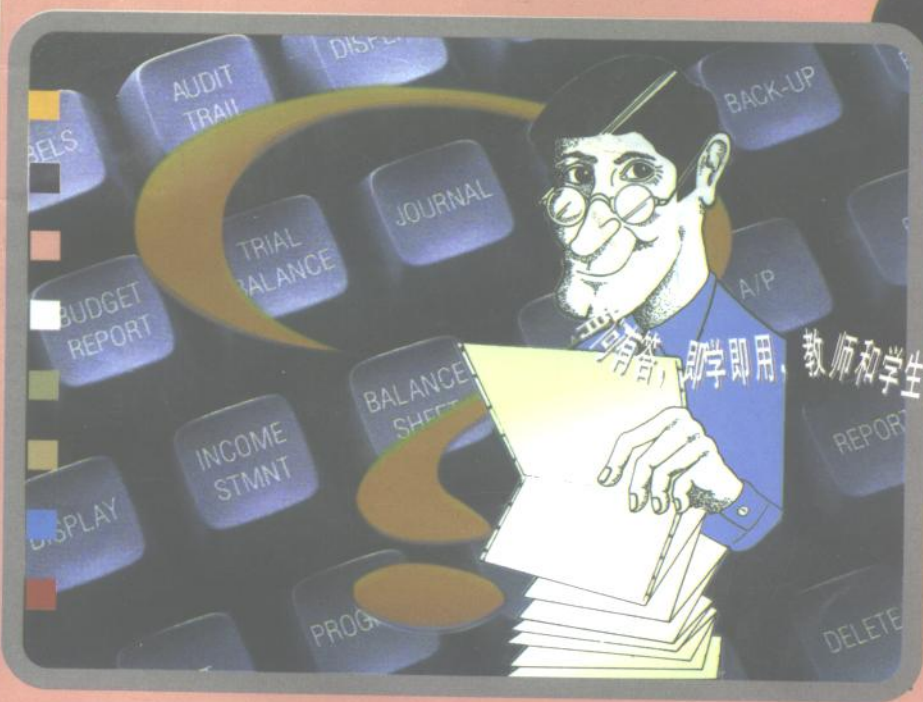


微型计算机 **实用问答** 丛书

吕凤翥 吕涛 编著

C 语言程序设计实用问答



问答即学即用，教师和学生都能使用的参考书

2-44

z/1

北京大学出版社

PEKING UNIVERSITY PRESS

TP312-44
LFZ/1
微型计算机实用问答丛书

C 语言程序设计实用问答

吕凤翥 吕 涛 编著

北京人民邮电出版社

JS362/3833

内 容 简 介

本书作者根据十多年来从事C语言教学的经验,按C语言程序设计的基本内容,将在教学、答疑、批改作业和审阅试卷中收集和整理的带有普遍性的231个问题,通过精心编排,用概念和实例相结合的方式、通俗易懂的语言做了详尽的解答。此外,还整理了125个测试题,并附有答案,供读者自我测验学习效果。

本书内容丰富,具有很强的实用性和可读性,无论是教师或学生,都能从中学到许多有关C语言程序设计的基础知识和实用技巧。本书既可作为初学C语言各类学生和从事C语言程序设计的技术人员的辅导教材,也可作为讲授C语言课程的教师的教学参考书。

图书在版编目(CIP)数据

C语言程序设计实用问答/吕凤翥,吕涛编著. —北京:北京大学出版社,1997.10

(微型计算机实用问答丛书)

ISBN 7-301-03534-9

I. C… I. ①吕… ②吕… III. C语言-程序设计-问答 N. TP312-44

书 名: C语言程序设计实用问答

著作责任者: 吕凤翥 吕 涛

责任编辑: 杨锡林

标准书号: ISBN 7-301-03534-9/TP·375

出 版 者: 北京大学出版社

地 址: 北京市海淀区中关村北京大学校内 100871

电 话: 出版部 62752015 发行部 62559712 编辑部 62752032

排 印 者: 北京飞达印刷厂印刷

发 行 者: 北京大学出版社

经 销 者: 新华书店

787×1029 16开本 14.125印张 350千字

1997年9月第一版 1997年9月第一次印刷

定 价: 21.00元

前 言

随着微机在各行各业的广泛应用,使用 C 语言编程的人越来越多,想学习和掌握 C 语言的人也越来越多,C 语言目前已成为了一种“热门”语言。C 语言与其他高级语言有着相似之处,但又有其自身的特点,一般的初学者总觉得 C 语言难以掌握,学深学透很不容易。为了帮助部分学习 C 语言的读者解决学习中所遇到的问题,作者根据十多年来从事 C 语言教学的经验,收集和整理了各类学生在学习提出的各种问题和考试出现的较为普遍的错误,汇集成此书,提供给初学 C 语言的读者学习参考,希望对读者能有所帮助。

