普通高等教育“十三五”规划教材
新工科建设之路·计算机类规划教材

基于搜索策略的问题求解

——数据结构与 C 语言程序设计综合实践

李国和　编著

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

内 容 简 介

本书面向新工科教育，以计算思维为指导、以程序设计为主线、以智能搜索应用为背景、以提高程序实践技能为目的组织编写，并采用标准 C 语言编写程序。同时以人工智能状态空间和产生式系统问题求解为背景，从盲目与启发式、局部与全局、递推与递归、可回溯与不可回溯、最优与随机、个体与群体等多个维度对比介绍搜索算法。以问题为出发点，问题驱动贯穿全书，各章节依次从浅到深、从易到难递进介绍，并通过模块化程序实例，增强内容的可读性和可理解性。

本书可以作为本科生 C 语言课程设计用书，或人工智能导论参考书。通过对本书的学习，使读者不仅可以提高 C 语言编程和数据结构应用能力，而且可以掌握人工智能基于搜索策略的若干问题的基本求解方法。

图书在版编目（CIP）数据

基于搜索策略的问题求解：数据结构与 C 语言程序设计综合实践 / 李国和编著. — 北京：电子工业出版社，2019.10
ISBN 978-7-121-36966-7

I. ①基… II. ①李… III. ①C 语言－数据结构－高等学校－教材 IV. ①TP311.12 ②TP312.8

中国版本图书馆 CIP 数据核字(2019)第 128623 号

责任编辑：孟 宇
印 刷：
装 订：
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编：100036
开 本：787×1092 1/16 印张：11 字数：282 千字
版 次：2019 年 10 月第 1 版
印 次：2019 年 10 月第 1 次印刷
定 价：39.00 元

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

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

本书咨询联系方式：mengyu@phei.com.cn。

前　言

素质教育是当今大学教育的重要任务之一，而计算思维已成为现代素质教育的基本组成部分，其教学内容包括从单纯的计算机基本概念到思想与方法的提升，即以解决问题（计算机应用）为目标，涵盖问题描述、形式化表示、数据结构和算法设计、编码实现及验证全过程。目前很多高校开设了与计算思维教育密切相关的程序设计、数据结构和程序设计综合实践课程，但主要存在以下问题：

（1）停留于基础内容，应用性与实践性不强。程序设计教学往往注重语言的词法、语法、语义，而经常忽视问题求解的内容。数据结构的教学重点在抽象化的数据及其联系和数据操作（算法）上，缺少具体化的应用。程序设计综合实践课程应用性不强，大多以复杂数据结构的练习为主。

（2）实践主题复杂，系统性与前沿性不足。在实践类课程中，实践内容还有相当一部分是数据结构的内容，而不是开展数据结构的应用，此外还有一些模拟数据管理的简单应用，缺少系统化和紧扣时代特征的应用。

（3）基础教材丰富，实践教材较少。程序设计和数据结构教材的种类很多，这些教材的主题明确并且强调应用（包括综合程序设计、数据结构、问题求解），但是与社会发展需求密切相关的教材很少，尤其与人工智能主题相关的实践类教材更少。

针对上述存在的问题，本书以面向新工科教育、提升计算思维认识、落实应用技能培养为宗旨，以确保基础、注重联系、增强应用、提高技能为指导。本书具有以下特点：

（1）主题先进：以智能搜索技术及其实现为应用背景，内容紧扣智能时代的主流，人工智能研究领域与核心技术如表 1 所示。

表 1　人工智能研究领域与核心技术

序　号	研 究 领 域	核 心 技 术
1	机器定理证明	逻辑推理
2	机器翻译	自然语言理解
3	专家系统	知识表示、推理
4	博弈	树/图的搜索
5	模式识别	特征提取、优化
6	机器学习	模型结构、学习方法
7	机器人和智能控制	优化策略

（2）课程目的明确：面向最短路径、组合优化与函数优化、专家知识推理等不同问题的求解，采用不同搜索技术，培养基于搜索策略的计算机应用能力。

（3）内容主线清晰：从树搜索、图搜索、启发式搜索到局部搜索、全局搜索，再到规则（与

/或）树搜索，内容逐渐深入。每种搜索算法均可以解决特定的问题，并且可以引出新的问题，进而改进算法，这样可以提高创新能力。

（4）学习内容丰富：除计算机语言知识和编程技能外，本书综合应用了数据结构知识和智能搜索算法问题求解。所有程序案例均采用自顶向下、逐步细化的模块化设计，程序编写规范、注释详细，提高案例程序的可读性和可理解性，潜移默化地培养读者软件工程思想。

本书主要包括以下内容：

第 1 章，C 语言及其程序设计基础：对 C 语言及其程序设计的内涵和要点进行回顾与总结；

第 2 章，树搜索：可回溯盲目搜索，路径求解；

第 3 章，图搜索：可回溯图搜索、启发式搜索，路径求解；

第 4 章，启发式搜索：局部性、随机性、可回溯搜索，路径求解；

第 5 章，局部最优搜索：启发性、随机性、局部性、不可回溯搜索，目标求解；

第 6 章，全局最优搜索：并行性、启发性、随机性、全局性、不可回溯搜索，目标求解；

第 7 章，规则树搜索：规则树正向搜索和逆向搜索，目标求解、过程（路径）求解。

本书所有程序均采用标准 C 语言实现，并且均在 Visual C++ 6.0 环境中调试通过。若使用其他 C 语言编译系统，请参考相关资料，略加修改程序即可。

本书作者多年从事本科生和研究生的 C 语言、数据结构和人工智能教学，在深入了解学生对 C 语言和人工智能知识的渴求以及希望达到的应用水平后才逐步确定本书的内容。在本书编写过程中，得到中国石油大学（北京）教务处、信息科学与工程学院，中国石油大学（北京）克拉玛依校区教务处与国际交流部、文理学院和石油学院的大力支持，以及校级 C 语言优秀教学团队的大力帮助，在此表示衷心的感谢。同时，也感谢教育部—中锐网络产学合作协同育人项目“面向新工科教育的计算思维培养教学改革与实践——以 C 语言程序设计混合教学模式为例（201801181004）” 和新疆维吾尔自治区教改项目“面向新工科教育的计算机基础教学研究与实践（2017JG094）”的支持。另外，赵建辉、吴卫江、张岩、段毛毛、董丹丹等老师对本书提出很多建议，在此向他们表示衷心的感谢。由于作者水平有限，书中不完善之处甚至错误在所难免，敬请读者批评、指正。

李国和

2019 年 7 月

于中国石油大学（北京）

目　录

第 1 章

C 语言及其程序设计基础

C 语言是当今经典、流行的计算机高级语言之一。本章主要回顾、总结 C 语言及其程序设计的内容要点，其核心思想、编程要点贯穿在本书中。通过后续章节学习，可进一步提高编程技能。

1.1 C 语言与程序设计

计算机语言是人与计算机进行交互的一种协议，涵盖了词法、语法、语义及句法。通过计算机语言表达人对客观问题的求解过程（程序设计包括数据结构、算法），可以让计算机理解人的意图，进而实现问题的求解。从计算机语言的发展来看，计算机语言分为机器语言、汇编语言、高级语言和更高级语言，也就是在问题求解过程中，从关注求解问题和计算机资源分配管理的面向机器到只关注求解问题而无须参与计算机资源分配管理的面向问题的发展；从二进制代码的可移植性差、可读性差到类自然语言的可移植性好、可读性好的发展；从目前关注“做什么”和“怎么做”到未来只关注“做什么”的发展。

从计算机语言发展历程来看，C 语言不仅具有高级语言的编程风格，而且具备低级语言的功能，这个特点使 C 语言成为“中间语言”。根据高级语言的编程特点，计算机高级语言分为过程型、函数型、逻辑型和面向对象型 4 种。过程型语言的核心是变量，该语言具有顺序、分支和循环三种控制程序过程的结构，并且它适用于数值处理；函数型语言的核心是函数定义及其调用，该语言具有弱类型变量和顺序、分支及无条件转向控制程序过程的结构，但它强调递归结构控制，适用于人工智能的符号处理；逻辑型语言的核心是谓词，该语言主要描述事物及事物的联系，具有弱类型变量，并且只有递归、自动搜索匹配和 CUT 等控制结构，它适用于人工智能的符号处理；面向对象型语言的核心是对象，该语言的信息（数据）和处理（方法）是统一的，同时它具有类、继承、消息、多态等概念，可以实现代码共享，它适用于单机或网络软件系统的集成开发。

C 语言既是过程型计算机语言又是函数型计算机语言。程序设计的核心是算法和数据结构的设计，算法用于描述问题求解过程，其具有输入、输出、有效性、确定性和有穷性的特点，算法可用程序流程图、N-S 图等表示。采用符合计算机语言的词法、语法及句法的规范进行算法描述就是计算机语言的程序设计。从软件系统实现来看，针对功能实现采用结构化程序设计（包括顺序、分支和循环结构），针对系统集成采用模块化程序设计（模块定义及其调用）。

C 语言既是结构化程序设计语言（体现在顺序、分支和循环结构语句上）又是模块化程序设计语言（体现在函数定义和函数调用上）。程序设计形成的文档称为程序。由高级计算机语

