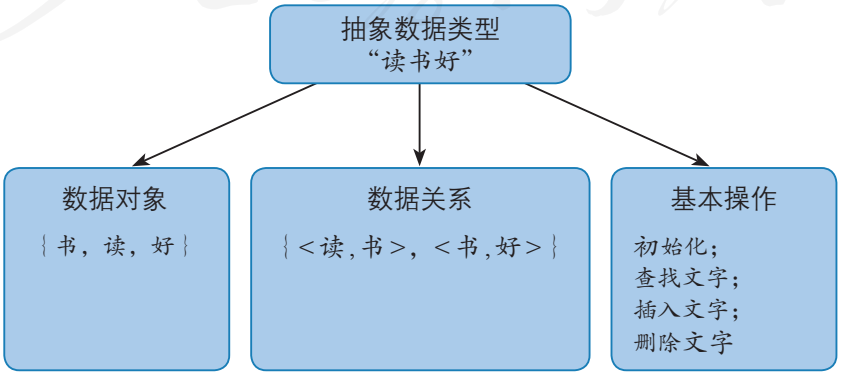


图2.1.11　抽象数据类型“读书好”的定义示意图

抽象数据类型把实际生活中的问题分解为多个小规模且容易处理的问题，然后建立一个计算机能处理的数据模型，把每个功能模块的实现细节作为一个独立的单元，并将其具体实现过程隐藏起来。

高级程序设计语言中没有提供的数据类型需要人为定义，可以通过建立抽象数据类型来实现，即抽象数据类型给程序员提供了构造自定义数据类型的手段。

实践活动

组合数的计算

从n个不同元素中，任取$m(m \leqslant n)$个元素并成一组，叫作从n个不同元素中取出m个元素的一个组合；从n个不同元素中取出$m(m \leqslant n)$个元素的所有组合的个数，叫作从n个不同元素中取出m个元素的组合数，用符号C_n^m表示。

组合数的计算公式为$C_n^m=\dfrac{n!}{(n-m)!m!}$

分析：本程序要求用户输入n、m（$m \leqslant n$）的值，同时这里定义n、m为正整数，根据求组合数的计算公式，要三次用到阶乘（$n!=1\times2\times3\times\cdots\times n$，$0!=1$）计算。因此，可以自定义一个函数来计算阶乘，然后在程序中调用三次。

理解组合数计算算法的思路，并将如下程序补充完整。

```
# 自定义函数计算x!
def factorial(x):
    t = 1
    for i in range(2, x + 1):
        t = t * i
    return t

if __name__ == "__main__":
    # 输入n的值
    n = _____________
    # 输入m的值
    m = _____________
    if m > 0 and n > 0 and n >= m:
        # 求组合数
        c = factorial(n) / _____________
        # 打印结果
        _____________________________________
    else:
        print("您的输入有误")
```

技术支持

自定义函数

随着程序的功能愈加强大，程序的规模也将随之扩大，若把所有代码写在一起，就会使程序的可读性越来越差、维护的难度越来越大。另外，当程序中需要反复编写同样的语句段以实现相同的功能时，程序就会变得冗长。此时，就可以使用自定义函数实现程序的优化。

Python提供了许多内建函数，此外，我们还可以创建自定义函数。

在Python中，定义一个函数的标准方法是使用def语句，依次写出函数名、括号、括号中的参数和冒号“:”，然后在缩进块中编写函数体，函数的返回值用return语句返回。

其格式如下：

```
def 函数名(参数):
    函数体
    return 返回值
```

以自定义一个判断是否为成年人的函数get_age()为例：

```
# 判断年龄
def get_age(age):
    if age >= 18:
        return "成年人"
    else:
        return "未成年人"
```

注意，函数体内部的语句在执行时，一旦执行到return，函数就执行完毕，并返回结果。

自定义函数需要先定义才能使用。在程序中使用自定义函数的方法与内建函数相同。函数的设计体现了抽象的思想，对函数的调用者而言，不必关心函数的实现过程，即函数将具体实现过程隐藏了起来。

项目实施

阅读活动中的数据结构

一、项目活动

书籍是可以按照一定的标准进行分类的，如分为文学类图书和科技类图书等。文学类图书又可以分为小说、诗歌、散文和报告文学等，科技类图书也可以再细分为数学、物理、化学和生物等。围绕阅读活动中存在的数据结构实例，探究它们和数据结构类型的对应关系，并填入表 2.1.3 中。

表2.1.3　阅读活动中的数据结构

数据逻辑结构类型	名称	文字简述	图例
集合结构	学校图书馆的图书	这些图书仅属于同一学校的图书馆，除此之外，没有其他关系	① ② ③ ④ ⑤ ⑥
线性结构			
树形结构			
图状结构			

提示：如果图例位置偏小，可使用思维导图单独绘制。

二、项目检查

1. 探究所绘制图例和数据结构类型的对应关系是否一致。

2. 小组之间交流、展示成果并相互点评。

3. 完善表格内容，进一步理解数据的逻辑结构。

练习提升

1. 将本章的知识脉络用恰当的数据结构类型表示出来。

2. 编写程序进行求解：已知一元二次方程$ax^2+bx+c=0$，从键盘输入a、b和c的值，求该一元二次方程的解。

提示：需要自定义一个函数。

2.2
数组与链表

学习目标

- 能结合生活实际理解数组、链表的含义，并能编程实现其相关操作。
- 理解数组、链表的区别，能根据不同的应用场景选择合适的存储方式。

体验探索

方队与数据存储

我们的祖先曾创造了无与伦比的文化，而“和合”文化正是这其中的精髓之一。“和”指的是和谐、和平、中和等，“合”指的是汇合、融合、联合等。这种“贵和尚中、善解能容，厚德载物、和而不同”的宽容品格，是我们民族所追求的一种文化理念。

2008年北京奥林匹克运动会开幕式中的表演方队（图2.2.1）就完美展示了“和合”理念，向世界传达出中国人民希望构建一个和平、和谐而更加美好世界的期待。

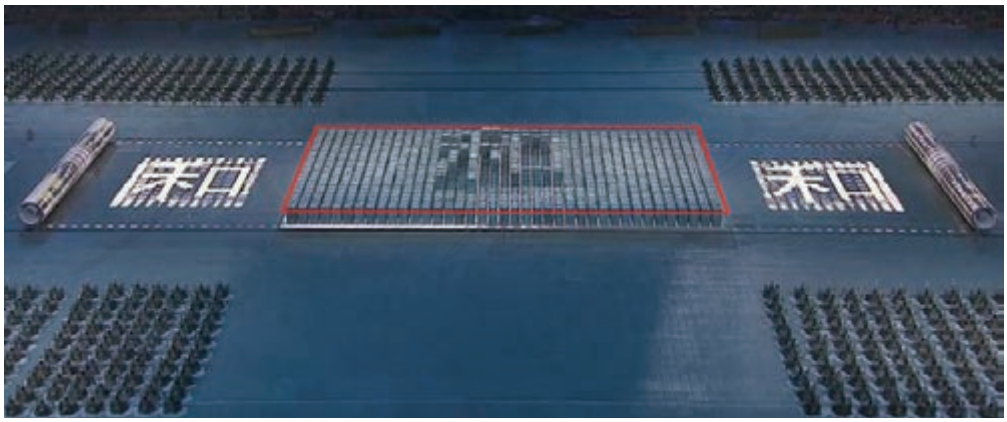


图2.2.1　2008年北京奥林匹克运动会开幕式中的表演方队

思考：

如果把图2.2.1红框方队中的每个表演者视为一个数据，在计算机中如何存储这些数据？

2.2.1 存储结构

存储结构，也称物理结构，是逻辑结构在计算机中的存储形式，它包括数据元素的存储和数据元素之间关系的存储。逻辑结构与存储结构的关系为：逻辑结构是面向问题的，而存储结构是面向计算机的。逻辑结构是数据结构的抽象，存储结构是数据结构的实现，两者综合起来建立了数据元素之间的结构关系。

思考活动

医院的秩序管理

在医院大厅里，常会看到这样的情形：有人在缴费窗口前排队，也有人零零散散坐在座位上等待叫号。

思考：

如果把每个排队缴费或等待叫号看病的人均抽象为一个数据元素，在计算机中采取什么方式来存储这些数据元素更为方便？

按照数据元素之间关系的不同存储方式，存储结构可分为两种基本类型：顺序存储结构和链式存储结构。

例如，假设每个数据元素占用2字节，<24,56>可用两种方式存储，如图2.2.2所示。在链式存储中，“∧”表示数据元素“56”没有后继元素。

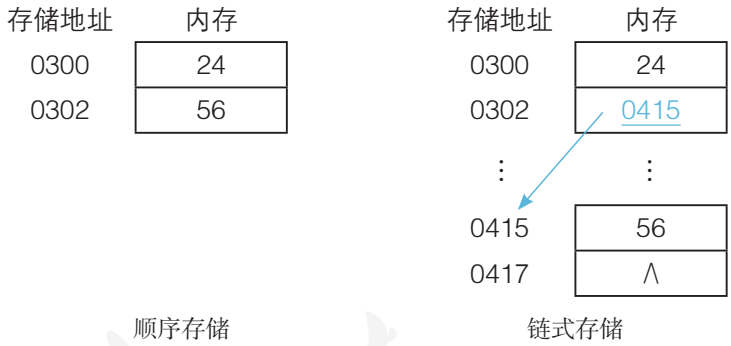


图2.2.2 <24,56>的两种存储方式示意图

顺序存储结构是把逻辑上相邻的数据元素存放在地址连续的若干存储单元中，数据元素之间的逻辑关系由存储单元的邻接关系来体现。顺序存储结构是一种最基本的存储结构，在高级程序设计语言中通常用数组类型来实现。

假定有一个由n（n为正整数）个数据元素构成的线性结构（a_1，a_2，…，a_i，…，a_n）（$1 \leqslant i \leqslant n$），采用顺序存储结构，每个数据元素占$k$（$k$为正整数）字节的存储单元。第一个数据元素$a_1$的存储地址记为$\mathrm{loc}(a_1)$，称为基地址，则可以通过如下公式计算数据元素$a_i$的存储地址$\mathrm{loc}(a_i)$：

$$\mathrm{loc}(a_i) = \mathrm{loc}(a_1) + (i-1) \times k$$

该线性结构的顺序存储结构示意如图2.2.3所示。

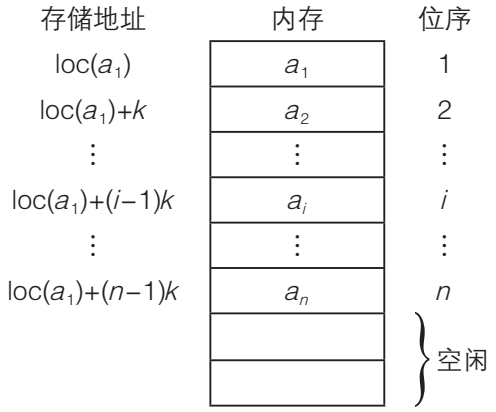


图2.2.3　顺序存储结构示意图

链式存储结构是把数据元素存放在任意的存储单元中，这些存储单元可以是连续的，也可以是不连续的。链式存储结构在高级程序设计语言中通常用指针来实现。

在顺序存储结构中，每个数据元素只需存储数据元素信息即可。但在链式存储结构中，除了存储数据元素信息（数据域）外，还要存储它的后继元素的存储地址（指针域），指针域中存储的信息称为指针或链。这两部分信息组成链式存储结构中的节点，如图2.2.4所示。

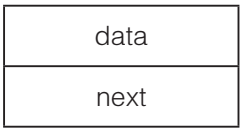


图2.2.4　链式存储结构中的节点

假设每个数据元素的数据域和指针域均占用2字节，线性结构（a_1，a_2，a_3，a_4）的链式存储结构示意如图2.2.5所示，因为“0208”是存储首元素a_1的地址，故而地址“0208”被称为头指针，这里的存储地址采用十六进制表示。

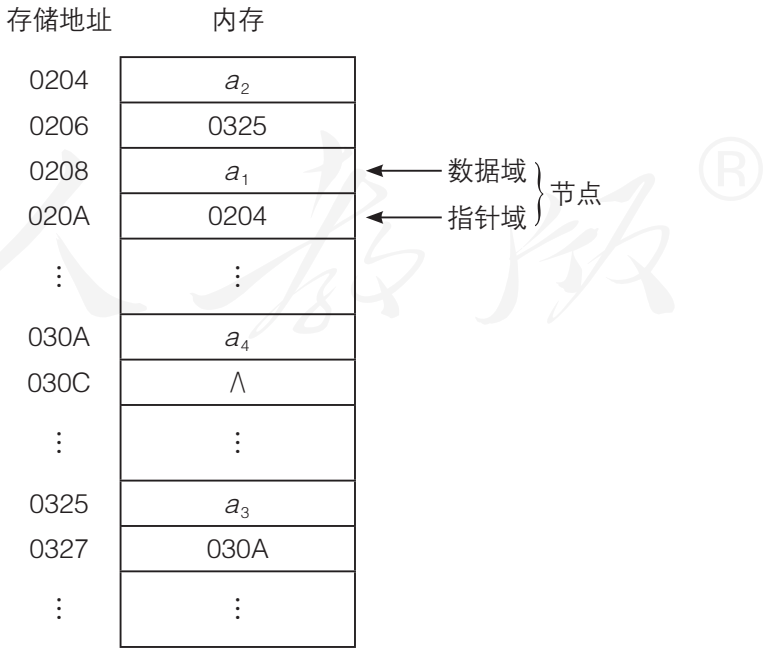


图2.2.5　链式存储结构示意图

2.2.2 数组——顺序存储

大多数实际问题中涉及的数据元素都有很多个，数组是存储多个数据元素的重要方法。

思考活动

身高数据处理

整型、浮点型等数据类型只能表示单一数据，现已采集10位女生的身高数据，如表2.2.1所示，需要计算这10位女生的平均身高。

表2.2.1　10位女生的身高数据表

学号	身高/cm
1	162
2	165
3	167
4	155
5	162
6	168
7	159
8	166
9	164
10	160

思考：

1. 如何在程序中定义变量来表示这些数据？

2. 如果有更多的数据，比如一个班或一个年级的学生身高数据，在程序中又怎么用变量来表示？

数组

数组是一组具有相同数据类型的数据元素的集合，占用一段连续的存储空间，常用来实现数据的顺序存储。用下标代表数据元素在数组中的序号，一般地，数组下标自0开始编号。用数组名和下标来唯一地标识数组中的一个数据元素。

例如，表2.2.1中的数据可用数组*a*来存储，*a*[0]，*a*[1]，*a*[2]，…，*a*[9]分别存储学生1，学生2，学生3，……，学生10的身高，如表2.2.2所示。其中，*a*是数组名，0，1，2，…，9是数组下标。

表2.2.2　存储在数组中的学生身高数据

a[0]	*a*[1]	*a*[2]	*a*[3]	*a*[4]	*a*[5]	*a*[6]	*a*[7]	*a*[8]	*a*[9]
162	165	167	155	162	168	159	166	164	160

标识数组中数据元素所需的下标个数可能不止一个，下标个数称为数组的维数。只有一个下标的数组称为一维数组，如上面介绍的数组*a*就是一维数组。有两个下标的数组称为二维数组，也常称为矩阵。

如图2.2.6所示的围棋棋盘需要用一个二维数组（如*m*）来表示，棋盘中的一个具体位置需要指定两个下标才能唯一确定，如用*m*[0][0]来表示图中左上角的位置，则*m*[18][18]表示图中右下角的位置，其中第一个下标表示行号，第二个下标表示列号。一般而言，可以设置数组元素值为0，表示该位置没有棋子；设置数组元素值为1，表示该位置为一方棋子；设置数组元素值为2，表示该位置为另一方棋子。

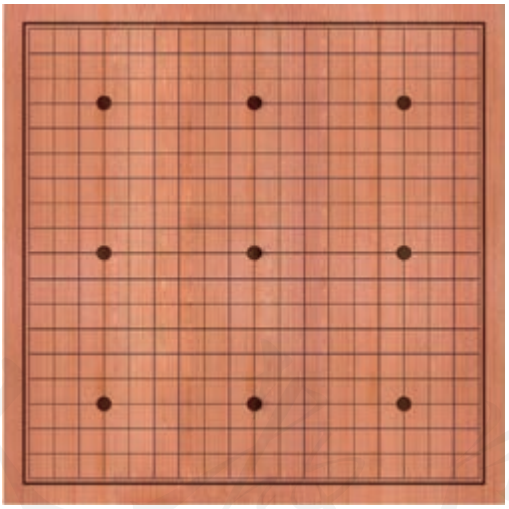


图2.2.6　围棋棋盘示例

数组的操作

1. 数组的初始化和赋值

在Python中，可以通过列表类型“list”来实现对数组的操作。在程序中，可以使用以下方法为数组进行初始化和赋值操作。

例如，*a*=[ ]表示数组*a*是一个空数组；*b*=[1，2，3，4，5，6，7，8，9]，表示数组*b*中有9个数据元素。

例1：初始化一个具有100个元素且元素值均为0的一维数组a。

```
a = []
for i in range(100):
    a.append(0)
```

例2：初始化一个具有10个元素且元素值均为1～100的随机整数的数组a。

```
import random
a = []
for i in range(10):
    x = random.randint(1, 100)        # 产生随机整数
    a.append(x)
print(a)
```

程序运行结果：

```
[83, 67, 69, 41, 66, 94, 75, 81, 54, 68]
```

需要注意的是，该程序的运行结果是随机的。

例3：输入10名女生的身高数据并存入数组a中，求她们的平均身高。

```
a = []
s = 0
for i in range(10):
    print("请输入第", i + 1, "名女生身高的数据")
    x = int(input())
    a.append(x)
    s = s + a[i]
print("这10名女生的身高分别为：", a)
print("这10名女生的身高平均值为：", s / 10)
```

例4：初始化一个具有8行2列且每个元素都是0的数组a。

```
a = [ [ 0 for i in range(0, 2)] for j in range(0, 8) ]
print(a)
```

程序运行结果：

```
[[0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0]]
```

实践活动

斐波那契数列问题

斐波那契数列是这样一个数列：1，1，2，3，5，8，13，21，34，…在数学上，斐波那契数列可以用这样的方法定义：$f(n)=f(n-1)+f(n-2)$（$n>2$，$n\in N$）。用文字来说，就是斐波那契数列的第1项和第2项均为1，从第3项起每一项等于它的前两项之和。

分析：用数组处理问题时，每一个数组元素表示数列中的一个数值，设计程序时，只需依次求出各数值并保存在相应的数组元素中即可。用数组的方法求解斐波那契数列问题的程序如下：

```
f = []
# 求斐波那契数列前20项
for i in range(20):
    f.append(_____)
for i in range(_______):
    f[i] = _______________
print(f)
```

程序运行结果：

```
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584,
4181, 6765]
```

2. 插入数组元素

在有n（n 为正整数）个数组元素的一维数组a中插入一个元素e，使之成为第i（$0 \leqslant i \leqslant n$）号元素的基本操作为：

■ 若元素e插入在数组的尾部（即$i=n$），那么数组中原来的所有元素的存储位置都不会改变，直接将元素e赋值给$a[n]$即可；

■ 否则，从$a[i]$到$a[n-1]$的所有元素必须依次向后移动一个位置，以空出第i号位置，用于存放元素e。

实现$a[i]$到$a[n-1]$向后移动一个位置的方法是：先将b_n移动到$a[n]$的位置上，再将b_{n-1}移动到$a[n-1]$的位置上。依此类推，直到将b_{i+1}移动到$a[i+1]$的位置上。

数组元素的位置变化如图2.2.7所示。

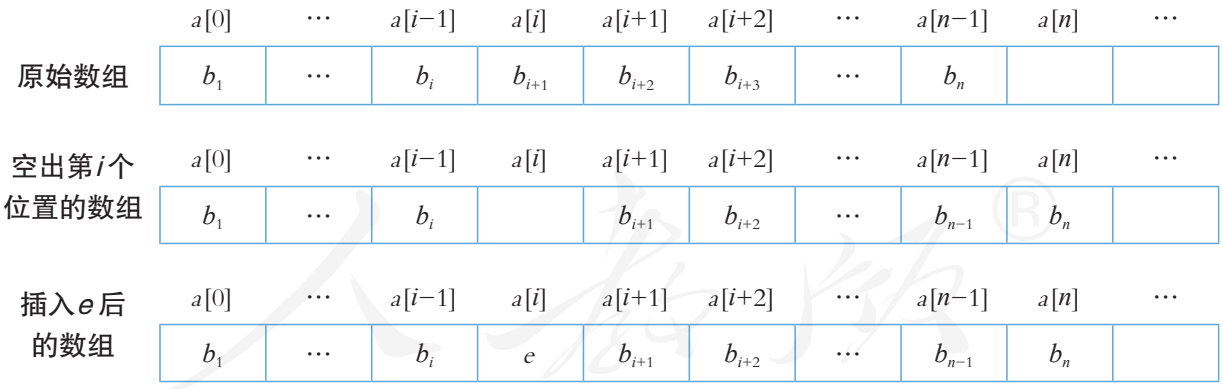


图2.2.7　数组元素的位置变化示意图

3. 数组元素的删除

在有n（n为正整数）个数组元素的一维数组a中删除第i号数组元素$a[i]$（$0 \leqslant i \leqslant n-1$）的基本操作为：

■ 若$i=n-1$，直接删除$a[n-1]$，数组中剩余的所有元素的存储位置都不会改变；

■ 否则，从$a[i+1]$到$a[n-1]$的所有元素必须依次向前移动一个位置。

由于数组在内存中占用一段连续的存储空间，一旦在程序中定义了数组，则它的维数和能容纳的元素个数就不再轻易改变。因此，对数组经常进行存取和查找操作，一般不进行数组元素的插入或删除操作。

实践活动

利用二维数组描述“亘”字字模

编写程序，使用二维数组描述如图2.2.8所示的汉字“亘”的字模。

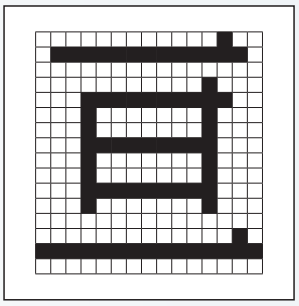


图2.2.8　15×16点阵汉字“亘”的字模

2.2.3　链表——链式存储

数组的优点和缺点都在于元素存储的集中方式和连续性。它的缺点具体表现为数组元素的插入和删除需要大量移动数组中已有的元素，当数组中存储的数据元素个数较多时，操作量将会很大。

思考活动

“老鹰捉小鸡”游戏与数据存储

如图2.2.9所示的场景中，由老师身后的孩子们组成的队伍有时会发生一些变化，例如，某个孩子插进队伍中或单独从队伍中跑出来等。

图2.2.9　“老鹰捉小鸡”活动场景

思考：

如果我们把每个孩子抽象为一个数据元素，在计算机中采取哪种结构存储这些数据元素更合适？

链表

链表指由多个节点（由数据域和指针域组成）链接成的序列，通过节点的指针域将多个节点按数据元素的逻辑顺序链接在一起。每个节点只有一个指向后继的指针域的链表称为单链表。通常，我们将链表示意为用箭头相链接的节点的序列，节点之间的箭头表示指针域中的指针。

假设有一个由n（n为正整数）个数据元素构成的线性结构（a_1，a_2，…，a_i，…，a_n）（$1 \leqslant i \leqslant n$），则由$n$个节点链接成的单链表如图2.2.10所示，其中，Head 指示链表中第一个节点的存储位置，称为头指针。

图2.2.10　单链表示意图

可以看出，单链表中每个节点的存储地址是放在其前驱节点的指针域中的，因此，对整个链表的存取必须从头指针Head开始，这样就可以由头指针找到第一个节点，再由第一个节点的指针域找到第二个节点……依次顺着指针域找到每个节点。因为最后一个节点的数据元素a_n没有后继，所以其指针域为“空指针”，用“∧”表示。

例如，表2.2.1中的数据可用单链表来存储，如图2.2.11所示。

图2.2.11　女生身高数据单链表存储示意图

链表的操作

在Python中，可以通过“类”来实现对链表的操作。每个节点都是链表中的一个实例，链接在一起形成一个完整链表。

技术支持

类简介

类：用来描述具有相同属性和方法的对象的集合。它定义了该集合中每个对象所共有的属性和方法。对象是类的实例。类中定义的函数称为方法。

对象：通过类定义的数据结构实例。对象包括两种数据成员（类变量和实例变量）和方法。

类的定义：

```
class ClassName:
    "类的帮助信息"              # 类文档字符串
    class_suite                 # 类体
```

说明：类名首字符一般要大写；class_suite 由类成员的属性和方法组成。

以下是一个简单的创建类的Python程序：

```
class Student:
    "所有学生的基类"
    stu_count = 0
    def __init__(self, name, age):
        self.name = name
        self.age = age
        Student.stu_count = Student.stu_count + 1
    def display_count(self):
        print("学生人数 : " ,  Student.stu_count)
    def display_student(self):
        print("姓名 : ", self.name, ", 年龄: ", self.age)
```

说明：

■ stu_count 变量是一个类变量，它的值将在这个类的所有实例之间共享；

■ __init__( ) 方法是初始化方法，当创建类的实例时就会调用该方法；

■ self 代表类的实例，在定义类的方法时self 是必须有的，虽然在调用时不必传入相应的参数。

创建对象：

```
# 创建Student类的第一个对象
stu1 = Student("李莉", 15)
# 创建Student类的第二个对象
stu2 = Student("张明", 16)
```

访问属性：可以使用点“.”来访问对象的属性。

```
stu1.display_student()
stu2.display_student()
print("学生人数: " ,  Student.stu_count)
```

链表的操作包括定义节点、初始化链表、插入节点和删除节点等。

1. 定义节点类

```
# 建立节点的类
class Node(object):
    # 类的初始化方法
    def __init__(self, data, next = None):
        self.data = data
        self.next = next
```

2. 初始化链表

链表建立的过程为：输入节点的值（如身高数据），如果输入不是“-1”，则新建一个节点实例，并为数据域和指针域赋值；重复上述操作，当输入为“-1”时，结束并返回当前链表的头指针。

其核心代码如下：

```
class LinkList(object):
    # 初始化方法，当创建类的实例时就会调用该方法；
    def __init__(self):
        self.head = None

    # 初始化链表
    def initlist(self):
        print("请输入身高数据，当输入“-1”时，表示输入结束。")
        data = input()
        if data != "-1":
            # 新建第一个节点实例
            self.head = Node(int(data))
            p = self.head
        # 重复输入身高数据
        while data != "-1":
            data = input()
            if data == "-1":
                break
            else:
                p.next = Node(int(data))
                p = p.next
        print("输入结束！")
```

3. 插入节点

在有n（n为正整数）个节点的单链表中插入一个数据元素e成为第i（$1 \leqslant i \leqslant n+1$）个节点的基本操作为：

首先，查找定位a_{i-1}；

然后，生成一个新节点，将其数据域置为e；

最后，调整指针，完成节点的插入。

调整指针过程中的指针变化如图2.2.12所示。

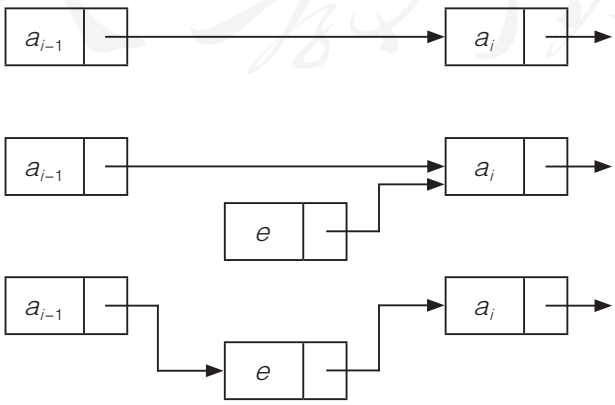


图2.2.12　在单链表中插入节点的指针变化示意图

当$i \neq 1$时，链表节点的插入需要进行两次指针调整：首先将e的后继指针指向a_i，然后再将a_{i-1}的后继指针指向e。注意：两次指针调整的顺序不能颠倒。

当i=1时，不存在a_{i-1}，此时数据元素e插入为链表的第一个节点。

其核心代码如下：

```
# 插入节点
def insert(self, index, item):
    p = self.head
    if index == None:                          # 如果要把节点插入在第一个位置
        q = Node(item)
        q.next = self.head
        self.head = q
    else:
        post = self.head
        j = 0
        while p.next != None and j < index: # 查找插入的位置
            post = p
            p = p.next
            j = j + 1
        if index == j:
            q = Node(item, p)                  # 新建一个节点实例
            post.next = q
            q.next = p
```

4. 删除节点

在有n（n为正整数）个节点的单链表中删除第i个数据元素a_i（$1 \leqslant i \leqslant n$）的基本操作为：

首先，查找定位a_{i-1}（若i=1，则不存在a_{i-1}，要删除的是链表的第一个节点）；

其次，修改其指针。

指针变化如图2.2.13所示。

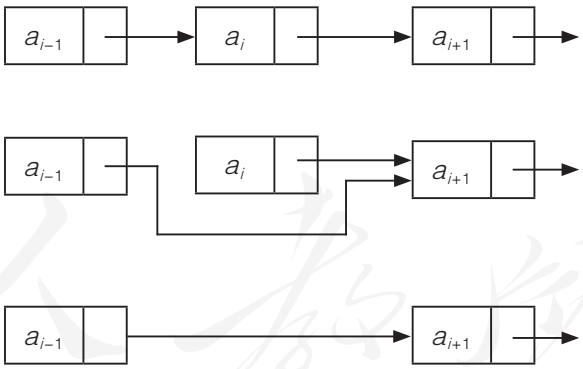


图2.2.13　删除节点的指针变化示意图

其核心代码如下：

```
# 删除节点
def delete(self, index):
    p = self.head
    if index == None:                          # 如果要删除的节点在第一个位置
        q = self.head
        self.head = q.next
        p = self.head
```

```
        else:
            post = self.head
            j = 0
            while p.next != None and j < index:
                post = p
                p = p.next
                j = j + 1
            if index == j:
                post.next = p.next
```