本书共分 10 章。第一至八章是按 C 语言语法的基本内容分为 8 个方面,每章就其某一个方面提出问题并进行解答。第九章收集了前 8 章中没有包含进去的问题和解答。第十章提供了自我测试题,读者可以在看这本书之前或者之后,用所提供的自我测试题检查一下自己对 C 语言掌握的情况,测试题包括 50 个填空题和 75 个分析程序输出结果的题目,并附有答案。

本书是作者教学经验的总结,书中所提出的问题具有一定的代表性和广泛性,这些问题较全面的包含了学习 C 语言时所能遇到的疑难问题,搞清楚这些问题,便可以较好地掌握 C 语言词法、语法的基本概念、基本操作和基本技巧。多数问题是先讲道理,再用实例进行回答的。概念与实例结合得比较密切,有较强的说服力。语言通俗易懂,有较好的可读性。

欢迎读者对本书中存在的缺点和问题提出批评和意见。

作 者

1997.7 于燕北园

目 录

第一章 C 语言概述	(1)
1.1 C 语言的特点	(1)
• 为什么说 C 语言是一种不完全的结构化程序设计语言?	(1)
• 如何理解“C 语言是一种弱类型的语言”?	(1)
• 有人说,“C 语言是一种中级语言”,这是为什么?	(3)
• C 语言语句简练表现在哪些方面?	(3)
• 为什么说 C 语言的移植性强?	(4)
• C 语言本身有哪些不足? 在实际编程中如何克服这些不足?	(5)
1.2 C 语言程序的格式	(6)
• C 语言中,程序、文件、函数三者有何关系?	(6)
• C 语言程序在书写格式上有何特点?	(7)
• 组成 C 语言程序的最小可执行单元是什么? 最小模块单元是什么?	(7)
• 在多个文件的程序中,主函数是否只能有一个?	(8)
• C 语言程序的书写格式与可读性有何关系?	(8)
1.3 C 语言的实现	(9)
• 如何实现一个 C 语言的程序?	(9)
• C 语言程序在编译中的致命错和警告错有什么区别? 编译错和连接错有何不同?	(10)
• 多文件程序如何实现?	(10)
• 主函数带参数的程序如何实现?	(11)
• 如何调试一个 C 语言的程序?	(13)
1.4 C 语言词法及其规则	(13)
• C 语言有哪些单词?	(13)
• 标识符是如何规定的? 它与关键字有何区别?	(14)
• C 语言中常用的分隔符有哪些? 各自的作用是什么?	(14)
• C 语言中空白符的含义是什么? 它有何作用?	(15)
1.5 语言的应用	(15)
• C 语言主要应用在哪些方面?	(15)
• C 语言与 C++ 语言的关系如何?	(15)
第二章 常量、变量、运算符和表达式	(17)
2.1 常量类型和使用方法	(17)
• C 语言中有哪些常量? 如何表示?	(17)
• 符号常量是怎么回事? 为什么要用符号常量?	(17)
• 常量的前缀和后缀表示是如何规定的?	(18)
• 不可打印的字符常量如何表示? \n 和 \\ 表示什么意思?	(18)