言编码的程序称为源程序，需要对源程序进行编译形成目标程序，再进行连接形成可执行程序。上机实践过程包括编辑（形成源程序.c）、编译（形成目标程序.obj）、连接（形成可执行程序.exe）、运行，对应系统软件依次为编辑器、编译器、连接器、操作系统。为了便于软件开发，常把编辑器、编译器和连接器集成在一个开发环境中，如 Visual C++。软件开发过程包括问题分析（完成功能分解、形式化描述）、研究算法（完成问题求解过程）、程序实现（完成计算机语言编码）及测试运行（完成软件测试、交付使用）。

1.2 C 语言基础

C 语言程序结构一般包括#include 头文件包含、#define 符号常量定义和#if 条件编译的编译预处理命令。程序运行的唯一入口函数 main（主函数）是被调用户自定义函数，并且涉及用户自定义函数中的变量定义、使用及函数的输入/输出、返回等重要概念。

C 语言规定了一组基础字符，即 C 语言字符集（类似于英文字母集），包括小写英文字母、大写英文字母、数字和空格（包括空格、制表符、换行符）等一般字符，也包括运算符、分隔符（空格、分号等）、转义字符及注释符等特殊字符。在一般字符集的基础上，定义了以字母或下画线“_”开头的字母、数字、下画线构成的标识符（类似于英文单词），用于表示或命名变量、符号常量、函数名等。标识符中的英文字母的大小写是相互区分的。C 语言还有一些预先定义的标识符，包括基本数据类符、预处理命令名、系统函数名及控制语句符等，这些标识符是固定的。这些特殊标识符称为关键字或称为保留字，即其有明确的内涵和作用，不可作为他用。

计算机内存最小的数据单元为字节（8 位二进制位）。以字节为基本单位构成连续、顺序的内存，用于存储程序和数据。根据作用的不同，进一步可把内存划分为系统区、应用程序区和数据区。计算机在做任何处理时，其程序、数据都必须在内存的相应区域中。数据是过程型语言程序处理的对象，数据可以包括常量、变量和表达式，同时可以参加各种运算，因此数据可以称为数据对象，也可以称为运算数。常量分为常数、符号常量及常变量，常量在程序运行中，不可被修改，常量（常变量除外）在程序区中是程序的构成部分。变量对应数据单元且在内存数据区中，该区域具有可读、可写的特点，因此程序中的变量就成为了可变的量。变量在程序中用变量名表示。变量值的改变是通过赋值运算、复合赋值运算符或自增和自减运算完成的。

C 语言任何数据（包括常量、变量和表达式）均属于某种数据类型（字符串除外），数据类型具有决定数据精度、数据存储单元大小和数据有效取值范围的属性。同一种数据类型的数据具有数据类型所规定的数据属性。基本数据类型包括字符型、整型和浮点型，分别用 char，int、float 类型关键字进行表示。变量必须先定义后使用，在定义中对变量进行初始化，变量值具有“挤得掉、取不尽”的特点。

运算体现对数据的加工和处理功能，由运算符表示。运算符涉及运算符功能、运算数的数目、运算数数据类型、运算优先级和结合性 5 个方面。其中结合性是指运算从左到右（左结合性）或从右到左（右结合性）依次运算。基本运算符包括算术运算符、赋值运算符、复合赋值运算符、逗号运算符、圆括号运算符、取地址运算符、数据单元大小运算符（sizeof）。为了叙述方便，对有些符号进行约定：«XXX»表示 XXX 必不可少，即必选；XXX⊥YYY 表示 XXX 和 YYY 二者选一；⌊XXX⌋表示 XXX 可选可不选，即 XXX 可省略；⌊XXX⌋^{n} 表示 XXX 不出现或多次出现。

表达式是由运算数和运算符构成且符合 C 语言规范的式子，表示对数据的加工和处理，其求值按运算符的优先级和结合性进行。通过赋值运算符和复合赋值运算符及自增和自减算术运算符的运算可改变变量的值。表达式中不同数据类型的数据进行混合运算时，编译系统利用临时单元对数据进行数据类型的自动转换，转换后再进行运算，其转换原则是低精度转换为高精度、数据单元小的转换为数据单元大的，该原则确保数据运算不失真。但有些运算符具有特殊要求，如求余（%）运算符要求两个运算数为整数。若自动转换的结果不能符合运算要求，即需要进行数据类型强制转换。

关系运算也称为比较运算，主要包括等于、不等于、小于、小于等于、大于、大于等于 6 种，对应的运算符分别为==、!=、<、<=、>、>=。这些运算符都是二元（或称二目）运算符，而且都具有左结合性，即从左到右依次运算。由于 C 语言没有逻辑数据类型（即没有逻辑型的数据），因此 C 语言规定关系运算的结果用 1 或 0 表示，且 1 表示逻辑“真”，0 表示逻辑“假”，故关系表达式的值为 1 或 0。关系运算符的优先级都低于算术运算符的优先级，因此可以简单记为只有进行算术计算后得到的数据值才能进行比较，而比较的结果为真（1）或假（0）。C 语言尽管没有逻辑数据，但提供了逻辑非运算、与运算、或运算，对应的逻辑运算符分别为!、&&、||。C 语言规定参与逻辑运算的运算数分为非 0 和 0，且非 0 相当于逻辑“真”，而 0 相当于逻辑“假”。逻辑表达式的值也为 1 和 0，即相当于逻辑“真”和“假”。除了非运算符，与运算符和或运算符的优先级都低于关系运算符，简单记忆为得到关系运算的结果真（1）或假（0）后，才能进行逻辑运算（非运算除外），结果为真（1）或假（0）。注意：！的优先级高于算术运算符。！为右结合性逻辑运算符，&&和||为左结合性逻辑运算符。

表达式中的运算符若要进行运算则必须通过语句进行。语句是程序可执行的基本单位，C 语言语句包括完成数据处理的表达式语句、启动程序模块的函数调用语句、控制程序语句流程走向的控制语句、为完成一定功能多个语句构成的复合语句及没有实际意义作用的空语句。任何语句都以分号结束，即分号是语句的分隔符。编译系统对 C 语言的编译是“逐行从上到下，同行从左到右”进行的。因此，在编写程序时，要充分利用 C 语言的有效分隔符、缩列（实际上是空格分隔符）、英文字母大小写的差异、“见名知意”命名方式及注释，增强程序的可读性和可理解性。

1.3　结构化程序设计

程序结构是指语句的执行流程，也就是一条语句执行完毕后决定如何执行下一条语句，可分为顺序结构、分支结构（选择结构）和循环结构。顺序结构是最简单的结构，一条语句执行完毕后不做任何判断就“自上而下、自左到右”依次执行，而分支结构和循环结构则需要进行条件判断后再决定如何执行下一条语句。三种结构的组合可以求解任何复杂的问题。

C 语言输入/输出功能的实现是通过调用标准函数库中输入/输出函数完成的。以键盘和屏幕为标准输入/输出设备，数据的输入/输出标准函数可分为格式化输入/输出函数和字符输入/输出函数以及字符串输入/输出函数。格式化输入/输出函数为 scanf 输入函数和 printf 输出函数。字符输入/输出函数为 getchar 输入函数和 putchar 输出函数。每个函数都有一个返回值，除 getchar 函数的返回值为字符 ASCII 码外，其他返回值表示函数调用的正确性。由于在程序中都是简单使用输入/输出函数，调用基本不会出错，因此只关注函数功能，不关注函数返回值。printf 输出函数的调用形式为 printf(«格式说明串»⌊,«运算数列表»⌋)，其中⌊ ⌋表示可选项，即

若没有输出数据，则可以省略；“运算数列表”可以是若干常量、变量、表达式及有返回值的函数；“格式说明串”包括原样输出的普通字符和格式控制串，即“%⌊±⌋0⌋m⌋⌊.n⌋⌊l⌋«格式符»”，其中“格式符”与输出数据的数据类型对应，“±”表示在输出数据所占列数较大时数据靠右对齐还是靠左对齐，“m”表示输出数据所占的列数（当列数小时，数据原样输出，不失真），“n”表示数据的小数位（对浮点型数据而言）或从字符串头部所取的字符数（对字符串而言），“l”是指输出数据为 double 型或 long int 型数据。另外，要求输出数据与“格式控制串”一一对应，包括数据类型和数据个数。scanf 输入函数的调用形式为 scanf(«格式说明串», «地址列表»)。“地址列表”为数据单元的地址或称为变量的指针。由于 C 语言无须参与内存管理，因此只关注变量指针的存在和使用，不关心具体值。变量指针由&取地址运算符表示，其中“格式说明串”包括原样输入的普通字符和格式控制串为“«%»⌊*⌋⌊m⌋⌊l⊥h⌋«格式符»”，其中“格式符”与输入数据的数据类型对应，*表示输入时跳过数据，或指忽略相应位置上的数据，“m”表示读取 m 列的数据，“l”表示读取的数据为 double 型或 long int 型数据，“h”表示读取的数据为 short int 型数据。字符输出函数的调用形式为 putchar(«字符数据»)，其中“字符数据”可以是字符常量、字符变量、整常数或整型变量。除数据单元大小不同外，在 ASCII 码范围内整型数据与字符型数据是等效的。字符输入函数的调用形式为 getchar()，返回一个字符的 ASCII 码。若不保留 getchar 函数值，则该函数在程序中起到暂停程序运行的作用，直到敲击回车键继续执行。