由上面两种操作可知，在单链表中插入或删除一个节点时，仅需修改指针而不需要移动元素的位置。

5. 求链表长度

从链表的头指针开始，利用while循环，指针逐一后移，直至为空，则退出循环，返回链表的节点个数。

```
    # 求链表的长度
    def  get_length(self):
        p = self.head
        length = 0
        while p != None:
            length = length + 1
            p = p.next
        return length
```

6. 查找节点

从链表的头指针开始查找，验证边界条件，当链表为空时，显示“这是空链表”。当链表非空时，利用while循环逐个节点比较，若找到节点，则显示其位序；若查找到链表的最后一个节点仍没有找到，则显示“查找失败！”。

```
    # 查找节点
    def index(self, value):
        if  self.get_length() == 0:
            print("这是空链表")
            return
        p = self.head
        i = 0
        while p.next != None and  p.data != value:
            p = p.next
            i = i + 1

        if p.data == value:
            print("要找的值在链表中的第%d位。" %(i + 1))
        else:
            print("查找失败!")
        return
```

实践活动

链表程序应用

体验式户外拓展训练获得了越来越多年轻人的欢迎，其训练项目丰富多样，图2.2.14所示为“毕业墙”项目：所有队员按照教官的指示，利用人梯爬上4 m的高墙，它强调学员之间的团结合作，共同完成同一个目标。

图2.2.14　户外拓展训练项目——毕业墙

某学校要组织学生参加这样的户外拓展活动，预案按照时间顺序拟定了一些项目，但在活动过程中因为天气的原因要将其中的两个项目改为室内活动项目。

查找资料，设计若干户外和室内活动项目名称。编写程序，用链表的方式保存拓展项目名称，并编写程序实现增加、删除的功能。

2.2.4　数组与链表的比较

数组和链表是两种基本的存储结构，它们在内存分配与使用上是不一样的，有各自的特点。

思考活动

旅游景点名称的存储

小明打算利用假期与家人外出度假，他们旅游的目的地是位于我国西南边陲，有“彩云之南”美称的云南省。他们有两种可选方案：跟团游或自由行。

思考：

如果用数组或链表来存储他们将要游览的每个旅游景点的名称，你会选择哪种方案？请说明原因。

数组和链表在存储分配方式、空间性能和时间性能三方面都各有其特点，如表2.2.3所示。

表2.2.3　数组与链表的对比

比较项目	数组	链表
存储分配方式	数组用一段地址连续的存储单元依次存储数据元素，数据元素之间的逻辑关系通过存储位置来体现	链表用一组地址不要求连续的存储单元存放数据元素，用指针来反映数据元素之间的逻辑关系
空间性能	数组需要一段连续的存储空间，因此对内存的要求较高；由于数组中数据元素的逻辑结构可通过物理上的相对位置表现出来，故数组的存储密度高	链表不需要一段连续的存储空间，因此对内存的要求不高；由于链表中数据元素的逻辑结构需通过指针来体现，故链表的存储密度低
时间性能	数组是一种随机访问结构，对数组中任一元素都可以直接存取。而在数组中进行插入和删除，平均要移动近一半的元素，当每个元素的信息量较大时，移动元素需要消耗较长的时间	链表是一种顺序访问结构，要查找链表中的任一元素，都需从头指针起开始查找，平均需要搜索半个链表。而在链表中的任何位置上进行插入和删除，都只需要修改指针，不需要移动元素

具体程序设计中，应该如何选择存储结构呢？

■ 当对数据元素进行的操作主要是元素查找，而很少做插入和删除时，宜采用数组作为存储结构。如果需要频繁进行数据元素的插入和删除操作，宜采用链表作为存储结构。

■ 当程序中需要的元素个数变化较大或者不知道数据量有多大时，建议选择链表结构，这样可以不需要考虑存储空间大小的问题。但如果事先知道元素的大致个数，采用数组的效率会高很多。

总之，需要根据实际情况，客观考虑采用哪种数据结构更能满足程序功能和性能需求。在具体程序实现中，要综合考量数组和链表的优缺点，才能最终选定比较适宜的实现方法。

例：编程解决“约瑟夫环”问题。

有41个人围坐在一起排成一个圆圈，由第1个人开始报数，每数到3，此人就必须出列，然后再由下一位重新从1开始报数，直到所有人都出列为止。请问，最后一个出列的人所在的初始位置是什么？

这其实是一个数学应用问题，可以描述为：已知n个人（以编号1，2，3，…，n分别表示）围坐在一张圆桌周围。约定从编号为1的人开始报数，数到k的那个人出列；下一个人又从1开始报数，数到k的那个人又出列……依此规律重复下去，直到圆桌周围的人全部出列。简化的“约瑟夫环”如图2.2.15所示。

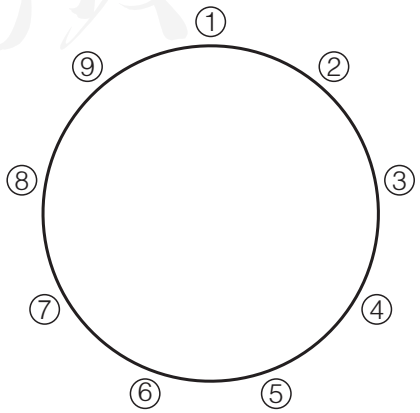


图2.2.15　简化的“约瑟夫环”

要解决此问题，要求用户输入的内容包括：

a. 参加活动的人数，即n的值（编号1，2，3，…，n需要存放在数组或链表中）；

b. 出列的间隔数，即k的值。

要求输出的内容包括：

a. 出列人员的序号；

b. 最后出列人员的序号。

根据上面的问题分析及输入、输出参数分析，可以选择一种数据存储结构，然后用算法来实现。

1. 利用数组编程求解。为了解决这一问题，在Python中，可以用长度为n的数组作为顺序存储结构，来存储活动中参与人员的编号，利用Python中的取模（%）运算，计算每次出列人员的序号。利用数组求解“约瑟夫环”问题的程序如下：

```
def josephus(n, k):
    if k == 1:                              # 如果是数到1的人出列，则n是最后出列的人
        for i in range(1,n):
            print("出列人员的序号为：", i)
        print("最后一个出列人员的序号为：", n)
        return
    p = 0
    people = list(range(1, n + 1))
    while True:
        if len(people) == 1:                # 如果数组只有一个元素，退出循环
            break
        p = (p + (k - 1)) % len(people) # 取模运算，计算出列的序号
        print("出列人员的序号为：  ", people[p])
        del people[p]                       # 删除list中相应的元素
    print("最后一个出列人员的序号为：", people[0])

if __name__ == "__main__":
    print("请输入总人数n=:(n>1)")
    n = int(input())
    print("请输入第几人出列k=:")
    k = int(input())
    josephus(n, k)
```

实践活动

理解程序，跟踪数组的变化

通过填写下列数组元素值的方法，体会程序中用取模运算来计算出列人员的设计思路。

例如，设定n=9，k=3，列出数组的变化。

数组初始状态：

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]
1	2	3	4	5	6	7	8	9

第 1 次删除元素的编号是 3，数组变为：

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
1	2	4	5	6	7	8	9

第 2 次删除元素的编号是 6，数组变为：

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
1	2	4	5	7	8	9

第 3 次删除元素的编号是_____，数组变为：

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
1	2				

第 4 次删除元素的编号是 4，数组变为：

a[0]	a[1]	a[2]	a[3]	a[4]
1	2	5	7	8

第 5 次删除元素的编号是_____，数组变为：

a[0]	a[1]	a[2]	a[3]

第 6 次删除元素的编号是_____，数组变为：

a[0]	a[1]	a[2]

第 7 次删除元素的编号是_____，数组变为：

a[0]	a[1]

第 8 次删除元素的编号是_____，数组变为：

a[0]

2. 利用链表编程求解。可以通过单循环链表的数据结构来模拟“约瑟夫环”的结构。所谓单循环链表，就是将单链表的尾元素指针指向首元素。

将n个人的编号组织成具有n个节点的单循环链表，从第一个节点开始数，数到第k个节点，将该节点从循环链表中删除，然后再从删除节点的下一个节点开始数，数到第k个节点，再将其删除，如此进行下去，直到所有人都出列为止。

当k=1时，最后出列的就是最后一个人；当$k \geqslant 2$时，构造一个n个元素的单循环链表，然后依次出列第k个人，最后一个为最后出列的人。

实践活动

理解程序思路，完成编程任务

理解利用单循环链表求解“约瑟夫环”问题的思路，并将如下程序补充完整。

```
class Node():                                          # 建立节点的类
    def __init__(self, data, next = None):
        self.data = data
        self.next = next

def create_link(n):                                    # 产生n个元素的链表
    root = Node(1)                                     # 新建第一个节点
    tmp = root                                         # 指针指向第1个节点
    for i in range(2, n + 1):
        tmp.next = ____________________                # 产生第i个节点，请你填写
        tmp = ____________________                     # 指针指向该节点，请你填写
    tmp.next = root              # 将链表的尾指针指向root，从而构成单循环链表
    return root

def josephus(n, k):
    # 如果是数到1的人出列，则n是最后出列的人
    if k == 1:
        for i in range(1, n):
            print("出列人员的序号为：", i)
        print("最后一个出列人员的序号为：  ", n)
        return
    root = create_link(n)
    tmp = root
    while True:
        for i in range(k - 2):
            tmp = tmp.next
        print("出列人员的序号为：", tmp.next.data)
        tmp.next = tmp.next.next                       # 删除相应节点
        tmp = tmp.next
        if tmp.next == tmp:
            break
    print("最后一个出列人员的序号为：  ", tmp.data)

if __name__ == "__main__":
    print("请输入总人数n=:(n>1)")
    n = ____________________                           # n为整数
    print("请输入第几人出列k=:")
    k = ____________________                           # k为整数
    ____________________                               # 调用函数josephus
```

项目实施

编写程序，管理个人书目

一、项目活动

请将喜欢的图书列出一张书单。

1. 根据需求，选择合适的数据结构对其进行存储。

2. 编写程序对该数据结构进行初始化、插入、删除和查找等操作。

3. 给程序添加注释，同时形成一份描述工作记录的文字资料。

4. 整理程序及文档，形成项目报告。小组之间开展交流。

二、项目检查

完成“管理个人书目”的程序设计，并检查程序段所实现的功能，程序没有明显错误，能正确运行。

练习提升

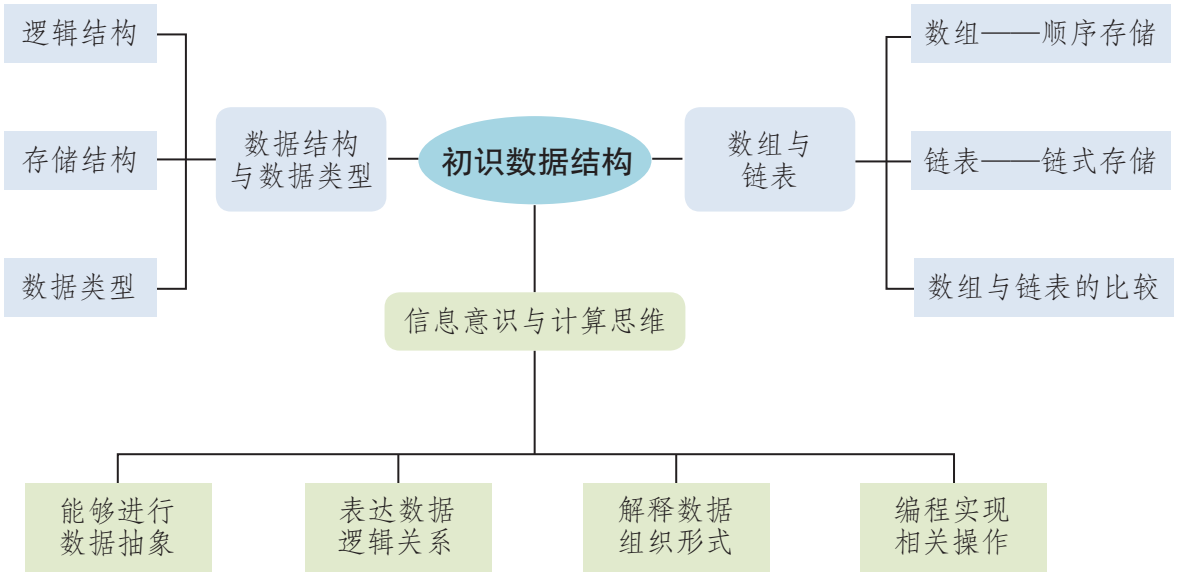
1. 编写程序，利用随机函数，生成10 个0 ~ 10 的随机整数存储在数组中，并判断其中是否有数字“5”，若有，则输出它在数组中的下标（如有多个，也一并输出）；否则，输出“NO DATA”。

2. 编写程序，使用二维数组构造出以下的杨辉三角形（要求输出n行，$n>9$）。杨辉三角形是南宋数学家杨辉所著的《详解九章算术》一书中用三角形解释二项式系数的乘方规律，是二项式系数在三角形中的一种几何排列。

```
1
1   1
1   2   1
1   3   3   1
1   4   6   4   1
1   5   10  10  5   1
1   6   15  20  15  6   1
⋮   ⋮   ⋮   ⋮   ⋮   ⋮   ⋮   ⋮
```

总结 评价

1. 下图展示了本章的核心概念与关键能力，请同学们对照图中的内容进行总结。



2. 根据自己的掌握情况填写下表。

学习内容	掌握程度
数据结构的基本概念	□不了解 □了解 □理解
抽象数据类型的概念	□不了解 □了解 □理解
数组、链表等基本数据结构的概念	□不了解 □了解 □理解
数组、链表的相关操作	□不了解 □了解 □理解
数组、链表的区别	□不了解 □了解 □理解

第 3 章
数据结构基本类型

戏曲是中国传统艺术瑰宝之一，在中华民族的历史长河中绽放着夺目光彩。中国戏曲不但种类繁多有趣，表演形式也是集“唱、念、做、打”于一体，精彩绝伦。面对不同戏曲剧种及其发展历程、艺术特色、戏曲行当和各式脸谱等纷繁复杂的要素，如何才能尽快抓住诸要素之间的关系并读懂戏曲这门艺术呢?

在计算机中，数据元素并不是孤立、杂乱无序的，而是相互之间存在着某种联系。分析它们的特性及之间存在的关系，把数据组织为合理的结构，正是学习与研究数据结构的意义所在。数据结构不但在计算机原理、操作系统、算法与程序设计中有着重要应用，而且在现实社会的学习与生活中也随处可见。学习应用数据结构，结合问题需求进行抽象，可以降低问题解决方案的复杂度。例如，在了解戏曲艺术时，分析、挖掘戏曲诸要素之间关系的基本类型，将对读懂戏曲艺术起到事半功倍的作用。

本章将以主题项目“数解传统戏曲”开展学习，通过体验探索戏曲要素中蕴含的数据结构类型，理解基本数据结构的概念。编写程序实现数据结构的基本操作，进一步理解抽象数据类型对于数据处理的重要意义，深入体会数据结构的实际应用。

03 主题学习项目：数解传统戏曲

项目目标

围绕主题“数解传统戏曲”，从日常生活与学习中的现象出发，探索基本数据结构类型的本质，并在此基础上总结数据结构的应用规律，利用规律创新使用数据结构，完成项目报告。

1. 结合生活实际，理解线性表、栈、队列、字符串和二叉树的基本概念，掌握它们各自的基本特性。

2. 通过解决实际问题，掌握线性表、栈、队列和字符串的基本操作，编程实现项目学习任务。

3. 了解二叉树的基本操作及应用，深入体会数据结构与日常生活的息息相关，提高利用数据结构解决实际问题的能力。

项目准备

为完成项目，需做如下准备。

- 全班分成若干小组，建议每组3 ~ 5人，自拟主题（如民族服饰、饮食文化、交通管理和戏曲赏析等），明确目标和分工。
- 准备相关资料，初步了解线性表、栈、队列、字符串和二叉树的基本特征。
- 利用思维导图、Python语言环境和多媒体制作软件等工具更好地完成项目学习。

项目过程

在学习本章内容的同时开展项目活动。为了保证本项目的顺利完成，要在以下各阶段检查项目的进度。

1 确定主题

确定项目学习主题，并选用恰当工具梳理线性表的相关知识。

P63

2 完善方案

通过栈的学习，完善选定主题的学习方案，分析栈在生活中的应用实例。

P73

3 巩固新知

比较线性表、栈和队列的联系与区别，进一步梳理知识结构图，明确栈和队列都是操作受限制的线性表。

P83

4 阶段总结

总结线性数据结构的知识体系，完成项目学习报告的初步框架，提炼、整理报告素材，选定制作工具。

P94

5 总结表达

完成本章学习的数据结构知识体系图，以小组为单位制作、完善并提交项目学习报告。

P102

项目总结

完成本章学习，可采用演示文稿、视频和思维导图等多种形式提交学习报告。报告中应包括项目主题、项目学习内容与所学数据结构的内在联系，以及对数据结构多元化应用的体会。全班各小组之间进行交流展示。

3.1 线性表

学习目标

- 理解线性表的基本特征和操作，感受其应用价值。
- 能用自然语言和流程图描述线性表的基本操作，并编程实现。

体验探索

戏曲艺术中的数据结构

中国的戏曲艺术由文学、音乐、舞蹈、美术、武术、杂技以及表演等融合而成，其特点是将众多艺术形式以一种标准聚合在一起，在共性中展现各自的特性。经过长期的发展演变，逐步形成了以京剧（图3.1.1）、豫剧、越剧、评剧和黄梅戏五大剧种为核心的中华戏曲百花苑。

图3.1.1　京剧脸谱艺术

思考：

1. 用列举法描述戏曲包含的艺术门类以及戏曲核心剧种。
2. 用列举法描述的事物个体之间存在怎样的关系。

3.1.1 线性表的概念

日常学习和生活中，具有相同属性的事物常被归为一类，放到一个集合中。例如，{1，2，3，4，5，6，7，8}、{1996 年 6 月，1971 年 5 月，2001 年 6 月 }、{apple，pear，mango，orange}等集合的共性在于集合中的每个元素都具有相同的属性，并且是以线性排列方式呈现的。有一种数据结构，它的某些特点与集合类似，这就是线性表。

思考活动

线性表与集合的区别

京剧、豫剧、越剧、评剧和黄梅戏是我国群众基础较为广泛的五大剧种。如果用集合来表示，可以表示为：{京剧，豫剧，越剧，黄梅戏，评剧}、{豫剧，评剧，京剧，越剧，黄梅戏}、{评剧，京剧，黄梅戏，豫剧，越剧}等多种形式。

思考：

1. 五大剧种的多种集合表示方法对应了几个集合？说明理由。

2. 如果将五大剧种书写成（京剧，豫剧，越剧，黄梅戏，评剧）、（豫剧，评剧，京剧，越剧，黄梅戏）、（评剧，京剧，黄梅戏，豫剧，越剧）线性表的方式，它们对应了几个线性表？请思考原因。

线性表的定义

线性表是由n（$n \geqslant 0$且为整数）个相同类型的数据元素组成的有限序列，其中，n称为线性表的长度。当n=0时，称线性表为空表。通常，我们将非空的线性表表示为（a_1，a_2，…，a_n）。线性表是一种简单且常用的数据结构，其抽象数据类型定义如图3.1.2所示。

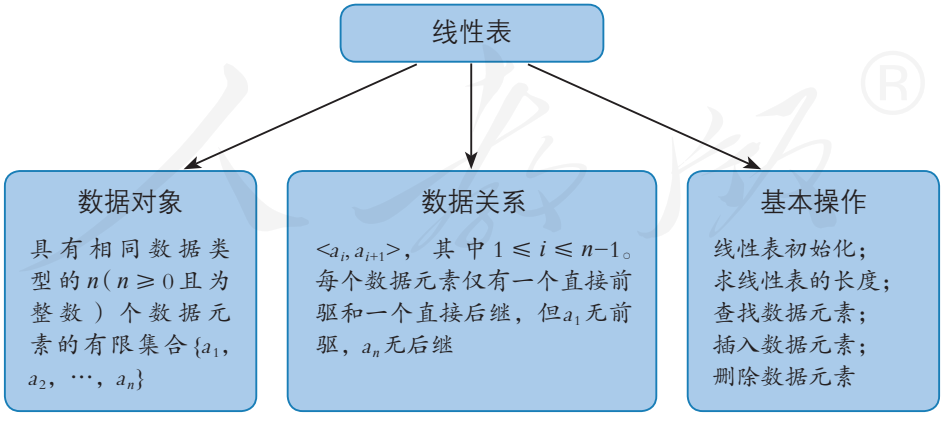


图3.1.2　线性表的抽象数据类型定义示意图

在线性表中，除常见的数字和字符之外，数据元素还可以是一本书、一出戏或一首歌。例如，（A，B，C，D）就是一个由4个英文字母构成的线性表，其中每一个英文字母都是一个数据元素。又如，可用线性表将部分高中科目表示为（语文，数学，英语，物

理，化学，生物，政治，地理，历史，信息技术）。在较为复杂的线性表中，每个数据元素还可以包含若干数据项。

线性表的特点

一个具有 n（$n>0$ 且为整数）个数据元素的非空线性表（a_1，a_2，…，a_n）具有以下特点：

- 有且仅有一个开始元素 a_1，a_1 没有前驱；
- 有且仅有一个终端元素 a_n，a_n 没有后继；
- 除 a_1 外，每个数据元素均有一个直接前驱；除 a_n 外，每个数据元素均有一个直接后继。

例如，线性表（语文，数学，英语，物理，化学，生物，政治，地理，历史，信息技术）中，数据元素之间的相互关系如图 3.1.3 所示。

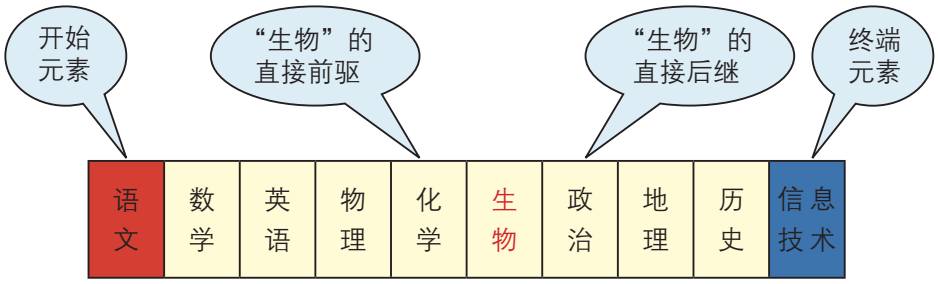


图 3.1.3　线性表中数据元素间的关系

“语文”是该线性表的开始元素，它没有前驱，有且仅有一个直接后继“数学”；“信息技术”是该线性表的终端元素，它没有后继，有且仅有一个直接前驱“历史”；其余元素有且仅有一个直接前驱和一个直接后继，如“生物”的直接前驱是“化学”，直接后继是“政治”。

实践活动　戏曲角色与线性表

尝试用线性表表示图 3.1.4 的相关信息，并从数据对象、数据关系和基本操作三个方面分析构造的线性表。

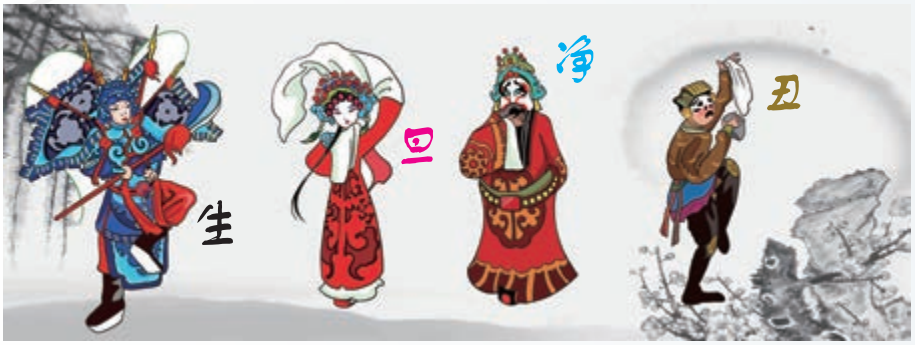


图 3.1.4　戏曲角色图

3.1.2 线性表的实现

线性表的基本操作包括线性表的初始化、求线性表的长度、查找线性表中的数据元素、在线性表中插入和删除数据元素等。通过这些基本操作，可以实现对数据的有效管理。

思考活动

利用线性表管理演出信息

对于戏迷票友来说，亲临剧场欣赏名家名段是美好的享受。为方便观看演出，他们可使用演出表管理演出信息。演出表的主要内容有演出时间、演出地点和演出团体等。

思考：

演出表的制作过程与图3.1.2中线性表的基本操作有何对应关系？完成表3.1.1。

表3.1.1　演出表的制作过程与线性表的基本操作对应关系表

演出表制作过程	线性表基本操作
准备一张便笺纸	建立一个空线性表
收集近期演出剧目信息并记录在便笺纸上	逐个向线性表中插入元素
统计信息表中的演出次数	
将看完的演出从列表中删除	
在划掉信息之前，需要查找到相关信息	

线性表的顺序存储

线性表的顺序存储指用一段地址连续的内存单元依次存储线性表中的数据元素。顺序存储的线性表称为顺序表。

1. 顺序表的初始化

构造一个容量为n（n为正整数）的空顺序表，即系统为线性表分配一段地址连续的内存单元。在高级程序设计语言中，一般用一维数组来实现顺序表。若数组名为a，则可存放8个数据元素的空顺序表如图3.1.5所示。注意，$a[0]$对应于线性表中的数据元素a_1。

图3.1.5　空顺序表示意图

例如，近期演出表（定军山，四郎探母，穆桂英挂帅，空城计）目前只有4个数据元素，后期还会有新的剧目加入。如果对该表采用顺序表方式存储，初始化时定义的数组元素个数需大于4，为后期进行插入等操作预留足够的存储空间，防止数组“溢出”。

在Python中，可以用“类”来定义一个空顺序表，其核心代码如下：

```
class Seqlist:  # 定义一个顺序表
    def __init__(self, max_size): # 初始化顺序表
        self.max_size = max_size  # max_size为能够存储的最多元素个数，即顺序表的容量
        self.data = []            # 当前顺序表为空
        self.last = -1            # 用于记录当前元素的下标
```

2. 求顺序表的长度

返回顺序表中存储的数据元素的个数，即顺序表的长度。图3.1.6所示的顺序表的长度为7。

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
36	78	16	25	88	9	16	

图3.1.6　有7个数据元素的顺序表

例如，用数组a存放近期演出表（定军山，四郎探母，穆桂英挂帅，空城计），则a[0]中存放“定军山”、a[1]中存放“四郎探母”、a[2]中存放“穆桂英挂帅”、a[3]中存放“空城计”，对该线性表执行求长度操作，则返回值应为“4”。

要定义求顺序表长度的函数，其核心代码如下：

```
def length_of_seqlist(self):      # 返回顺序表的当前元素个数，即当前的长度
    return self.last + 1
```

3. 在顺序表中查找某个数据元素

在顺序表中从第1个（数组下标为“0”）数据元素开始，查找值为“x”的数据元素，若查找成功，则返回该数组元素在顺序表中首次出现的数组下标；否则，返回特殊值“−1”，标志查找失败。

例如，若要在图3.1.6所示的顺序表中查找“16”，则返回值为“2”。该表中，a[2]和a[6] 中存放的数据元素均为“16”，但返回的是“16”在该表中首次出现的数组下标。因此，本次查找“16”的操作返回值应为“2”。若查找“49”，则返回值为“−1”，表示查找失败，该表中没有“49”这个数据元素。

又如，在近期演出表（定军山，四郎探母，穆桂英挂帅，空城计）中查找“定军山”，返回值应为“0”，查找“智取威虎山”则返回值为“−1”。

要定义在顺序表中查找某个数据元素的函数，其核心代码如下：

```
def search_data(self, x):         # 在顺序表中查找数据元素x
    for i in range(self.last + 1):
        if self.data[i] == x:
            return i
    print("你所查找的元素不存在")
    return -1
```

4. 在顺序表中插入一个数据元素

在已存有n（$n \geqslant 0$且为整数）个数据元素的顺序表a的第i（$1 \leqslant i \leqslant n+1$）个位置上插入一个值为“$x$”的新元素，插入后，表的长度增加1。为保持顺序表的存储特点，当$i \neq n+1$时，必须先将表中从a_i到a_n的$n-i+1$个元素从原来的存储位置依次后移一个存储单元，以为新插入的元素腾出存储空间。

在顺序表中插入元素是较为复杂的操作，插入位置分为表头、表中和表尾三种不同的情况。图3.1.7、图3.1.8、图3.1.9分别展示了在如图3.1.6所示的顺序表中的不同位置插入数据元素“56”时数组的变化情况，图中蓝色部分为存储位置发生变化的数据元素。

■ 若在表头即第1个位置上插入数据元素“56”，则插入后的顺序表如图3.1.7所示。

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
56	36	78	16	25	88	9	16

图3.1.7　在顺序表的表头位置插入数据元素“56”

■ 若在表中第4个位置上插入数据元素“56”，则插入后的顺序表如图3.1.8所示。

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
36	78	16	56	25	88	9	16

图3.1.8　在顺序表的表中位置插入数据元素“56”