• C 语言中浮点数是怎样表示的?	(19)
2.2 变量的名字、类型和值	(19)
• C 语言中变量名字是如何规定的? 变量名必须都用小写字母吗?	(19)
• 给变量赋值和给变量赋初值是一回事吗?	(20)
• C 语言中变量类型有哪些? 变量类型的高低是怎样规定的?	(20)
• 在给数组赋值和赋初值时是否都有“越界”问题?	(21)
• 使用初始值表给数组赋初值时,对数组元素的个数和初始值表中数据项的个数有何要求?	(21)
• 可以通过赋初值将一个字符串常量赋给一个字符数组。能否将一个字符串常量直接赋值给一个字符数组?	(21)
• 在程序中出现没有定义的变量会发生什么现象? 定义过的变量没有赋值和赋初值能否使用?	(22)
2.3 运算符的种类、用法、优先级和结合性	(22)
• 如何记住运算符的 15 种优先级和两种结合性?	(22)
• C 语言中有哪几种逻辑运算符? 由逻辑运算符组成的表达式的值如何表示?	(23)
• C 语言中有哪几种逻辑位运算符? 逻辑位运算符与逻辑运算符有何区别?	(23)
• 增 1 减 1 运算符只能作用在变量上吗? 能否作用在表达式上?	(23)
• 移位运算符中左移和右移补位规则是否相同?	(24)
• 在 C 语言中,“非 0 为真,0 为假;真用 1 表示,假用 0 表示”这句话如何理解?	(24)
• 在 C 语言程序中,如何判断逗号“,”是分隔符还是运算符?	(24)
• 运算符 sizeof 只能作用在类型说明符上吗? 为什么?	(25)
• 三目运算符有何特点? 使用时应注意些什么?	(25)
2.4 表达式的种类和求值方法	(25)
• 如何计算表达式的值和确定表达式的类型?	(25)
• 逻辑表达式在计算值上有什么特点?	(26)
• 条件表达式的类型如何确定?	(27)
• 逗号表达式中由逗号分隔的各个表达式的类型是否要求一致? 如果不一致,逗号表达式的类型如何确定?	(27)
• 在表达式中连续出现两个以上(含两个)运算符时应注意些什么?	(27)
• 书写表达式时应该注意些什么?	(27)
• 表达式的二义性是怎么回事? 如何避免?	(28)
• 在 $!5$ 和 $!-5$; ~ 5 和 ~ -5 两组表达式中,每组的表达式的值相同吗?	(29)
• 已知: $\text{int } a=8$; 请分析表达式 $a \gg 2$ 和 $a \gg= 2$ 计算后, a 的值是否相同? 为什么?	(29)
2.5 类型转换	(29)
• C 语言中有哪些类型转换的机制?	(29)
• 一个已定义的变量经过强制类型转换后,它的类型是保留原来的类型还是被转换的类型?	(30)
• 能否将一个 int 型数值赋给 float 型变量? 又能否将一个 float 型数值赋给 int 型变量?	(31)

• 在一个表达式中,有几种不同类型的变量如何处理? 表达式 $5.76 < 3$ 和 $64 > 'A'$ 是合法的吗?	(31)
第三章 语句	(32)
3.1 表达式语句、空语句和复合语句	(32)
• 表达式和表达式语句有何不同?	(32)
• 什么是空语句? 它有何作用?	(32)
• 什么是复合语句? 它与分程序有何不同?	(33)
• 在 C 语言程序中复合语句有何作用?	(33)
3.2 分支语句	(34)
• 在 if 语句嵌套的程序段中,如何判断 else 是属于哪个 if 的?	(34)
• 在 if 语句中,if 条件和 else if 条件只能用逻辑表达式和关系表达式吗?	(35)
• 在 if 语句中,else 短语和 else if 短语的个数是否有限制?	(35)
• 写出 switch 语句的格式,并说明该语句是如何执行的?	(36)
• 在 switch 语句中,关键字 default 出现的位置是否有所规定?	(37)
• 在 switch 语句中,break 语句起什么作用?	(39)
• 在多路分支的情况下,是选择 if 语句还是选择 switch 语句的主要依据是什么?	(39)
3.3 循环语句	(40)
• C 语言中规定使用哪些循环语句? 这些循环语句各自的特点是什么?	(40)
• while 循环和 do-while 循环的区别是什么?	(41)
• 当 while 循环条件是赋值表达式时,会出现什么现象?	(42)
• 写出 for 循环语句的格式,并说明该语句是如何执行的?	(42)
• 任何一个 for 循环语句都可以写成 while 循环的格式吗?	(44)
• 如何判断 for 循环中循环体的执行次数?	(45)
• 如何判断循环结束? 当循环条件为真时如何退出循环?	(46)
• break 语句和 continue 语句在循环体中起什么作用?	(47)
• 循环体如何判界? 循环体为空语句意味着什么?	(48)
• 同一种格式的循环可以嵌套,不同种格式的循环能否嵌套?	(48)
• 在开始无法确定循环次数的情况下,应选用哪种循环? 用 for 循环是否可以?	(49)
• 在循环嵌套使用时,应注意些什么问题?	(50)
3.4 转向语句	(51)
• C 语言提供哪些转向语句? 它们各自的特点是什么?	(51)
• C 语言中对使用 goto 语句有何限制?	(52)
• 如何定义语句标号? 它有何作用?	(52)
• 返回语句 return 在一个函数体内必须使用吗? 且只能使用一次吗?	(53)
第四章 函数和存储类	(55)
4.1 函数的定义和说明	(55)
• 怎样定义一个函数? 函数的定义和说明有何不同?	(55)
• 所有的函数是否在调用之前都必须说明?	(56)
• 如何说明函数? 函数的简单说明和原型说明有何区别?	(56)