选择程序结构（分支程序结构）是结构化程序设计中三种结构之一。选择程序结构表示根据判断条件选择相应的处理方法。另外，在程序中，存在分支程序结构，判断条件作为分支程序结构的重要组成部分，它主要采用关系运算（比较运算）和逻辑运算来构造表达式。

条件运算符（？：）是 C 语言中唯一的三元运算符，其构成的条件表达式形式为«选择条件»？«表达式 1»：«表达式 2»，其中当“选择条件”为非 0（逻辑“真”）时，“表达式 1”为条件表达式的结果，反之，当“选择条件”为 0（逻辑“假”）时，“表达式 2”为条件表达式的结果，即通过“选择条件”来选择“表达式 1”或者“表达式 2”为条件表达式的结果，而且条件表达式的数据类型为“表达式 1”和“表达式 2”中精度高的、数据单元大的类型。“选择条件”“表达式 1”“表达式 2”可以为常量、变量或表达式（包括条件表达式），而“选择条件”常为关系表达式或逻辑表达式。当一个条件表达式中嵌套着另外一个条件表达式时，嵌套的条件表达式为右结合性运算符，这样有助于理解嵌套条件表达式的含义。

C 语言是由 if 语句、switch 语句和 break 语句联合实现选择程序结构的，其中 if 语句多用于实现二分支的选择结构，而 switch 语句和 break 语句联合用于实现多分支的选择结构。if 语句的基本形式为 if（«选择条件»）«语句 1» ⌊else «语句 2»]。“选择条件”为判断条件，可以是常量、变量和表达式（尤其是关系表达式或逻辑表达式），若其值为非 0（逻辑“真”），则执行“语句 1”，否则，其值为 0（逻辑“假”）时，执行“语句 2”。“语句 1”和“语句 2”均可以是 if 语句，从而构成多选择的 if 语句。if 语句的 else 子句是可选子句。C 语言程序书写具有很强的灵活性，故可以使用多空格对齐语句，但在多选择的 if 语句中，else 子句只与最近的 if 子句配对。switch 语句的基本形式为

```
switch(«运算数»)
{
  ⌊case «变量: » ⌊«语句»⌋n⌋   ⌊ break;⌋
  ⌊default   :   ⌊«语句»⌋n⌋
}
```

实际上，switch 语句是条件转向语句，而“常量”为跳转的目的“标号”，即当“运算数”等于“常量”时，跳过“语句列表”，故单独用 switch 语句没办法实现多分支选择功能。为了实现多分支选择功能，需要结合无条件跳转 break 语句，故 break 语句只能结合 switch 语句使用，不能单独使用 break 语句。通过 switch 语句的条件跳转和 break 语句的无条件跳转，可以实现多分支选择功能。在 switch 语句中，case 后的语句还可以是 switch 语句，进而构成 switch 语句的嵌套，并且当执行 break 语句时，只能跳转出当前 switch 语句。

循环程序结构是结构化程序的三种结构之一，可以解决具有相同变换规律的现实问题。在实现过程中，循环程序结构体现在反复执行一段相同功能的程序段，即循环程序。为了确保循环程序在有限的步骤内完成问题求解，避免陷入死循环，循环程序通常包括 4 个方面：有关问题初始化、循环条件、循环求解目标和循环控制变量变化。有关问题初始化包括循环控制变量、累加或累乘和标记符号的初始化等；循环条件决定结束或继续循环；循环求解目标需要反复执行，体现有规律的问题求解，此外还包括循环控制变量的变化和标记符号的变化等；循环控制变量的变化很重要，随着循环的进行，循环控制变量与循环条件联合使循环控制变量的变化一定朝着循环条件不能满足的方向逐渐逼近，并且使循环能够在有限的步骤内终止，确保问题得解。循环程序结构中循环体还可嵌入顺序程序结构、分支程序结构和循环程序结构。在循环程序结构中，嵌入循环程序结构构成了嵌套循环。若外层循环进行一次循环，则内层循环进行一个完整循环，也就是外层循环变化慢，内层循环变化快。

实现循环的结构包括 if 语句和 goto 语句、for 语句、while 语句和 do-while 语句。for 语句、while 语句和 do-while 语句统称为循环语句，这三个循环语句都只能内嵌一种语句（称为内嵌语句）作为循环体的主要部分，并且内嵌语句可以是任何语句。若内嵌语句为复合语句，则循环执行了一个程序段；若内嵌语句包含循环语句，则构成嵌套循环。当三种循环语句的循环条件都为“真”（非 0）时，执行内嵌语句。各种循环语句可以相互嵌套，也可以并列（顺序结构），但不能相互交叉。在实际应用中，for 语句使用最多，while 语句其次，do-while 语句使用最少。

for 语句的形式为 for(⌊«表达式 1»⌋;⌊«循环条件»⌋;⌊«表达式 2»⌋)«语句»，其中“循环条件”也是表达式，三个表达式依次表示有关问题的初始化、满足循环的条件和循环控制变量的变化，这种形式结构清晰，可读性好。三个表达式可以省略，但是循环结构中所要求的有关问题初始化、循环条件和循环控制变量的变化不能省略，只能在程序的其他位置出现。另外，for 语句一般用在循环次数已知的情况下。while 语句的形式为 while(«循环条件») «语句»。for 语句和 while 语句的特点是先判断循环条件、再执行内嵌语句，故内嵌语句有可能一次都不执行。do-while 语句的形式为 do «语句» while(«循环条件»)。do-while 语句的特点是先执行内嵌语句、再判断循环条件，故内嵌语句至少执行一次。

从实现功能角度看，无论三种循环语句，还是由 if 语句和 goto 语句构成的循环程序，它们都是等价的，故它们可以相互代替，但从编程风格来看，针对不同的问题，编程的简易性和可读性有所不同，需要注意以下问题：

（1）一般不提倡使用由 if 语句和 goto 语句构成的循环。由于 goto 语句可跳转到函数内任意执行语句，即转向具有随意性，因此导致程序可读性、可维护性差，另外 goto 语句不是结构化程序设计的语句。

（2）for 语句一般用于有循环控制变量且事先知道循环次数的程序中。

（3）while 语句一般用于不知道循环次数，或者没有循环控制变量的程序中。

（4）do-while 语句一般用于需要至少执行一次循环的程序中。

（5）while 语句和 do-while 语句的内嵌语句中应包含使循环趋于结束的语句。若有循环控制变量，则循环控制变量初始化的操作应在 while 语句和 do-while 语句之前完成。

goto 语句、break 语句和 continue 语句都是无条件转向语句（无条件跳转语句）。通过无条件转向语句可以改变程序的执行顺序走向。在使用这些无条件转向语句时，经常需要结合 if 语句来决定跳转的去向。goto 语句通过语句标号决定转向位置，该位置只要在同一个函数（模块）内都是合法的，因此转向的范围很大，随意性强，进而破坏程序结构化，同时导致程序（尤其是循环程序）的可读性、维护性差。break 语句和 continue 语句的转向位置是默认的、固定的，无须语句标号，其默认位置在内嵌语句之后。goto 语句可以结束 if 语句与 goto 语句构成的循环、循环语句构成的循环及任何嵌套循环。break 语句和 continue 语句只能用在循环语句的内嵌语句中，不能用在 if 语句和 goto 语句构成的循环中。break 语句跳转出当前循环语句，终止循环，而 continue 语句结束循环语句的当前次循环，提前进入下一次循环，并没有终止循环。

1.4 构造类型数据（一）

数据类型是数据共有的性质，决定数据单元大小、数据精度及操作运算等。除整型、浮点型与字符型等基本数据类型外，还可以根据需要构建新的数据类型，即构造数据类型，或称自定义数据类型。所谓构造数据类型就是根据数据类型构造原则和已有的数据类型自定义的数据类型。已有的数据类型可以是基本类型或构造数据类型，故构造数据类型也是递归定义的。下面内容涉及的构造数据类型包括指针类型和数组类型。

指针与计算机内存紧密相关，计算机内存单元按线性连续编址。程序中定义变量 a，内存中就有数据单元，也就有相应的地址，该地址在程序中称为变量的指针。在指针使用上，只关心指针的存在，不关心指针值，因此，C 语言提供取地址运算符“&”用于获取变量指针&a。变量的指针&a 是常量，称该指针&a 指向变量 a。指针可用指针变量 p 进行保留和运算，该指针变量 p 称为指向变量的指针变量。变量 a 与指针变量 p 就建立指向关系，即指针变量指向变量。对变量取值或赋值称为对变量的直接访问，如“a=10, p=&a;”都是直接访问，即直接访问 a 和 p，而通过指针（包括指针变量）对变量取值或赋值称为间接访问，而间接访问必须通过指针的指向运算符“*”才能进行，如*(&a)=10 和*p=10 都是间接访问，即通过指针&a 和指针变量 p（对 p 是直接访问）间接访问 a。指针变量 p 也有指针&p，同样可以定义保留该指针的指针变量 pp，称为指向指针变量的指针变量，通过**pp 也可以间接访问 a。相对 a 而言，p 为 a 的一级指针，pp 为 a 的二级指针。指针变量可以递推定义多级，指针变量是变量，具有变量的访问等基本特性。指针可以进行算术运算，如+、–、++、––，其表示以数据单元的个数为单位进行指向的改变。取地址运算符“&”和指向运算符“*”为单目运算符，其优先级为 2 级，并且它们是左结合性的运算符。由于指针指向以数据单元为单位，因此定义指针变量时必须指明数据类型，以限定指针指向单元的大小。