■ 若在表尾即第8个位置上插入数据元素“56”，则插入后的顺序表如图3.1.9所示。

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
36	78	16	25	88	9	16	56

图3.1.9　在顺序表的表尾位置插入数据元素“56”

要定义在顺序表中插入一个数据元素的函数，其核心代码如下：

```
def insert_list(self, i, x):                    # 在顺序表中的i位置插入数据元素x
    self_length = self.length_of_seqlist()      # 求顺序表的长度
    # 当插入位置小于0或大于顺序表长度+1时，返回错误
    if i < 0 or i > self_length + 1:
        return false
    if i < self_length:
        self.data.append();
        # 循环，将指定位置后面的数据元素依次向后移一位
        for k in range(self_length, i - 1, -1):
            self[k] = self[k-1]
        self[i - 1] = x
        self.last = self.last + 1
```

5. 在顺序表中删除一个数据元素

在已存有n（$n \geqslant 0$且为整数）个数据元素的顺序表a中删除第i（$1 \leqslant i \leqslant n$）个位置上的数据元素$a_i$，删除后，表的长度减少1。为保持顺序表的存储特点，删除操作必须将表中从a_{i+1}到a_n的$n-i$个元素从原来的存储位置上依次前移一个存储单元。

与插入元素操作相同，在顺序表中删除元素同样分为表头位置、表中位置和表尾位置三种不同的情况。图3.1.10、图3.1.11、图3.1.12分别展示了在如图3.1.6所示的顺序表中的不同位置删除数据元素时数组的变化情况，图中蓝色部分为存储位置发生变化的数据元素。

■ 若删除表头即第1个位置上的数据元素“36”，则删除后的顺序表如图3.1.10所示。

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
78	16	25	88	9	16		

图3.1.10　删除顺序表的表头位置数据

■ 若删除表中第4个位置上的数据元素“25”，则删除后的顺序表如图3.1.11所示。

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
36	78	16	88	9	16		

图3.1.11　删除顺序表的表中位置数据

■ 若删除表尾即第7个位置上的数据元素“16”，则删除后的顺序表如图3.1.12所示。

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
36	78	16	25	88	9		

图3.1.12　删除顺序表的表尾位置数据

要定义在顺序表中删除一个数据元素的函数，其核心代码如下：

```
def delete_data(self, x):                    # 从表中删除一个数据元素x
    if self.length_of_seqlist() == 0:
        print ("表已空，无法删除元素！")
    else:
        index = -1
        for item in self.data:
            index = index + 1
            if x == item:
                break
        for i in range(index, self.last):
            self.data[i] = self.data[i + 1]
        self.last = self.last - 1
```

采用顺序存储方式时，存在插入、删除操作需要移动大量元素的问题，并且一旦分配了存储空间就难以改变，因此分配较多的存储空间会造成浪费，分配空间不足难以扩充，此时可以采用链式存储解决这些问题。

线性表的链式存储

线性表的链式存储指用一组任意（不要求连续）的内存单元存储线性表中的数据元素，元素之间的逻辑关系用指针表示。由多个节点链接成的序列称作链表。由于每个节点只含一个指针域，所以称为单链表。

例如，有线性表（京剧，越剧，评剧，豫剧，川剧，曲剧），若采用链式存储方式，则6个节点可以链接成如图3.1.13所示的单链表，其中Head指向第一个元素节点，称为头指针。

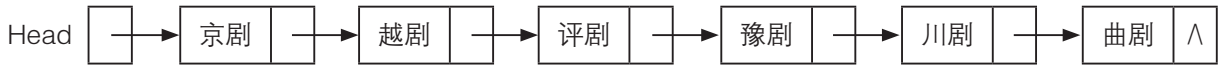


图3.1.13　戏曲种类单链表示意图

与顺序存储不同，链式存储有如下特点：

■ 链式存储不要求存储空间连续，也无须事先指定最大长度，因此链表的初始化不需要考虑“溢出”问题。

■ 链式存储通过指针域完成节点的链接，因此插入和删除操作均不需移动节点的存储位置，只需修改指针，操作十分方便。

■ 可以在首元素节点之前加一个“头节点”，并充分利用头节点的数据域存放数据元素节点个数，便于链表求长度等操作的高效实现。

■ 链式存储中的指针占用了较多的存储空间，存储密度低。

实践活动　探究链式存储线性表的基本操作

结合第2章关于链表的基本操作的介绍，自主探究实现线性表链式存储的基本操作，将表3.1.2填写完整。

表3.1.2　线性表链式存储基本操作汇总

基本操作	操作思路	操作示意图	核心代码
线性表初始化			
求线性表的长度			
在线性表中查找某个数据元素			
在线性表中插入一个数据元素			
在线性表中删除一个数据元素			

阅读拓展　双链表和循环链表

单链表在实现数据存储和管理方面虽然有一定的优势，但也存在局限性。例如，查找操作需要从表头开始逐一进行元素比较，操作效率非常低。在实际应用中，我们可以通过双链表和循环链表来提高链表的操作效率。

1. 双链表

■ 若链表中每个节点都有两个指针域，分别指向直接后继和直接前驱，这样的链表就是双链表，如图 3.1.14 所示为具有 n 个节点的双链表，其中“prior”表示指向直接前驱的指针，“next”表示指向直接后继的指针。

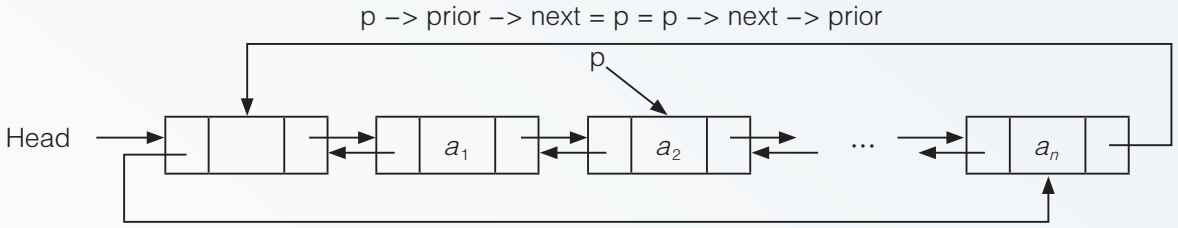


图 3.1.14　具有 n 个节点的双链表示意图

■ 双链表在查找、删除前驱节点时通过向前的指针可以直接实现，非常方便。

■ 双链表在简化某些操作的同时，由于每个节点均要增加一个前驱节点的指针域，增加了存储空间。

2. 循环链表

■ 在单链表中，将最后一个节点的指针域指向头节点，使整个链表形成一个环，这样的链表就是单循环链表，如图 3.1.15 所示为具有 n 个节点的单循环链表。

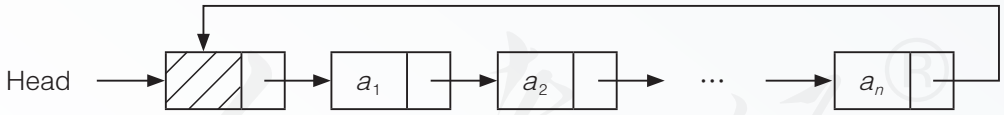


图 3.1.15　具有 n 个节点的单循环链表示意图

■ 单循环链表中没有空指针，当进行遍历操作时，其终止条件就不再是判别当前节点的指针域是否为空，而是判别它们是否等于某一指定指针（如头指针或尾指针等）即可。

■ 在单链表中，只能访问到当前节点及其后续节点，无法直接找到该节点之前的其他节点。而在单循环链表中，从任一节点出发都可访问到表中所有节点。

■ 在双链表中，将最后一个节点的后继指针域指向头节点，将头节点的前驱指针指向尾节点，使整个链表形成一个环，这样的链表就是双循环链表。

3.1.3 线性表的应用

利用线性表中数据元素类型相同的特点可以指导信息分类，如戏曲分类、化学元素分类等；利用线性表的基本操作可以实现信息管理，如通信录、备忘录等。

思考活动

利用线性表制作旅游攻略

旅游是日常生活中的休闲方式之一。旅游之前，通常会制作旅游攻略，合理安排出行。

思考：

参考表3.1.3中的提示信息，尝试利用线性表相关知识制作一份旅游攻略。

表3.1.3 旅游攻略内容与线性表的关系

旅游攻略内容	用线性表表达主要内容
旅游线路	景点名称（西湖，苏州园林，太湖……）
日程安排	
必备物品	
特产信息	
紧急通信	
⋮	

利用线性表特点制作通信录的主要过程

1. 需求分析

确定简单通信录的主要功能，包括：建立空通信录、添加联系人、删除联系人、查找联系人等。

2. 选择数据结构

逻辑结构：根据需求特点选择线性表。

物理结构：数组，在Python中，可通过列表类型“list”来实现。

核心代码如下：

```
class Person:                          # 定义Person类
    contacts = []                      # 定义Listcontacts
```

3. 定义主要功能

■ 添加联系人

核心代码如下：

```
def add(self):
    # 按照姓名的顺序，在适当的位置插入一条新联系人记录
```

```
        name = input("请输入要添加的联系人姓名：")
        index = len(Person.contacts)
        for i in range(len(Person.contacts)):
            if Person.contacts[i]["姓名"] > name:
                index = i
            telephone = input("请输入联系人电话号码：")
            addr = input("请输入联系人地址：")
            label = {"姓名": name, "电话": telephone, "地址": addr}
            Person.contacts.insert(index, label)
            self.write()
```

■ 删除联系人

核心代码如下：

```
    def delete(self):
        # 删除第一个名字为输入内容的联系人记录
        name = input("请输入要删除的联系人姓名：")
        index = -1
        for i in range(len(Person.contacts)):
            if Person.contacts[i]["姓名"] == name:
                index = i
        if index > -1:
            Person.contacts.remove(Person.contacts[index])
            print("%s" % "\n".join([x["姓名"] for x in Person.contacts]))
            self.write()
        else:
            print("联系人 %s 不存在" % name)
```

■ 查找联系人

核心代码如下：

```
    def search(self):
        # 查找符合搜索姓名的联系人记录
        name = (input("请输入要搜索的联系人姓名："))
        contacts = self.find_by_name(name)
        if len(contacts) > 0:
            for contact in contacts:
                print("联系人 %s 的电话号码是 %s , 地址是 %s" % (
                    contact["姓名"],
                    contact["电话"],
                    contact["地址"]))
        else:
            print("联系人 %s 不存在" % name)
```

除以上主要功能外，完整的通信录还应具备修改联系人信息、显示联系人信息等功能。

4. 定义用户界面

编写程序满足用户需求，应该通过简洁、明确和友好的程序界面为用户提供服务。定义通信录用户界面的核心代码如下：

```
    def menu():
        print('''系统提供以下功能
        1: 添加
        2: 删除
        3: 修改
        4: 搜索
        5: 退出
        6: 显示全部联系人信息''')
```

5. 调试程序，修改完善。

实践活动

完善通信录制作程序

结合前面的程序设计，正确填写①、②中缺失的内容，将下列程序补充完整，实现通信录管理。

```
people = Person()
people.read()
while ____①____:
    try:
        menu()
        choice = int(input("请输入相应数字操作："))
        if choice == 1:
            people.add()
        elif choice == 2:
            people.delete()
        elif choice == 3:
            people.modify()
        elif choice == 4:
            people.search()
        elif choice == 5:
            people.write()
            ____②____
        elif choice == 6:
            people.show()
        else:
            print("输入不合法，请输入合法数字")
    except ValueError:
        print("请输入数字选项")
```

技术支持

Python 中的文件读写功能

Python 中文件操作的基本方法：

■ open( ) 方法用于打开文件

open( ) 可以实现对文本文件和二进制文件的打开，如果文件存在，则打开文件对象；否则，返回错误信息。

■ close( ) 方法用于关闭文件

close( ) 关闭当前使用的文件，释放程序运行过程中，该文件数据所占据的内存空间。

■ read( ) 方法用于读取文件

read( ) 可以读取整个文件，通常用于将文件内容放到一个字符串变量中。

■ write( ) 方法用于写文件

write( ) 方法用于向文件中写入指定字符串。

项目实施

确定主题

一、项目活动

1. 在充分做好项目准备的基础上确定小组项目学习主题，找到该主题中与线性表对应的内容，参照图 3.1.16 的样例将图 3.1.17 填写完整。

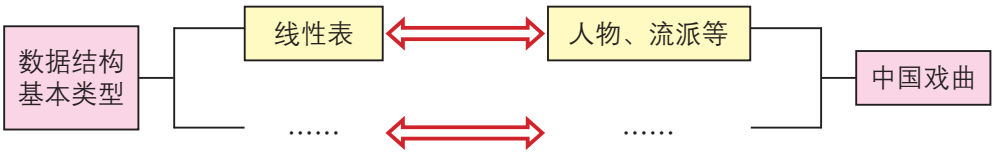


图 3.1.16　中国戏曲内容与数据结构类型对应关系示意图

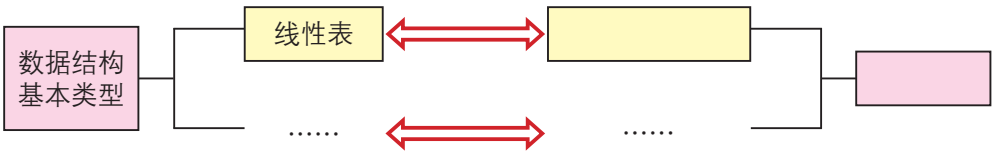


图 3.1.17　自拟主题内容与数据结构类型对应关系示意图

2. 在确定小组项目学习主题的基础上，总结线性表的学习经验，以中国戏曲项目学习方案（图 3.1.18）为例，自行设计项目学习方案。

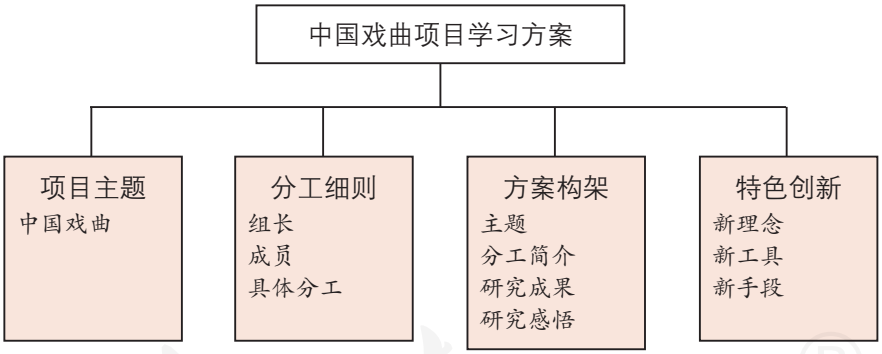


图 3.1.18　中国戏曲项目学习方案示例

二、项目检查

1. 梳理线性表相关知识，撰写阶段性项目学习报告。

2. 在班内交流各组提交的项目学习报告。

练习提升

1. 为了使市民出行更加便利，相关部门经过调查研究，准备在某条公交线路上新增若干站点。如果选择线性表来设计、规划线路，请问：是否合理？应选择哪种存储结构？说明理由。

2. 现有5个数据元素组成的线性表（35，19，56，22，43），采用链式存储的方式存放在200 ~ 219号内存单元中，每个数据元素的数据域和指针域分别占用2字节，如图3.1.19所示。

图3.1.19　内存中的某单链表示意图

其中，A、B、C、D、E均为指针，它们的值分别是多少？

3.2
操作受限的线性表——栈

学习目标 ▶▶▶

- 理解栈的概念及其特点。
- 掌握栈的基本操作，学会利用栈来解决实际问题，感受栈的应用价值。

体验探索

变脸绝活，先贴后扯

变脸是中国戏曲绝活中的代表之一，浓缩了戏曲文化的精华。变脸的方法大体分为抹脸、吹脸和扯脸三种。其中，扯脸比较复杂，要事先将脸谱画在一张张绸子上，然后一张一张地将脸谱贴在脸上，表演时再在舞蹈动作的掩护下逐张扯下来。

思考：

根据图3.2.1所示，结合表3.2.1进行对比，进一步体会贴脸顺序和扯脸顺序的先后关系，在表3.2.1中继续添加其他可能情况。

①　②　③　④　⑤

图3.2.1　面具及其编号

表3.2.1　贴脸顺序与扯脸顺序对比表

贴脸顺序	扯脸顺序
①②③④⑤	⑤④③②①
②④⑤①③	③①⑤④②
④①③②⑤	⑤②③①④
⋮	⋮

3.2.1 栈的概念

在本节体验探索提到的变脸表演中，先被贴在脸上的脸谱会后被扯下来，后被贴在脸上的脸谱则会先被扯下来。生活中有很多现象与变脸类似，如图3.2.2所示为在碗柜中放碗和取碗的示意图。

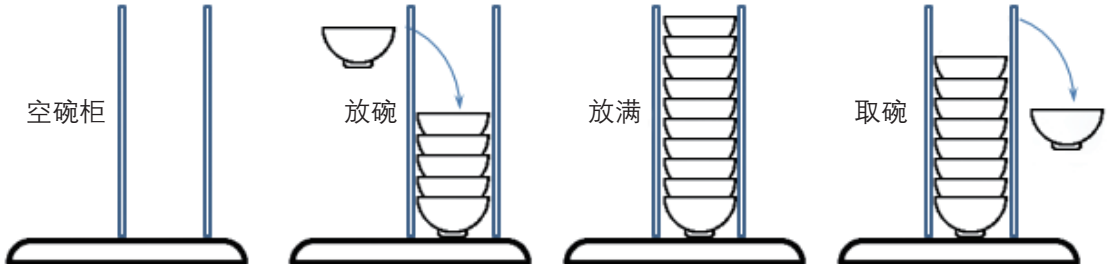


图3.2.2　放碗、取碗示意图

变脸表演和放碗、取碗的过程均与数据结构中栈的操作特点类似，即“后进先出”。

思考活动

生活中的栈

生活中的许多现象都具有栈的特性。例如，调整举重器械杠铃（图3.2.3）的重量，杠铃的重量可以通过更换不同重量的杠铃片实现。更换杠铃片的顺序总是先从最外侧的一片开始，不难想象，最外侧的杠铃片是当初最后装入的那一片，符合栈“后进先出”的特性。

图3.2.3　举重器械杠铃

思考：

日常生活中还有哪些类似现象？

栈是限定仅能在一端进行插入和删除操作的特殊线性表。允许插入和删除的一端称为栈顶，另一端称为栈底，如图3.2.4所示，其中，a_n是栈顶元素，a_1是栈底元素。不含任何数据元素的栈称为空栈。

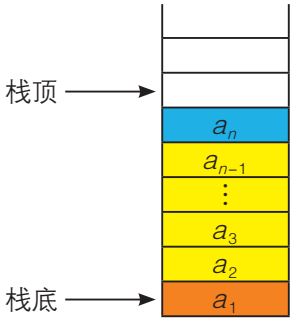


图3.2.4　栈的示意图

向栈中插入元素的操作称为入栈，删除栈顶元素的操作称为出栈。入栈、出栈的示意如图3.2.5所示。

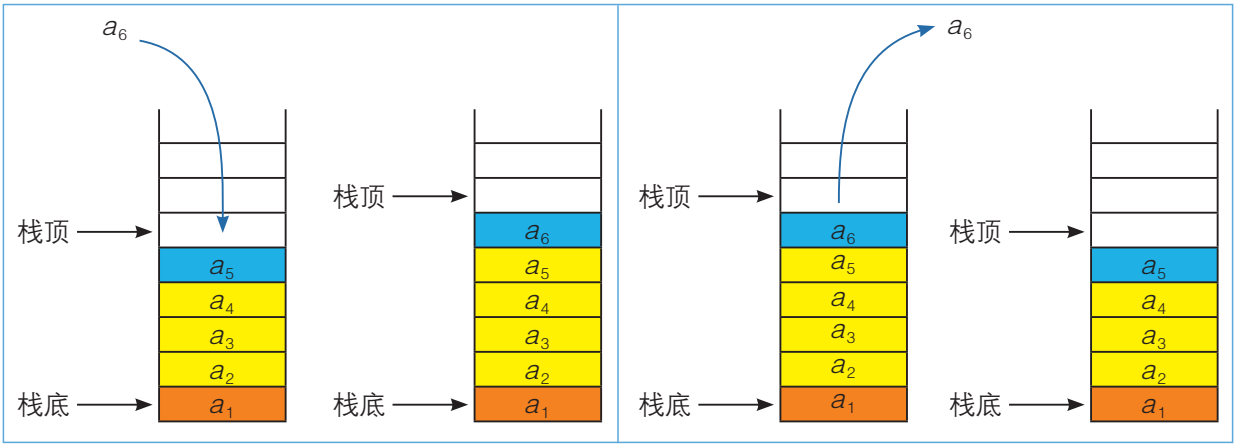


图3.2.5　入栈、出栈示意图

由于栈限定了只能在栈顶进行插入和删除操作，即总是栈顶的数据元素成为操作对象，因而形成了这样一个特点：后入栈的元素先出栈，先入栈的元素后出栈，即“后进先出”或“先进后出”。栈是操作受限制的线性表，因此，在栈的抽象数据类型定义中，数据对象和数据关系的定义与线性表相同，而基本操作的定义与线性表不同。栈的抽象数据类型定义如图3.2.6所示。

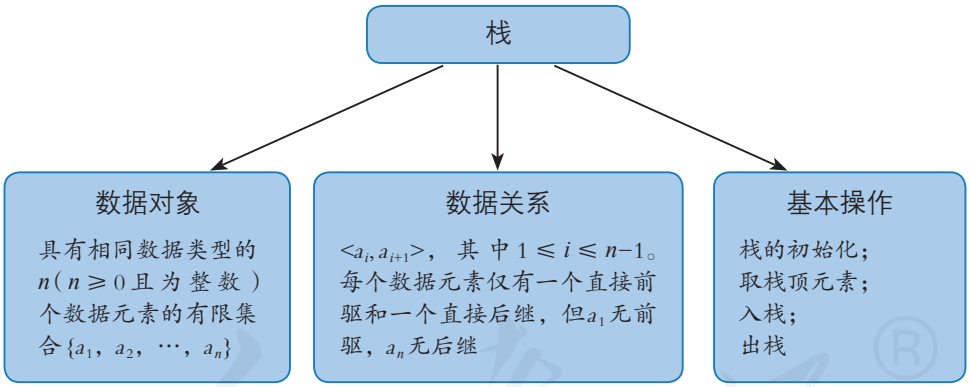


图3.2.6　栈的抽象数据类型定义示意图

实践活动

栈的先进后出

假设入栈序列为A、B、C、D，入栈过程中允许出栈，则可以得到“D、C、B、A”和“A、B、C、D”等出栈序列。

1. 画出“D、C、B、A”和“A、B、C、D”两个出栈序列的入栈、出栈过程示意图。

2. 判断能否得到“A、D、B、C”和“A、D、C、B”出栈序列，并说明原因。

3.2.2 栈的实现

栈在生活中的应用十分广泛，通过限定线性表数据元素先进后出可解决很多实际问题。

思考活动

列车调度与栈

列车进站、出站的调度采用了栈的原理，采用双车头“后进先出”完成调度任务，如图3.2.7所示。列车调度员根据时间和线路调度不同车次依次进站、出站，完成复杂的调度任务，保证运行安全。

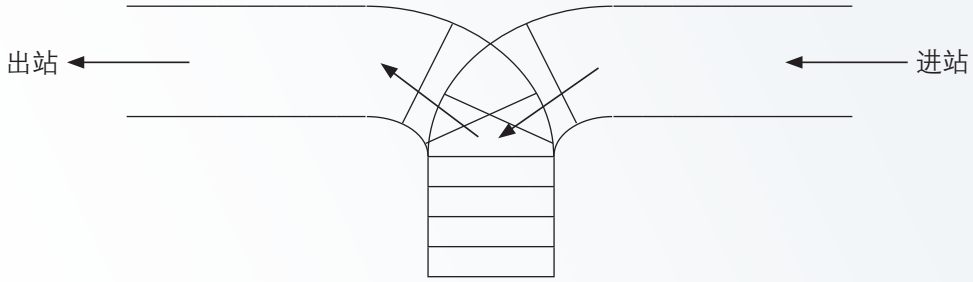


图3.2.7　列车调度示意图

思考：

如何利用栈的基本操作实现列车的调度？

栈的顺序存储

栈的顺序存储指用一段地址连续的内存单元依次存储栈中的数据元素。顺序存储的栈称为顺序栈。

1. 顺序栈的初始化

为顺序存储的栈分配连续的存储空间，同时置栈空。容量为5的空顺序栈如图3.2.8所示。

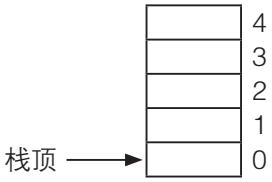


图3.2.8　容量为5的空顺序栈示意图

顺序栈的初始化操作的核心代码如下：

```
def __init__(self, size):
    self.data = []
    self.max_size = size
    self.top = 0
```

在顺序存储栈时，可以额外设定一个变量，用于判断栈的状态，如上段代码中的变量“top”。通常情况下，top用于标识栈顶元素的下一个位置，该变量值为“0”时，表示栈为空的状态；该变量等于栈的容量时，表示栈已满的状态。

2. 取顺序栈的栈顶元素

先判断栈是否为空，若栈为空，则输出栈空提示；若栈非空，则返回当前栈顶元素的值。该操作不改变栈的状态，即栈顶不变。核心代码如下：

```
def get_top(self):
    if self.top == 0:
        print ("栈为空")
    else:
        return self.data[self.top - 1]
```

3. 顺序栈的入栈

对于顺序栈来说，由于其容量已经确定，所以执行入栈操作时应先判断栈的状态。若栈已满，则输出栈满提示；若栈未满，将数据元素存入栈顶，然后标识栈顶的变量top的值加1。图3.2.9给出了数据元素“4”和“8”连续入栈后栈的变化示意。

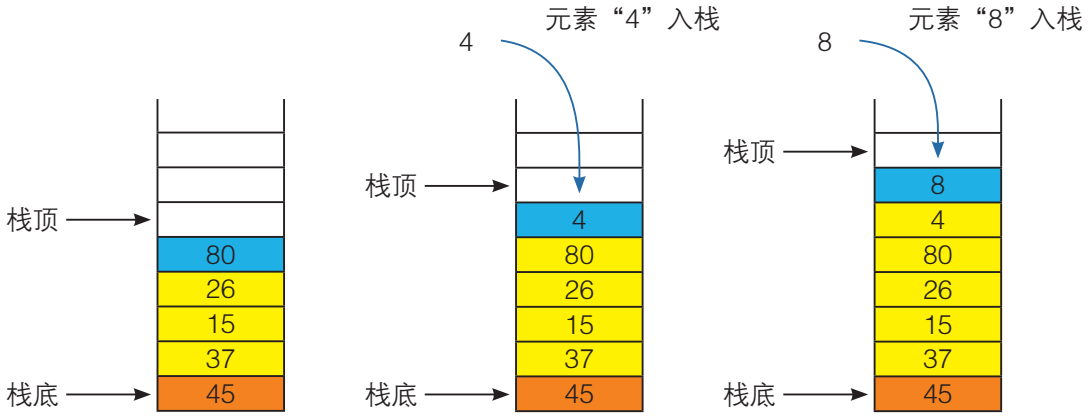


图3.2.9　连续入栈示意图

顺序栈的入栈操作的核心代码如下：

```
def push(self, element):
    if self.top  == self.max_size:
        print("栈已满")
        return
    else:
        self.data.append(element)
        self.top = self.top + 1
```

4. 顺序栈的出栈

执行出栈操作时，需先判断栈是否为空，若栈为空，则输出栈空提示；若栈非空，则返回栈顶元素的值，同时栈顶指针减1。图3.2.10给出了顺序栈中数据元素“8”和“4”连续出栈后栈的变化示意。

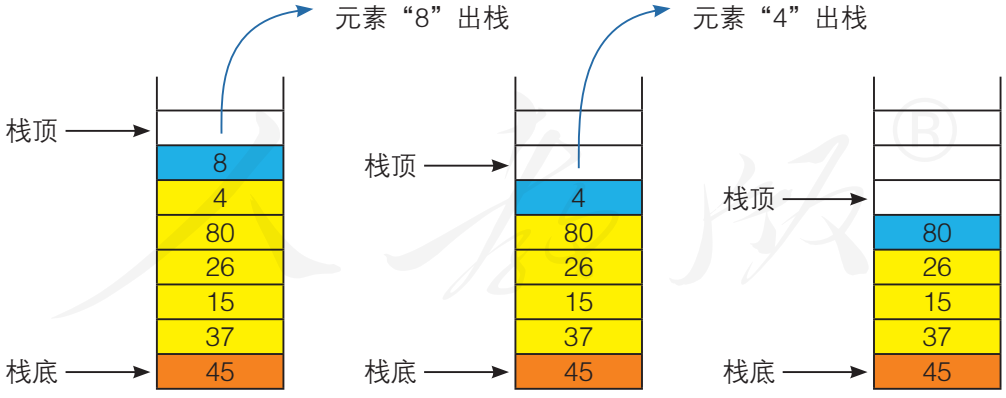


图3.2.10　连续出栈示意图

顺序栈的出栈操作的核心代码如下：

```
def pop(self):
    if self.top == 0:
        print("栈为空")
    else:
        self.top = self.top - 1
        return self.data[self.top]
```

栈的链式存储

栈的链式存储指用一组任意（不要求连续）的内存单元存储栈中数据元素及数据元素间的关系。链式存储的栈称为链栈。与单链表类似，链栈不需要像顺序栈那样在初始化时就指定最大长度，因此无须考虑栈满的情况。与单链表不同的是，链栈只能在头指针处进行操作。在Python语言中，可先通过类定义链栈的节点。

