

全国高等教育自学考试计算机信息管理专业应试指导  
全国高等教育自学考试计算机信息管理专业应试指导



# 高级语言程序设计 **上机**模拟试题分析 与解答

姚庭宝 封孝生 编著



173

TP312.C-44

y354

全国高等教育自学考试计算机信息管理专业应试指导

# 高级语言程序设计 上机模拟试题分析与解答

姚庭宝 封孝生 编著



A1016104

電子工業出版社.

Publishing House of Electronics Industry

北京 · BEIJING

## 《自考应试指导丛书》编委会

主 编 张维明 常兆成

编 委 王朝阳 姚庭宝 邓必雄

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

### 图书在版编目(CIP)数据

高级语言程序设计上机模拟试题分析与解答/姚庭宝,封孝生编著. —北京:电子工业出版社,2002.11

全国高等教育自学考试计算机信息管理专业应试指导

ISBN 7-5053-8038-9

I. 高… II. ①姚… ②封… III. 高级语言—程序设计—解题 IV. TP321-44

中国版本图书馆 CIP 数据核字(2002)第 075201 号

责任编辑:秦 梅 特约编辑:李 莉

印 刷:北京民族印刷厂

出版发行:电子工业出版社 <http://www.phei.com.cn>

北京市海淀区万寿路 173 信箱 邮编 100036

经 销:各地新华书店

开 本:787×1092 1/16 印张:25 字数:640 千字

版 次:2002 年 11 月第 1 版 2002 年 11 月第 1 次印刷

印 数:6 000 册 定价:28.00 元

凡购买电子工业出版社的图书,如有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系。  
联系电话:(010)68279077

## 丛 书 序

全国高等教育自学考试指导委员会在组织编写供读者学习用教材的前言中写道：“这种教材应当适合自学，应当有利于学习者掌握、了解新知识、新信息，有利于学习者增强创新意识、培养实践能力、形成自学能力，也应有利于学习者学以致用，解决实际工作中所遇到的问题。”

确实，有一套好的助学教材对广大考生无疑是一个福音。然而，针对自学考试的特点，广大考生还殷切期望能拥有与教材相配套的模拟试题训练与应试指导方面的书籍，以帮助他们加强基础训练，巩固所学知识，增强灵活运用能力，提高应试自信心，达到既顺利过线又增长能力的双重效果。

自 2001 年起，全国自学考试开始使用新的考试大纲。为了尽快适应新考试计划的要求，我们作为高等教育自学考试计算机信息管理专业主考院校之一，组织了一批多年来工作在大学本科、高教自考第一线的教学骨干，陆续编写了一套全国高等教育自学考试计算机信息管理专业应试指导丛书，内容包括专科、本科多门课程，分为笔试、机试两类。这套应试指导丛书力求体现下列特色：题型丰富，分析透彻，解答准确，实用性强。我们希望本丛书能够成为广大自学者的良师益友，通过对模拟试题的试做和自测，帮助自学者消化所学知识，巩固所学概念，充分理解考核知识点，提高应试技巧，使他们在考前扎扎实实地提高水平，增强信心，在考试时能顺利过线，甚至考高分。同时，更重要的是能促使他们在培养训练综合素质与能力，增进知识和才干，努力成为合格的计算机信息管理人才方面迈出坚实的步伐。

这套应试指导丛书的主要读者虽然是全国高等教育自学考试计算机信息管理专业专科、本科广大考生，但其中有些书籍对其他相近专业（如计算机及应用专业等）以及全国计算机等级考试的广大考生亦具有较大参考价值

《自考应试指导丛书》编委会

## 前 言

本书是与《高级语言程序设计模拟试题分析与解答》相配套的应试指导书。编者认为,学习、掌握并熟练应用一门计算机高级语言(现为 C 语言)是培养广大学生使用计算机语言及应用能力的重要环节。编写这两本书的目的就是希望广大学生能够通过它们巩固和加深对 C 语言各种概念与特性的理解,了解和掌握用 C 语言编程的基本方法和技巧,培养和训练良好的编程风格和习惯,锻炼和提高使用计算机解决应用问题的能力,为后续课程和进一步的程序设计实践打下较扎实的基础。此外,本书还包括应试指导,以帮助、促进广大学生顺利过线,甚至考高分。编者殷切期望,这两本书能成为广大学生学用 C 语言进行编程的良师益友。