数组是具有相同数据类型的变量集合，数组中的变量称为数组元素。在定义数组时，可以全部或部分对数组元素依次进行初始化。对数组的访问有直接访问和间接访问，也称下标法访问和指针法访问。对于一维数组 a，长度为 N，下标 i 取值为 0, 1, …, N–1，数组元素 a[i]表达了直接访问。一维数组元素的指针&a[i]，数组名 a 是数组的首地址也是指针。a 与&a[i]一样，都是指向数组元素的指针，因此对数组元素 a[i]的间接访问为*(a+i)或*(&a[i])。通过定义指

向变量的指针变量 p，p=a+i 或 p=&a[i]使得指针变量 p 指向一维数组第 i 个元素，也可称 p 指向一维数组 a。由于一维数组数元素对应数据单元在内存中是线性连续存储的，因此下标 i 或指针变量 p 的算术运算标识不同的数据单元，下标 i 或指针变量 p 的连续变化可以直接或间接访问数组元素。数组下标 i 和指针变量 p 可以理解为数组元素的索引，数组下标 i 为相对索引，指针变量 p 为绝对索引。通过数组名 a（绝对索引）下标 i 和指针变量 p 可以相互换算。

n 维数组 a[N1][N2]…[Nn]是一个一维数组 a[N1]（注：N 表示长度，数字表示维数），其每个元素 a[i1]是 n–1 维数组（注：i 表示下标，数字表示第几维），这是递归定义。只有数组元素 a[i1][i2]…[in]在内存中有相应的数据单元，对于一维数组元素 a[i1]只是一个逻辑单元，并且代表 n–1 维数组。通过下标法 a[i1][i2]…[in]可直接访问数组元素，也可以通过指针法间接访问数组元素。对于 n 维数组，a+i1、&a[i1]是指向 n–1 维数组的指针；*(a+i1)+i2 和&a[i1][i2]是指向 n–2 维数组的指针；依此类推，*(…*(*(a+i1)+i2)…)+in 和&a[i1][i2]…[in]是指向数组元素 a[i1][i2]…[in]的指针。可以定义相应的指针变量保留相应的指针，即(*p1)[N2]…[Nn]为指向 n–1 维数组的指针变量，即 p1=a+i1 或 p1=&a[i1]；(*p2) [N3]…[Nn] 为指向 n–2 维数组的指针变量，即 p2=*(a+i1)+i2 或 p1=&a[i1][i2]；依此类推，*pn 为指向 n 维数组元素的指针变量，即 pn=*(…*(*(a+i1)+i2)…)+in 或 pn=&a[i1][i2]…[in]。

对于二维数组 a[N1][N2]，第一维称为行，第二维称为列。数组名 a 和指针变量(*p1)[N2]是指向行的指针，因此 p1=a+i1 表示第 i1 行的指针，即指针 a 或 p1 的指向是以行为单位的（即数组元素个数为 N1 的一维数组）。a[i1]和 p2 是指向二维数组元素的指针（a[i1]相当于数组元素个数为 N2 的一维数组名），因此 p2=a[i1]+i2 或 p2=*p1+i2 或 p2=*(a+i1)+i2 或 p2=&a[i1][i2]表示第 i1 行、第 i2 列元素的指针，即指针 a[i1]或 p2 的指向是以二维数组元素为单位的，或者使用下标法 a[i1][i2]直接访问第 i1 行、第 i2 列元素，还可以使用指针法*p2 或*(*(a+i1)+i2)或*(a[i1]+i2)或*(&a[i1][i2]) 间接访问第 i1 行、第 i2 列元素。

指针是一种数据类型，根据指针类型也可以定义数组，称为指针数组，即数组元素是指针变量，如*ps[N]为长度为 N 的一维指针数组，即数组元素 ps[i]为指针变量，可保留其他变量的指针，数组名 ps 也是指针，对于其他变量而言，ps 就是二级指针。指针数组具有数组的一般特性，既可以方便进行批量处理，又可以保留指针，故指针数组常作为批量数据的索引。

C 语言有字符串常量而没有字符串数据类型，故没有字符串变量，但可用一维字符型数组保留字符串常量。由于字符串关心其有效长度，因此保留字符串的字符数组需要足够大，并以'\0'作为字符串结束标记。通过字符型数组的处理方法或针对字符串的处理函数可以实现对字符串的比较、复制、连接等处理。

数组必须先定义、后使用，也就是数组的数据类型、维数和每一维长度都必须在程序中指定，以便程序编译阶段为其分配数据单元，在一定程度上也限制数组的应用。动态内存分配和撤销函数可以在程序运行阶段分配和撤销连续数据单元，等同于在程序运行期间动态生成和管理一维数组。通过数据类型的强制转换、直接计算指针或下标方式，一维数组可以当多维数组使用。

void 空类型也是基本类型，可以作为函数返回值类型，表示该函数没有返回值。若 void 空类型作为变量类型，则表示变量具体类型待定（可以是 void 空类型除外的其他类型），在实际应用中，需要进行强制类型转换，这样变量才有意义，才可以使用。

1.5 构造类型数据（二）

除指针类型、数组类型外，构造数据类型还包括结构体类型、共用体类型和枚举类型。结构体类型及其变量、数组的形式可定义为«struct»⌊«结构体类型名»⌋{ «成员声明表列»}⌊«变量名列表»⊥«数组名列表»⌋;，其中，struct 是结构体类型的关键字，“结构体类型名”为自定义标识符，“成员声明表列”由若干个成员声明组成，即«成员声明 1»⌊, «成员声明 2»⌋[n];，而每个“成员声明”形式为«数据类型符» «成员名 1»⌊, «成员名 2»⌋[n];。结构体类型可以汇集各种类型的成员，这些成员可以是变量成员、数组成员、字符串成员和指针成员等。由结构体类型定义的变量包含成员变量、成员数组、成员字符串和成员指针等。结构体变量的数据单元是由各成员数据单元构成的，即是结构体变量的数据单元的大小是各成员数据单元大小之和，而且各成员的顺序与定义结构体类型成员顺序一致，另外在定义结构体变量时可以初始化。结构体变量初始化时，构造类型成员对应一对花括号。对于嵌套多层的结构体类型，定义变量并初始化可以省略内部的花括号，其初始化是花括号内的常量依次保留到对应的成员数据单元中。对于部分初始化，只能默认后面的常数项。对于默认的数据内容，系统自动初始化为 0（对于整型和浮点型成员）或'\0'（对字符型或字符串成员）或 NULL（空指针，对于指针类型成员）。对结构体类型变量的访问有两种方式，即直接访问和间接访问。直接访问通过成员运算符“.”实现，形式为«结构体变量»«.»«成员名»，而间接访问通过指向成员运算符“->”实现，形式为«结构体指针»«->»«成员名»。成员运算符或指向成员运算符都是左结合性、优先级为 1 级。结构体类型不仅可以定义结构体变量，而且可以定义数组和指针变量等。

共用体类型及其变量的定义和访问形式与结构体类型及其变量的定义和访问形式相同，但具有不同含义。共用体类型及其变量、数组的形式可定义为«union»⌊«共用体类型名»⌋{ «成员声明表列»}⌊«变量名列表»⊥«数组名列表»⌋;，其中，union 是共用体类型的关键字，“共用体类型名”是自定义标识符，“成员声明表列”由若干个成员声明组成，其形式为«成员声明 1»⌊, «成员声明 2»⌋[n];，而每个“成员声明”形式为«数据类型符» «成员名 1»⌊, «成员名 2»⌋[n];。共用体类型包括各种类型的变量成员、数组成员和指针成员等，共用体类型变量的数据单元大小是成员中数据单元最大的成员，各成员的指针值相同，共享共用体数据单元，因此对某个成员的赋值将全部或部分覆盖其他成员以前的值，使得原成员的值不可预见，而只知道当前成员值。通过成员运算符直接访问共用体的成员变量、成员数组和成员指针等，通过指向成员运算符间接访问成员变量、成员数组和成员指针等。

在定义枚举类型时，使用 enum 关键字标识枚举类型，并由用户给定的常量构成，其形式为«enum» «枚举类型名»{«枚举常量列表»};，另外枚举类型变量的取值只能是枚举常量。枚举常量由系统自动定义从 0 开始的序列号，但序列号不等于枚举常量，因为它们分别属于整型数据和枚举类型数据。整型数据与枚举类型数据可以通过强制类型转换实现两种类型变量之间的赋值。

typedef 语句可以对没有构造类型名的构造类型命名，也可以对已有数据类型名的数据类型重新命名。通过 typedef 语句命名的数据类型可以增强程序的可读性和程序移植的灵活性。