• 函数的调用可以嵌套,函数的定义能否嵌套?	(57)
• 函数定义中的函数类型与参数类型有何关系? 函数类型与返回值类型有何关系?	(57)
4.2 函数的参数和返回值	(58)
• 函数的实参和形参分别指的是什么?	(58)
• 一个有返回值的函数,调用时不使用其返回值会出现什么现象?	(58)
• 一个函数的类型为 double 型,而它的返回值表达式的类型为 int 型,这样定义的函数有问题吗?	(58)
• 用 void 类型定义的函数意味着什么? void 型函数调用之前是否一定要说明?	(58)
• 函数定义时不作任何类型说明,这种定义意味着什么? 这种函数在调用之前是否一定要说明?	(58)
• 调用一个函数时是否能获得多个返回值? 在一个函数中是否可出现多个带有返回值表达式的 return 语句?	(59)
• 主函数 main()带有参数意味着什么? 对主函数的参数有何规定?	(60)
4.3 函数的调用方式	(61)
• C 语言中有哪些不同的传值调用形式? 它们有何区别?	(61)
• 函数的传值调用中,对形参和实参有何要求? 如果不满足会出现什么现象?	(63)
• 什么是函数的嵌套调用? 举例说明.	(63)
• 什么是函数的递归调用? 举例说明.	(64)
• 一个函数的若干个参数中,有的采用传递变量值的形式,有的采用传递变量地址值的形式. 这样使用是否可以?	(65)
4.4 作用域	(66)
• C 语言中标识符的作用域是怎样规定的? 具体指出语句标号、变量和函数的作用域是如何规定的?	(66)
• 在一个变量的作用域内,使用分程序对该变量进行重新定义后,新定义的变量与原变量在作用域上有何关系?	(67)
• 作用域和寿命有何关系? 某个变量在其作用域外是否还可以存在?	(68)
4.5 存储类	(68)
• 存储类的含义是什么? 不同存储类的变量有何不同?	(68)
• C 语言中,哪些变量属于全局变量? 哪些变量属于局部变量? 函数参数属于哪类变量?	(68)
• 自动类变量和内部静态类变量在作用域和寿命上有何不同?	(68)
• 寄存器类变量有何特点? 使用时应注意些什么?	(69)
• 外部静态变量和内部静态变量在作用域上有何不同? 在定义或说明上有何区别?	(70)
• 外部变量的定义和说明是不是一回事? 何时需要说明外部变量? 如何说明?	(71)
• 自动类变量在赋值或赋初值之前不能使用,为什么? 其他存储类变量是否也如此?	(72)
• 外部函数和内部函数是如何规定的?	(73)
• 在一个多文件的程序中,如何定义一个只被某个文件引用的变量和函数?	(73)
• 外部变量用作函数之间信息传递有何弊病?	(73)
第五章 预处理和类型定义	(74)