本书共分九个单元,它们是顺序结构程序设计,分支结构程序设计,循环结构程序设计,数组与字符串,函数,指针,结构体与线性链表,文件,通用程序设计。每一单元有四节,即基本概念、典型例题、模拟试题、模拟试题分析与参考解答。第一节简要介绍与本单元有关的基本概念。第二节给出四个典型例题,包括两个改错题和两个编程题。第三节提供十个模拟试题,包括五个改错题和五个编程题。第四节则对这十个试题作试题分析并给出完整的 C 程序以及运行示例。本书还有两个附录。附录 A 为 Turbo C 2.0 集成开发环境上机指南,简要介绍 Turbo C 2.0 的安装和启动, Turbo C 2.0 的集成开发环境, Turbo C 2.0 的配置文件, C 程序的开发和运行, C 程序的调试以及 C 编程常见错误。附录 B 为 C 语言程序设计题库,提供一批具有一定技巧性和趣味性,且富于启发性的 C 语言程序设计试题,以鼓励学有余力者进一步发展程序设计技巧,提高灵活运用 C 语言解决应用问题的本领,并深入体现 C 语言程序设计的潜在能力。

本书力求体现下列特色:内容翔实、题型丰富、信息量大、覆盖面广、分析透彻、解答准确、重在应用、实用性强;文字叙述流畅易读、富于启发性、针对性、直观性、技巧性。本书不仅适用于参加全国高等教育自学考试计算机信息管理专业、计算机及应用专业以及相应专业的广大考生,也适用于全国各高校本科生和参加全国计算机等级考试的考生。对从事 C 语言教学的广大教师和用 C 语言编程的技术工作者,亦有一定的参考价值。

非常感谢周予滨先生在本书的审校过程中提出的宝贵意见。

竭诚欢迎广大读者特别是教师、学生以及技术工作者对本书提出批评和建议。

编 者

# 目 录

|                          |      |
|--------------------------|------|
| 第一单元 顺序结构程序设计 .....      | (1)  |
| 1.1 基本概念 .....           | (1)  |
| 1.1.1 C 程序的基本结构 .....    | (1)  |
| 1.1.2 基本字符集与标识符 .....    | (1)  |
| 1.1.3 简单数据类型 .....       | (2)  |
| 1.1.4 运算符与表达式初探 .....    | (4)  |
| 1.1.5 赋值语句及简单输入、输出 ..... | (5)  |
| 1.2 典型例题 .....           | (7)  |
| 1.2.1 改错题 .....          | (7)  |
| 1.2.2 编程题 .....          | (10) |
| 1.3 模拟试题 .....           | (13) |
| 1.3.1 改错题 .....          | (13) |
| 1.3.2 编程题 .....          | (15) |
| 1.4 模拟试题分析与参考解答 .....    | (17) |
| 1.4.1 改错题 .....          | (17) |
| 1.4.2 编程题 .....          | (23) |
| 第二单元 分支结构程序设计 .....      | (30) |
| 2.1 基本概念 .....           | (30) |
| 2.1.1 if 语句 .....        | (30) |
| 2.1.2 switch 语句 .....    | (32) |
| 2.2 典型例题 .....           | (33) |
| 2.2.1 改错题 .....          | (33) |
| 2.2.2 编程题 .....          | (37) |
| 2.3 模拟试题 .....           | (40) |
| 2.3.1 改错题 .....          | (40) |
| 2.3.2 编程题 .....          | (43) |
| 2.4 模拟试题分析与参考解答 .....    | (47) |
| 2.4.1 改错题 .....          | (47) |
| 2.4.2 编程题 .....          | (55) |
| 第三单元 循环结构程序设计 .....      | (66) |
| 3.1 基本概念 .....           | (66) |
| 3.1.1 while 语句 .....     | (66) |
| 3.1.2 do-while 语句 .....  | (66) |
| 3.1.3 for 语句 .....       | (67) |