1.6 模块化程序设计

C 语言模块化程序设计体现在函数概念上。根据函数的来源，函数可以分为系统函数和自

定义函数；根据函数的参数，函数可以分为有参函数、无参函数及空函数。C 语言模块化程序设计表现在自定义有参函数和无参函数上。函数定义形式为

```
⌊«extern»⌋⊥⌊«static»⌋«数据类型符» «函数名»（⌊«形式参数声明列表»⌋）
{
    ⌊«函数声明»⌋n
    ⌊«数据类型定义»⌋n
    ⌊«数据定义»⌋n
    ⌊«语句»⌋n
    ⌊return ⌊（«运算数»）⌋⊥⌊«运算数»⌋⌋;
}
```

函数调用体现函数之间的联系，它是模块组织形式的集中体现，涉及函数声明形式和函数参数的结合形式。函数参数可以分为函数定义中的形参和函数调用中的实参，其中形参可以是变量（包括基本类型变量和构造类型变量）和数组（实际上是指针变量），形参的数据单元是动态数据单元，即在函数调用时，由系统动态分配数据单元，函数执行结束后，由系统回收数据单元。实参可以是常量、有值的变量和表达式。在函数调用过程中，实参和形参必须一一对应，即参数个数和对应位置的参数数据类型必须一致。实参和形参的结合过程是实参传递复制数值给形参的过程，由于形参和实参不共享相同的数据单元，因此形参的变化不影响实参。当指针（包括变量指针、数组和函数名）作为参数时，复制的数值是指针（地址），虽然属于直接访问，但强调直接访问获得指针后进行的间接访问。C 语言程序的运行方式是串行运行，即主调函数调用完被调函数后，转向被调函数运行，而主调函数处于等待状态，只有被调函数运行结束后并返回，主调函数才可以接着运行。任何函数调用都要返回，只有 void 函数类型不返回值，或 void *函数类型返回指针值但指向单元性质待定，而其他函数都可以返回一个数据（基本类型或构造类型数据）。函数调用可以嵌套调用，而且不受调用层次的限制。递归调用是一种特殊的嵌套调用，即直接或间接调用函数本身。在递归调用时，参数的变化尤为重要，为了避免无限递归耗尽内存资源，递归参数变化一定朝着递归出口进行，这样递归才可能结束。动态内存管理是在程序运行中进行维护和管理的，这非常有利于对规模数据单元不确定的数据进行管理和处理，但也导致程序效率变低。为了体现模块化的特性，函数可以在一个文件或多个文件中定义，采用内部函数和外部函数有效范围机制来规定函数的有效范围，确保函数的调用安全。在函数定义时，static 修饰符表示所定义的函数为内部函数，只能在本文件内被调用，而 extern 修饰符或省略修饰符表示所定义的函数为外部函数，可以跨文件被调用。在处理文件包含或联合编译时，需要确定函数之间调用的合法性，并且加强函数模块化程度和安全性。内部函数和外部函数是根据源程序文件是否可以各自编译后再进行连接来划分的，而文件包含不区分内部函数和外部函数。这是由于在编译预处理后形成统一的源程序文件，所有函数本质上都在一个文件中，或者说所有函数都只是内部函数。为了确保函数的正确调用，往往需要对函数进行声明，函数声明形式为：

```
⌊«extern»⌋ «数据类型符» «函数名»（⌊«形式参数声明列表»⊥«数据类型符列表»⌋）;
```

为了保证函数成功调用，被调函数必须存在。当被调函数不在主调函数调用的有效范围内，需要通过函数声明扩大函数被调用的有效范围，使其能被主调函数调用。一个文件内函数定义必须唯一，但函数声明可以不唯一。

主函数 main 是特殊的用户自定义函数，其定义形式为

```
⌊«数据类型符»⌋ «main» (⌊«int argc, char **argv»⌋){ «函数体» };
```

其特殊性主要体现在 main 是关键字；main 函数只能是主调函数，不能成为被调函数；main 函数是程序运行的入口，由操作系统启动（或调用）；main 函数可以是有参函数也可以是无参函数；对于有参函数，形参只能有两个，分别为 int 型和指向字符变量的指针数组。在运行命令时，第 1 个实参为命令及参数个数（参数个数+1），第 2 个实参为每个元素指向命令及实参的字符串。

1.7 变量有效范围与存储类别

变量是计算机高级语言的核心，其涉及决定存储单元大小和数据精度的数据类型，决定变量可访问的有效范围和决定变量生存期的存储类别，即变量的属性包括数据类型、有效范围和存储类别。在一个程序文件中，函数内及其复合语句内定义的变量为内部变量，内部变量在函数内或复合语句内可以被访问，而函数外定义的变量为外部变量，外部变量从定义处开始到文件结束所有语句均可以被访问。把变量的有效范围理解为一种区域，其覆盖到的函数和语句均可以访问变量。在文件中可以定义有效范围不同的变量，可以出现区域之间的交叉或覆盖的情况。一旦在交叉或覆盖区域含有同名的变量时，被覆盖内的变量访问优先权高于覆盖的变量优先权，也就是被覆盖屏蔽了覆盖，语句访问到的是被覆盖的变量。在一个文件中，若外部变量不是从文件开头处开始定义，则其有效范围只覆盖到部分文件，即从文件开头处到定义处的函数无法访问该外部变量，故该外部变量为局部变量；若外部变量从文件开头处开始定义，其有效范围覆盖了整个文件，则该外部变量为全局变量。从覆盖角度看，内部变量也可以理解为局部变量，通过外部变量声明可以改变局部变量（函数外定义）的有效范围，其声明形式为

```
«extern» «数据类型符» «外部变量名 1»⌊, «外部变量名 2»⌋ⁿ;
```

若外部变量声明在函数内，则其有效范围覆盖到整个函数；若外部变量声明在函数外，则其有效范围覆盖到从其声明处到文件结束。尤其是当外部变量声明在文件开头处时，外部变量的有效范围覆盖到整个文件，故该变量成为全局变量。在多个文件中的外部变量的有效范围与是否跨文件有关，这种外部变量的定义形式为

```
⌊«extern»⌋⊥«static» «数据类型符» «外部变量名 1»⌊, «外部变量名 2»⌋ⁿ;
```

其中，extern 可以省略，表示所定义的外部变量可以被跨文件访问，即这样的外部变量有效范围可以覆盖到另外文件。而 static 表示所定义的外部变量只限于被本文件内访问，即这样的外部变量有效范围最多只覆盖到本文件。对许可跨文件访问的外部变量进行定义，另一个文件需要对该变量进行变量声明。变量声明与变量定义不同，其可以多处出现。由于存在跨文件的外部变量定义和声明，因此变量有效范围改变了，故可能出现同名变量有效范围覆盖重叠，出现这种情况时需要遵循被覆盖的变量访问优先权大于覆盖的变量访问优先权的原则。可以看到，变量的有效范围提供了正确访问变量的安全性措施。

变量的生存期是指变量数据单元在程序运行过程中存在的时间，变量可分为动态变量和静态变量。动态变量在函数调用后开辟数据单元，在函数结束后数据单元撤销。静态变量在程序运行期间数据单元始终存在，直至程序结束。动态变量的动态特性是对动态变量的初始化重复进行，而静态变量的静态特性是对静态变量只初始化一次。在程序设计中，变量存储类别又可

细分为 4 种：自动（auto）、静态（static）、寄存器（register）和外部（extern）的存储方式。这 4 种变量的形式为

```
⌊«auto»⌋⊥«static»⊥«register»⊥«extern» «数据类型符» «变量名 1»⌊, «变量名 2»⌋ⁿ;
```

内部变量有自动、静态和寄存器 3 种存储方式；外部变量有静态和外部两种存储方式；外部变量只有静态存储，而采用静态或外部存储方式定义外部变量只是决定外部变量的有效范围是否跨文件。

总之，根据变量对应数据单元的生存期，变量分为动态变量和静态变量；根据变量对应数据单元的存储位置，变量分为内存变量和寄存器变量；根据变量在函数内外，变量分为内部变量和外部变量；根据变量可访问的范围（有效范围），变量分为局部变量和全局变量；根据变量是否允许跨文件访问，变量分为文件内部访问和跨文件访问。变量所属的数据类型和存储类别是通过关键字进行标识的，而变量的有效范围是通过变量在程序中的位置体现的。在编写程序过程中应该正确理解变量的数据类型、有效范围和存储类别这 3 个属性。