定义链栈节点的核心代码如下：

```
# 声明一个节点类
class Node(object):
    def __init__(self, data = None):
    self.data = data
    self.next = None
```

1. 链栈的初始化

定义一个栈顶指针，置栈顶指针为空，同时置链栈数据元素个数为“0”。

链栈的初始化操作的核心代码如下：

```
# 声明一个链栈类
class LinkStack(object):
    def __init__(self):
        self.top = Node(None)
        self.count = 0
```

2. 取链栈的栈顶元素

若栈非空，则返回栈顶元素节点数据域的值，栈顶指针不变；否则，输出栈空提示。

取链栈的栈顶元素的核心代码如下：

```
def get_top(self):
    return self.top.data
```

3. 链栈的入栈

由于链栈是一种操作受限制的单链表，其插入和删除操作都在栈顶指针位置进行，因此，入栈操作即在栈顶指针位置插入一个数据元素。插入数据元素为新的栈顶元素，栈顶指针发生改变。

链栈的入栈操作的核心代码如下：

```
def push(self, element):
    tmp = Node(element)
    if self.count == 0:
        self.top = tmp
    else:
        tmp.next = self.top
        self.top = tmp
    self.count = self.count + 1
```

4. 链栈的出栈

先判断栈是否为空，若栈非空，返回栈顶元素节点数据域的值，并将栈顶元素节点从栈中删除，栈顶指针发生改变；否则，输出栈空提示。

链栈的出栈操作的核心代码如下：

```
def pop(self):
    if self.count == 0:
        print("栈为空!")
    else:
        self.count = self.count- 1
        element = self.top.data
        self.top = self.top.next
        return element
```

实践活动

利用栈判断回文句

“青岛绿草坡草绿岛青”正着读和反着读是相同的，这样的句子称为“回文句”。试写出运用栈实现“判断语句是否为回文句”的基本思路。

3.2.3 栈的应用

文档编辑软件和网页浏览器都是利用栈的“后进先出”特点记录用户的操作顺序，从而为用户提供后退或撤销操作，以方便用户使用，如图3.2.11所示。

图3.2.11　文档编辑软件和网页浏览器按钮示意图

思考活动

利用栈实现表达式求值

“a*b”称为中缀表达式，其特点是操作符位于操作数中间；“ab*”称为后缀表达式，其特点是操作符位于操作数之后。中缀表达式的求值过程就是将中缀表达式转换为后缀表达式，然后利用栈对后缀表达式进行计算。

思考：

1. 如何将中缀表达式“15+2*3”转换为后缀表达式？
2. 为什么可以使用栈来实现后缀表达式的计算？

在计算机中，数制转换的基本方法是“除R反向取余法”，其中，R代表进制，如2、8、16等。在数制转换过程中，先得到的余数后输出，可用栈作容器，暂存余数。

表3.2.2为采用“除8反向取余法”将$(1348)_{10}$转换为$(2504)_8$的计算过程。

表3.2.2　将（1348）$_{10}$转换为（2504）$_{8}$的计算过程

除以8后的商	除以8后的余数
168	4
21	0
2	5
0	2

将运算过程中的余数4，0，5，2依次入栈，当商为0时，对栈进行出栈操作，直至栈空为止，就得到了“2，5，0，4”这个数字序列，正好是转换后的八进制数“2504”。

实践活动

编程实现十进制到八进制的转换

1. 选择合适的数据结构，并画出流程图。

2. 将下列程序补充完整，并上机调试。

```
bb = []                          # 新建列表bb，用于存储转换后的八进制数
# 输入要转换的十进制数赋值给aa
aa = int(input("请输入要转换进制的数值："))
while ____①____:                 # 循环，判断aa是否大于0，若不大于0，则退出循环
    bb.append(aa % 8)            # 将数值对8取余，值存入bb列表中
    ____②____
bb.reverse()                     # 将列表反序
print("该数字转换为八进制后是：")
print(bb)  # 打印列表bb
```

阅读拓展

栈在编译中的作用

计算机程序通常需要经过编译才能执行，栈在编译中起着特殊作用。利用栈可以实现表达式的求值和表达式中的括号是否配对的检查。例如，“[()]”就是配对的，而“{[}]”就不配对。具体检查过程如下：

1. 若出现的是“左括号”，则进栈；

2. 若出现的是“右括号”，首先判断栈是否为空，若栈空，则表明该右括号是多余的；否则，与当前栈顶元素进行比较，若相匹配，则栈顶的左括号出栈，否则表明不匹配；

3. 当没有括号时，若栈空，则表明表达式匹配正确；否则，表明栈中的左括号是多余的。

项目实施

完善方案

一、项目活动

1. 根据图3.2.12中的提示，补充完善项目主题内容与基本数据结构的对应关系。

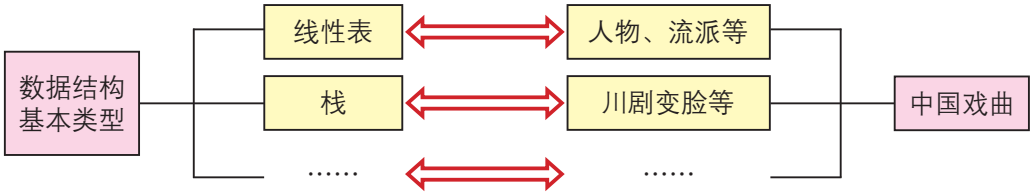


图3.2.12　中国戏曲内容与数据结构类型对应关系示意图

2. 以小组为单位，总结对栈的理解，寻找生活中栈的典型实例，填写表3.2.3。

表3.2.3　生活中栈的典型实例

学习方面	交通方面	饮食方面	个人爱好	其他方面
整理书籍	火车调度	糖葫芦	表演川剧变脸	……

二、项目检查

1. 进一步完善项目方案。

2. 通过分析栈在项目主题活动中的作用，丰富项目报告内容。

练习提升

1. 编写程序，利用栈将从键盘输入的十进制整数转换为十六进制数，并输出。

2. 一般情况下，求自然数n的阶乘$n!$，需要知道$(n-1)!$，依此类推，思考这一解题思路中是否用到栈的原理，小组讨论，说明理由。

3.3
操作受限的线性表——队列

学习目标 ▶▶▶

- 理解队列的概念，感受其应用价值。
- 掌握队列的基本操作，能用队列解决实际问题。

体验探索

龙套助阵，先进先出

龙套是传统戏曲中由演员扮演的仪仗、卫士、宫娥等角色，在戏曲中象征着千军万马或众多随从，对故事情节起到烘托和渲染的作用。以战场为例，双方的士兵排成两队，分别于舞台的上、下场门以“二龙出水”的动作列队出场，为主将加油助阵，如图 3.3.1 所示。

图 3.3.1　龙套助阵

思考：

龙套的退场顺序和进场顺序之间有什么关系？生活中还有哪些现象与戏曲中的龙套类似？

3.3.1 队列的概念

在日常生活中，我们经常会遇到为了维护社会正常秩序而需要排队的场景，如图3.3.2所示。

图3.3.2 排队购物、景区购票示意图

人们在购物结账、等待电梯等场合中，往往会遇到多人等待的情景。一般情况下，大家都会自觉排队，维持公共秩序。排队的规则是：新加入的成员只能排在队尾，而且队中全体成员只能按顺序向前移动，在到达队头得到服务后离队。以排队结账为例，后来的人在队伍的最后等待，排在最前面的人先付账，结账后从队伍的最前端离开。数据结构中的队列结构与上述排队情景类似。不同的是，在现实生活中排队时，队中任何成员可以中途离队；而对数据结构中的队列来说，是不允许“成员”中途离队的。

思考活动

汽车排队加油

随着城市里汽车数量的急速增长，汽车加油站也逐渐多了起来。通常，汽车加油站的入口和出口均为单行道，加油车道会有若干条，如图3.3.3所示。

思考：

汽车是如何在加油前、加油中和加油后利用队列实现有序加油的？

图3.3.3 汽车排队加油

队列是只允许在线性表的一端进行删除，而在另一端进行插入的特殊线性表。允许删除的一端称为队头，允许插入的一端称为队尾。当队列中没有数据元素时称为空队列。队列也称为先进先出表，如图3.3.4所示。

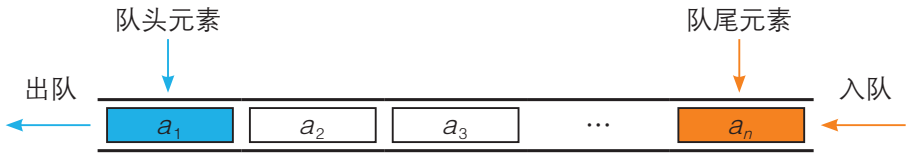


图3.3.4 队列示意图

队列也是操作受限制的线性表。因此，队列的抽象数据类型定义中，数据对象和数据关系的定义也与线性表相同，而基本操作的定义与线性表不同。队列的抽象数据类型定义如图3.3.5所示。

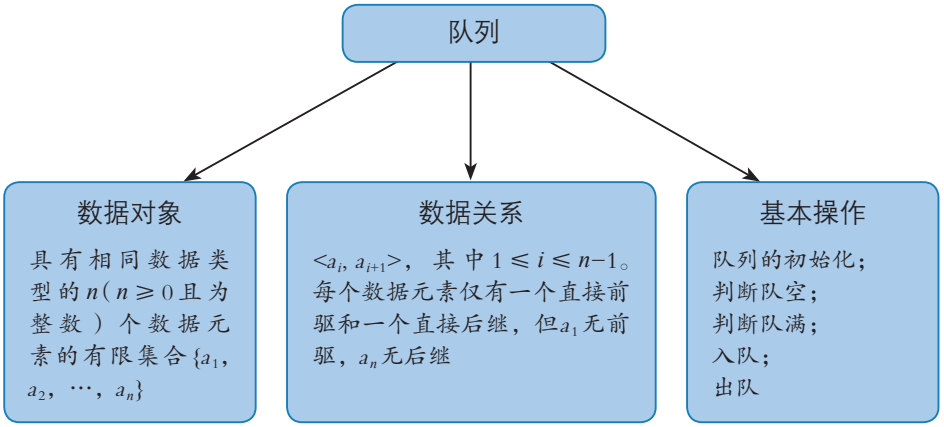


图3.3.5　队列的抽象数据类型定义示意图

例如，京剧中一般都把整出戏分成若干场。家喻户晓的《智取威虎山》共分10场，可以将这10场戏看作一个队列（第一场 乘胜进军，第二场 夹皮沟遭劫，第三场 深山问苦，……，第十场 会师百鸡宴），场次顺序也就是演出顺序，演员按照故事情节从第一场到第十场顺序演出。

实践活动

队列的应用

在实际生活中，队列的应用非常广泛，每个人都是队列的使用者和创造者。根据个人实际生活体验，梳理日常生活中队列的应用，感受队列应用的广泛性和实用性，填写表3.3.1。

表3.3.1　日常队列的应用

学习中的队列	生活中的队列	项目学习中的队列
体育课排队跳绳	排队点餐	龙套

3.3.2　队列的实现

队列的相关操作与其存储方式密不可分，其存储结构与线性表的存储结构基本相同。根据队列的特征，可以将队列的入队操作转化为在线性表的表尾插入数据元素，将队列的出队操作转化为在线性表的表头删除数据元素。

思考活动

操作系统管理与队列机制

操作系统是计算机系统软件的核心，实现对计算机整体的控制与管理。在作业、进程、线程的管理中，操作系统通常引入队列机制，以实现高效管理和统一调度。

思考：

结合上面的材料收集、整理信息，了解操作系统如何利用队列机制管理计算机，进一步体会队列的重要作用。

队列的顺序存储

队列的顺序存储指用一段地址连续的内存单元依次存储队列中的数据元素。顺序存储的队列称为顺序队列。

1. 顺序队列

■ 队头位置不变的入队与出队

假设一个队列有n（$n \geqslant 0$且为整数）个元素，则顺序存储的队列需要建立一个容量大于n的数组，并把队列的所有元素存储在数组的前n个单元，数组下标为“0”的一端即为队头。

入队，就是在队尾追加一个元素，只要此时数组中有空闲存储单元，不需要移动任何元素，即可完成入队操作，如图3.3.6所示。

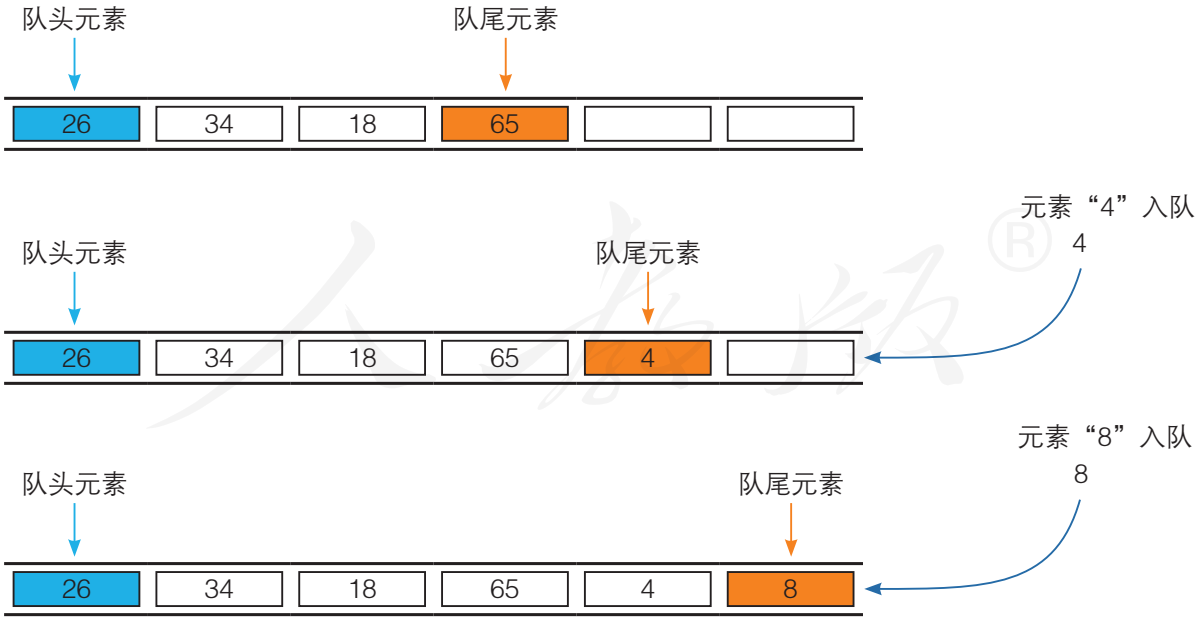


图3.3.6　数据元素连续入队示意图

出队操作是将队头元素删除，为了保持队头的位置不变，则队头之后的所有元素都要前移一个位置，如图3.3.7所示。

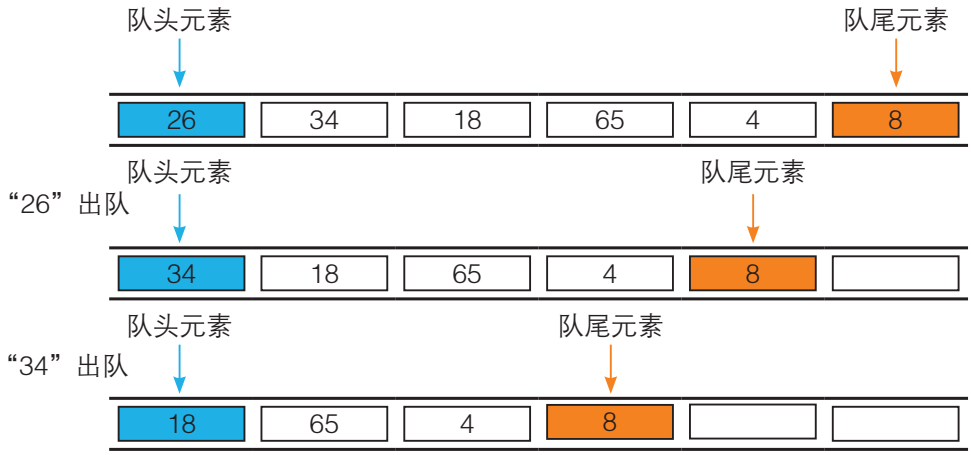


图3.3.7　队头位置不变时连续出队示意图

由此可见，在需要保持队头位置不变的情况下，进行出队操作将引发整个队列中的数据元素全部向队头方向移动，增加系统负担。此时，可以考虑第二种方法，即队头位置随操作而变化。

■ 队头位置变化的入队与出队

此时，入队操作跟队头不变的情况相同，如图3.3.6所示。

出队操作不需要移动数据元素，只需改变队头位置即可，如图3.3.8所示。

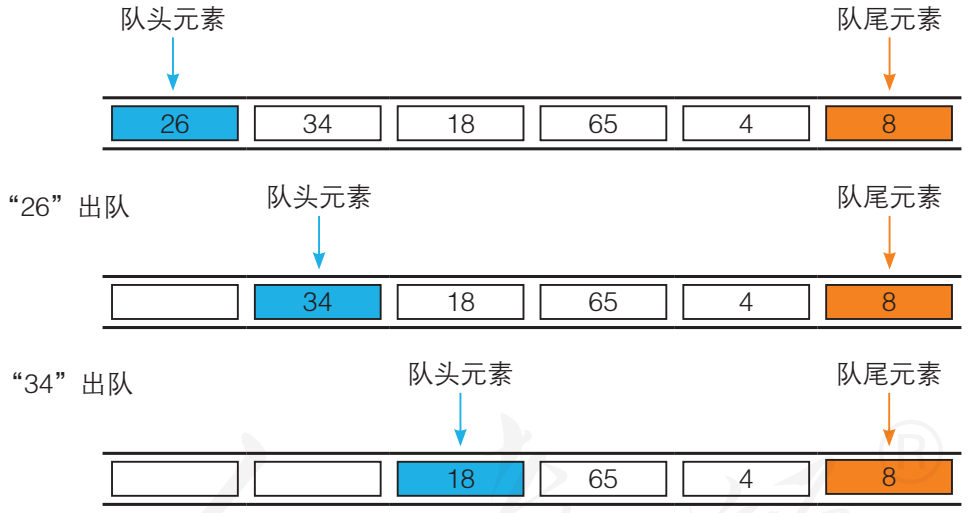


图3.3.8　队头位置变化时连续出队示意图

显然，队头位置变化有效减少了数据元素移动所带来的系统负担，弥补了队头位置不变的不足。但是，这种情况却隐含着另一个危机：当队满时，即使所有数据元素都出队，因队头位置移动而空出的所有存储空间也不能再进行入队操作了，造成了存储空间的极大浪费，这种现象称为“假溢出”。如图3.3.9所示就是一种“假溢出”现象。

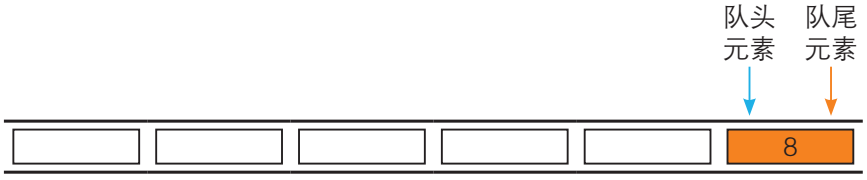


图3.3.9　队列“假溢出”示意图

2. 循环队列

顺序存储的队列由于频繁地进行入队、出队操作，队头位置和队尾位置只能后移而不能前移，导致出队元素留下的空间无法得到重新利用，出现“假溢出”的现象。为了解决这一问题，当队尾元素后面没有空闲存储空间时，可以利用前面的空闲空间继续存放新入队的数据元素，这样就形成首尾相接的顺序存储队列，称为循环队列。如图3.3.10所示，可以把新入队的数据元素“16”和“6”存储到前面的空闲空间。

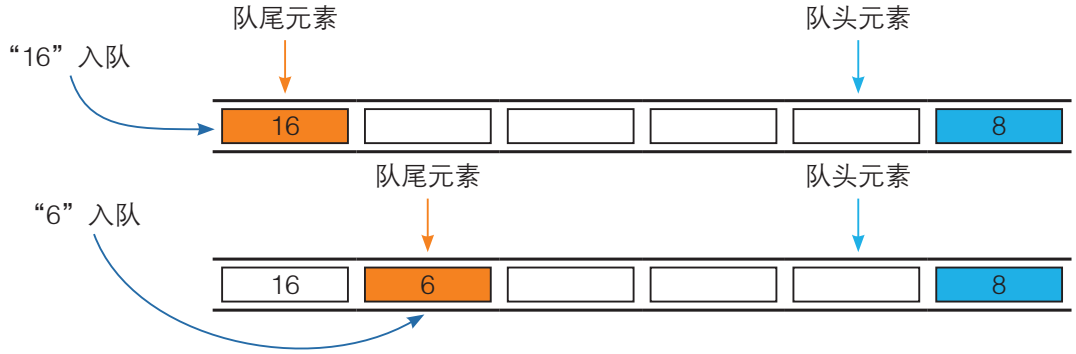


图3.3.10　循环队列连续入队示意图

循环队列将顺序存储空间看作首尾相接的环，充分利用了存储空间，有效解决了“假溢出”的问题。为了便于判断循环队列的队空、队满状态，可预留一个空闲存储单元，如图3.3.11所示。注意，在非空队列中，头指针始终指向队列头元素，而尾指针始终指向队列尾元素的下一个位置。

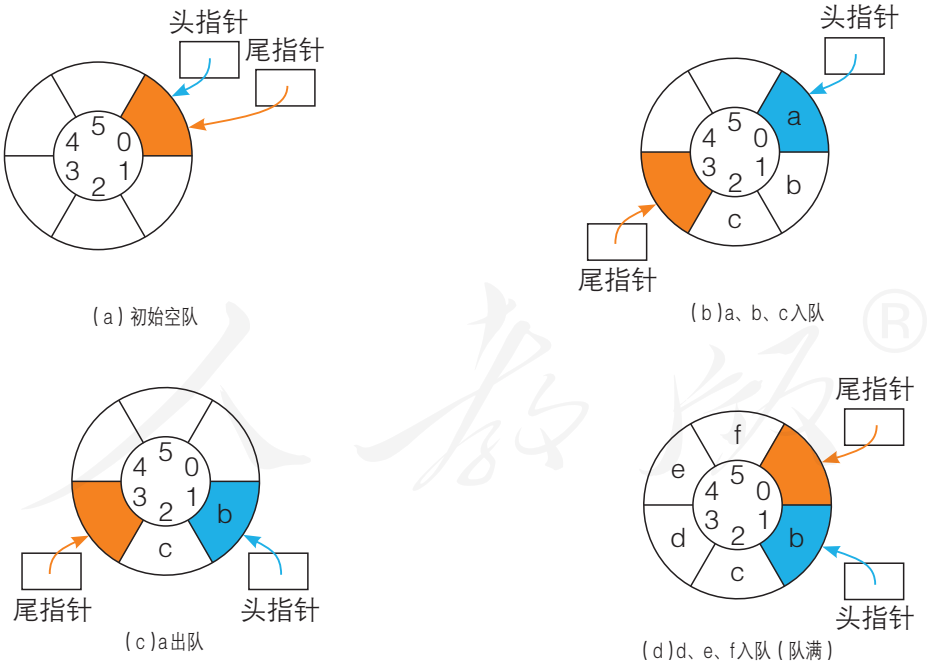


图3.3.11　循环队列示意图

队列的链式存储

队列的链式存储指用一组任意（不要求连续）的内存单元存储队列中的数据元素及

数据元素间的关系。链式存储的队列称为链队列。由于链式存储的队列的存储空间不需要连续，因此在链式存储方式下一般不考虑队满的情况。

由于队列本身的特点是在队头进行删除操作，在队尾进行插入操作，对于单链表而言，仅有一个表头指针不便于完成在表尾的插入操作，因此，需要增加一个队尾指针指向链表的最后一个元素。所以，一个链队列由一个头指针和一个尾指针共同确定，如图3.3.12所示。

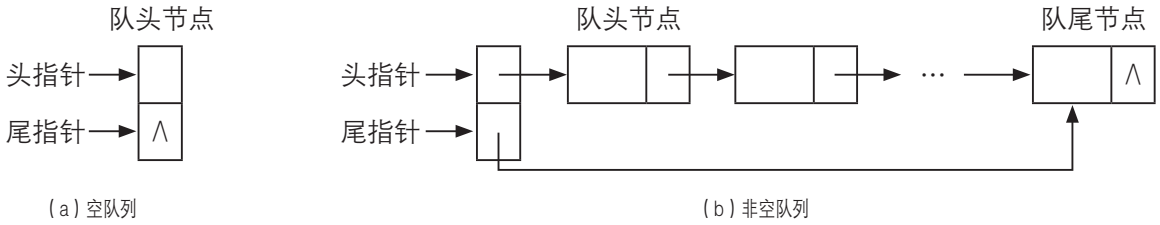


图3.3.12　链式队列示意图

定义链队列的节点的核心代码如下：

```
# 声明节点类
class Node(object):
    def __init__(self, data = None):
        self.data = data
        self.next = None
```

1. 初始化队列

设置一个空队，将头指针和尾指针都设置为空，队列长度记为0。

核心代码如下：

```
def init_queue(self):
    self.front = None
    self.rear = None
    self.length = 0
```

2. 判断队空

当队头（或队尾）指针为空时，返回队空。

核心代码如下：

```
def is_empty(self):
    if self.front == none:
        return true
    else:
        return false
```

3. 链队列的入队

若队列为空，将节点插入队列中，使头指针、尾指针均指向该节点；若队列非空，将尾指针指向该节点，队列长度加1。

核心代码如下：

```
def enqueue(self, element):
    tmp = Node(element)
```

```
        if self.is_empty():
            self.front = tmp
            self.rear = tmp
        else:
            self.rear.next = tmp
            self.rear = tmp
        self.length = self.length + 1
```

4. 链队列的出队

当队列为空时，输出“队列为空！”的提示信息；否则修改头指针，队列长度减1，并返回删除的队头元素。

核心代码如下：

```
    def dequeue(self):
        if self.is_empty():
            print("队列为空!")
        else:
            del_element = self.front.data
            self.front = self.front.next
            self.length = self.length - 1
            return del_element
```

实践活动

物流配送单与队列

物流公司通常按照货物到达的先后顺序生成配送单，快递员按照配送单的先后顺序依次完成送货任务。

利用队列的基本操作描述配送单的管理过程。

3.3.3 队列的应用

生活中有很多排队的场景，通过抽象可将这些场景转化成队列，如医院网上挂号、银行电子取号、网上叫车等。应用队列解决实际问题时，要从队列的“先进先出”特点入手，思考什么样的环境、条件、需求适合运用队列来解决实际问题。

思考活动

计算机中的打印队列

队列在学习和生活中的应用十分广泛。例如，使用打印机打印多份文档时，用户先后提交了多个打印任务，管理程序就将待打印的文档按照先来先服务的规则依次入队，形成打印队列。

思考：

在日常计算机操作中，还有哪些类似的应用是运用队列进行系统资源的分配与管理的？

银行通过自动取号机为客户提供“先来先服务”的体验。根据用户使用银行卡的种类（VIP卡或普通卡），按照取号的先后顺序，形成VIP队列和普通用户队列。普通用户根据取号的顺序自动排队办理相关业务。当银行VIP用户出现时，如果此时排队人数较少，可以采用VIP优先的原则，让VIP用户变为队头，优先办理业务；当排队人数较多时，则可以为VIP开辟独立窗口，VIP用户同样根据“先来先服务”的规则在VIP队列中排队、等待办理。银行等服务业利用队列原理解决现实问题，满足了不同用户的实际需求。

实践活动

编写银行取号程序

1. 将下列程序补充完整，并上机调试。

```
# 定义普通队列mylist，现在有三个人在排队
mylist = ["Tommy", "Alica", "Rock"]
# 定义VIP排队队列viplist，现在为空
viplist = []
# 定义队列VIP，VIP人员名册
VIP = ["WangGang", "LiBing", "Terry", "Tom"]
_____①_____                              # 重复以下操作
    # 新来人员输入姓名
    ss = input("请输入姓名：");
    # 查看新来人员是否属于VIP
    _____②_____
        # 属于VIP，将来人放在VIP队列尾
        viplist.append(ss)
    else:
        # 不属于VIP，将来人放在普通队列尾
        mylist.append(ss)
    # 输出VIP排队顺序
    print("VIP排队顺序：", viplist)
    # 输出普通排队顺序
    print("普通排队顺序：", mylist)
```