5.1	宏定义	(74)
	• 什么是预处理? 预处理包含哪些内容? 预处理命令有何特点?	(74)
	• 什么是宏定义? 宏定义和宏替换二者有何关系?	(74)
	• 宏定义中宏名必须用大写字母吗?	(75)
	• 宏定义的作用域是怎样规定的?	(75)
	• 宏定义中的嵌套是怎么回事?	(76)
	• 不同编译系统中宏替换的结果是否相同?	(76)
	• 为什么说宏定义中正确运用括号是十分重要的?	(77)
	• 使用宏定义和宏替换时最容易犯的错误是什么?	(78)
	• 带参数的宏定义与函数在形式上很相似,但二者有何区别?	(79)
	• 在编程中,有些功能既可用宏定义实现又可用函数实现,这时应该如何选择?	(80)
5.2	文件包含和文件编译	(81)
	• 什么是文件包含? 文件包含只能包含头文件(.h)吗?	(81)
	• 如何建立.h文件? 建立.h文件时应注意些什么?	(82)
	• 文件包含的嵌套是什么意思?	(82)
	• 文件包含在程序设计中有何作用?	(83)
	• 什么是条件编译? 它有什么用途?	(83)
	• 常用的条件编译的格式有哪几种?	(83)
5.3	类型定义	(85)
	• “类型定义不是定义新的类型”这句话如何理解?	(85)
	• 类型定义有何用处?	(85)
	• 类型定义是否允许嵌套?	(86)
	• 宏定义与类型定义有何不同?	(86)
第六章	指针	(89)
6.1	指针的定义、赋值和运算	(89)
	• 指针是一种变量,它与一般变量的区别是什么?	(89)
	• 如何给指针赋值或赋初值? C语言中变量地址值如何表示?	(89)
	• 指针被定义后不赋值为什么不能使用?	(90)
	• 给一个指针赋值为NULL,这表示什么意思?	(91)
	• 不同类型的指针能否相互赋值?	(91)
	• 不同类型的指针所占内存的大小是否相同?	(91)
	• 指向数组的指针、指向函数的指针和指向指针的指针如何表示? 如何赋值?	(92)
	• 指针本身的值和指针所指向的变量的值是否相同?	(93)
	• 指针的运算有哪些? 任意两个指针做相减运算是否有意义?	(93)
	• “指针运算实际上是地址运算,但是两者又不相同”这句话如何理解?	(95)
6.2	指针和数组	(95)
	• C语言中规定数组名是一个指针,这种指针与一般指针有何不同?	(95)
	• 指向数组元素的指针与指向数组的指针相同吗? 为什么?	(96)
	• C语言中数组既可用下标方式表示又可用指针方式表示,这两种方式如何选择?	(97)

• 写出一维数组、二维数组和三维数组的指针和下标的各种表示形式。	(97)
• 在 <code>int a[2][3]</code> 中, <code>a+1</code> 的含义与 <code>a[1]</code> 的含义是否相同?	(98)
• 什么是指针数组? 如何给它赋值?	(99)
• 在 <code>int *p1[2], (*p2)[3]</code> 中, <code>p1</code> 和 <code>p2</code> 有何不同?	(99)
• 在 <code>int a[] = {1, 2, 3}, *p = &a[1]</code> 中, <code>*p</code> 的值与 <code>a[1]</code> 的值相等吗? <code>a[0]</code> 的值和 <code>p[0]</code> 的值相等吗? <code>p[0]</code> 的值与 <code>*(a+1)</code> 的值相等吗?	(100)
• 在 <code>int a[][2] = {1, 2, 3, 4, 5, 6}, *p[3]</code> 中, <code>p=a</code> ; 这样赋值是否可以? <code>p[0]=a</code> ; 这样赋值是否可以? 如果 <code>p[1]=a[2]</code> ; 后, <code>*(*(p+1)+1)</code> 的值是多少?	(100)
6.3 字符指针和字符串	(101)
• 什么是字符指针? 为什么要规定字符指针?	(101)
• 什么是字符串? 一维字符数组就是字符串吗?	(102)
• 在 <code>char *p="abcdef"</code> 中, 如何使用 <code>printf()</code> 函数输出该字符串、串中某一字符和某字符的 ASCII 码值?	(102)
• 在 <code>char b[] = {"abc", "mnp"}</code> 中, <code>b[0]</code> 的含义是什么? <code>printf("%s\n", b[0])</code> ; 输出结果是什么? <code>b[1][2]</code> 的含义是什么? <code>printf("%c\n", b[1][1])</code> ; 输出结果是什么?	(103)
• 已知 <code>char *s1="123", *s2="456", *s[2], *p</code> ; 又知 <code>s[0]=s1, s[1]=s2, p=*s</code> ; 试写出 <code>printf("%c\n", *p)</code> ; <code>printf("%s\n", p+1)</code> ; <code>printf("%s\n", s[1]+2)</code> ; 各自的输出结果?	(104)
• 在判断两个字符串 <code>s1</code> 和 <code>s2</code> (<code>s1</code> 和 <code>s2</code> 为字符指针) 相等时, 使用表达式 <code>s1==s2</code> 是否可以?	(105)
• 在使用 <code>strcat(s1, s2)</code> 函数时, 如果第一个参数 <code>s1</code> 的字符数组不够大时会发生什么现象?	(105)
6.4 指针与函数	(106)
• 用指针作函数形参实现传址调用中, 当在被调用函数中改变形参时, 实参一定被改变吗?	(106)
• 指向数组的指针作函数参数与数组名作函数参数其作用相同吗?	(107)
• 什么叫指针函数? 它与指向函数的指针在说明中有何不同?	(108)
• 怎样给一个指向函数的指针赋值? 试举例说明。	(109)
• 指向函数的指针作函数参数时, 如何调用指针所指向的函数?	(109)
第七章 结构、联合和枚举	(111)
7.1 结构变量的基本概念	(111)
• 如何定义结构变量? 结构变量与结构有何关系?	(111)
• 定义一个结构时对其成员的类型是否有所限制? 结构变量可否作结构成员?	(112)
• 结构名、结构成员名和结构变量名三者有何关系?	(113)
• 结构变量的成员和指向结构变量指针的成员在表示上有何不同?	(113)
• 结构变量成员为数组和结构变量为数组元素有何不同?	(114)
• 结构的定义可以嵌套, 在嵌套时有何规定?	(115)
• 结构变量的赋值与数组赋值有何相同之处?	(115)
• 指向结构变量的指针赋值与指针赋值是否相同?	(116)