1.8　数据位运算

位运算是 C 语言作为计算机高级语言实现低级语言功能之一的运算方式，在应用软件、系统软件和支撑软件开发中得到广泛应用。位运算是以二进制位为单位的运算，位运算符包括按位逻辑运算符和移位运算符两类，按位复合赋值运算符包括按位逻辑复合赋值运算符和移位复合赋值运算符。移位运算包括左移位运算（<<）和右移位运算（>>）。左移位运算的形式为«运算数»«<<»«移位次数»，表示“运算数”所有二进制位向左进行“移位次数”次移位，左边最高位丢失，右边补 0。每左移位 1 次，相当于对“运算数”乘 2。右移位运算的形式为«运算数»«>>»«移位次数»，表示“运算数”所有二进制位向右进行“移位次数”次移位，右边最低位丢失，左边补 0（无符号数）或补符号位（有符号数，符号位不变）。每右移位 1 次，相当于对“运算数”除以 2，若“运算数”为奇数，则移位次数取不大于商的整数。移位运算的“运算数”可以是字符和整数（有符号整数按补码形式表示和存储）。移位复合赋值运算符为“<<=”和“>>=”，其“运算数”必须是有值的变量。按位逻辑运算是把运算数的二进制 1 和 0 分别当成逻辑真和逻辑假。按位逻辑运算包括按位求反（～）、按位逻辑与（&）、按位逻辑或（|）和按位异或（^）四种。按位求反（～«运算数»）是把“运算数”的每位二进制位 0 和 1 互换，其他按位逻辑运算（«运算数 1»op«运算数 2»，其中“op”可以是“&”“|”“^”之一）。对于“&”运算，若两个运算数对应的二进制位均为 1，则运算结果为 1，其他情况运算结果均为 0。对于“|”运算，若两个运算数对应的二进制位均为 0，则运算结果 0，其他情况运算结果均为 1。对于“^”运算，若两个运算数对应的二进制位不同，则运算结果为 1；若二进制位相同，则运算结果为 0。在运用上，位运算可完成低级语言的某些功能，如置位（置 1）、清零（清 0）和移位等，该运算也可用于数据加解密和压缩存储等。在本质上，位域类型是特殊的结构体类型，其成员按二进制位分配。位运算的形式为

```
«struct» «位域类型名» {
                        «数据类型符 1»«位域名 1»«：位域长度 1»;
                        ⌊«数据类型符 2»«位域名 2»«：位域长度 2»;⌋ⁿ
                      };
```

位域类型及其变量定义和访问与结构体类型及其变量定义和访问相同，只是多个位域共享同一个字节。位域变量直接访问的形式为«位域变量»«.»«位域名»，位域变量间接访问的形式为«位域指针»«->»«位域名»，这两种形式均表示位域变量的成员变量。由于指针指向的最小数据单元为字节，因此只有位域变量的指针而没有位域成员变量的指针，也就不能对位域成员变量取地址。位域类型提供了实现数据压缩、节省存储空间及提高程序效率的手段。

1.9 数据文件处理

文件是存储在计算机外部介质上有序的数据集合，通过文件名唯一标识。文件名一般包括盘符、目录路径、文件主干名和文件扩展名。根据存储内容文件可分为程序文件和数据文件；根据存储形式（编码）文件可分为文本文件和二进制文件。文本文件以 ASCII 码形式存储数据，由于内存数据（二进制）和外存数据（ASCII）形式不一致，因此在进行文件读/写时需要进行文本数据和二进制数据的转换。二进制文件以二进制形式存储数据，由于内存数据（二进制）与外存数据（二进制）形式一致，因此在进行文件读/写时无须进行数据转换。本书重点讨论数据的文本文件和二进制文件的访问。C 语言文件系统可分为缓冲文件系统（高级文件系统）和非缓冲文件系统（低级文件系统），前者为编译系统开辟输入/输出缓冲区，与操作系统无关，可移植性好；后者为程序自设输入/输出缓冲区，与操作系统相关，可移植性差。

关于缓冲文件系统的文件缓冲区和文件读/写等信息，C 语言给出了 FILE 结构类型进行描述。对文件进行访问之前必须先打开文件，打开文件的主要目的是建立文件指针和开辟输入/输出缓冲区。打开文件后，动态分配 FILE 类型的数据单元，并充填和维护相应信息，返回文件指针。程序中涉及的文件读/写过程使用该文件指针即可。在程序结束运行之前，必须关闭文件，其主要目的是在读/写所有缓冲区的数据后回收缓冲区，并在文件尾部插入文件结束标记 EOF。由于文件的访问可分为读、写和追加，并且文件又分为文本文件和二进制文件，因此在文件打开时会涉及文件名和文件打开模式，其中文件打开模式包含文件的类型（文本文件、二进制文件）和访问（读、写、可读可写、追加、可读可追加）信息。实现文件打开和关闭的函数有 fopen 函数和 fclose 函数。

文件打开后，形成唯一一个可以变动的文件“访问位置”标记，该标记与编辑软件中的光标类似，该标记的位置就是文件读/写位置。根据文件“访问位置”的变动，文件访问过程可分顺序访问和随机访问。文件顺序访问就是从文件“访问位置”开始（一般从文件头）依次读/写文件，每次读/写操作结束后，文件“访问位置”自动后移。而随机文件访问就是从文件“访问位置”开始，然后重新定位文件“访问位置”，再进行读/写。文件“访问位置”的定位由 rewind 函数和 fseek 函数实现。

文本文件的访问函数包括读/写文件字符（fgetc 函数和 fputc 函数）、读/写文件字符串（fgets 函数和 fputs 函数）、按格式读/写文件数据（fscanf 函数和 fprintf 函数）。二进制文件的访问函数包括按数据块读/写文件数据（fread 函数和 fwrite 函数）。

若文件打开、关闭和文件访问函数出错，则返回错误表示值，可以通过出错检测函数 ferror 获取当前文件访问的错误表示值。对当前错误表示值可以通过 clearerr 函数进行清除处理。

对于非缓冲文件系统，通过打开或创建文件，并由操作系统统一分配唯一的文件柄来标识正在访问的文件。输入/输出缓冲区（变量、数组数据单元）由程序自行设置。文件、文件柄和输入/输出缓冲区关联在一起由操作系统维护。

程序对文件的读/写本质上是对缓冲区的读/写。对于缓冲文件系统，缓冲区数据与外存的文件数据之间的交互由缓冲文件系统进行维护。通过缓冲文件系统可以解决内存与外存数据访问速度不匹配的问题。缓冲文件系统以字符、字符串、格式方式和数据块（二进制）方式对文件进行读/写，这丰富了文件访问形式，但是该系统的文件访问效率比非缓冲文件系统的文件访问效率低。非缓冲文件系统由程序在程序数据区中开辟缓冲区，并且只能按数据块方式对二进制文件进行读/写，故该系统具有文件高效访问的特点。

1.10　C 语言学习体会

软件系统经常是现实物理系统（也可以是虚拟系统）的一种模拟，如财务软件系统模拟真实财务管理业务和流程，同时充分发挥计算机高速运转、大量存储及实时交互等优点，使计算机成为工作、学习和生活中高效、有效的应用工具。计算机硬件的性能（主频高低、内存大小、总线宽度等）是计算机工作的基础，而计算机软件的功能（如科学计算、人事管理、游戏等）确保计算机的应用，硬件和软件构成了完整的计算机系统。在硬件确定的情况下，可以配备不同的软件，使得计算机具有不同的功能，并扮演不同的角色，如游戏机、学习机和办公系统等。在此意义上，软件成就了计算机的普适性，反过来说，任意一款软件都是针对不同应用背景产生的。

软件的模拟性、针对性使得软件开发必须深入研究现实世界的物理系统，尤其物理系统中的客观对象及其关系，如人事管理中张三（男，20 岁，月薪 5000 元）、李四（女，18 岁，月薪 6000）等；地层评价中检测深度 1000m（自然电位 20mv，自然伽马 500c/s，电阻率 0.4mΩ）、1000.125m（自然电位 20.5mv，自然伽马 501c/s，电阻率 0.42mΩ）等。进一步对客观对象及其关系的共性进行抽象，成为概念世界（人脑）中的对象类型及其关系类型，如人事管理中“人员”，地层评价中“深度点”，尤其对对象属性进行抽象，如人事管理中对象类型为“人员（姓名，性别，年龄，月薪）”，地层评价中对象类型为“测井系列（深度，自然电位，自然伽马，电阻率）”，也可以形式化为 Person(name, sex, age, salary)，Logging(depth, sp, gr, r)，其中 name、sex、depth 等为属性。在概念世界中，还要完成数据变换描述，即进行数据建模，如人事管理中平均工资 avg=f(salary)，地层评价含水饱和度 s=g(sp, gr, r)等。在完成现实世界到概念世界转变后，接着需要通过编程进入计算机世界进行问题求解、事务管理和事务处理等，即首先进行数据结构设计、算法设计和编码，完成程序设计，最终生成软件。

计算机高级语言（如 C 语言）是问题求解、事务管理、事务处理等进入计算机世界的工具，而核心是基于数据结构和算法的程序设计，即“数据结构+算法=程序设计”。采用 C 语言进行程序设计为 C 程序设计，还可以用其他语言进行程序设计，如 FORTRAN 程序设计、BASIC 程序设计和 Java 程序设计等。

为了在计算机世界中进行问题求解、事务管理、事务处理等，通过利用计算机语言的基本数据类型、构造数据类型对概念世界的对象类型及其关系类型一一对应描述，如性别为字符型（char）、年龄为整型（int）、月薪为浮点型（float）、自然电位为浮点型（float）等、Person 为结构体类型（struct Person{char name[20]; char sex; int age; float salary;}）、Logging 为结构体类型（struct Logging{ float depth, sp, gr, r;}）等。根据客观世界中单个对象或对象集合，采用变量、数组和链表等对其进行管理、处理。这些变量、数组和链表节点等都是根据数据类型进行定义生成的。数据类型定义、变量定义、数据定义及链表定义创建就是数据结构设计过程。数