2. 上述程序中的循环结构没有终止条件，即没有出口，请进一步修改程序，在程序功能中增加“结束排队”的功能。

项目实施

巩固新知

一、项目活动

1. 根据图3.3.13中的提示，补充完善项目主题内容与基本数据结构的对应关系。

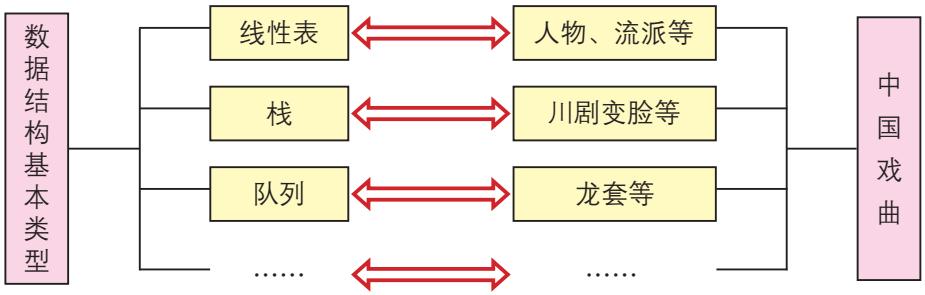


图3.3.13　中国戏曲内容与数据结构类型对应关系示意图

2. 比较线性表、栈和队列之间的区别和联系。

（1）用恰当工具制作线性表、栈和队列的知识结构图。

（2）填写表格3.3.2，比较线性表、栈和队列之间的区别与联系。

表3.3.2　线性表、栈与队列比较

比较项目	线性表	栈	队列
插入位置			
删除位置			
数据结构特点			
适用情况			
相同点			
不同点			

二、项目检查

进一步梳理知识结构图，明确栈和队列都是操作受限制的线性表，进行阶段性知识小结。

练习提升

1. 假设用长度为11的数组作为顺序队列，初始为空。分别绘出做完下列操作后的队列示意图。若有元素不能入队，试说明理由。操作序列如下：

（1）A，B，C，D，E依次入队；　　（2）A，B依次出队；

（3）F，G，H，I，J依次入队；　　（4）O，P，Q，R依次入队。

2. 利用两个顺序栈S_1和S_2模拟一个队列。利用栈的基本操作，实现队列的入队和出队操作，并说出基本思路。

3.4
元素受限的线性表——字符串

学习目标 ▶▶▶

- 理解字符串的概念，感受其应用价值。
- 理解字符串的基本操作，并用于解决实际问题。

体验探索

字雕句镂，成就经典

唱 脸 谱

外国人把那京戏叫作 Beijing Okra，
没见过那五色的油彩楞往脸上画，
“四击头”一亮相，（哇……）
美极了，妙极了，简直 OK 顶呱呱！

蓝脸的窦尔敦盗御马，
红脸的关公战长沙，
黄脸的典韦，白脸的曹操，
黑脸的张飞叫喳喳！
……

图 3.4.1 《唱脸谱》部分歌词

图 3.4.1 中的文字部分是《唱脸谱》的部分歌词，从中可看出中国戏曲也深深地吸引着外国友人。实际上，无论是脍炙人口的戏文，还是传唱不衰的歌词，大都经历过反复修改才得以最终呈现。在计算机中，对文字修改可以看作是对字符串进行的操作。

思考：

若要在计算机中将歌词里的“Beijing Okra”修改为“Beijing Opera”，在修改之前需要进行哪些基本操作？

3.4.1 字符串的概念

《唱脸谱》的歌词是由多个汉字和英文字母组成的，在数据结构中将其称为字符串。例如，"abcde"和"美丽的中国梦"及"+-*/=?"均为字符串。字符串是一种特殊的线性表，其特殊性在于数据元素均为字符。在程序设计中，通常将字符串作为一个整体来处理。

思考活动

单词拼拼乐

现有一个字母数组[a, b, c, d, e, r, s, t, u, v, w]。

思考：

利用数组中的字母可以组合成哪些单词？这些单词可以组合成哪些词组或语句？

字符串是由n（$n \geqslant 0$且为整数）个字符组成的有限序列，可简称为串。n为字符串中字符的个数，称为串的长度。n=0时，称字符串为空串，可表示为""。$n>0$时，字符串可表示为"$a_1a_2\cdots a_n$"，其中，1，2，…，n这些数字为字符在串中的位置；a_i（$1 \leqslant i \leqslant n$）可以是字母、数字或其他可显示字符。例如，字符串"data_structure@126.com"中既包含数字、字母，又包括“@”和“.”字符。注意，串值必须用一对半角双引号括起来，但双引号本身不属于串。

字符串的抽象数据类型定义如图3.4.2所示。

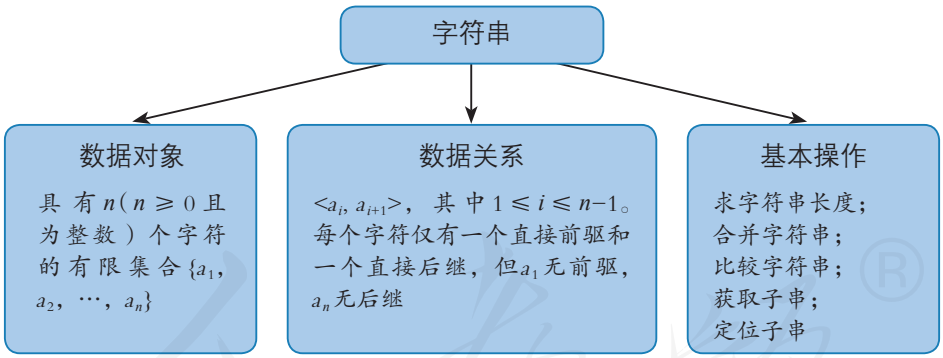


图3.4.2 字符串的抽象数据类型定义示意图

实践活动

戏文中的字符串

“桂英我十年未曾离鞍马，咱杨家四代俱是军中人。”“老太君为国把忠尽，她命我挂帅平藩臣。一不为官二不为宦，为的大宋江山和黎民。”这两句话是豫剧《穆桂英挂帅》中的戏文。

1. 请以字符串的形式写出上述两句戏文。
2. 分别统计这两个字符串的长度。

3.4.2 字符串的基本操作

在电子文档中，可能存在拼写错误的文字，如将“读诗”误写为“读史”。如果文章的篇幅不长，可从文档首行首字开始，将找到的每一处“读史”手动改为“读诗”。但若文章篇幅过长，这种方式就太费时费力。此时，可以利用文档编辑软件中的“查找”和“替换”功能快速完成修改，这些功能的实现，本质上都是对字符串的基本操作。

思考活动

作文修改与字符串操作

唐代诗人卢延让在《苦吟》中写道“吟安一个字，捻断数茎须”。同理，一篇好的作文大多会经过反复修改才能定稿。

思考：

在作文的初稿、修改和定稿的三个阶段中，我们一般要对文章中的字、词进行哪些操作？

求字符串长度

求字符串的长度即统计字符串中字符的个数。例如，S="Opera"，对其中的字符进行计数，得出S的长度为5。

合并字符串

对两个字符串进行合并指将第二个字符串加入到第一个字符串的尾部，得到一个新的字符串。

例如，有两个字符串S_1="$a_1a_2\cdots a_n$"和S_2="$b_1b_2b_3\cdots b_m$"（n、m均为正整数），S_1与S_2合并后得到的S_3="$a_1a_2\cdots a_nb_1b_2b_3\cdots b_m$"，如图3.4.3所示。

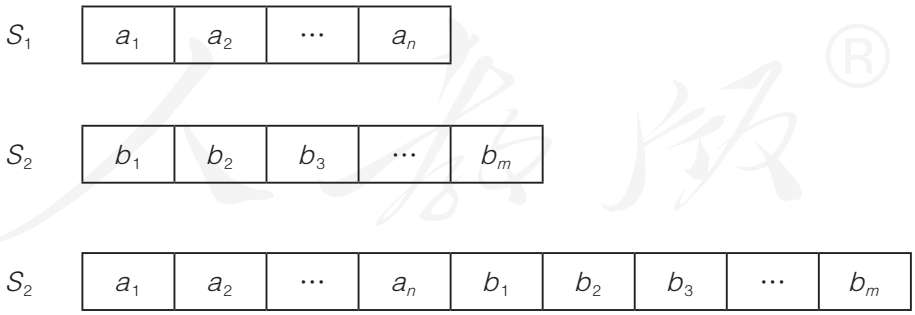


图3.4.3 字符串合并示意图

比较字符串

字符串的比较实际上就是对组成串的字符编码的比较，字符的编码是字符在相应字符集中的序号。unicode编码以16位二进制数表示一个字符，该二进制数即为字符在字符集中的序号。

设存在两个字符串S="$a_1a_2\cdots a_n$"和T="$b_1b_2\cdots b_m$"（n、m均为正整数）。

当S与T同时满足下面两个条件时，S=T。

① 两个字符串的长度相等。

② 两个字符串对应位置的字符都相同。

也就是说，当n=m，且a_1=b_1，a_2=b_2，…，a_n=b_m时，S=T。

例如，若S="Opera"，T="Opera"，字符串的比较过程如图3.4.4所示。

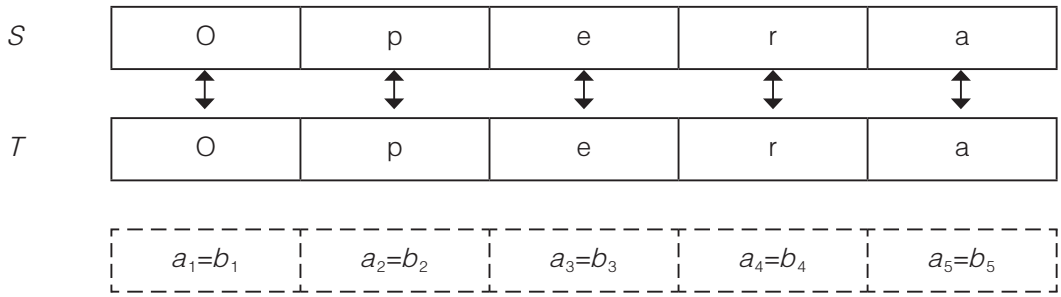


图3.4.4　字符串比较（1）

因此，S=T。

当n＜m且a_i=b_i（i=1，2，…，n）时，S＜T。

例如，若S="Oper"，T="Opera"，字符串比较过程如图3.4.5所示。

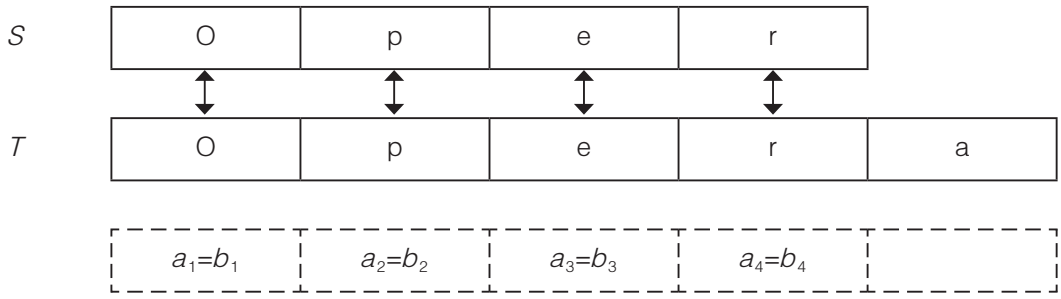


图3.4.5　字符串比较（2）

因此，S＜T。

若存在k，k≤min(n，m)，使得a_i=b_i（i=1，2，…，k–1），a_k＜b_k，则S＜T。

例如，若S="OpEra"，T="Opera"，则S＜T。字符串比较过程如图3.4.6所示。

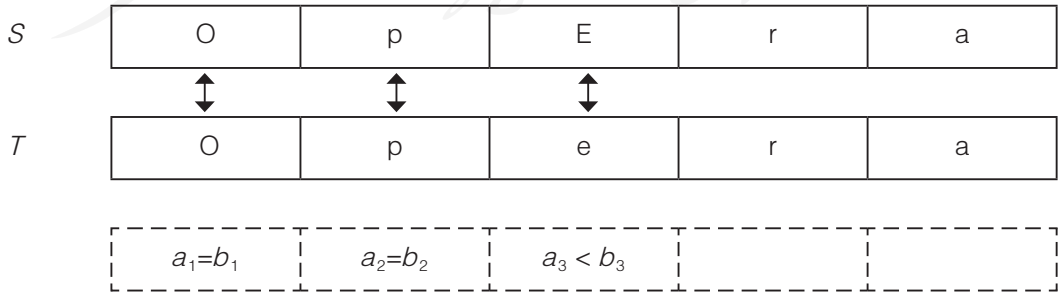


图3.4.6　字符串比较（3）

因此，S＜T。

实践活动

字符串大小的判定

"OperA"与"Opera"狭路相逢，彼此互不相让，双方约定谁的值大谁先走。在图3.4.7中写出两个字符串比较的过程，判定谁先走。

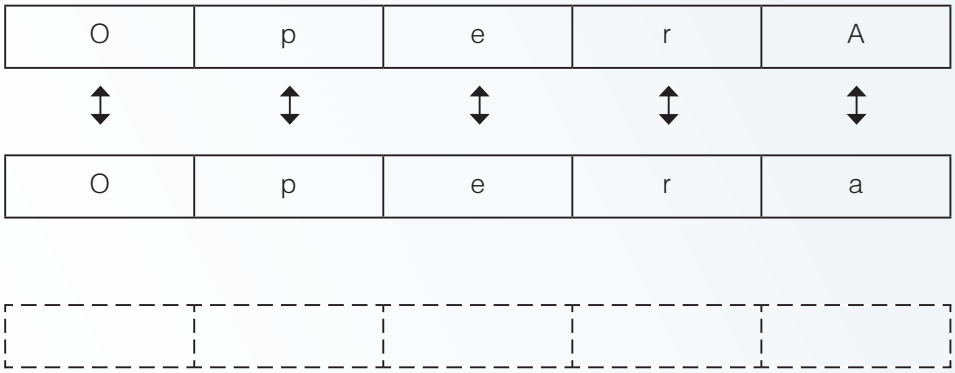


图3.4.7　字符串比较（4）

获取子串

字符串中任意个数的连续字符组成的子序列称为该串的子串，包含子串的串则称为主串。子串在主串中的位置以子串的第一个字符在主串中的序号来表示。获取子串即复制字符串的开始位置至结束位置的字符。注意，结束位置是子串最后一个字符的后面一个位置。

例如，现有字符串"structure"，取开始位置为2、结束位置为7的子串"truct"，如图3.4.8所示。

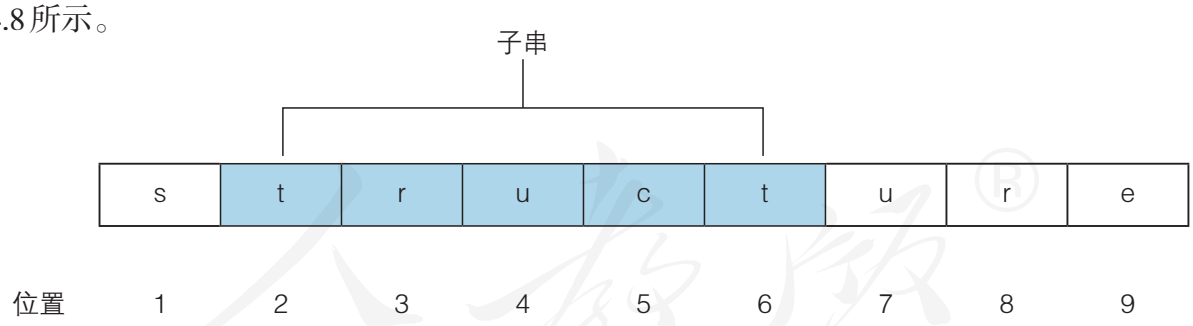


图3.4.8　获取子串

定位子串

在文档编辑软件中，利用查找功能搜索某个词语或句子；垃圾邮件过滤器根据邮件的主题、内容是否含有某些特定的字符序列而决定是否过滤并标记为垃圾邮件；利用网络搜索引擎，查找包含检索关键字的内容……以上场景均应用了字符串的定位操作。

定位子串过程为：首先将主串和子串分上下两行左对齐，然后从左向右逐个字符进行比较，如果不成功，则子串向右移动一个字符，继续进行比较。

假设主串S="Peking Opera"，要定位的子串S_1="ing"，则其定位过程如下：

（1）主串S从第一位开始，其首字母为“P”，要匹配的S_1字母是“i”，匹配失败；

（2）主串S从第二位开始，其首字母为“e”，要匹配的S_1字母是“i”，匹配失败；

（3）主串S从第三位开始，其首字母为“k”，要匹配的S_1字母是“i”，匹配失败；

（4）主串S从第四位开始，3个字母全部匹配成功。

上述定位操作的全过程如图3.4.9所示。

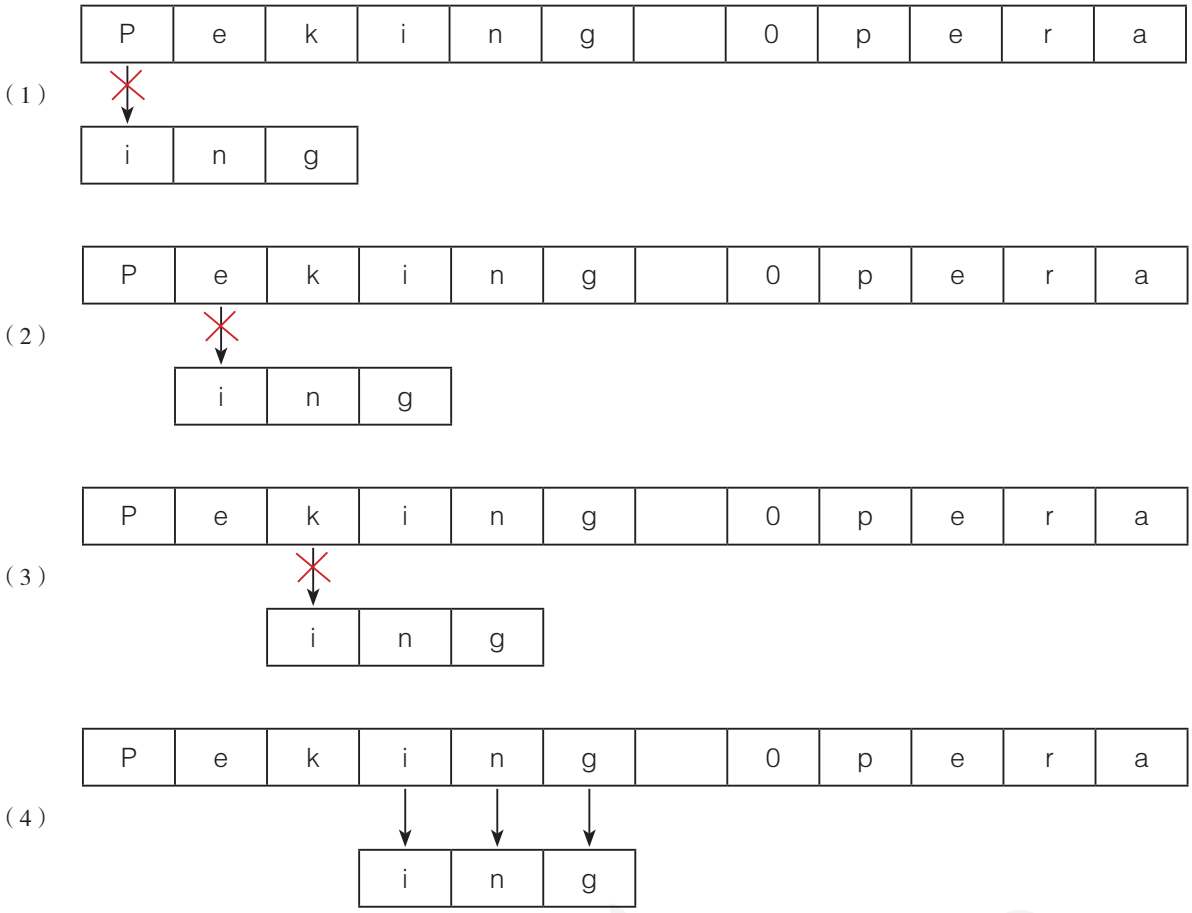


图3.4.9　定位子串

3.4.3　字符串的常用函数

在Python中有很多内建函数，利用内建函数可以方便快捷地实现相应的功能，现对字符串常用的内建函数与基本操作做简要介绍。需要注意，在Python中，字符串的位置从0开始。

求字符串长度

可利用函数len(string)计算字符串、列表、字典等的长度。例如：

```
s = "Information technology"
print(len(s))
```

运行结果为：

```
22
```

合并字符串

合并字符串有如下的两种操作方式。

1. 利用“+”操作符。例如：

```
a = "Information"
b = " "
c = "technology"
d = a + b + c
print(d)
```

运行结果为：

```
Information technology
```

2. 利用函数join()将数组中的字符串以指定的字符（分隔符）连接生成一个新的字符串。语法格式为symbol.join(seq)，其中，symbol为字符串的分隔符，可以为空；seq为要合并的各个字符串。例如：

```
s = ["Information", " ", "technology"]
print("".join(s))
```

运行结果为：

```
Information technology
```

比较字符串

若想比较两个字符串的大小，需导入operator模块，使用其中的operator.eq(str1, str2)、operator.lt(str1, str2)和operator.gt(str1, str2)等函数进行两个字符串的比较。例如：

```
import operator
str1 = "Information"
str2 = "information"
print(operator.eq(str1, str2))
print(operator.lt(str1, str2))
print(operator.gt(str1, str2))
```

运行结果为：

```
False
True
False
```

技术支持

operator模块

operator模块是Python中内置的操作符函数接口，它定义了一些算术和比较操作的函数，如表3.4.1所示。

表3.4.1　operator模块常用函数

函数/方法	功能说明
operator.lt(a, b)	函数返回a是否小于b
operator.le(a, b)	函数返回a是否小于或等于b
operator.eq(a, b)	函数返回a是否等于b
operator.ne(a, b)	函数返回a是否不等于b
operator.ge(a, b)	函数返回a是否大于或等于b
operator.gt(a, b)	函数返回a是否大于b

获取子串

获取子串需先确定该子串在字符串中的开始和结束位置，再利用截取字符串的方式（str[start:end]）获取子串。str[start:end]表示取出字符串中下标start到下标end之间的字符（顾头不顾尾，不包括下标end的字符）。例如：

```
str = "Information technology"
s = str[2:11]
print(s)
```

运行结果为：

```
formation
```

定位子串

利用函数find()确定字符串中是否包含子字符串，如果指定了查找范围，则检查是否包含在指定范围内，如果包含子串，则返回开始的位置；否则，返回“-1”。例如：

```
str = "Information technology"
print(str.find("o"))
print(str.find("o", 5, 10))
```

运行结果：

```
3
9
```

3.4.4　字符串的应用

文档编辑软件可用于文字的录入、修改和排版，实现导入、合并文档等功能。各类不同的文档编辑软件虽有差异，但其基本的操作原理相同，均为对字符串进行相关操作。

思考活动

文档编辑软件中的字符串操作

通过前面的学习，我们可以了解到文档编辑软件中的查找功能应用了子串定位的基本操作。

思考：

1. 替换功能的原理是什么？其过程是怎样的？
2. 文档编辑软件中还应用了字符串的哪些操作？

日常学习和办公过程中，文档编辑软件的查找、替换等是较为常用的功能。此外，还有统计字数、清除空格和拼写检查等，这些功能都涉及对字符串的操作。下面以文档编辑小程序为例，了解这些功能是如何实现的。

```
if __name__ == "__main__":
    a = input("请输入原始字符串：")
    print("原始字符串为：", a)
    print('''对该字符串进行哪些操作？
        1：计算长度
        2：定位子串
        3：替换子串
        4：统计某字符串出现的次数''')
    try:
        choice = int(input("请输入相应数字进行操作："))
        if choice == 1:
            print("该字符串的长度为：", len(a))
        elif choice == 2:
            b = input("请输入要定位的子串：")
            print(b, "在", a, "中的位置为：", a.find(b))
        elif choice == 3:
            c = input("请输入要更改的子串：")
            d = input("请输入要替换的子串：")
            print("替换后的字符串为：", a.replace(c,d))
        elif choice == 4:
            e = input("请输入要统计的字符串：")
            print(e, "在", a, "中出现的次数为：", a.count(e))
        else:
            print("输入不合法，请输入合法数字")
    except ValueError:
        print("请输入数字选项")
```

实践活动

程序优化设计

1. 思考上述文档编辑小程序能否进一步优化，若可以，请优化后上机调试。

2. 上述程序初步实现了字符串的查找、替换、定位和统计功能，请根据需要选用表3.4.2中的函数，为该程序增加新的功能。

表3.4.2　字符串函数

功能分类	函数	功能说明
字母处理	string.upper()	全部大写
	string.lower()	全部小写
	string.swapcase()	大小写互换
	string.capitalize()	首字母大写，其余小写
	string.title()	首字母大写
格式化相关	string.ljust(width)	获取固定长度，左对齐，右边不够用空格补齐
	string.rjust(width)	获取固定长度，右对齐，左边不够用空格补齐
	string.center(width)	获取固定长度，中间对齐，两边不够用空格补齐
	string.zfill(width)	获取固定长度，右对齐，左边不足用“0”补齐
字符串去空格	string.strip()	去两边空格
	string.lstrip()	去左边空格
	string.rstrip()	去右边空格

项目实施

阶段总结

一、项目活动

根据图3.4.10中的提示，补充完善项目主题内容与基本数据结构的对应关系。

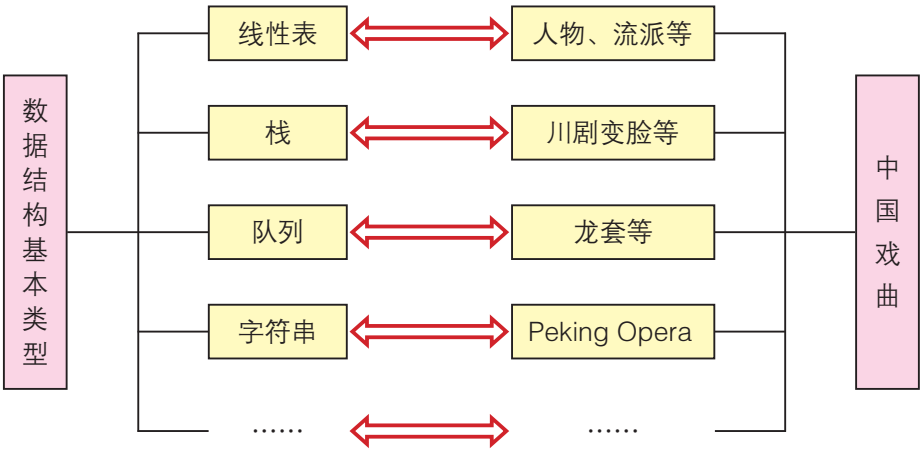


图3.4.10　中国戏曲内容与数据结构类型对应关系示意图

二、项目检查

1. 构建线性数据结构的知识体系，总结其特点及基本操作。

2. 完成项目学习报告的初步构架，提炼、整理报告素材，选定制作工具。

练习提升

1. 编写程序从键盘输入密码，判断是否正确。

2. 有一种可以倒读或反复回旋阅读的诗体，称为回文诗。唐寅的《咏冬》正是回文诗的代表之一。

咏冬

梅枝几点雪花开，点雪花开春信来。

来信春开花雪点，开花雪点几枝梅。

在这首诗的阅读过程中用到了字符串的哪些操作？

3.5
二叉树

学习目标 ▶▶▶

- 了解二叉树的概念和特点，并能结合实例进行分析。
- 了解二叉树的基本操作方法，感受二叉树组织数据的高效性。

体验探索

行当分支，支支精彩

“生、旦、净、丑”是京剧的四个角色行当（图3.5.1），每个行当又有若干分支，各有固定的扮演人物和表演特色。例如，“生”可分为老生、小生、武生等；“旦”可分为青衣、花旦、老旦、武旦等；“净”可分为正净、副净、武净等；“丑”可分为文丑、武丑等。

图3.5.1　生、旦、净、丑漫画图

思考：

能否利用线性结构表示行当间的关系？说明原因，并用分支图表示行当间的关系。

3.5.1　二叉树的概念

线性表、栈、队列和字符串均属于线性结构，用来表达数据元素之间一对一的关系。然而，日常生活中很多数据元素之间的关系往往不是简单的线性关系，而是更为复杂的结构，如人类社会中的族谱、各种社会组织机构、交通道路和通信网络等。

思考活动

数的分类

在数学中，数可以分为实数和虚数，实数可分为有理数和无理数，有理数又可分为整数和分数……

思考：

1. 如图3.5.2所示的数的分类的分支图有什么特点？
2. 该分支图与本节体验探索中的行当分支图有何异同？

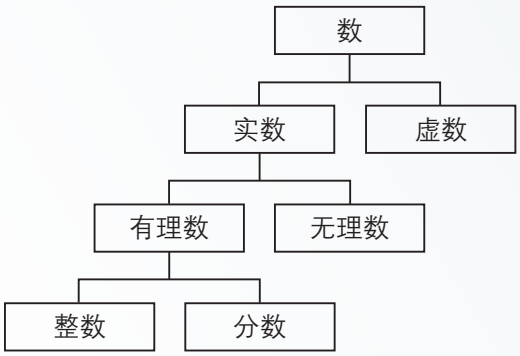


图3.5.2　数的分类示意图

如图3.5.2所示的结构称为树形结构。

树形结构的基本术语

树形结构的节点包含数据元素的值及指向其子树的分支，节点拥有的分支数称为节点的度。在图3.5.3中共有6个节点，它们是A、B、C、D、E、F，其中，A节点共有2个分支，故A节点的度为2；B节点共有2个分支，故B节点的度为2；C节点只有1个分支，故C节点的度为1。

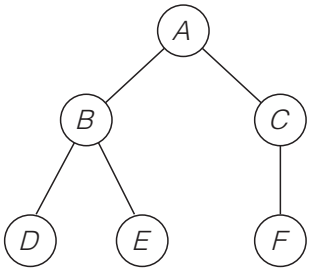


图3.5.3　树形结构示意图

树形结构中，节点的度的最大值称为树的度。图3.5.3所示的树的度为2。

度为0的节点称为终端节点或叶节点。例如，在图3.5.4中，节点H、I、E、J、G均为终端节点。

度不为0的节点称为非终端节点或分支节点，除根节点以外的分支节点也称为内部节点。例如，在图3.5.4中，节点A、B、C、D、F均为非终端节点，其中B、C、D、F均为内部节点。