|             |                               |              |
|-------------|-------------------------------|--------------|
| 3.1.4       | 用于循环控制的几个特殊运算符                | (67)         |
| 3.1.5       | 多重循环结构                        | (68)         |
| 3.1.6       | break 语句、continue 语句和 goto 语句 | (68)         |
| 3.2         | 典型例题                          | (69)         |
| 3.2.1       | 改错题                           | (69)         |
| 3.2.2       | 编程题                           | (73)         |
| 3.3         | 模拟试题                          | (76)         |
| 3.3.1       | 改错题                           | (76)         |
| 3.3.2       | 编程题                           | (79)         |
| 3.4         | 模拟试题分析与参考解答                   | (83)         |
| 3.4.1       | 改错题                           | (83)         |
| 3.4.2       | 编程题                           | (91)         |
| <b>第四单元</b> | <b>数组与字符串</b>                 | <b>(101)</b> |
| 4.1         | 基本概念                          | (101)        |
| 4.1.1       | 一维数组                          | (101)        |
| 4.1.2       | 多维数组                          | (102)        |
| 4.1.3       | 字符数组与字符串                      | (103)        |
| 4.2         | 典型例题                          | (104)        |
| 4.2.1       | 改错题                           | (104)        |
| 4.2.2       | 编程题                           | (108)        |
| 4.3         | 模拟试题                          | (112)        |
| 4.3.1       | 改错题                           | (112)        |
| 4.3.2       | 编程题                           | (116)        |
| 4.4         | 模拟试题分析与参考解答                   | (119)        |
| 4.4.1       | 改错题                           | (119)        |
| 4.4.2       | 编程题                           | (129)        |
| <b>第五单元</b> | <b>函数</b>                     | <b>(141)</b> |
| 5.1         | 基本概念                          | (141)        |
| 5.1.1       | C 程序中的函数                      | (141)        |
| 5.1.2       | 函数定义                          | (142)        |
| 5.1.3       | 函数调用与函数说明                     | (142)        |
| 5.1.4       | 通过函数参数传递数据                    | (143)        |
| 5.1.5       | 函数的嵌套调用与递归调用                  | (144)        |
| 5.1.6       | 数组作为函数参数传递                    | (144)        |
| 5.2         | 典型例题                          | (145)        |
| 5.2.1       | 改错题                           | (145)        |
| 5.2.2       | 编程题                           | (149)        |
| 5.3         | 模拟试题                          | (153)        |
| 5.3.1       | 改错题                           | (153)        |
| 5.3.2       | 编程题                           | (157)        |

|                              |              |
|------------------------------|--------------|
| 5.4 模拟试题分析与参考解答 .....        | (161)        |
| 5.4.1 改错题 .....              | (161)        |
| 5.4.2 编程题 .....              | (170)        |
| <b>第六单元 指针 .....</b>         | <b>(184)</b> |
| 6.1 基本概念 .....               | (184)        |
| 6.1.1 指针变量的说明与赋值操作 .....     | (184)        |
| 6.1.2 指针运算 .....             | (186)        |
| 6.1.3 指向数组的指针变量 .....        | (187)        |
| 6.1.4 指向字符串的指针变量 .....       | (187)        |
| 6.1.5 指向函数的指针变量 .....        | (188)        |
| 6.1.6 返回指针值的函数 .....         | (188)        |
| 6.1.7 指针数组 .....             | (189)        |
| 6.1.8 指向指针的指针变量 .....        | (189)        |
| 6.2 典型例题 .....               | (190)        |
| 6.2.1 改错题 .....              | (190)        |
| 6.2.2 编程题 .....              | (192)        |
| 6.3 模拟试题 .....               | (197)        |
| 6.3.1 改错题 .....              | (197)        |
| 6.3.2 编程题 .....              | (200)        |
| 6.4 模拟试题分析与参考解答 .....        | (205)        |
| 6.4.1 改错题 .....              | (205)        |
| 6.4.2 编程题 .....              | (213)        |
| <b>第七单元 结构体与线性链表 .....</b>   | <b>(226)</b> |
| 7.1 基本概念 .....               | (226)        |
| 7.1.1 用 typedef 定义类型 .....   | (226)        |
| 7.1.2 结构体变量说明与赋值操作 .....     | (226)        |
| 7.1.3 结构体数组和结构体指针 .....      | (228)        |
| 7.1.4 动态存储分配及释放 .....        | (228)        |
| 7.1.5 线性链表概念 .....           | (229)        |
| 7.1.6 线性链表生成与遍历 .....        | (230)        |
| 7.1.7 在已知线性链表中的插入与删除操作 ..... | (230)        |
| 7.1.8 带头结点的线性链表结构 .....      | (231)        |
| 7.2 典型例题 .....               | (231)        |
| 7.2.1 改错题 .....              | (231)        |
| 7.2.2 编程题 .....              | (234)        |
| 7.3 模拟试题 .....               | (239)        |
| 7.3.1 改错题 .....              | (239)        |
| 7.3.2 编程题 .....              | (243)        |
| 7.4 模拟试题分析与参考解答 .....        | (249)        |
| 7.4.1 改错题 .....              | (249)        |