据类型、变量和数组等都是由计算机语言规定的字符集生成的标识符表示。标识符进一步分成保留字标识符（如 int、float 等）和自定义标识符（如 Person、Logging，变量名、数组名等）。现实世界中对对象进行增加、删除、修改或排序等操作，概念世界中的建模表达了数据的变换，在进入计算机世界前，需要进行算法设计，进而表达问题求解、事务管理、事务处理过程。算法中每个步骤是确定无异议的，这些步骤除决定算法的跳转或循环走向外，更多是表达对象或对象属性的变换处理。采用计算机语言（如 C 语言）描述数据结构设计和算法设计就是程序设计（如 C 语言程序设计）。为了增强程序的可读性和提升软件质量，算法设计、程序设计都是结构化的。无论多么复杂的算法或程序，都是采用结构化，包括顺序结构、选择（分支）结构和循环结构。一条语句没有任何判断依次执行相邻下一条语句构成了顺序结构程序，而选择语句（if 语句、switch 与 break 语句）或循环语句（for 语句、while 语句、do-while 语句）根据是否满足条件来决定跳转到指定位置再接着执行相关语句，break 语句和 continue 语句无条件跳转到默认指定位置再接着执行相关语句，这些语句都改变了程序语句执行的走向。求解问题的流程为客观问题、事务→抽象表示、建模→数据结构+算法到结构化程序→程序运行→完成问题求解、事务管理、事务处理等。可以看到，程序中数据类型反映描述问题的能力，而运算符反映处理问题的能力，结构化语句反映问题的处理，程序反映问题处理的流程。

除结构化外，软件研发还常常需要分工协作，要求先按功能划分，再进行集成，即模块化程序设计——把大功能分解为小功能，再进一步分解为更小的功能，直到分解为最简单的基本功能。每个子功能均为功能模块且在 C 语言中体现为函数。模块化程序设计除可以增强程序可读性外，还可以增强程序的可修改性和可重用性。模块的连接通过模块接口进行，具体到 C 语言就体现在函数调用和函数返回上，这是模块间唯一的联系方式，故模块具有相对独立性。这个特性使一个模块的变化不影响到其他模块，也使模块具有很好的可修改性。一方面，一个模块可以被多个模块调用，即这个模块的代码被其他模块共享，模块具有可重用性（复用性），可以提高软件开发的高效性；另一方面，相同功能的模块可由不同算法实现，导致执行效率不同，模块单一接口形式可以确保选择高效的模块，进而提高软件的性能。这种“强功能、弱耦合”是模块化的优点，但是单一的数据通道等缺点降低了软件的工作效率，故模块化提供了外部变量作为模块间数据交互的通道。为了数据访问安全，外部变量增加了有效范围。为了程序安全，在不同文件间，对外部变量、函数也增加了有效范围，使得变量和函数访问更加安全。此外，对变量还增加了动态和静态的处理方式，提高内存空间的管理效率。

计算思维是运用计算机科学的基础概念、原理和方法进行问题求解、系统设计，以及人类行为理解等涵盖计算机科学广度的一系列思维活动，也就是基于计算机科学的思维，其与理论思维（以数学为代表）和实验思维（以物理为代表）构成当代思维体系，因此计算思维是当今社会人们必须具备的一种普适的思维能力。计算机基础教育承载计算思维教育，这主要体现在计算机基础知识和应用技能的培养，其中计算机基础知识包括计算机文化基础与使用、计算机硬件和软件技术基础和计算机应用基础 3 个层次。计算机语言及其程序设计是计算机基础技术的重要组成部分，它涵盖了问题描述、数据结构设计、算法设计、程序设计及程序运行与问题验证，因此它是计算思维教育的核心内容组成部分。对于 C 语言的学习不能只简单理解为工具性的掌握，而是要以 C 语言为载体来掌握基于计算机的问题求解方法，即计算思维培养。计算机应用针对具体应用，而采用计算思维进行软件

的规划、设计，并采用计算机语言（如 C 语言）和开发环境（如 Visual C++）进行编写，实现问题求解。就本书而言，基于人工智能搜索策略，采用数据结构和 C 语言，实现通用问题求解，提高读者计算机的应用技能。

1.11　本章小结

本章主要回顾、总结了 C 语言及其程序设计的内容，包括 C 语言、算法及程序、程序设计、数据、变量、常量、运算及复合运算、基本数据类型及构造类型、结构化程序设计、模块化程序设计、变量和程序有效范围、变量存储类别和数据文件处理。通过 C 语言的学习体会，站在软件系统实现的角度，把 C 语言的知识点串联起来。

习题 1

1．根据计算机高级语言的编程风格，计算机高级语言可以分为几种？各有什么特点？

2．叙述算法的定义、特点及与程序的关系。

3．函数

$$y=\begin{cases}2+3x, & \text{当 } x\leqslant 0 \text{ 时}\\ \sum_{i=1}^{5}(i^{2x}+5), & \text{当 } x>0 \text{ 时}\end{cases}$$

分别用自然语言、伪语言、流程图和 N-S 图描述算法。

4．什么是结构化程序设计？在结构化程序中有哪 3 种基本控制结构？

5．什么是模块化程序设计？在 C 语言程序中是如何体现模块化特性的？

6．采用结构化和模块化程序设计各有什么优点？

7．叙述软件开发过程。

8．叙述 C 语言上机实践过程。

9．叙述下列名词的含义

（1）标识符、标识符的作用

（2）数据、常量、变量、表达式、运算数

（3）数据类型、整型、浮点型、字符型

（4）运算、运算符

（5）函数、程序构成

（6）变量属性

（7）预处理命令

10．在学习运算符时，需要掌握运算符的什么要点？

11．按照优先级从高到低的顺序写出所学的运算符。

12．表达式和语句有什么不同？

13．为什么在 C 语言程序设计中变量必须先定义后使用（访问）？

14．判断下面标识符的合法性（正确√，错误×）：

a.b	Data_base	arr()	x−y	_1_a	$dollar	_Max
fun(x)	3abc	Y3	No:	(Y/N)?	J.Smith	a[1]

Yes/No　ox123　0x123　x=y　a+b–2　_1_2_3　funx

15．(a=5)&&a++||a/2%2，表达式的值为_____，a 值为_____。

16．判断下面常量合法性（正确√，错误×）：

'Abc'	2^4	–0x123	10e	077	088	\'n'	"A"
+2.0	0xab	10e–2	0xef	\'111'	"x/y"	π	'\ff'
35C	'?'	e3	–085	xff	'\aaa'	10:50	"#"
3.	–85	ff	'\xab'	"10:50"	'\\'	“\\”	'\t'

17．对于整数 10 和–10，给出其所属不同整型在内存数据单元中的存储形式。

数据类型	10	–10
int		
short int		
long int		
unsigned int		
unsigned short		
unsigned long		

然后把存储形式转换为八进制数和十六进制数，编程验证正确性。

18．定义 int x=10, y, z;执行 y=z=x;x=y==z;后，变量 x 的值为_____。

19．以下运算符中优先级最低的运算符为_____，优先级最高的为_____。

A．&&　B．!　C．!=　D．||　E．>=　F．==

20．若 w=1, x=2, y=3, z=4，则条件表达式 w<x?w:y<z?y:z 的结果为_____。

A．4　B．3　C．2　D．1

21．根据题意写出表达式

（1）设 n 是一个正整数，写出判断 n 是偶数的表达式为_____。

（2）设 a、b 是实数，写出判断 a、b 同号的表达式为_____。

（3）设 a、b、c 是一个三角形的三条边，分别写出判断直角三角形、等边三角形和等腰三角形的条件为_____。

22．已知 int a=10, b=2;float c=5.8;，分别求下面表达式的值。

（1）a+'a'–100*b%(int)c

a+++b++–a– – –b– –

b++%a++*(int)c

（2）a>b–4*c!=5

c<=a%2>=0

（3）a&&b||c–6

c–6&&a+b

!c+a&&b

（4）a>b%3?a+b:a–b

a>b?a>c?a:c:b>c?c:b

23．顺序程序设计

（1）将华氏温度转换为摄氏温度和绝对温度，其转换关系为

$c=\frac{5}{9}(f-32)$　　（摄氏温度）

$k=273.16+c$　　（绝对温度）

（2）把极坐标(r, θ)（θ 的单位为度）转换为直角坐标(x, y)，其转换关系为

$x=r*\cos\theta$

$y=r*\sin\theta$

（3）求任意 4 个实数的平均值、平方和、平方和开方。

24．分支程序设计

（1）break 语句与 switch 语句配合使用能起到什么作用？

（2）利用程序实现如下函数。

$$y=\begin{cases}\dfrac{\sin(x)+\cos(x)}{2}, & x\geqslant 0\\ \dfrac{\sin(x)-\cos(x)}{2}, & x<0\end{cases}$$

（3）字符判断、转换输出：小写英文字母转换为大写英文字母输出；大写英文字母转换为小写英文字母输出；数字字符保持输出不变；其他字符输出“other”。

（4）由键盘输入 3 个数 a、b、c，输出其中最大的数。

（5）由键盘输入 4 个数 a、b、c、d，将 4 个数由小到大排序输出。

（6）将百分制成绩转换为成绩等级：90 分以上为 A，80～89 分为 B，70～79 分为 C，60～69 分为 D，60 分以下为 E。

25．循环程序设计

（1）循环程序由几个部分组成？

（2）goto 语句与 if 语句在实现循环的过程中有什么优缺点？

（3）break 语句与 continue 语句在循环语句中起到什么作用？

（4）从跳转位置来看，goto 语句、break 语句和 continue 语句有什么不同？在循环嵌套中的应用有什么不同？

（5）求数列 12、22、32…202 的和，要求用三种循环语句实现。

（6）求和 $s_n=\frac{1}{1}+\frac{1}{1+2}+\frac{1}{1+2+3}+\cdots+\frac{1}{1+2+3+\cdots+n}$。