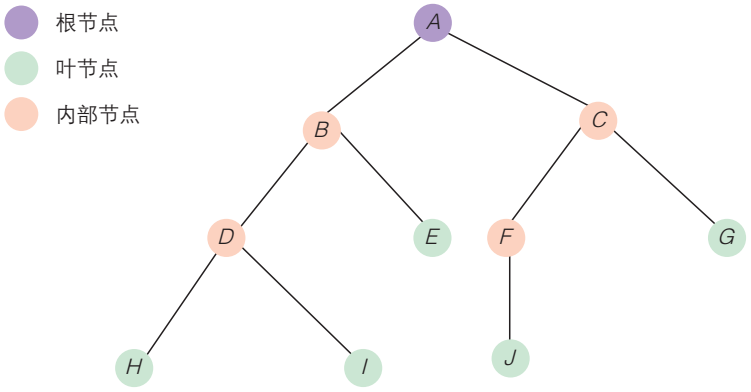


图3.5.4　节点分类示意图

二叉树的定义

二叉树是n（$n \geqslant 0$且为整数）个节点的有限集，n=0时为空树。在任意一棵非空二叉树中有且仅有一个特定的称为根的节点。当$n > 1$时，除根节点外的所有节点可分为两个互不相交的有限集，分别称为根的左子树和右子树，左子树和右子树也是二叉树，如图3.5.5所示。

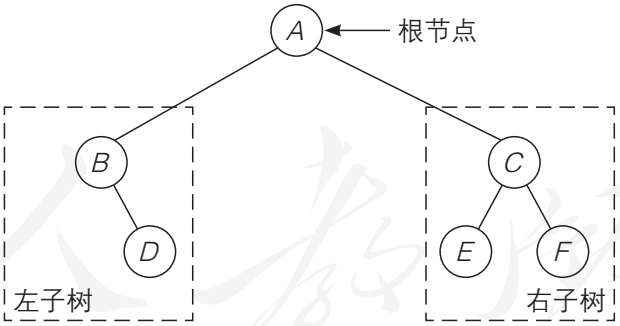


图3.5.5　二叉树结构示意图

某节点的子树的根称为该节点的孩子，该节点称为孩子的双亲，同一双亲的孩子之间互称兄弟。例如，在图3.5.5中，A是B、C的双亲，B、C是A的孩子，B、C互为兄弟。

节点的层次从根开始定义，根为第一层，根的孩子为第二层，以此类推。若某节点在第L层，则其孩子就在第L+1层。

二叉树中节点的最大层数称为二叉树的深度或高度，图3.5.6中树的深度为3。

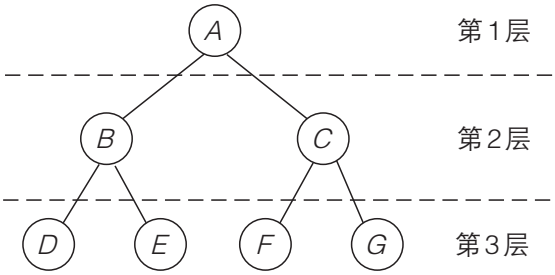


图3.5.6　二叉树的层次示意图

二叉树的特点

二叉树具有如下特点：

■ 二叉树中不存在度大于2的节点。在二叉树中，每个节点最多有两棵子树，没有子树或只有一棵子树都是允许的。

■ 二叉树的子树有左右之分，且次序不能颠倒。若二叉树只有一棵子树，也需要明确它是左子树还是右子树。

根据二叉树的定义可知，二叉树的子树也是二叉树。由此可以推出，非空二叉树存在4种基本形态，如图3.5.7所示，只有一个根节点（a）、根节点只有左子树（b）、根节点只有右子树（c）、根节点既有左子树又有右子树（d）。

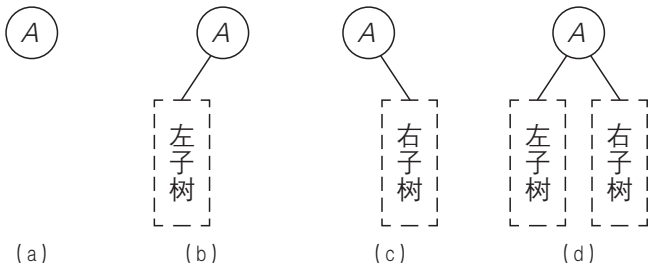


图3.5.7　二叉树的基本形态

在一棵二叉树中，若每一个分支节点都存在左、右子树，且所有的叶节点均在同一层上，这棵二叉树称为满二叉树，如图3.5.8所示。

对于如图3.5.8所示的满二叉树，可将所有节点按照“从上到下、同层从左到右”的次序从1开始连续编号，如图3.5.9所示。按从大到小的顺序连续删除该满二叉树的若干节点后剩下的二叉树称为完全二叉树，如图3.5.10所示。

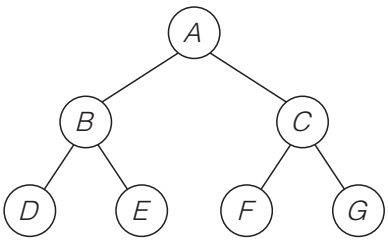


图3.5.8　满二叉树

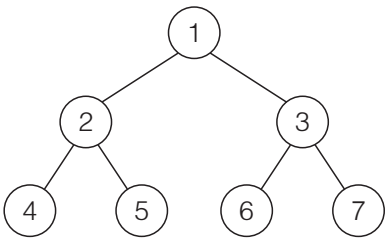


图3.5.9　满二叉树编号

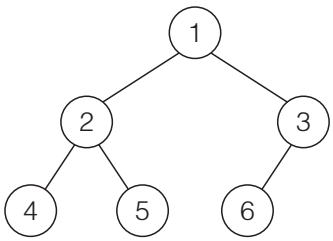


图3.5.10　完全二叉树

实践活动

了解二叉树的结构

图3.5.11中，二叉树的根是哪个节点？哪些节点是叶节点？二叉树的深度是多少？哪些节点有子树？节点B的子树有几棵？节点E的双亲节点、孩子节点、兄弟节点分别是哪些节点？

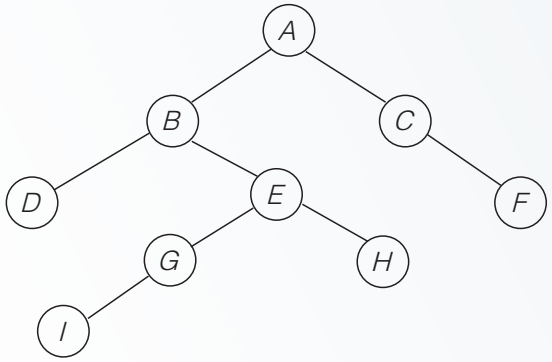


图3.5.11　二叉树结构

3.5.2　二叉树的基本操作

二叉树的基本操作包括构造空二叉树、查找二叉树中的某个节点、删除二叉树中的某个节点和遍历二叉树等，而二叉树的很多操作都以遍历二叉树为前提和基础。

思考活动

游览动物园

某动物园有9个馆舍，馆舍之间的道路有8条，动物园的出入口在长颈鹿馆处，如图3.5.12所示。

图3.5.12　动物园馆舍示意图

思考：

如何规划一条能将每个馆舍都游遍且每个馆舍仅游览一次的路线？

二叉树的遍历指从根节点出发，按照某种次序依次访问二叉树中的所有节点，使得每个节点仅被访问一次。

根据前面的学习可知，一棵非空二叉树由根节点、左子树和右子树构成，因此，遍历二叉树需要完成三项工作：访问根节点、遍历左子树和遍历右子树。如果规定先遍历左子树，再遍历右子树，则二叉树的遍历方法有三种，分别称为先根序遍历、中根序遍历和后根序遍历。下面，以图3.5.13中的表达式二叉树为例进行说明。

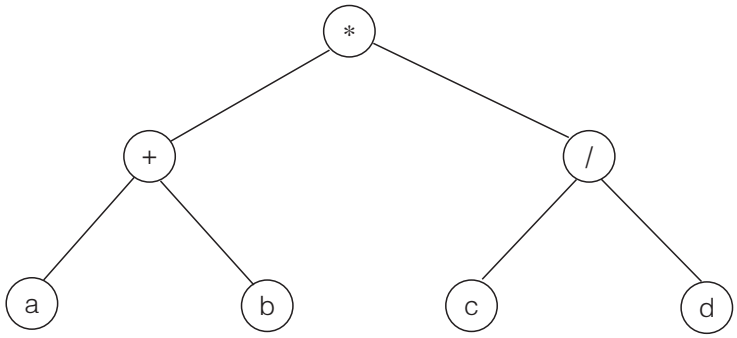


图3.5.13　表达式二叉树

1. 先根序遍历

若二叉树为空，则空操作返回；否则，①访问根节点，②先根序遍历左子树，③先根序遍历右子树。

首先，访问根节点*，再先根序遍历*的左子树，之后再先根序遍历*的右子树。图3.5.14中的编号表示节点的访问顺序。

先根序遍历序列为：*+ab/cd。

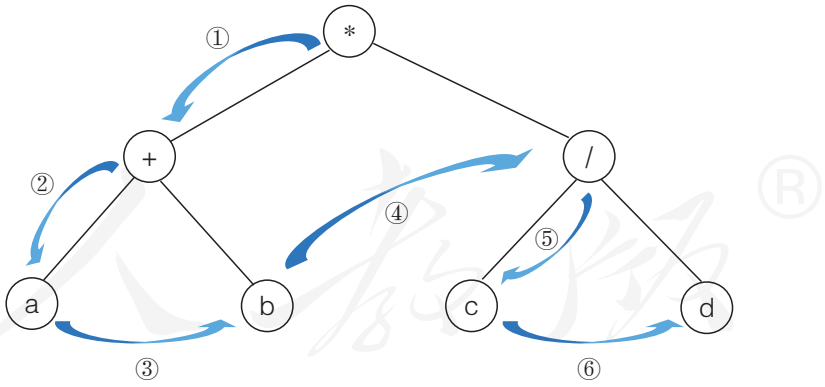


图3.5.14　先根序遍历示意图

2. 中根序遍历

若二叉树为空，则空操作返回；否则，①中根序遍历左子树，②访问根节点，③中根序遍历右子树。

首先，中根序遍历*的左子树，再访问根节点*，之后再中根序遍历*的右子树。图3.5.15中的编号表示节点的访问顺序。

中根序遍历序列为：a+b*c/d。

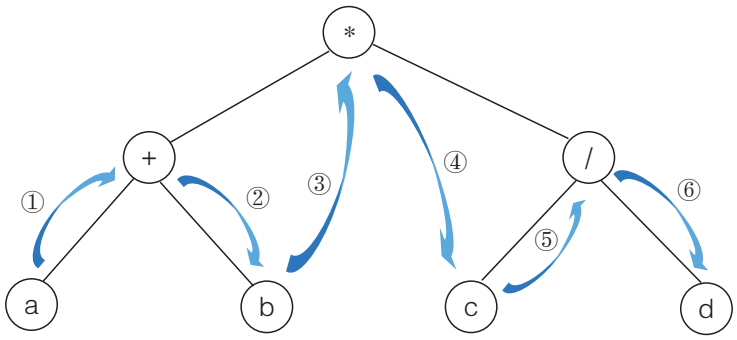


图3.5.15　中根序遍历示意图

3. 后根序遍历

若二叉树为空，则空操作返回；否则，①后根序遍历左子树，②后根序遍历右子树，③访问根节点。

首先，后根序遍历*的左子树，再后根序遍历*的右子树，最后再访问根节点*。图3.5.16中的编号表示节点的访问顺序。

后根序遍历序列为：ab+cd/*。

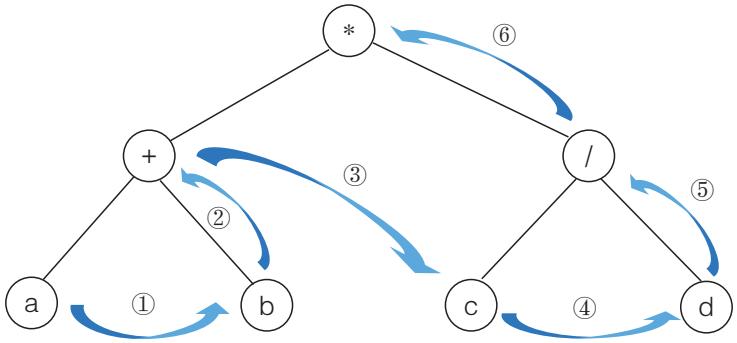


图3.5.16　后根序遍历示意图

实践活动

解决游览动物园馆舍问题

学习二叉树的遍历后，我们就可解决上文中提出的“如何规划一条能将每个馆舍都游遍且每个馆舍仅游览一次的路线”的问题了。

将动物园的馆舍抽象为节点，节点之间的路径抽象为边，请用二叉树表示动物园的馆舍分布图。

用 ⊗、☆和△三种符号分别标记出先根序、中根序和后根序遍历时各节点的访问顺序，如图3.5.17所示。请按照符号提示，写出遍历动物园馆舍的三种访问序列。

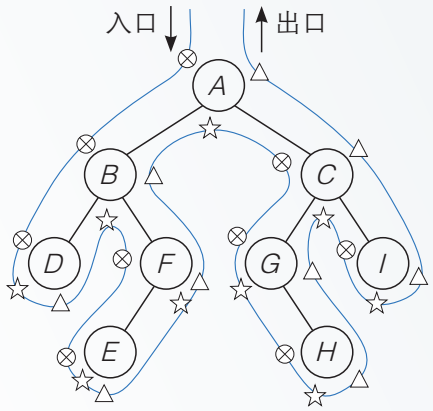


图3.5.17　动物园馆舍遍历示意图

■ 先根序遍历：_____。

■ 中根序遍历：_____。

■ 后根序遍历：_____。

项目实施

总结表达

一、项目活动

1. 根据图3.5.18中的提示，补充完善项目主题内容与基本数据结构的对应关系。

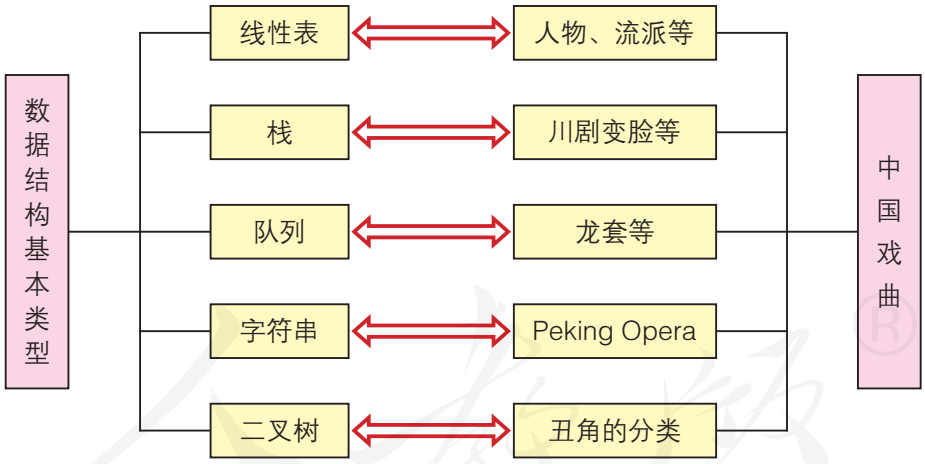


图3.5.18　中国戏曲内容与数据结构类型对应关系示意图

2. 总结二叉树的定义、基本术语、遍历方式及生活中的应用实例，并绘制思维导图。

二、项目检查

1. 分析二叉树的特点、应用范围，总结线性结构与非线性结构的区别，并绘制数据结构思维导图。

2. 完成阶段性项目学习报告并上传至班级共享空间。

练习提升

1. 根据定义，找出图3.5.19中的二叉树。

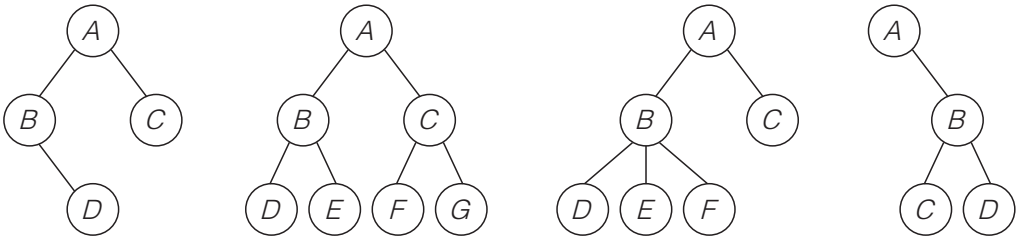


图3.5.19　找出符合定义的二叉树

2. 某二叉树有3个节点，两位同学就这棵二叉树的形态产生了争执，一位同学认为该二叉树的形态如图3.5.20(a)，另一位同学则认为该二叉树的形态如图3.5.20(b)。你赞同哪位同学的看法？还有其他的形态吗？

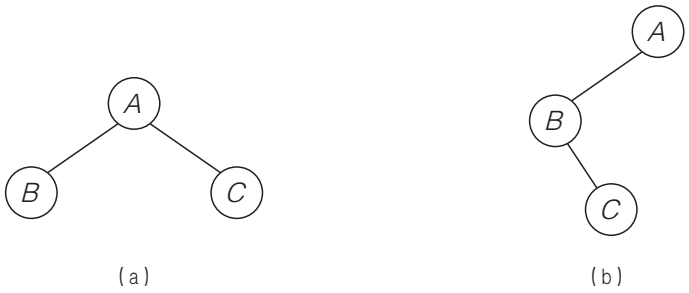
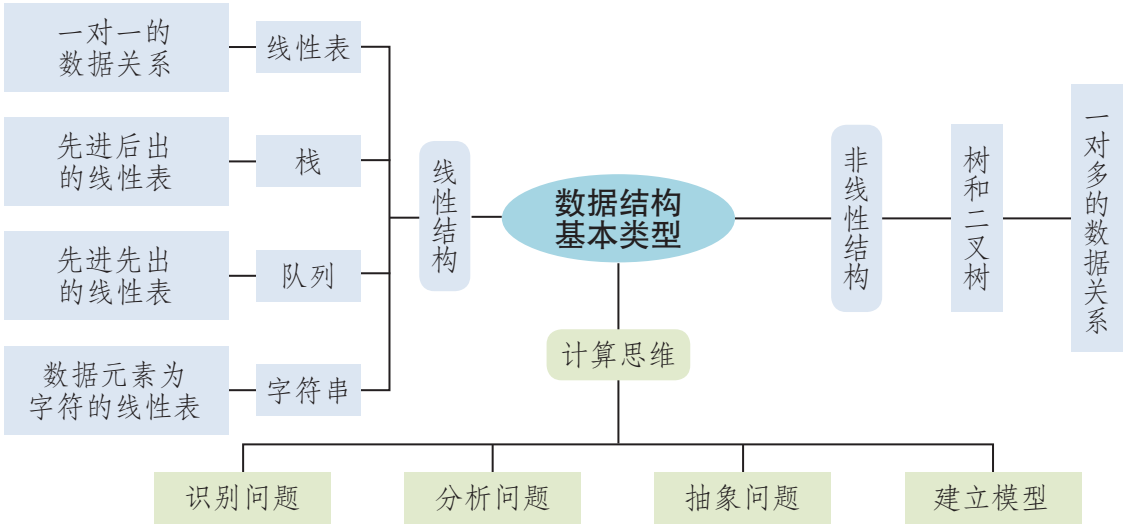


图3.5.20　3个节点的二叉树形态

总结 评价

1. 下图展示了本章的核心概念与关键能力，请同学们对照图中的内容进行总结。



2. 根据自己的掌握情况填写下表。

学习内容	掌握程度
线性表的基本概念	□不了解 □了解 □理解
栈的基本概念	□不了解 □了解 □理解
队列的基本概念	□不了解 □了解 □理解
字符串的基本概念	□不了解 □了解 □理解
二叉树的基本概念	□不了解 □了解 □理解
线性表、栈、队列、字符串和二叉树的基本操作	□不了解 □了解 □理解
编程实现线性表、栈、队列和字符串的基本操作	□不了解 □了解 □理解
结合实际发现现实生活中的数据结构	□不了解 □了解 □理解
在生活中运用数据结构	□不了解 □了解 □理解

第 4 章
算法与数据结构

随着互联网的发展，搜索引擎改变了我们的生活，我们获取信息的速度得到极大提升。你知道搜索引擎的工作原理吗？当你在互联网上发布文章、上传照片或视频时，来自互联网世界的无数程序便会像“蜘蛛”一样蜂拥而至，抓取并复制你的网页，并通过网页上的链接顺藤摸瓜抓取更多的页面，将所有信息纳入到搜索引擎网站的索引数据库，并将拆解后的网页内容、关键词的位置、字体和颜色等信息，生成庞大的索引记录。当有人在搜索引擎上输入一个关键词进行搜索时，它会在很短时间内，在索引数据库中检索到所有包含关键词的网页，并依据人们的浏览次数与关联性等，通过一系列算法确定网页级别，排列出顺序，最终呈现在网页上。

算法是描述解决问题过程的方法。现实世界中的问题千差万别，算法自然也是千变万化的。由于不同搜索引擎使用的算法不同，呈现的搜索结果页面也不相同。此外，购物推荐、新闻推送、旅游出行和社交通信等各种信息服务在生活中随处可见，背后起着重要作用的就是算法。算法是计算机科学领域最重要的基石之一，是数字化生活的基本保障，而数据结构又是算法研究的基础。

本章将通过项目“编写对弈程序”开展学习，理解算法与数据结构的关系，体验问题解决的一般过程。通过实现数据的排序和查找，体验迭代和递归的方法，解决生活中的实际问题。

4 主题学习项目：编写对弈程序

项目目标

本章通过“编写对弈程序”项目，引领同学们开展自主选题，建议从确定主题、选择数据结构、设计算法和编写程序等几个方面入手，完成小组项目，发布应用程序，详细记录研究过程并形成研究报告。

1. 通过项目学习，体验算法与数据结构的重要关系，深化对数据结构与算法的理解。

2. 在研究问题的过程中，能借助数字化工具协同学习，利用数据结构与算法解决实际问题。

项目准备

为完成项目，需做如下准备。

- 以小组为单位，确定积极有意义的项目研究主题，对可行性和前瞻性做简要分析。
- 收集与项目相关的资料，进行适当的筛选与加工，确保资料真实有效。
- 搭建Python语言基本运行环境，并安装项目需要的模块。

项目过程

在学习本章内容的同时开展项目活动。为了保证本项目的顺利完成，要在以下各阶段检查项目的进度。

制订方案，分析问题 1

确定小组研究的问题，规划使用的数据结构，并进行基本分析。 P114

设计算法 2

分析分解后的每一个问题，设计其求解算法。 P126

完成项目，汇报交流 3

完整解决问题，发布项目应用程序，并形成研究报告。 P135

项目总结

完成本章学习，发布项目应用程序，形成研究报告，并在全班范围内展示、汇报小组研究成果，进行评价。进一步理解算法与数据结构的关系。

4.1 算法

学习目标 ▶▶▶

- 掌握解决问题的一般过程，能对生活中的实际问题进行抽象，并选择合适的数据结构。
- 理解算法与数据结构的关系，能在确定数据结构的基础上设计算法，并编程实现。

体验探索

“常胜将军”与算法

生活中，有一类二人对弈问题，在对弈中，有一方总能获胜，因而常被称为“常胜将军”。

例如，小明和小华二人玩取火柴的游戏，共有21根火柴，如图4.1.1所示。二人轮流取，每人每次最多取4根，最少取1根。谁取到最后一根火柴则视为告负。小华让小明先取，结果每次都是小华赢。

图4.1.1 “常胜将军”游戏

思考：

1. 小明每次分别取1根、2根、3根、4根火柴时，小华需要取几根才能保证自己必然获胜？

2. 如果让计算机模拟解决这一问题，需要将解决问题的过程转化为可执行的若干步骤。试分析如何在计算机中分别表示小明和小华取的火柴数，如何设计这些步骤，并让计算机执行，才能实现这一问题的解决。画出流程图表示这一过程。

4.1.1 算法与问题解决

以计算机为核心的现代科学技术帮助我们解决了各种各样的问题，如网上挂号、即时通信、网上购物和信息查询等，极大地方便了我们的生活。

思考活动

对弈程序中的数据结构与算法

开发五子棋对弈程序，关键在于让计算机能够“认识”棋盘和棋子，并让计算机识别出“五子相连”的情况以终止对弈。

思考：

1. 如何设计数据结构使得计算机可以存储整张棋盘的棋子状态？

2. 针对已选择的数据结构，能设计哪些算法来实现计算机判断“五子相连”？

使用计算机解决问题，首先要为这个问题建立数学模型，将问题转化为计算机可处理的对象，然后再把解决问题的过程描述为若干操作步骤，并将这些步骤转化为计算机可识别、可计算的基本操作。使用计算机解决问题的一般过程通常包括提出问题、分析问题、设计方案、编程调试和解决问题等环节，如图4.1.2所示。根据问题求解的需要，整个过程可能要反复修正，直至问题得到有效解决。

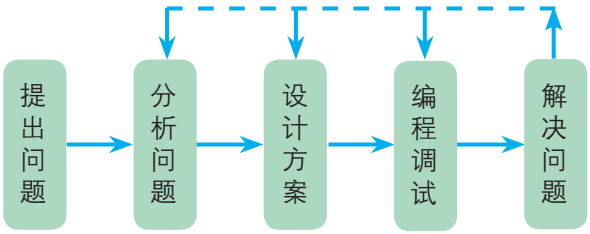


图4.1.2　使用计算机解决问题的一般过程示意图

图4.1.2中，“设计方案”环节是对解决问题的方法与步骤进行规划和设计，对这些步骤的描述就是算法。

算法的描述方法与计算机程序设计语言无关，它可以用流程图、伪代码语言、数学语言和自然语言等方式描述，但算法的执行必须通过具体的计算机程序设计语言编程来实现。

4.1.2 算法与数据结构的关系

计算机科学家尼古拉斯·沃斯提出了计算机科学领域著名的公式：

算法 + 数据结构 = 程序

思考活动

汉字卡片的检索

现有两套内容相同的汉字卡片，每套卡片都是100张。其中一套按照汉语拼音顺序排列，另一套无序。

思考：

1. 如果用计算机分别存储这两套卡片，需要分别选择哪种数据结构？
2. 若要在两套卡片中分别检索汉字“能”，检索算法有何不同？

通过“汉字卡片的检索”思考活动，我们发现，同样使用数组存储两套卡片，但由于两套卡片一套有序，一套无序，即两套卡片的数据关系不同，因而必然会采用不同的检索算法。

通俗地说，数据结构是数据组织和存储的方式，算法是数据的处理过程。数据结构是算法实现的基础，算法总是要依赖于某种数据结构来实现。例如，上述思考活动中的案例，脱离了卡片的数据结构，根本无法设计算法。不仅如此，一般对于同一个问题，如果选用不同的数据结构，则解决问题的算法也不同，即使已选择了认为合适的数据结构，在算法设计和程序实现的过程中，数据结构也可能被修改。算法是计算机科学的重要基石，数据结构又是算法研究的基础。

在实际进行方案设计时，算法的设计通常会同时伴有数据结构的设计，二者的目的都是解决问题。要解决问题，数据结构和算法缺一不可，而程序可以说是算法与数据结构的有机结合。因此，算法与数据结构是问题求解中相辅相成、不可分割的两个方面。

4.1.3 算法分析

同一问题可用不同算法解决，不同算法的执行效率可能不同，而算法的效率直接决定程序的效率。算法分析就是研究算法，以达到提高计算机解决问题效率的目的。

思考活动

素数判断算法及其优化

计算机在数据存储与传输中常常需要用到加密算法。由于大位数的素数（大于 1 且除了 1 和它本身以外不再有其他因数的自然数）相乘容易，但乘积的素因数分解困难，所以素数判断常被用于加密算法中。素数判断算法如图4.1.3所示。

思考：

除了2以外的素数都是奇数，那么，如何利用这一特性对图4.1.3所示的算法进行优化？

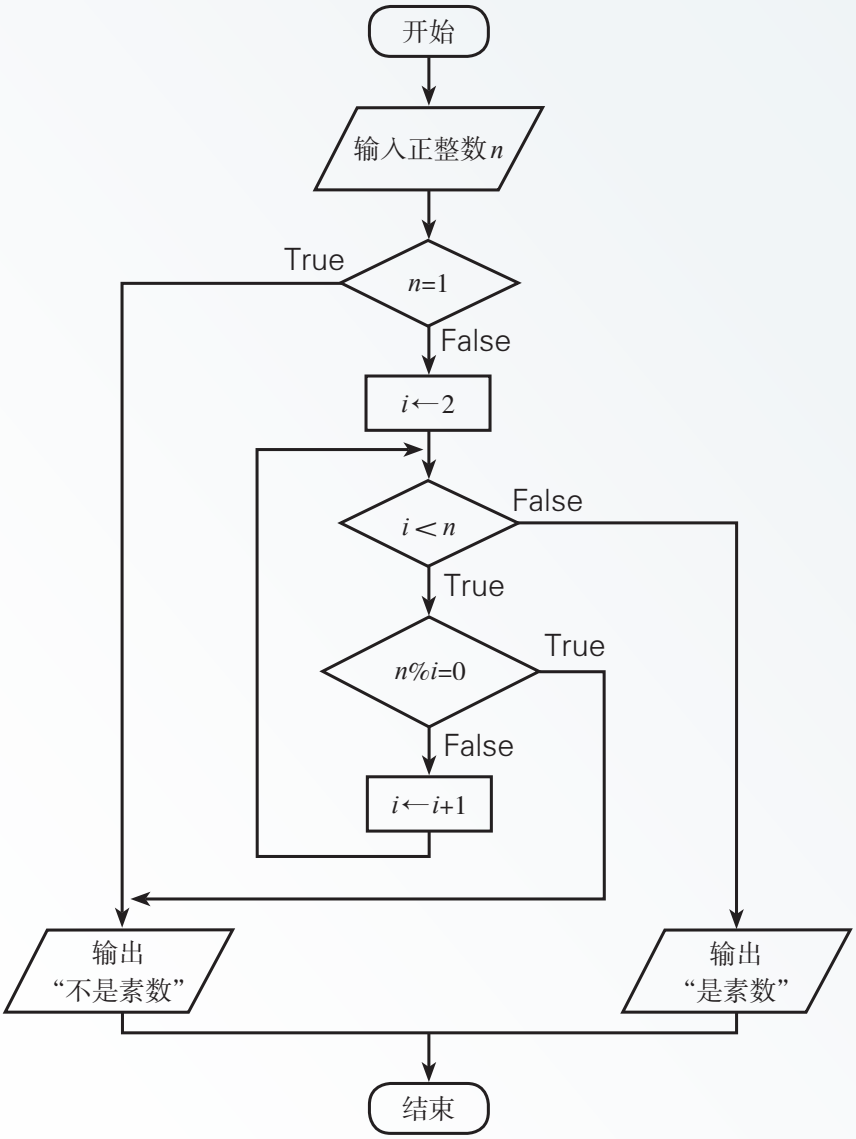


图4.1.3　素数判断算法流程图

实际上，对于任意大于1的正整数n，如果n不是素数，则$\frac{n}{2}$是除了n本身以外可能存在的最大因数，即n不可能被大于$\frac{n}{2}$的整数整除。因此只需判断$1\sim\left(\frac{n}{2}+1\right)$是否存在$n$的因数即可，可据此对如图4.1.3所示的算法进行优化。