|                                          |       |
|------------------------------------------|-------|
| 7.4.2 编程题 .....                          | (260) |
| <b>第八单元 文件</b> .....                     | (277) |
| 8.1 基本概念 .....                           | (277) |
| 8.1.1 文件指针 .....                         | (277) |
| 8.1.2 文件的打开与关闭 .....                     | (278) |
| 8.1.3 文本文件操作 .....                       | (279) |
| 8.1.4 二进制文件操作 .....                      | (280) |
| 8.1.5 文件的定位与随机读写 .....                   | (280) |
| 8.1.6 文件检测函数 .....                       | (280) |
| 8.2 典型例题 .....                           | (280) |
| 8.2.1 改错题 .....                          | (280) |
| 8.2.2 编程题 .....                          | (284) |
| 8.3 模拟试题 .....                           | (288) |
| 8.3.1 改错题 .....                          | (288) |
| 8.3.2 编程题 .....                          | (291) |
| 8.4 模拟试题分析与参考解答 .....                    | (297) |
| 8.4.1 改错题 .....                          | (297) |
| 8.4.2 编程题 .....                          | (306) |
| <b>第九单元 通用程序设计</b> .....                 | (321) |
| 9.1 基本概念 .....                           | (321) |
| 9.1.1 程序设计风格 .....                       | (321) |
| 9.1.2 结构化程序设计方法 .....                    | (321) |
| 9.1.3 通用程序设计 .....                       | (323) |
| 9.2 典型例题 .....                           | (323) |
| 9.2.1 改错题 .....                          | (323) |
| 9.2.2 编程题 .....                          | (327) |
| 9.3 模拟试题 .....                           | (331) |
| 9.3.1 改错题 .....                          | (331) |
| 9.3.2 编程题 .....                          | (336) |
| 9.4 模拟试题分析与参考解答 .....                    | (340) |
| 9.4.1 改错题 .....                          | (340) |
| 9.4.2 编程题 .....                          | (352) |
| <b>附录 A Turbo C 2.0 集成开发环境上机指南</b> ..... | (363) |
| A.1 Turbo C 2.0 的安装和启动 .....             | (363) |
| A.2 Turbo C 2.0 的集成开发环境 .....            | (363) |
| A.2.1 File(文件)菜单项 .....                  | (364) |
| A.2.2 Edit(编辑)菜单项 .....                  | (365) |
| A.2.3 Run(运行)菜单项 .....                   | (365) |
| A.2.4 Compile(编译)菜单项 .....               | (366) |
| A.2.5 Project(项目)菜单项 .....               | (366) |

|                                   |       |
|-----------------------------------|-------|
| A.2.6 Options(选项)菜单项 .....        | (367) |
| A.2.7 Debug(调试)菜单项 .....          | (369) |
| A.2.8 Break/watch(断点及监视)菜单项 ..... | (369) |
| A.3 Turbo C 2.0 的配置文件 .....       | (370) |
| A.4 C 程序的开发和运行 .....              | (370) |
| A.5 C 程序的调试 .....                 | (371) |
| A.6 C 语言编程常见错误 .....              | (372) |
| 附录 B C 语言程序设计题库 .....             | (374) |
| 参考书目 .....                        | (386) |

# 第一单元 顺序结构程序设计

## 1.1 基本概念

本单元的内容涉及 C 语言的一些基础知识,包括 C 程序的基本结构,基本字符集与标识符,简单数据类型,运算符与表达式,赋值语句及简单输入、输出等。它们在 C 语言编程中被经常用到。一个 C 程序大体上可分为输入数据、计算或处理数据以及输出数据、文字信息这三大部分,而这些都往往离不开诸如常量定义,变量说明,使用输入、输出标准函数和表达式、赋值语句等。在本单元的示例程序和模拟试题中,均仅涉及顺序结构程序设计。

### 1.1.1 C 程序的基本结构

可执行的 C 程序系由一个或多个函数组成,每个函数有其自己的名字,实现一定的功能。函数是 C 程序的基本构件。任何 C 程序必须有一个名为 main 的主函数。C 程序总是从主函数 main 开始。

C 程序中的函数由函数首部与函数体组成。函数首部说明了该函数的名字和类型特征等。函数体是一个由左、右花括号括起来的复合结构,其中可包含若干变量说明(亦称变量声明)及执行语句。变量必须先说明后使用。执行语句包括赋值语句、格式输出语句等。

为了输出指定的字符串信息或计算结果,可以使用格式输出语句(`printf` 语句)。这实际上是个函数调用语句。`printf` 函数是系统提供的一个标准库函数,它在系统预定义头标文件 `stdio.h` 中定义。在程序开始位置处写如下包含命令

```
#include <stdio.h>
```