（7）求方程 $3x+5y+7z=100$ 的所有的非负整数解。

26．解释批量数据存储方式、数组、下标、指针、指针变量的含义。

27．在一维数组中找出最大的数，然后与第一个数交换，然后输出数组所有元素。分别用数组下标法和指针法实现。

28．对一维数组元素逆序存放，然后输出数组元素。分别用数组下标法和指针法实现。

29．已知数组中若干个整数从小到大排序，在插入一个数后，数组的顺序性保持不变。分别用数组下标法和指针法实现。

30．已知两个数组元素均从小到大排序，在合并两个数组后，还保持新数组元素的有序性不变。分别用数组下标法和指针法处理。

31．用选择法对一维数组元素按照从大到小的顺序排序。分别用数组下标法和指针法实现。

32．对同一个 4×4 二维数组进行转置，如

转置前 $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix}$ 转置后 $\begin{pmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{pmatrix}$

分别用数组下标法和指针法实现。

33．从键盘输入一个字符串，统计其中字母、数字和空格的个数。分别用数组下标法和指针法实现。

34．把一个字符串中的元音字母都删除后输出字符串。分别用数组下标法和指针法实现。

35．用选择法实现对 10 个英文单词按字典中的顺序排序。分别用数组下标法和指针法实现。

36．将 10 个分数（按照分子/分母的顺序输入），按分数值从小到大的顺序排序。分别用指针法和数组下标法实现。

37．已知 8 名学生的基本信息，输入每名学生的姓名和 2 门课程的成绩，按照总成绩从高到低的顺序排序。分别用指针法和数组下标法实现。

38．解释基本概念

函数定义与函数声明、函数调用与函数返回、 函数嵌套调用与函数递归调用、函数形参与实参、内部函数与外部函数、内部函数与外部函数的有效范围、include 预处理命令

39．已知标识符 p、p1、p2、p3、p4、p5、p6、p7，其定义形式为

float p, *p1, *p2[5], (*p3)[5], *p4[3][4], (*p5)[3][4], **p6, (*p7)();

解释说明 p、p1、p2、p3、p4、p5、p6、p7 的含义。

40．函数有哪些划分形式？

41．文件包含、编译和连接都可以实现两个文件的联合，它们有什么不同？

42．主调函数和被调函数的关系在程序执行过程中是如何体现控制与被控制的关系？

43．在函数调用关系中，函数参数可分为几种？各有什么特点？

44．根据文件中函数的位置，函数的有效范围是如何划分的？

45．根据函数在文件中的位置，函数可分为几类？函数声明有什么作用？

46．主函数 main 有哪些特点？

47．从程序安全角度解释：在函数定义中，关键字 static 和关键字 extern 起什么作用？

48．用牛顿迭代法求方程 $ax^3+bx^2+cx+d=0$ 的根，其中 a、b、c、d 和第一个根的近似值由键盘输入。

49．用梯形法求定积分 $\int_a^b f(x)\mathrm{d}x$，其中 $f(x)=5x^2+6x-3$，上下限 a 和 b 从键盘输入。

50．用递归方法求一个自然数的最大公约数。

51．用递归方法求 $n!$。

52．已知分数数列 $\frac{1}{2},\frac{2}{3},\frac{3}{5},\frac{5}{8},\frac{8}{13},\ldots$ 用结构体描述分数，采用递归方法求第 n 项，n 从键盘输入。

53．输入 3 个数，调用一个函数同时可得 3 个数中的最大值和最小值。

54．变量定义需要涉及哪三个属性？各有什么含义？

55．从程序、数据安全角度解释：什么是变量的有效范围？根据变量与函数定义中的位置关系，变量可分为哪两种变量各有什么特点？若变量在同一个文件内，则变量可分为哪两种变量各有什么特点？若变量在不同文件内，则变量可分为哪两种变量各有什么特点？

56．变量声明与变量定义有什么不同？为什么需要进行变量声明？根据变量在同一个文件和不同文件内，变量是如何声明的？

57．什么是变量的存储类型？变量存储类型可分为哪两大类？根据变量与函数定义中的位置关系，变量可细

分为哪些存储类型及其使用什么关键字修饰声明？

58．在定义变量时，关键字 static 和关键字 extern 起什么作用？

59．选择题

（1）以下运算符中优先级最低的是_____，最高的是_____。

A．&&　　B．&　　C．‖　　D．|

（2）若有运算符<<、sizeof、^、&=，则按优先级由高到低的顺序排列正确的是_____。

A．sizeof, &=, <<, ^　　B．sizeof, <<, ^, &=

C．^, <<, sizeof, &=　　D．<<, ^, &=, sizeof

（3）以下叙述中不正确的是_____。

A．表达式 a&=b 等价于　a=a&b

B．表达式 a|=b　等价于　a=a|b

C．表达式 a!=b　等价于　a=a!b

D．表达式 a^=b　等价于　a=a^b

（4）若 x=2, y=3，则 x&y 的结果是_____。

A．0　　B．2　　C．3　　D．5

（5）在位运算中，运算数每左移一位，则结果相当_____。

A．运算数乘以 2　　B．运算数除以 2

C．运算数除以 4　　D．运算数乘以 4

60．解释题

（1）写出以下每个代码段的输出结果，其中，i 、j 和 k 都是 unsigned int 类型的变量。

```
(a) i=8; j=9;
    printf(" %d", i >> 1 + j >> 1);
(b) i=1;
    printf ("%d", i&～i);
(c) i=2; j=1; k=0;
    printf(" %d", ～i&j^k);
(d) i=7; j=8; k=9;
    printf ("%d", i^j&k);
```

（2）编写一条语句将变量 i 的第 4 位进行转换（即 0 变为 1、1 变为 0）。

（3）函数 f 定义为

```
unsigned int f(unsigned int i , int m, int n)
{return (i >> (m+1-n) & ～(～0<<n));}
```

① ～(～0<<n)结果是什么?

② 函数 f 的作用是什么?

61．程序设计

（1）在计算机图形处理中，红、绿、蓝 3 种颜色组成显示颜色（三基色），每种颜色由 0～255 灰度表示，并且将 3 种颜色存放在一个长整型变量中，请编写名为 MK_COLOR 的宏，包含 3 个参数（红、绿、蓝的灰度），宏 MK_COLOR 需要返回一个长整型值，其中后 3 个字节分别为红、绿和蓝，且红在最后一个字节。

（2）定义字节交换函数

```
int swap_byte (unsigned short int i) ;
```

函数 swap_byte 的返回值是将 i 的两个字节调换后的结果。例如 i 的值是 0x1234（二进制形式为 00010010 00110100），swap_byte 的返回值为 0x3412（二进制形式 00110100 00010010）。程序以十六进制读入数，然后交换两个字节并显示

```
Enter a hexadecimal number: 1234
Number with byte swapped: 3412
```

提示：使用&hx 转换来读入和输出十六进制数。

另外，试将 swap_byte 函数的函数体化简为一条语句。

（3）定义函数

```
unsigned int rotate_left(unsigned int i , int n);
unsigned int rotate_right(unsigned int i , int n);
```

函数 rotate_left (i, n)的值是将 i 左移 n 位并将从左侧移出的位再移入 i 的右端。如整型占 16 位，rotate_left(0x1234, 4)将返回 0x2341。函数 rotate_right 也类似，只是将数字中的位向右循环移位。

62．C 程序处理的文件类型是哪些？

63．高级文件系统（缓冲文件系统）与低级文件系统（非缓冲文件系统）有什么不同?

64．将文件读/写时，为什么要关注当前访问位置？

65．将文件类型指针有什么作用？

66．为什么要对文件打开和关闭？

67．将 10 个整数写入数据文件 f.dat 中，再读出 f.dat 中的数据并求其和（分别用高级文件系统和低级文件系统实现）。

68．使用函数 scanf 从键盘输入 5 个学生的基本数据（包括学生姓名、学号、3 门课程的成绩），然后求出平均成绩。用 fprintf 函数输出所有信息到磁盘文件 stud.rec 中，再用函数 scanf 从 stud.rec 中读入这些数据并在显示屏上显示出来。

69．将 10 名职工的基本数据（取工号、职工姓名、性别、年龄和工资）从键盘输入，然后送入磁盘文件 worker1.rec 中保存，再从磁盘调入这些数据并依次打印出来（使用 fwrite 函数）。

（1）将 worker1.rec 中的职工的基本数据按工资由高到低排序，将排好序的记录存放在 worker2.rec 中（使用 fread 函数）。

（2）在文件 worker2.rec 中插入一个新职工的数据，并使数据在插入后仍保持原来的顺序（按工资高低顺序插入到原有文件中），然后写入 worker3.rec 中。

（3）删除 worker2.rec 中某个编号的职工记录，再存入原文件中（使用 fread 和 fwrite 函数）。