事实上，该算法在此基础上还可以进一步优化。对于任何大于1的正整数n，如果n不是素数，那么，必然存在$n=m\times p$（m和p皆为正整数）。其中，m和p必有其一小于$\sqrt{n}$，故只需判断$2\sim\sqrt{n}$是否存在n的因数即可，可据此对如图4.1.3所示的算法进行进一步优化。

从直观上判断，如图4.1.3所示的算法与优化后的两个算法的执行时间是逐渐缩短的，而实际的执行效果究竟有多大差距需要编写程序进行验证。在相同条件下，让不同算法对应的程序对规模相同的数据进行操作，然后比较它们所用的时间，显然，用时少的程序效率高。

实践活动

比较素数判断程序效率

1. 编写3个函数，分别实现3个素数判断算法。

2. 编写程序，分别调用前面编写的函数计算100 000以内的素数个数，并使用日期时间函数比较3个程序的效率。

技术支持

Python语言中的日期时间函数

Python语言的datetime模块封装了很多日期时间的类和对象，用以完成一些特定的功能。datetime模块中常用的一些方法如表4.1.1所示。

表4.1.1　datetime模块中的常用方法

方法/属性	功能说明
`datetime.datetime.now()`	返回当前本地时间
`datetime.datetime.fromtimestamp(timestamp)`	返回指定时间戳对应的时间
`datetime.timedelta()`	返回一个时间间隔对象，可以直接与datetime.datetime对象做加减运算

1. 导入模块

```
import datetime
```

2. 引用当前时间

```
t1 = datetime.datetime.now()
```

3. 计算时间差

```
t1 = datetime.datetime.now()
t2 = datetime.datetime.now()
n = t2 - t1
```

一般来说，算法设计应考虑达到以下目标。

■ 正确性

算法的正确性指对任意一个合法的输入经过有限步操作后，算法应给出正确的结果，这是算法必须要达到的首要目标。例如，判断素数的算法应当能准确判断出“5”是素数，“25”不是素数。

■ 可读性

算法的可读性指一个算法可供人们阅读的容易程度。为增加算法的可读性，通常要在算法中正确使用缩进和注释信息。

■ 健壮性

算法的健壮性指一个算法对不合理数据输入的反应能力和处理能力，也称为容错性。例如，素数只能是大于1的自然数，如果输入“1”就属于不合理数据，此时算法应能判断出“1”是不合理数据，并输出类似于“1不在处理范围之内，请重新输入”的提示信息。这样，就增强了算法的容错性。算法的健壮性是在不断调试和改进中逐渐增强的。

■ 时间效率高和存储空间消耗低

对于解决同一问题的多个算法，运行时间越短、运行速度越快的算法，时间效率就越高，算法性能越优异。例如，素数判断的三个算法中，经过第二次优化后的算法时间效率最高。另外，算法在运行过程中占用和消耗的存储空间越少，算法的空间性能越高。

计算机科学中，可以用时间复杂度描述算法的运行时间量。

简单地讲，算法的时间复杂度指算法的运行时间，通常用函数描述（称为时间复杂度函数）。时间复杂度函数通常可以描述为输入问题规模n的某个函数$f(n)$，记作：$T(n)=O(f(n))$。它表示随着问题规模n的增大，算法运行时间的增长率和$f(n)$的增长率相同。使用这种方法，时间复杂度可被称为是渐近的，亦即考查输入值大小趋近于无穷大时的情况。例如，如果一个算法对于任何大小为n的输入，它至多需要$5n^3+3n$的时间运行完毕，那么，它的渐近时间复杂度是$O(n^3)$（不包括这个函数的低阶项和最高阶项系数）。表4.1.2列出了常见的函数阶，并按照n趋近于无穷大时函数阶增长的快慢进行排序，增长得慢的函数列在表的前面。

表4.1.2　常见的函数阶

函数阶	级别	算法示例
$O(1)$	常数级	一般的公式计算，如$a=b+c$；判断一个二进制数的奇偶
$O(\log_2 n)$	对数级	折半查找（二分查找）
$O(n)$	线性级	按照素数定义编写的素数判断；无序数组的搜索
$O(n^2)$	平方级	冒泡排序；插入排序

项目实施

制订方案，分析问题

一、项目活动

1. 依据项目目标，从项目分工、项目内容、项目过程和项目检查等方面制订项目方案。

2. 参考“编写对弈程序”主题，根据小组项目方案，确定要解决的问题是哪些，再把每个问题的解决过程分解为若干步骤，找到算法中的输入项和输出项。分析这些输入项中哪些可以由算法自行获得，哪些由用户手工输入，及各自具有的特征。参考表4.1.3～表4.1.5，自行设计并填写相应表格。

表4.1.3　问题分解表（样表）

主题	五子棋对弈
问题1	对弈双方的表征
问题2	棋子、棋盘的表征
问题3	对弈双方落子的实现
问题4	棋盘的绘制
问题5	屏幕上的位置与棋盘点位的对应
问题6	棋子的绘制
问题7	对弈胜负的判定
⋮	

表4.1.4　算法步骤分析表（样表）

问题7	对弈胜负的判定
算法7	五子棋对弈判胜算法
步骤1	确定当前落子方（黑方或白方）
步骤2	检测水平方向落子方是否形成五子相连
步骤3	检测垂直方向落子方是否形成五子相连
⋮	

表4.1.5　算法的输入项和输出项表（样表）

算法7	五子棋对弈判胜算法
输入项1	鼠标点击位置
输入项2	落子点位
输入项3	落子颜色
⋮	
输出项1	点击棋盘位置后在棋盘上显示棋子
输出项2	落子后获胜的提示信息（黑方或白方获胜）
⋮	

3. 将研究问题按照数据结构和算法的研究范畴分别解决。参考下面给出的五子棋对弈问题的解决示例，根据项目问题分解表，列出与数据表示相关的问题，确定数据结构。

例如，问题1“对弈双方的表征”，五子棋对弈的规则是执黑先行，黑方和白方轮流落子。使用一个变量step来记录对弈的步数，step初值为0，每有一方落子，step值增加1。这样，通过step值的奇偶性可以表征对弈的双方，如表4.1.6所示。

表4.1.6　对弈步数变量step的取值含义表

step的取值	含义
0	未落子
奇数	黑方落子
非0偶数	白方落子

二、项目检查

1. 完成主题的确定与小组分工。
2. 完成对应主题的问题分解。
3. 完成主要问题的算法步骤和输入、输出分析。
4. 完成数据结构的分析。

练习提升

1. 如果想对五子棋进行复盘，就需要记录对弈双方所走的棋位，设计一种结构，记录对弈过程。

2. 设计一种数据结构，可以存储年级每个同学每周的课程表，并设计方案，以实现按姓名和星期进行检索。

3. 任选一款感兴趣的移动应用程序，分析它所解决的问题、可能应用到的数据结构及其算法。

4.2 迭代法

学习目标 ▶▶▶

- 体验迭代法解决问题的过程，能解决实际问题，发展计算思维。
- 通过体验排序与查找过程，能对具体问题进行抽象，合理选择数据结构，设计算法解决问题。

体验探索

背单词

小明制订了一个英语学习计划，其中一项内容是背单词：第1天背1个单词，第2天背2个单词，第3天背3个单词……即每天都比前一天多背1个单词，如图4.2.1所示。经过100天的努力，小明觉得收获颇丰。

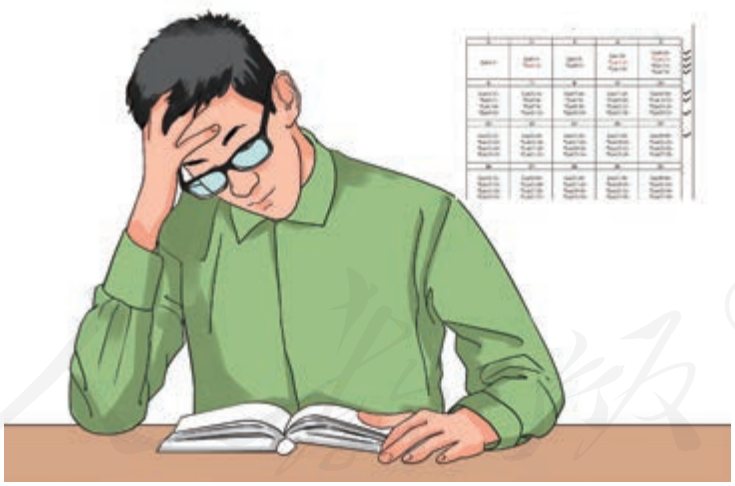


图4.2.1　小明背单词

思考：

1. 经过100天后，小明总共背了多少单词？你是如何算出来的？
2. 这个事例对你有什么启发？

4.2.1 迭代法的概念与特征

《荀子·劝学》中有云“不积跬步，无以至千里；不积小流，无以成江海。”要解决复杂问题，可以从简单的小问题开始，一步一步推进，最终解决复杂问题。背单词计数问题正是这样一个通过不断解决小问题，进而求解整个问题的典型示例。

思考活动

设计背单词计数问题的算法

使用计算机解决背单词计数问题，需要为这一问题设计算法。

思考：

如何设计算法实现背单词计数问题的求解？

例1：背单词计数。

首先，将背单词的过程描述如下：

第1天背了1个单词，累计背了1个单词；

第2天背了2个单词，比第1天多背1个单词，累计背了1+2=3个单词；

第3天背了3个单词，比第2天多背1个单词，累计背了（1+2）+3=6个单词；

第4天背了4个单词，比第3天多背1个单词，累计背了（1+2+3）+4=10个单词；

……

第100天背了100个单词，比第99天多背1个单词，累计背了（1+2+3+…+99）+100=5 050个单词。

这一过程可用表4.2.1记录。

表4.2.1 背单词过程记录表

时间	当天背的单词数	比前一天多背的单词数	累计背的单词数
第1天	1	1	1
第2天	2	1	1+2
第3天	3	1	1+2+3
第4天	4	1	1+2+3+4
⋮	⋮	⋮	⋮
第100天	100	1	1+2+3+4+…+100

通过分析，可以得到如下结论（设n为1～100的自然数）：

■ 第n天背的单词数为n；

■ 每天都比前一天多背1个单词，即每天背单词的增量为1；

■ 从第1天至第n天，总共背单词数就为1+2+3+…+n。

当n=100时，就可以统计出100天背的单词总数。

通过分析发现，背单词的计数问题可抽象并转化为连续自然数求和问题。

要使用计算机解决该问题，就要先选择合适的数据结构，再设计算法。

方案一：设计数组a，用循环将1至n依次存入数组，然后设计变量sum保存总和（初值为0），再把数组的每一个元素和sum相加，最后，sum值就是所求的结果。

方案二：使用变量sum保存总和（初值为0），再使用另一个循环变量i，让循环变量从1增加至100（步长为1），循环过程中执行sum与i相加。当循环结束时，sum值就是最终的结果。算法流程如图4.2.2所示。

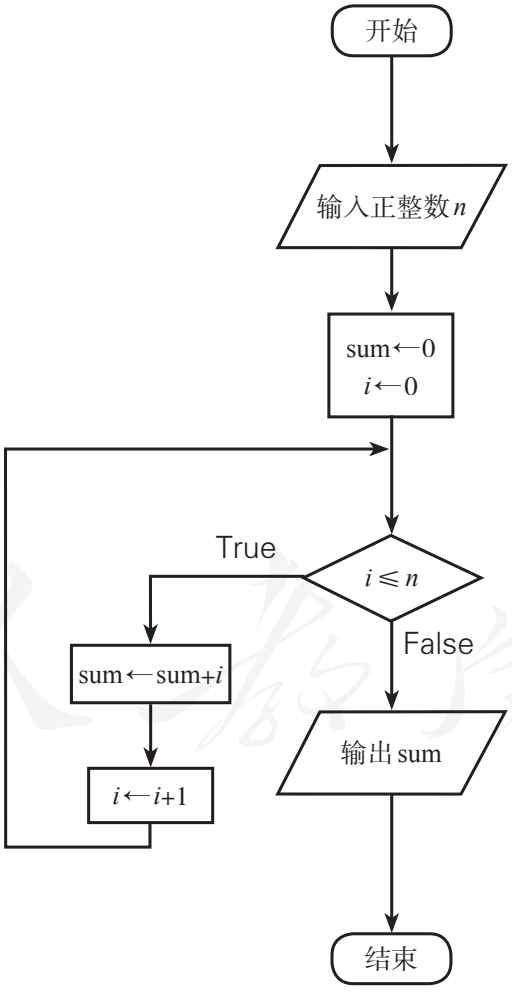


图4.2.2　背单词计数算法流程图

对比方案一和方案二，可以发现：方案二只使用变量来存储数据，比方案一使用数组更节省存储空间，效率也更高。

根据这一思路，按照图4.2.2，用Python语言实现背单词计数问题求解的程序如下：

```
# 输出显示提示信息，且光标不换行
print("请输入背单词的天数(为正整数)： ", end = "")
# 输入正整数，并转换为数值
n = int(input())
# 将和存放在sum中，初始值为0
sum = 0
for i in range(1, n + 1, 1):
    # 让i从1至n开始循环，步长为1即每天比前一天多背的单词数
    sum = sum + i
# 输出显示计算结果
print("第%d天背单词结束之后，背过的单词总数是%d。 " % (n, sum))
```

运行上述程序，输入背单词的天数100，运行结果如图4.2.3所示。

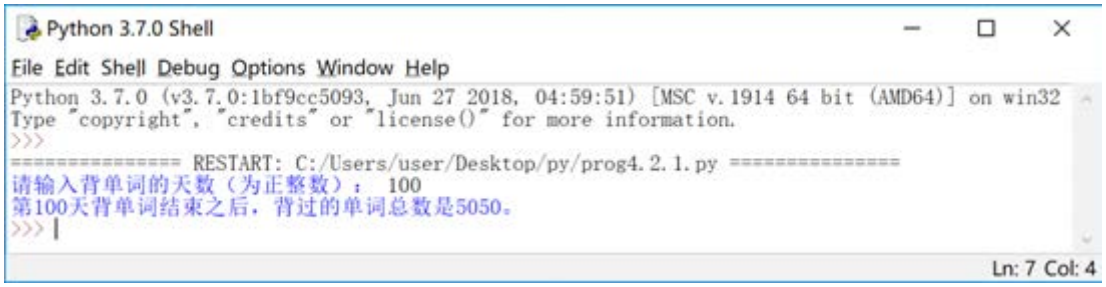


图4.2.3　求解背单词计数问题程序的运行结果

在求解背单词计数问题过程中，sum的值不断增加（也称为累加），而且每一次都用变量的旧值递推出变量的新值，直至得到最终的结果。这种不断用变量的旧值递推新值的过程称为迭代法，也称辗转法。使用迭代法解决问题需要具备三个要件：确定迭代变量、建立迭代关系式和确定迭代的终止条件。

■ 确定迭代变量

用迭代法解决问题时，迭代过程中至少存在一个直接或间接地不断由旧值推出新值的变量，这个变量就是迭代变量。在背单词计数问题中，sum的新值由旧值不断推出，因而sum就是迭代变量，它的初值为0，终值就是计算的最终结果。

■ 建立迭代关系式

迭代关系式指迭代变量由前一个值推出其下一个值的公式（或关系）。建立迭代关系式是解决迭代问题的关键。在背单词计数问题中，迭代关系式就是sum值的累加关系式：

$$\text{sum} = \text{sum} + i$$

■ 确定迭代的终止条件

通常，迭代的实现要使用循环，为防止循环无休止地重复执行下去，必须设置终止

条件。在背单词计数问题中，使用了for循环，循环条件是循环变量$i \leqslant n$。因此，终止条件就是循环变量$i > n$。

例2：求解扩展后的正三角形个数。

现有若干边长为1个单位的正三角形，如图4.2.4所示，由多个这样的正三角形可拼成边长为2、3、4个单位的正三角形，如图4.2.5所示。如果要拼合成边长为n的正三角形，需要多少个边长为1个单位的正三角形？

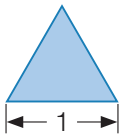


图4.2.4　边长为1个单位的正三角形

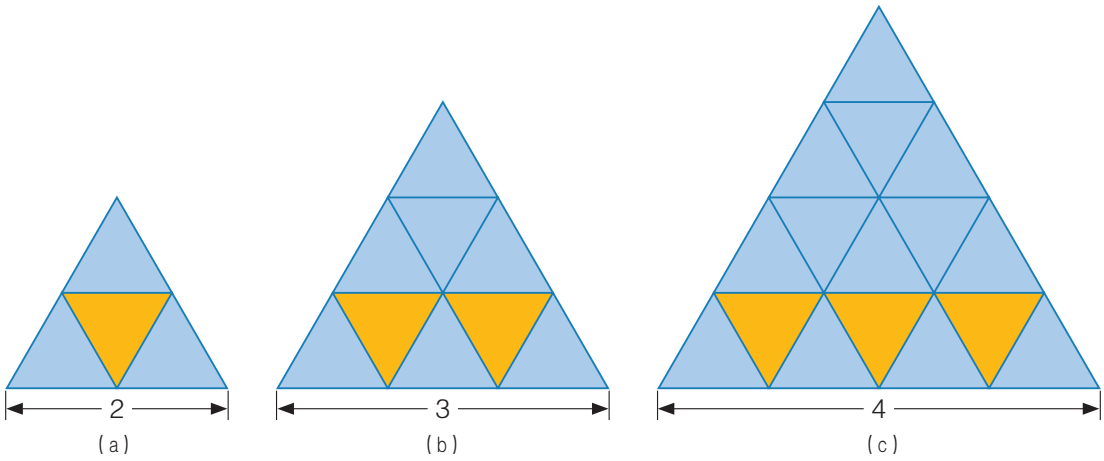


图4.2.5　边长分别为2、3、4个单位的正三角形

这里，用$f(n)$表示拼成边长为n的正三角形所需边长为1个单位的正三角形的数量。显然，$f(1)=1$。

通过观察，不难发现：

由边长为1个单位的正三角形，拼成边长为2个单位的正三角形，可视为增加2个蓝色正三角形和1个橘色正三角形，如图4.2.5(a)所示。因而，$f(2)=f(1)+2+1=4$；

由边长为2个单位的正三角形，拼成边长为3个单位的正三角形，可视为增加3个蓝色正三角形和2个橘色正三角形，如图4.2.5(b)所示。因而，$f(3)=f(2)+3+2=9$；

由边长为3个单位的正三角形，拼成边长为4个单位的正三角形，可视为增加4个蓝色正三角形和3个橘色正三角形，如图4.2.5(c)所示。因而，$f(4)=f(3)+4+3=16$；

……

同样地，由边长为$n-1$个单位的正三角形，拼成边长为n个单位的正三角形，可视为增加n个蓝色正三角形和$n-1$个橘色正三角形。因而，$f(n)=f(n-1)+n+n-1=f(n-1)+2n-1$。

这一过程可以用表4.2.2表示。

表4.2.2　正三角形扩展过程表

正三角形边长	正三角形个数	扩展后正三角形边长	增加的正三角形个数	扩展后正三角形个数
1	1	2	2+1=3	$f(2)=1+3=4$
2	4	3	3+2=5	$f(3)=4+5=9$
3	9	4	4+3=7	$f(4)=9+7=16$
⋮	⋮	⋮	⋮	⋮
$n-1$	$f(n-1)$	n	$n+(n-1)=2n-1$	$f(n)=f(n-1)+2n-1$

由此可见，扩展后的正三角形个数能够用迭代法求解，其迭代关系式为$f(n)=f(n-1)+2n-1$，且$f(1)=1$。

实践活动

补全程序，求解扩展后的正三角形个数

将下面的程序补充完整，运行程序，输入合理的数值，观察结果。

```
# 输出显示提示信息，且光标不换行
print("请输入正三角形的边长(为正整数)：", end = "")
# 输入正整数，并转换为数值
n = int(input())
# 将和存放在f中，初始值为0
f = _______
# 让i从__至__循环，步长为1
for i in range(_____, _____, 1):
    f = ____________
    # 输出计算结果
    print("拼成边长为%d的正三角形需要%d个边长为1的正三角形" % (i, f))
```

例3：求解最大公因数。

要解决求正整数m和n的最大公因数问题，不妨假设$m=q\times n+p$，即m除以n的商为q，余数为p（其中，m、p、q、n皆为正整数，且$m\geqslant n$，$n>p$）。

如果r为m和n的公因数，则必然存在$m=r\times a$、$n=r\times b$（r、a、b皆为正整数），代入$m=q\times n+p$，得到$r\times a=q\times r\times b+p$。移项、提取公因式可得$p=r\times(a-q\times b)$，即$r$必然也是余数$p$的因数。

因此，所有m和n的公因数，都是n与p的公因数。同样，所有n与p的公因数，也都是m与n的公因数。所以，m与n的最大公因数为n与m除以n后的余数p的最大公因数。

通过上述分析，可以发现：只要不断地求余数，当余数为0时，此时的除数就是最大公约数了，这就是求解最大公因数的算法。

实践活动

编写程序，求解最大公因数

根据算法流程图4.2.6，使用Python语言，编写求最大公因数的程序，并验证其正确性。

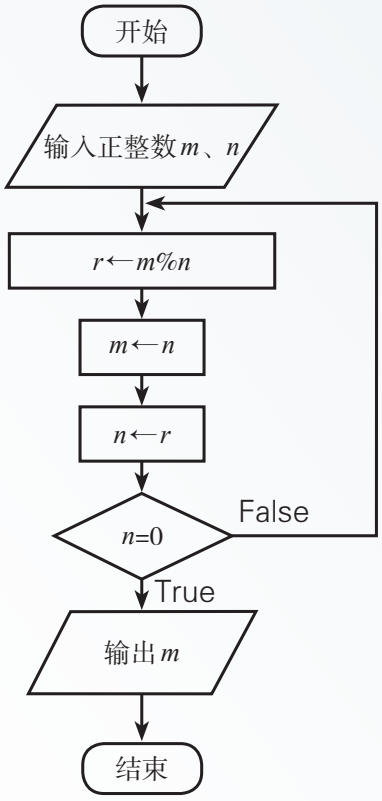


图4.2.6　求最大公因数的算法流程图

上述求最大公因数的算法，称为辗转相除法。我国古代数学专著《九章算术》中记载的“更相减损术”与此类似。辗转相除法属于迭代法，它同背单词计数问题求解的过程有所不同，不同之处在于辗转相除法的迭代次数未知，但是可以确定迭代的终止条件为“余数等于0”。

背单词计数、求解扩展后的正三角形个数和求最大公因数的共同特征是计算结果为精确值，这种迭代称为精确迭代。当无法计算出精确值时，使用的迭代称为近似迭代，如求算术平方根时常用的迭代公式 $x_1=\frac{1}{2}(x_0+\frac{a}{x_0})$。

实际上，迭代思想还可以用来指导我们的学习和生活。通过求解背单词计数问题，我们发现：只要每天努力多一点，日积月累，就可以产生惊人的变化。

4.2.2　迭代法的应用

迭代法除了可以解决前面提到的背单词计数、求最大公因数、求算术平方根近似值等问题外，其思想还可以用来解决计算机科学中的排序和查找等问题。

思考活动

设计排序算法

某校高一年级组织了 7 名学生参加演讲比赛，这 7 名学生的选手编号分别为 1 ～ 7 号。在观众投票环节，选手们获得的票数如表 4.2.3 所示。

表 4.2.3　参赛选手得票数统计表

选手编号	1 号	2 号	3 号	4 号	5 号	6 号	7 号
观众投票数	5	23	7	46	33	12	40

思考：

如果需要用计算机按照得票数对参赛选手们进行排序，如何设计算法？

互联网的信息量非常大，比如在如图 4.2.7 所示的手机购物应用中，输入“算法”查找到 6 005 条与“算法”有关的图书资料信息。要从中找到一本最贴近需求的书，往往需要对这些列出的数据按照一定的规则进行重新排列。例如，按照销量、价格、好评或者出版时间等排列。这个过程就是排序，而排序所依据的数据项称为关键字。

图 4.2.7　某款手机应用中的图书查找界面

排序是计算机中经常进行的一种操作，其作用是将一组“无序”的数据元素序列按关键字的顺序（升序或降序），调整为“有序”的数据元素序列。

为方便学习，我们约定数据元素存储在数组中，排序关键字为整型数据，对数据序列进行升序排列。

冒泡排序

在众多排序算法中，冒泡排序是最为常见的排序算法之一。它的基本思想是：两两比较相邻元素，如果反序则交换这两个元素，直到整个序列没有反序的元素为止。

如何根据冒泡排序的基本思想，对表 4.2.3 中的数据进行排序？显然，这个排序问题中的关键字是票数值。为简单起见，数组中只存储选手们获得的票数。设数组名为 a，则排序前的数组 a 如图 4.2.8 所示。

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
5	23	7	46	33	12	40

图4.2.8　排序前的数组a

根据冒泡排序的基本思想，整个排序过程分析如下：

第1趟冒泡排序从第1个关键字开始，依次比较相邻的两个关键字，如果第一个关键字大于第二个关键字，则交换这两个关键字。第1趟冒泡排序过程如图4.2.9所示。

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
初始状态	5	23	7	46	33	12	40
第1次比较后	5	23	7	46	33	12	40
第2次比较后	5	7	23	46	33	12	40
第3次比较后	5	7	23	46	33	12	40
第4次比较后	5	7	23	33	46	12	40
第5次比较后	5	7	23	33	12	46	40
第6次比较后	5	7	23	33	12	40	46

图4.2.9　第1趟冒泡排序过程

7个关键字经过6次两两比较后，第1趟冒泡排序结束。排序结果是最大关键字“46”被交换到a[6]。第2趟冒泡排序只需要对前6个关键字进行比较。完整的冒泡排序过程，如图4.2.10所示。

	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	比较次数
初始状态	5	23	7	46	33	12	40	
第1趟排序后	5	7	23	33	12	40	46	6
第2趟排序后	5	7	23	12	33	40	46	5
第3趟排序后	5	7	12	23	33	40	46	4
第4趟排序后	5	7	12	23	33	40	46	3
第5趟排序后	5	7	12	23	33	40	46	2
第6趟排序后	5	7	12	23	33	40	46	1

图4.2.10　冒泡排序每趟排序的结果

观察上述排序过程可以发现，7个数据元素需要进行6趟冒泡排序。第1趟需要对7个数据元素进行6次比较，而后的每一趟中需要比较的元素个数比上一趟少1。因此，每一趟中的元素比较次数也比上一趟少1次。

所以，如果待排序列中有n个数据元素，则需要进行$n-1$趟冒泡排序，第i趟排序中的元素比较次数为$n-i$（$1 \leqslant i \leqslant n-1$）。

冒泡排序的基本过程如下：

1. 比较第1个和第2个元素，如果第1个比第2个大，就交换这两个元素。再比较第2个和第3个元素，如果第2个比第3个大，就交换这两个元素。依此类推，最后比较倒数第二个元素和最后一个元素，如果倒数第二个元素比最后一个元素大，就交换这两个元素。此为第1趟排序，第1趟排序结束后，最后一个元素就是所有元素中最大的。

2. 按照第1趟排序的方法，对除最后一个元素外的其他所有元素进行第2趟排序。依此类推，持续对个数越来越少的元素进行比较，直到第$n-1$趟结束为止。

在这种排序方法中，数值较大的数据元素会像“冒泡”一样经由交换“浮”到数组的末端，冒泡排序法因此而得名。

实践活动

冒泡排序算法的编程实现与优化

1. 编写程序，用冒泡排序算法实现演讲比赛选手排序。

2. 仔细研究图4.2.10中冒泡排序每一趟的过程，可以发现第4趟没有发生数组元素的交换，说明此时数组已经是有序的，无须再进行第5趟和第6趟排序。请根据这一发现，编写程序，优化冒泡排序算法。

顺序查找

顺序查找指在一个无序（或有序）数组中，从头到尾逐一进行比较，找出关键字与给定值相同的数组元素。如果找到，则查找成功，返回该数组元素的下标；否则，查找失败，返回“-1”。