可为该程序提供一批有关输入、输出等方面的系统服务,包括可在程序中使用 `printf` 语句。

在 C 程序中,任一语句必须以分号作为其终止符。分号不仅仅起语句分隔符的作用,而且它是语句的必要组成成分,故不能省略。

为了增加 C 程序的可读性,可在适当位置插写注释。注释以“`/*`”开始,以“`*/`”结束,其间允许有任意字符串。在编译时它被 C 编译器忽略。

为便于理解并修改程序,应按一定的缩进格式来书写并键入程序。对于层次较多、结构较为复杂的程序而言,这样做就更显其重要性。

### 1.1.2 基本字符集与标识符

C 语言的基本字符集是 ASCII 字符集的一个子集。标准的 ASCII 字符集共有 128 个 ASCII 代码,其序号由 0 至 127。C 语言中使用的基本字符集包括数字字符“0”~“9”;英文大、小写字母字符“A”~“Z”、“a”~“z”;专用符号,如“+”、“-”、“\*”、“/”、“%”、“=”、“<”、“>”、“(”、“)”、“[”、“]”、“{”、“}”、“!”、“&”、“|”、“^”、“~”、“==”、“!=”、“<=”、“>=”、“&&”、“||”、“<<”、“>>”、“++”、“--”、“-”等。在 C 语言中还用到一些

表示特定功能的符号，如换行符用转义序列（或称转义符）“\n”表示等。

标识符用以标记名字。程序中要用到许多名字，有些名字在 C 语言中有特定含义，称为关键字（或保留字），如 `int`、`float`；有的名字也是由 C 语言预先规定的，用来标识预定义的命令、库函数等，称为标准标识符，如 `include`、`printf` 等；其余的则是应由程序员根据程序需要而自行选定的名字，如 `a`、`b`、`ratio`、`display_message` 等。所有上述名字都是用标识符来标识的。

在 C 语言中，一个标识符的形成规则是：它必须由英文字母字符或下划线字符（ASCII 序号 95）开头，后面可跟英文字母字符、数字字符和下划线字符的任意组合。C 语言对标识符中的英文字母大小写是严格区分的，或称是“敏感”的，英文大写字母和英文小写字母被认为是两个不同的字符。因此，`main` 和 `Main` 是两个不同的标识符。一个标识符的有效长度，即组成该标识符的有效字符个数按照 ANSI C 标准规定，为 31。

表 1.1 列出了 C 语言的 32 个基本关键字。

表 1.1 C 语言基本关键字表

|                       |                     |                       |                       |
|-----------------------|---------------------|-----------------------|-----------------------|
| <code>auto</code>     | <code>double</code> | <code>int</code>      | <code>struct</code>   |
| <code>break</code>    | <code>else</code>   | <code>long</code>     | <code>switch</code>   |
| <code>case</code>     | <code>enum</code>   | <code>register</code> | <code>typedef</code>  |
| <code>char</code>     | <code>extern</code> | <code>return</code>   | <code>union</code>    |
| <code>const</code>    | <code>float</code>  | <code>short</code>    | <code>unsigned</code> |
| <code>continue</code> | <code>for</code>    | <code>signed</code>   | <code>void</code>     |
| <code>default</code>  | <code>goto</code>   | <code>sizeof</code>   | <code>volatile</code> |
| <code>do</code>       | <code>if</code>     | <code>static</code>   | <code>while</code>    |

### 1.1.3 简单数据类型

在数据描述中要用到数据类型。数据类型有两类：预定义数据类型和程序员自定义数据类型。一个数据类型定义了变量可接受值的集合以及对它所能执行的操作。数据类型有三种主要用途：①指明对该类型的数据项（即变量）应分配多大内存空间；②定义能用于该类型数据项的操作；③防止数据类型不相匹配。

C 语言向程序设计者提供了一个丰富的数据类型集合，它们各用于专门目的，但却都是由各种简单的、非构造型的数据类型所构成。本节所涉及的是 C 语言中最为基本的几种数据类型：整数类型、实数类型以及字符类型，或将它们简称为整型、实型以及字符型。它们都是系统预定义的简单数据类型，或称标准标量类型。其中整型、字符型都属于离散类型或称序数类型，其特点是有序性。

#### 1. 整数类型

一个整数类型数据用来存放整数。C 支持多种形式的整数类型，以适应不同情况的需要。

C 语言有四种形式的整型变量（简称整变量）：基本整型（`int`）、短整型（`short int`）、长整型（`long int`）和无符号整型（`unsigned int`）。前三种同属于有符号整型（`signed int`）。这里 `int`、`short`、`long`、`unsigned` 以及 `signed` 均为 C 语言的关键字。每一种整数类型规定了相应整数的取值范围以及所占内存字节数（一个字节为 8 个二进制位）。