小明最近迷上了诗词。他制作了大量诗词卡片以便随时吟诵，并对每张卡片都进行了编号。部分卡片的编号存于如图4.2.11所示的数组中。现在，他要查找编号为“77”的卡片。如果采用顺序查找法，应该如何查找呢？

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
23	75	68	99	31	46	22	77	59	12

图4.2.11　存储部分卡片编号的数组

按照顺序查找法的思路，从下标为“0”的元素开始，逐一访问这个数组的所有数据元素，并与关键字“77”进行比较。如果相等，查找成功，并返回该元素的下标；如果不相等，就继续向前查找，当数组中没有元素可以访问时，说明数组中没有元素“77”，查找失败，返回“-1”。

实践活动　顺序查找算法的实现

1. 按照顺序查找法的思路，填写表4.2.4。

表4.2.4　顺序查找过程

步数	数组下标	关键字	与给定值“77”是否相等
1	0	23	否
2	1	75	否

2. 编程实现顺序查找算法，并输入数据，验证其正确性。

3. 小明和小清各自用Python语言编写了两个不同的顺序查找函数，小清觉得自己的算法比小明的更加实用一些。请设计一个方案，编写程序体验这两段程序的运行效果，分析哪一个算法更加合适，并阐明理由。

（1）小明的函数

小明设计的函数有两个参数：*a*表示数组；*n*表示要查找的值。若查找成功，返回True；否则，返回False。

```
def search_ming(a, n):
    for i in range(len(a)):
        if a[i] == n:
            return True
    return False
```

（2）小清的函数

小清设计的函数也有两个参数：*a*表示数组，且数组的第一个元素为空；*n*表示要查找的值。若查找成功，返回查找到的元素的位置；否则，返回“0”。

```
def search_qing(a, n):
    a[0] = n
    i = len(a) - 1
    while a[i] != n:
        i = i - 1
    return i
```

项目实施

设计算法

一、项目活动

在前面的项目研究中，已经对确定的主题进行了问题的分解，并为所研究的问题选择了数据结构。参考五子棋对弈的分析方式，根据表4.1.3，列举其中需要使用算法解决的问题，并设计算法。

例如，对表4.1.3中问题3“对弈双方落子的实现”的分析如下：

对弈的每一方落子相当于改变该点位对应的二维数组中元素的值。落子后要检查该方是否满足五子相连的条件，满足就是该方获胜，对弈结束；若不满足，对方继续落子……重复此过程。对弈的过程就转化为对弈结束条件是否满足的判定，即设计五子棋判胜算法。

设计五子棋判胜算法，首先要了解五子棋判胜和判负的规则，以最简单的判胜规则——五子相连为例，即相连的同一颜色棋子数等于5时即为获胜。

假如一方在棋盘上落子后获胜，那么，必然会出现以下四种相连情况之一：在同一行上，五子相连；在同一列上，五子相连；从左上至右下方向，五子相连；从右上至左下方向，五子相连。

五子相连的判胜算法即逐一判断上述四种情况是否存在，如存在，则提示落子方获胜，对弈结束。如果四种情况都不存在，即落子方没有取胜，则对弈继续。

用数组$a[14][14]$表示棋盘，落子点位对应数组元素$a[m][n]$，使用循环巧妙改变m和n的值，即可判断棋子相连的情况。

二、项目检查

设计已确定主题的主要算法，并编程实现。

练习提升

1. 若用链表存储表4.2.3所示的演讲比赛选手们的得票数据，如何编写程序实现冒泡排序算法？

2. 小明到校图书馆借阅《深度学习与Python》一书，图书管理员告诉他，这本书存放在图书馆X区的书架上。请设计算法模拟小明在书架上找书的过程，并编写Python程序实现这一算法。

4.3 递归法

学习目标

- 体验使用递归法解决问题的过程，解决实际问题，提升计算思维能力。
- 理解迭代法和递归法的异同，利用迭代或递归思想指导算法设计。

体验探索

汉诺塔问题

汉诺塔问题是由一位法国数学家在19世纪提出的，直到现在，简化版的汉诺塔玩具在玩具店中仍然随处可见。汉诺塔的假设是：在一个铜板上，插着三根宝石针，分别被编号为a、b、c。在a针上，从下到上放置了由大到小的 64 个金片，要把这些金片借助b针，从a针全部移动到c针。移动规则是：一次只能移动一片，并且无论在哪一根针上，小片必须放置在大片上面。

思考：

1. 若a针上只有3个金片，如图4.3.1所示，如何将3个金片从a针移到c针？

图4.3.1　汉诺塔示意图

2. 如果a针上有4个金片，如何将4个金片从a针移到c针？如何设计算法实现这个过程？

4.3.1 递归法的概念与特征

生活中有这样一种蔬菜——洋葱，它的可食用部分是一层一层包裹起来的，仅从外观上看，无从判断它的内里。同样，有这样一种问题，也是层层包裹起来的，我们只有像“剥洋葱”一样，层层分解，才能把一个复杂问题转化为简单问题来解决。

思考活动

小球的排列与阶乘

有5个颜色各不相同的小球，将这5个小球按顺序排列在一起。

思考：

共有多少种排列方法？

小球排列是数学中典型的排列问题，根据数学知识，可以得到共有$P_5^5=5!$种排列方法。

5!即5的阶乘。一个自然数的阶乘是所有小于及等于该数的正整数的积，并且规定0!=1。

若$f(n)=n!$，根据阶乘的定义，可以得出：

$n!=n\times(n-1)\times\cdots\times5\times4\times3\times2\times1$，即

$1!=1$，$2!=2\times1$，$3!=3\times2\times1$，$4!=4\times3\times2\times1$，$\cdots$

分析这些算式，可以发现它们存在着如下关系：

$1!=1$，$2!=2\times1!$，$3!=3\times2!$，$4!=4\times3!$，$\cdots$，$n!=n\times(n-1)!$。

因而，

$$f(n)=\begin{cases}1 & n=0,\ 1\\ n\times f(n-1) & n>1\end{cases}$$

为了计算$f(5)$，要先计算$f(4)$；而要计算$f(4)$，又得先计算$f(3)$；欲计算$f(3)$，又得先计算$f(2)$；计算$f(2)$，又得先计算$f(1)$。$f(1)$的值为1!=1。

这个计算过程，如图4.3.2所示。

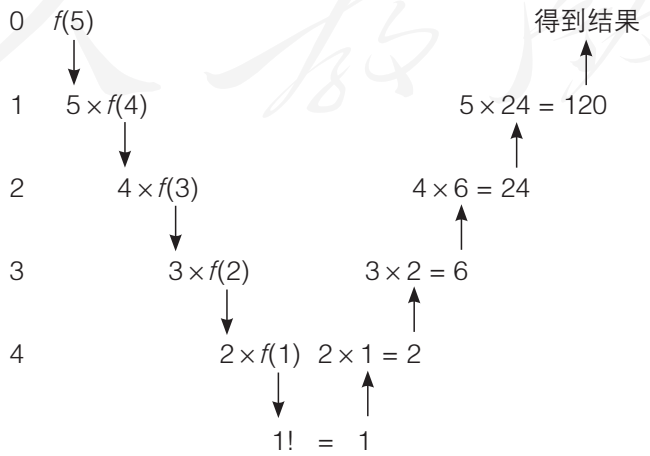


图4.3.2 自然数“5”的阶乘计算过程示意图

根据问题分析过程，使用Python语言计算自然数n的阶乘的程序如下：

```
# 定义函数f()，参数n为自然数，返回n的阶乘。
def f(n):
    if(n <= 1):
        # 如果n小于或等于1，那么返回n的阶乘为1
        return 1
    else:
        # 如果n大于1，那么返回n的阶乘为n*f(n-1)
        return n * f(n - 1)
# 输出提示信息，且光标不换行
print("请输入一个正整数：", end = "")
# 输入数字，并转换为整数
i = int(input())
# 输出计算结果
print("%d的阶乘是%d。" % (i, f(i)))
```

运行程序，按照提示信息，输入一个正整数，如输入“5”，运行结果如图4.3.3所示。

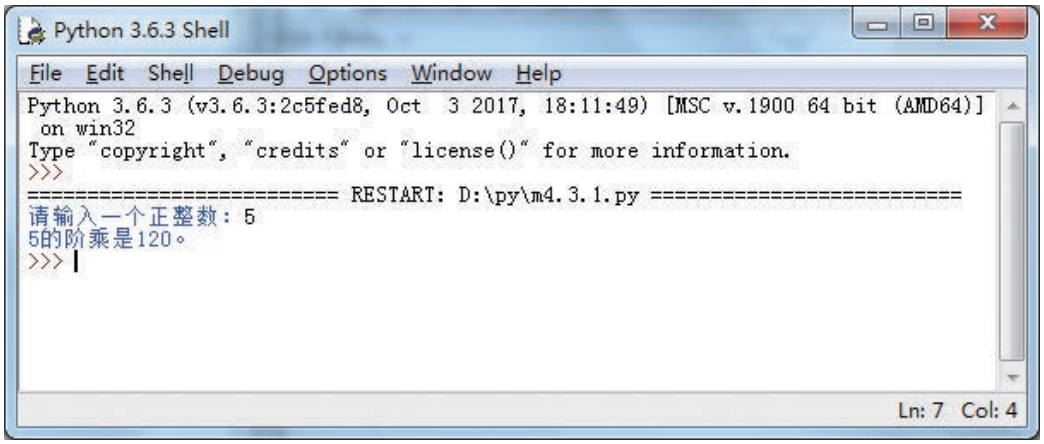


图4.3.3　计算自然数“5”的阶乘程序的运行结果

在上述程序中，函数f又调用了函数f，这种通过直接或间接地调用自身来解决问题的方法被称为递归。

使用递归算法解决问题需要具备两个要件：递归定义和递归的边界条件。

■ 确定递归定义

使用递归解决的问题都可以通过同一套规则（即相同的程序）转化为比该问题更为简单的子问题，这套规则被称为该问题的递归定义或递归公式。例如，$f(n)= n \times f(n-1)$就是计算阶乘的递归公式。

■ 确定递归的边界条件

经过不断缩小问题规模，问题最终能够得以解决。如图4.3.2所示，5!经过不断简化，最终转化为求1!，而1!=1。这种能直接得到结果，从而终止递归的情况，称为递归的边界条件。

实践活动

再谈斐波那契数列

在第2章学习数组时我们知道，斐波那契数列指的是这样一个数列：1，1，2，3，5，8，13，21，34，…即从第3项开始，每一项都是前面两项的和。如果用$f(n)$表示计算斐波那契数列的函数，那么，$f(n)=f(n-1)+f(n-2)(n>2, n\in \mathbf{N})$。

请根据斐波那契数列的递归定义，参考图4.3.2，画出其递归过程的示意图，并编写计算斐波那契数列的程序。

不仅是在数学问题中存在递归，生活中也存在很多类似递归的现象，如图4.3.4所示的玩具套娃。只要认真观察，就能发现更多生活中的递归，甚至还能创造更多的递归，体会生活之美。

图4.3.4　玩具套娃

4.3.2　递归法的应用

使用递归法可以解决许多复杂困难的问题，这些问题有一些共同特征：通过一套相同的规则，最终转化为一个相对比较简单的小问题，然后再一步一步地返回，最终使问题得以解决。本节体验探索中的汉诺塔就是这样一个问题。要想使用递归法解决问题，首先得证明这个问题的每次转化都能满足同一套规则。

思考活动

用递归法解决汉诺塔问题

本节的体验探索中提出了3个金片和4个金片的汉诺塔，如果用迭代法解决，将会十分复杂，甚至无从下手。

思考：

汉诺塔问题能否用递归法解决？请说明理由。

以汉诺塔问题为例，说明使用递归法解决问题的一般步骤。

1. 推断问题可用递归法解决

对于a上的64个金片，如果能把前63个金片借助c移动到b，那么就可以将a上的第64个金片直接由a移动到c。这样64个金片的移动问题转化为解决63个金片的移动问题。

对于b上的63个金片，如果能把前62个金片借助c移动到a，那么就可以将b上的第63个金片直接由b移动到c。这样63个金片的移动问题转化为解决62个金片的移动问题。

……

可以看出，汉诺塔问题是一个典型的递归问题。递归的终止条件是1个金片的移动。

2. 确定递归定义

对于a上的n个金片，需要把a上面的$n-1$个金片借助c移动到b上，a上面仅剩1个金片，将这个金片由a移动到c上即可。然后再将b上的$n-1$个金片，借助a移动到c上，就完成了金片移动。这是汉诺塔递归定义的文字表述。

如果用Hanoi表示求解汉诺塔问题的函数，那么函数Hanoi与金片的数量n以及a、b、c三根针的位置有关。

用Hanoi(n, a, b, c)表示n个金片从a通过b移动到c，move(a, b)表示金片从a移动到b，那么第n个金片的移动过程应如下：

将a上的$n-1$个金片通过c移动到b，即Hanoi($n-1$, a, c, b)；

将a上最后一个金片移动到c，即move(a, c)；

将b上的$n-1$个金片通过a移动到c，即Hanoi($n-1$, b, a, c)。

对于n=1的情况，只需move(a, c)，由此我们就得到了汉诺塔的递归定义。

实践活动

编程解决汉诺塔问题

1. 编程实现汉诺塔问题的解决，运行程序，并验证其正确性。

提示：move(a, b)在屏幕上输出“由a移动到b”。

2. 改变汉诺塔程序的输入参数，观察程序的运行结果，推断金片的移动次数与金片数之间的关系公式，推算64个金片的移动次数。

折半查找

顺序查找一般常用于被查找对象所在序列是无序的或排序情况未知的情况。实际生活中，我们经常遇到这样的情况：被查找对象所在序列是有序的。例如，新华字典中的汉字是按照汉语拼音顺序排列的。如果在字典中查找“希”字，我们通常不会从字典的第一页开始找起，而是先大致翻到后面的位置，如果找不到就继续推测是往前翻还是往后翻，而且每次翻动的页数越来越少，直到找到为止。

用计算机模拟类似查字典的过程，把汉字抽象转化为相应的编码数字，假设字典中部分汉字对应的编码数字已存入如图4.3.5所示的数组a中。现要查找编码数字为“112”的汉字。

$a[0]$	$a[1]$	$a[2]$	$a[3]$	$a[4]$	$a[5]$	$a[6]$	$a[7]$	$a[8]$	$a[9]$	$a[10]$
23	35	46	57	68	72	83	91	100	112	125

图4.3.5　数组a

数组a已经是有序的序列，如果按顺序查找法查找，则需要依次将给定值和数组中的每个元素进行比较。实际上，在和该数组中任一元素比较后，不仅能知道两者的大小关系，而且，在两者不相等时，还能确定下一步的查找范围。比如和$a[5]$比较，“112”大于“72”，所以下一步应当向大于$a[5]$的方向比较，而不是小于$a[5]$的方向。

那么，如何选择第一次需要比较的数组元素呢？最简单的方法就是选择数组的中间元素，如果中间元素等于待查的值，则查找成功，返回中间元素的下标；如果中间元素小于待查的值，则对右半部分数组继续进行查找；如果中间元素大于待查的值，则对左半部分数组继续进行查找。若数组中没有元素可查找，则查找失败。由于每一次比较都将查找范围缩小一半，因此这种查找称为折半查找。因每一次的查找都与原问题类似，只是规模减小一半，这一过程符合递归的思想。

分析折半查找过程，如图4.3.6所示。

在$a[0]$到$a[10]$这11个元素中，先找到中间元素$a[5]$，$72<112$，确定向较大元素方向查找；

在$a[6]$到$a[10]$这5个元素中，再找到中间元素$a[8]$，$100<112$，继续向较大元素方向查找；

在$a[9]$到$a[10]$这2个元素中，再找到中间元素$a[9]$，112=112，查找成功，返回下标9。

$a[0]$	$a[1]$	$a[2]$	$a[3]$	$a[4]$	$a[5]$	$a[6]$	$a[7]$	$a[8]$	$a[9]$	$a[10]$
23	35	46	57	68	72	83	91	100	112	125
23	35	46	57	68	72	83	91	100	112	125
23	35	46	57	68	72	83	91	100	112	125

图4.3.6　折半查找过程

实践活动　　**为“单词管理程序”增加查找功能**

小明经过100天的学习掌握了5 000多个英文单词。为了巩固记忆、掌握新单词，他打算编写一个管理单词的小程序，小程序中包括添加新单词和查找单词等多个功能。

请使用Python语言编写一个使用折半查找的程序，帮助他实现查找单词的功能。

分形

分形常用于形容部分与整体以某种形式相似的形状。比如雪花曲线是一个典型的分形，如图4.3.7所示。

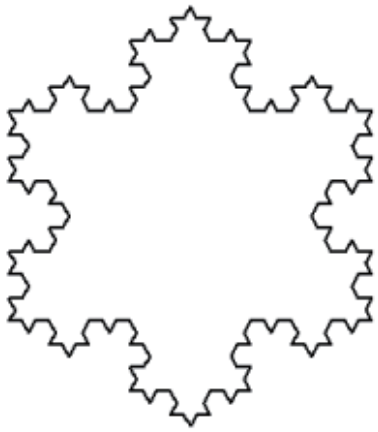


图4.3.7　雪花曲线

从一个正三角形开始，把每条边分成三等份，然后以各边的中间一段为底边，分别向外作正三角形，再把底边线段抹掉，这样就得到一个六角形，它共有12条边。再把每条边分为三等份，以各边的中间一段为底边，向外作正三角形后，抹掉底边线段。反复进行这一过程，就会得到一条形似雪花的闭合曲线，重复的次数越多，得到的曲线就越精细。

由于正三角形三条边长度相等，每个内角都是60°，所以只需研究一条边的情况即可。单边的雪花曲线如图4.3.8所示。仔细观察图4.3.8，不难发现：3阶的图是通过引用4次2阶的图完成绘制的；2阶的图通过引用4次1阶的图完成绘制；绘制0阶的一条直线则不再需要引用其他阶。因此，雪花曲线可以用递归来实现。

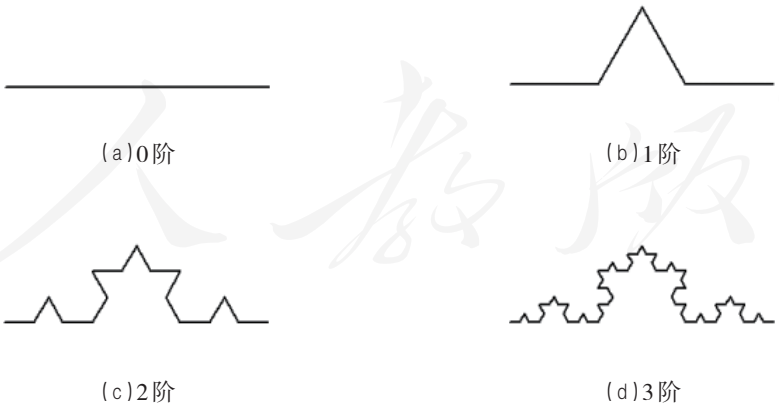


图4.3.8　单边的雪花曲线

在Python中，可引用turtle模块绘制图形。

对于0阶曲线，就是turtle前进一段长度turtle.forward(length)。

对于1阶曲线，把中间的三分之一的部分替换成没有底边的正三角形，且各部分的长

度相等，即

```
turtle.forward(length / 3)
turtle.left(60)
turtle.forward(length / 3)
turtle.left(-120)
turtle.forward(length / 3)
turtle.left(60)
turtle.forward(length / 3)
```

上述语句也可以表示为：

```
for angle in [60, -120, 60, 0]:
    turtle.forward(length / 3)
    turtle.left(angle)
```

根据前面得出的可以用递归方法绘制雪花曲线的结论，很容易得到一条边的雪花曲线函数。这种曲线有着极不寻常的特性，它的周长为无限大。

实践活动

编程绘制4阶雪花曲线

按照递归的思想，编写程序，并绘制出完整的4阶雪花曲线，如图4.3.9所示。

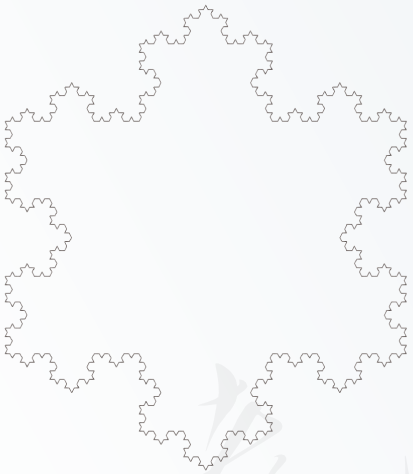


图4.3.9　4阶雪花曲线

分形还有很多种，如树、螺旋形等。有的分形用递归来实现，也有的分形由于有迭代关系式，也可以用迭代来实现。总之，世界表面上纷繁复杂，但都有一定规律可循，我们要善于发现问题，大胆地提出问题，科学严谨地分析问题，利用已有知识解决问题，从而促进新知识的产生。

项目实施

完成项目，汇报交流

一、项目活动

完成确定的主题项目，发布程序，撰写研究报告，小组交流展示。

在学习递归法后，综合运用递归、迭代、解析和枚举等各种方法，查阅相关的Python资料，完成各小组的程序。

以五子棋对弈项目为例，制作完成时，发布应用程序，界面如图4.3.10所示。

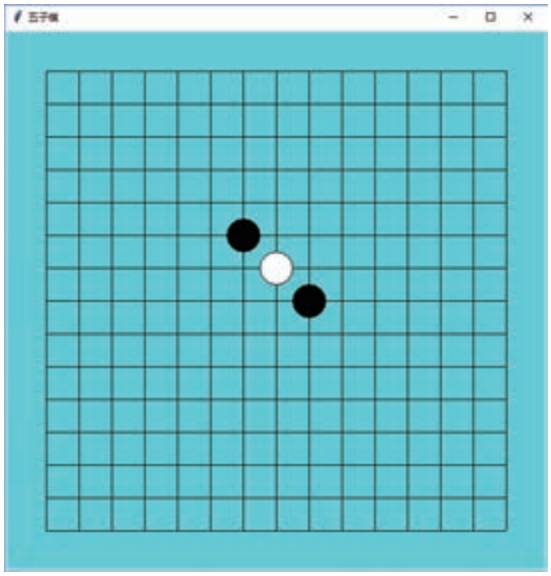


图4.3.10　五子棋对弈程序界面

改变编写的程序中的参数，反复调试，测试程序的健壮性。有条件应当使用多种操作系统测试制作完成的应用程序。

各小组在完成作品的基础上，形成主题研究报告，涵盖设计意图与特色、设计过程、主要算法和程序、体会与经验等各个方面。

召开班级展示会，各小组展示、交流各自编写的程序，讲评项目研究报告，取长补短。

二、项目检查

1. 完成主题项目后，正式发布程序，程序可正确运行，能实现项目功能。

2. 整理主题学习项目研究报告，召开各小组交流展示会。

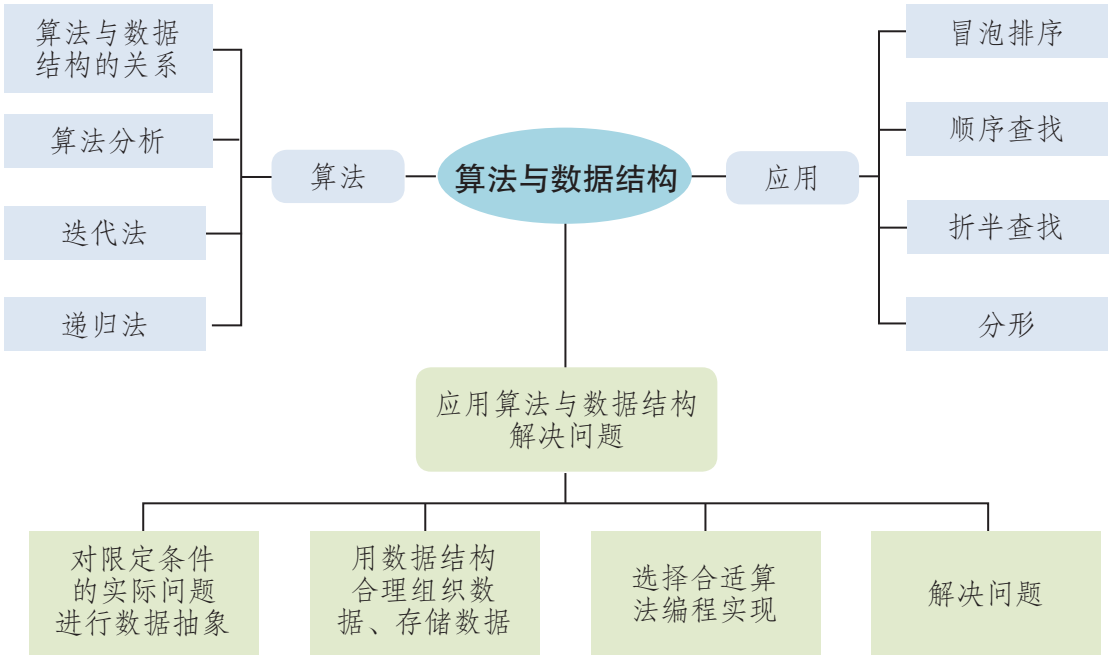
练习提升

1. 对于有序数组而言，查找算法一定要折半吗？能不能折三分之一、四分之一或者折更多呢？此外，还有黄金分割查找等方法。通过互联网查阅资料，并编写程序实现黄金分割查找。

2. 如果数据是用二叉树组织的，如何进行查找呢？尝试解决树形结构中数据的查找问题。

总结 评价

1. 下图展示了本章的核心概念与关键能力，请同学们对照图中的内容进行总结。



2. 根据自己的掌握情况填写下表。

学习内容	掌握程度
将需要解决的问题步骤化	□不了解 □了解 □理解
为需要解决的问题选取适当的数据结构	□不了解 □了解 □理解
设计迭代算法解决问题	□不了解 □了解 □理解
设计递归算法解决问题	□不了解 □了解 □理解

项目评价

在完成项目活动后，请各组对项目完成情况进行评价。评价实施围绕项目主题、实施过程、分工合作、项目成果和展示交流五方面进行。根据项目评价中的评分参考，结合项目实际完成情况，确定各项评分结果，给出评分理由。同时，对项目活动进行全面梳理，指出需要进一步改进的地方。将评价内容如实填写到项目评价表中。

评价项	评分参考	评分（1～5分）	评分理由	待改进之处
项目主题	项目主题能反映出学科核心素养的要求（信息意识、计算思维、数字化学习与创新、信息社会责任）；主题任务与学习目标保持一致			
实施过程	项目研究计划详细，准备充分；实施过程完整，过程记录翔实，资料丰富；研究数据来源渠道多，出处明确，收集方式多样，质量高；研究方法得当，技术手段适宜			
分工合作	小组成员分工明确，态度积极，参与度高；善于提出问题，分析问题、解决问题能力强；踊跃分享观点，交流充分；能在完成自己任务的前提下，乐意帮助他组完成任务			
项目成果	项目活动成果丰富，内容具体，符合项目目标要求；研究结论清晰准确，有价值，有创新，具有指导及建设意义；项目报告或作品内容完整，论述充分，表述清楚，整齐美观			
展示交流	项目展示形式新颖，综合运用多种技术呈现成果，表现力强；语言表达清晰准确，逻辑性好			
项目总分				

后　记

本册教科书是中国地图出版社与人民教育出版社依据教育部《普通高中信息技术课程标准（2017 年版）》，由双方共同组织团队联合编写的，经国家教材委员会 2019 年审查通过。

本册教科书的编写，集中反映了我国十余年来普通高中课程改革的成果，吸取了 2004 年版《普通高中课程标准实验教科书　信息技术》的编写经验，凝聚了参与课改实验的教育专家、学科专家、教材编写专家、教研人员和一线教师，以及教材装帧设计专家的集体智慧。本册教科书的编写人员还有刘利华、吕宝荣、李春、刘学瑞、罗维松、杜永刚，审校人员有刘利华、吴劲松、刘兆彬、李斌、张宇鹏。为本册教科书摄影或提供照片的有新华社记者等。

我们感谢所有对教科书的编写、出版、试教等提供过帮助与支持的同仁和社会各界朋友。同时，我们还要感谢 2004 年版《普通高中课程标准实验教科书 信息技术》的编写人员。

本册教科书出版之前，我们通过多种渠道与教科书选用作品（包括照片、画作）的作者进行了联系，得到了他们的大力支持。对此，我们表示衷心的感谢！恳请未联系到的作者与我们联系，以便及时支付稿酬。

我们真诚地希望广大教师、学生及家长在使用本册教科书的过程中提出宝贵意见。我们将集思广益，不断修订，使教科书趋于完善。

联系方式

电　　话：010–83543863　　010–58758866

电子邮箱：sinomaps@yeah.net　　jcfk@pep.com.cn

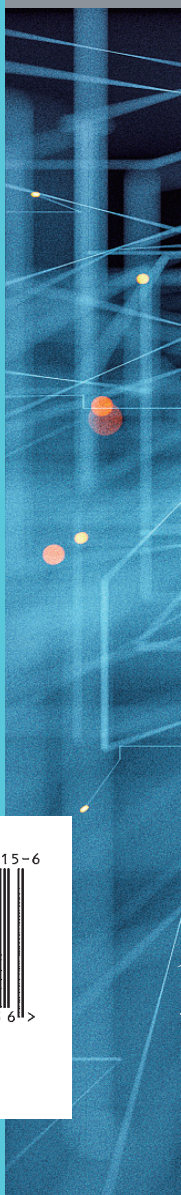
中国地图出版社教材出版分社

人民教育出版社课程教材研究所信息技术课程教材研究开发中心

2019 年 4 月

PUTONG GAOZHONG JIAOKESHU

XINXI JISHU



ISBN 978-7-107-34615-6

绿色印刷产品