最常使用的整型数据是基本整型（`int`）变量。

C 语言有三种表示整型常量的形式，它们分别是十进制形式、八进制形式以及十六进制

形式。整型常量简称整常量。

十进制整常量：以非 0 数字打头的十进制数字串，如 220、-560。

八进制整常量：以数字 0 打头的八进制数字串，如 06、0106，分别相当于十进制数 6、70。

十六进制整常量：以数字和字母的组合 0x 或 0X 打头的十六进制数字串，如 0x4e、0XFF 分别相当于十进制数 78、255。

在十六进制整常量中，英文小写字母 a~f 或对应地英文大写字母 A~F，分别表示十进制数的 10~15。

在整常量后添加一个英文字母 l 或 L，表示这是一个长整常量，如 0200000L 表示一个八进制长整常量，相当于十进制数 65536。

## 2. 实数类型

一个实数类型数据用来存放实数，C 语言支持多种形式的实数类型，以适应不同情况的需要。

C 语言有两种形式的实型变量（简称实变量），即单精度实型（float）与双精度实型（double）。这里 float 和 double 均为 C 语言的关键字。每一种实数类型规定了相应实数的取值范围、其所占内存字节以及它们所能达到的精度即有效数字位数。

C 语言有两种表示实型常量的形式，一种是十进制小数点形式，另一种是科学记数法的指数形式。实型常量简称实常量。例如：

6.365            十进制小数点形式实常量。

6.8e-5           科学记数法的指数形式实常量，其值等于  $6.8 \times 10^{-5}$ ，即 0.000068。

6.8E-5           与上数相同。

C 语言规定：浮点数的小数点两边至少一边要有数字。例如：0.22 可写为 .22，-0.0015E-3 可写为 .0015E-3。E 或 e 的左边必须有数字，右边必须有整数。.e6、1e1.5、2e 均为非法。

C 语言中最基本的实数类型是双精度实型（double）。若要表示一个单精度型（float）实常量，则需要对所写实数最后加上后缀字符 f 或 F。类似地，若在所写实数最后加上后缀字符 l 或 L，则表示这是一个长双精度型实常量。长双精度型实常量可具有约 19 位有效数字。

## 3. 字符类型

一个字符类型数据用来存放 ASCII 字符集内的某个字符。C 语言提供预定义字符类型 char。char 是一个关键字。C 语言支持扩展 ASCII 代码，共包括 256 个不同字符。

C 语言的字符型变量（简称字符变量）的形式比较简单，即字符型 char。但在 ANSI C 和 ISO C 中，还规定了 signed char 和 unsigned char 两种形式的字符类型，它们与 char 一样，在内存中占据一个字节。C 语言规定，字符型数据在操作时将按整型处理，在计算机内部，一个字符型数据所对应的整数值就是该字符的 ASCII 字符集中所对应的序数值或序号，即相应的 ASCII 代码值。但字符型 char、signed char 与 unsigned char 数据的取值范围有所不同，char 与 signed char 型字符数据的取值范围是 -128~127，而 unsigned char 型字符数据的取值范围是 0~255。

在 C 语言中，一个字符型常量代表 ASCII 字符集中的一个字符。将字符用单引号括起来，即成一个字符型常量或简称字符常量。例如：

'A'            表示英文大写字母 A 的字符常量。

'a' 表示英文小写字母 a 的字符常量。'a'与'A'不同。

" " 表示空格符的字符常量。

#### 1.1.4 运算符与表达式初探

表达式是描述计算机规则的一种算法结构，它是运算中最重要的成分。表达式由运算对象和运算符组成。根据需要可增加圆括号。运算对象又称操作数，包括变量、常量和函数等。运算符又称操作符，它是用于描述对数据进行运算的特殊符号。在 C 语言中，提供了非常丰富的各种运算符，从而构成了功能强大而又灵活多样的各类表达式，用以处理各种复杂的操作。运算符种类有十余种之多，而表达式种类亦丰富多彩。因此，人们甚至把 C 语言视做一种表达式语言。

##### 1. 算术运算符与算术表达式

基本的算术运算符除单目运算符“+”、“-”（表示正、负）外，还有双目运算符“\*”、“/”、“%”、“+”、“-”。

C 语言要求双目运算符左右两边的操作数的类型必须一致才能进行运算，所得结果类型与操作数类型相同。但如果双目运算符左右两边操作数的类型不一致，例如一边是整型数，另一边是实型数，则系统将自动把整型数转换为等值的实型数，然后再进行运算。另外，在 C 语言中，所有实型数的运算均按双精度（double）形式进行，以确保运算精度。

对于双目运算符“/”，当其左右两边操作数都是整型数时，其运算结果亦为整型数。对于不能整除的情形，则是截断取整（向零归整）。

对于双目运算符“%”，要求其左右两边的操作数均为整型数，运行结果是两数相除后所得的余数，亦为整型数。由于两操作数可正可负，当操作数中有负数时，C 语言规定，i%j 的运算结果值其正、负与被除数 i 的正、负相一致。

各个算术运算符的运算优先级按由高到低的顺序排列为：单目运算符“+”与“-”，双目运算符“\*”、“/”与“%”以及双目运算符“+”与“-”。单目运算符“+”、“-”的结合方向是从右到左，而双目运算符“\*”、“/”、“%”、“+”、“-”的结合方向都是从左到右。

算术表达式与一般数学式含义相似，其求值规则也有不少相同之处。比如，运算符有优先级高、低之分，同一级运算符一般按书写顺序自左至右计算，圆括号内的表达式以及函数优先计算等。

但是，C 语言中的算术表达式与一般数学式在表示形式上有所不同，比如  $b^2-4ac$  应表示为  $b*b-4*a*c$ ； $\sin 2\pi x$  应表示为  $\sin(2*3.1416*x)$ ；算术表达式中的所有括号包括多重括号一律使用圆括号等等。

##### 2. 关系运算符与关系表达式

关系运算符都是双目运算符。关系运算是一种比较关系运算符左右操作数大小的简单逻辑运算。比较结果表示这种关系成立与否。C 语言提供了“<”、“<=”、“>”、“>=”、“==”、“!=”六种关系运算符。

C 语言要求关系运算符左右两边的操作数的类型一致。如果进行关系运算的两个数据的类型不一致，系统将按相应规则先自动进行转换，然后再进行运算。关于关系运算的结果值，C 语言规定，当关系运算符所表示的关系成立时，其计算结果值取做 1，关系不成立时，其计算结果值取做 0。关系运算的结果值类型为 int 型。

关系运算符的运算优先级按由高到低的顺序排列为：“<”、“<=”、“>”、“>=”优先级较

高，“==”、“!=”优先级较低。关系运算符都是双目运算符，它们的结合方向都是从左到右。

由关系运算符及其操作数所组成的表达式，称为关系表达式。在关系表达式中，同样可以增加圆括号对。操作数自身亦可以是合法的表达式，包括算术表达式、关系表达式等。关系运算符的优先级低于所有算术运算符。

### 3. 逻辑运算符与逻辑表达式

逻辑运算是指用形式逻辑原则来建立数值间关系的运算。C 语言提供了三种逻辑运算符：“!”、“&&”、“||”。这里“!”是单目运算符，“&&”与“||”均为双目运算符。

C 语言中没有特别用于表示逻辑值的数据类型，它规定任意基本整型的值都可以作为逻辑值使用，任何非 0 的值都被当做“真”值，表示逻辑关系成立；而 0 值被当做“假”值，表示逻辑关系不成立。

逻辑运算符的运算优先级是：“!”最高，“&&”次之，“||”最低。单目运算符“!”的结合方向是从右到左，而双目运算符“&&”、“||”的结合方向都是从左到右。逻辑运算符“!”的优先级高于所有算术运算符与关系运算符，而“&&”与“||”的优先级则低于所有算术运算符与关系运算符。

由逻辑运算符及其操作数组成的表达式，称为逻辑表达式。在逻辑表达式中，同样可以增加圆括号对，其操作数自身亦可以是合法的表达式，包括算术表达式、关系表达式、逻辑表达式等。

### 4. 类型转换

C 语言不是一个强类型语言，在一个表达式中可以出现若干个属于不同数据类型的操作数。但 C 语言又要求，对于每一个双目运算符，其左右两边的操作数的类型必须一致才能进行运算。这时，就需要进行必要的类型转换。C 语言有两种形式的类型转换：隐式类型转换与显式类型转换。

隐式类型转换是指算术运算符当其左右操作数的类型不相一致时，由系统自动进行的一种类型转换，又称普通算术转换。其转换原则是类型向着两者之间占用单元多的类型转换，即所谓“类型提升”原则。比如，一个 int 类型数据（占 2 字节内存）与一个 long 类型数据（占 4 字节内存）相加，系统先将 int 类型数据自动转换为 long 类型数据，再进行相加运算。而运算结果则取较高数据类型，即 long 类型。

C 语言还提供了另一种类型转换的机制，称为显式类型转换或强制类型转换。其一般形式是

(类型名)表达式

其中类型名可以是一个任意基本数据类型名。这种类型转换形式的结果是：把上述表达式的结果类型转换成圆括号中给定的数据类型。

## 1.1.5 赋值语句及简单输入、输出

赋值语句及输入、输出在 C 程序中最常使用。程序员输入所需数据，借助赋值语句及其他语句对数据进行计算或处理，并输出结果。

### 1. 赋值语句

赋值语句是 C 程序中最基本的语句。它的一般形式为

变量=表达式;

其功能是：计算赋值运算符“=”右部表达式的值，将结果赋给左部变量。一般地，要求右部表达式结果值的数据类型应与左部变量的数据类型相一致，此时的赋值操作自然可顺利地进行。但是，当右部表达式与左部变量的数据类型不一致时，C语言规定，在转换允许的情况下，比如当它们都是数值型或字符型时，则表达式的结果值将被自动转换成左部变量数据类型，然后再进行赋值操作。

C语言允许进行多重赋值。

## 2. 赋值表达式

C语言中允许使用一种特殊的表达式，称为赋值表达式。其一般形式为

变量=表达式

其功能是：计算赋值符右部表达式的值，赋予左部变量，而该赋值表达式自身的值即为左部变量所得到的新值。赋值表达式可以出现在允许书写的表达式场合。

赋值表达式是C语言诸多表达式形式中的一个重要特性。对它的灵活使用，可以使C程序结构紧凑，书写简洁。但对初学者来说，开始时仍需慎用为妥。

## 3. 格式输出语句

C语言为输出数据提供了一个最常用的格式输出语句，它实际上是对系统预定义标准库函数printf的调用。printf函数是格式化输出函数，用于向标准输出设备按指定格式输出若干项数据、文字等信息。

格式输出语句的一般形式为

```
printf("格式控制字符串", 输出项列表);
```

这里，格式控制字符串或简称格式控制串可包含如下三类字符：一是普通字符，对于此类字符，将按原样输出；二是转义符，其实是转义序列，它由反斜线字符“\”后跟一个特定字符组成，表示将执行一种特定的功能，即用来输出转义符所代表的控制代码或特殊字符，比如转义符“\n”表示它是个换行符，即光标位置回车换行；三是格式转换说明符，简称格式符，它由百分字符“%”后跟1~2个英文小写字母字符组成，格式符指定了相对应的输出项数值的输出类型及格式。C语言允许使用多至十余种格式符。

printf中的格式符均以百分字符“%”开始，后跟表示某类格式的字符，其中大多数是英文小写字母字符。表1.2列出了printf中常用的格式符。

表 1.2 printf 中常用的格式符

| 格式符 | 作 用                       |
|-----|---------------------------|
| %d  | 输出有符号十进制整数（正数不输出符号）       |
| %f  | 输出小数点形式的十进制单双精度实型数，隐含6位小数 |
| %c  | 输出一个字符                    |
| %s  | 输出一个字符串                   |
| %o  | 输出八进制无符号整数（不带前导符0）        |
| %x  | 输出十六进制无符号整数（不带前导符0x或0X）   |
| %%  | 输出字符%本身                   |

另外，还可以在“%”与表示格式的字符之间增加m[n]以指定输出宽度，m指出数据宽度，n指出小数位数或截取的字符个数。输出数据采用右对齐形式。若欲指定输出宽度域

的同时还指定数据按左对齐形式，则可在宽度域之前加上一个减号“-”来实现。

#### 4. 格式输入语句

C 语言为输入数据提供了一个最常用的格式输入语句。它实际上是对系统预定义标准库函数 `scanf` 的调用。`scanf` 函数是格式化输入函数，用于从标准输入设备输入适当的数据赋值给相应的变量。

格式输入语句的一般形式为

`scanf ("格式控制字符串", 地址项列表);`

这里，格式控制字符串或简称格式控制串可包含如下三类字符：一是普通字符，对于此类字符，将要求按原样输入；二是转义符，用来输入转义符所代表的控制代码或特殊字符，此类字符在输入时较少使用；三是格式转换说明符，简称格式符，其含义与函数 `printf` 中的格式符基本相同。

地址项列表可包含若干个有待输入其相应值的变量的内存单元地址，各地址项之间用逗号隔开。C 中的地址量的表示是在变量前加上前缀字符“&”。

表 1.3 列出了 `scanf` 中常用的格式符。

表 1.3 `scanf` 中常用的格式符

| 格式符             | 作 用            |
|-----------------|----------------|
| <code>%d</code> | 输入十进制整型数       |
| <code>%f</code> | 输入小数点形式的十进制实型数 |
| <code>%c</code> | 输入一个字符         |
| <code>%s</code> | 输入一个字符串        |
| <code>%o</code> | 输入八进制整数        |
| <code>%x</code> | 输入十六进制整数       |
| <code>%%</code> | 输入字符%本身        |

另外，还可以在“%”与表示格式的字符之间增加 `m` 以指定欲输入整型数据的宽度。但不可以对实型数据指定小数位的宽度。

## 1.2 典型例题

### 1.2.1 改错题

【示例 1】 下述 C 程序有 4 处错误，请指出并改正它们（为便于指明位置，程序左边增写了行号，但实际的 C 程序中不应包含这些行号，下同）。

```
1  /* program demo1_01.c
2  #include <stdio.h>
3  main()
4  {
5      int i=10;
6          j;
7      printf("Enter an integer: ");
```