• 结构变量都有哪些运算? 它与数组的运算有相似之处吗?	(116)
7.2 结构的应用	(117)
• 举例说明结构与数组的关系,数组作结构成员和结构变量作数组元素。	(117)
• 结构变量作函数参数和指向结构变量的指针作函数参数有何不同?	(119)
• 较复杂的结构关系如何用图解法表示关系帮助理解?	(120)
• 什么是单向链表和双向链表? 举一个建立并显示单向链表的例子。	(122)
• 链表有哪些基本操作? 举例说明链表的插入和删除操作方法。	(124)
• 什么是位段? 为什么要引进位段?	(128)
7.3 联合和枚举	(129)
• 什么是联合? 它有何特点? 它与结构有什么区别?	(129)
• 如何验证联合变量的若干成员是共用内存单元的?	(130)
• 能否给联合变量赋初值? 试举例说明。	(131)
• 为什么说“枚举是若干个有名字的整型常量的集合,枚举变量的取值范围受到其枚举表中枚举符的限制”?	(132)
• 枚举符的值是如何确定的? 输出枚举变量的值是什么类型的?	(132)
• 能够将一个整数值直接赋给一个枚举变量吗? 能够直接输出枚举变量的枚举符吗?	(133)
• 枚举有何用途?	(134)
第八章 文件和读写操作函数	(135)
8.1 文件的概念	(135)
• C 语言文件有何特点?	(135)
• 标准文件和一般文件有何区别?	(135)
• 如何定义文件指针? 它是怎样获得的?	(136)
• 文件打开后为什么要及时关闭?	(136)
8.2 标准文件的读写操作	(137)
• 常用的标准文件的读写函数有哪些?	(137)
• getchar()函数与 getch()函数有何区别?	(139)
• scanf()函数有哪些参数? 该函数的返回值是什么?	(139)
• printf()函数有哪些参数? 该函数的返回值是什么?	(141)
• scanf()函数和 printf()函数中格式符的修饰符 * 有何不同?	(142)
• scanf()函数和 printf()函数中格式符 %md 的含意有何区别?	(142)
• scanf()函数的输入信息流中的输入项如何确定?	(143)
• scanf()函数输入数据时,常用的三种输入格式是什么?	(143)
• scanf()函数在输入信息流中遇到非法输入字符如何处理?	(144)
• scanf()函数中字符格式(%c)和字符串格式(%ls)有何不同?	(145)
• 在连续使用 scanf()函数时,缓冲区内的剩余字符有何影响? 应如何处理?	(145)
• 用 printf()函数的格式符 %u 能够输出一个负整数吗?	(147)
• 在 printf()函数的格式说明符中能否用大写字母? 举例说明用大写字母的含意	(148)
• 在 printf()函数中格式符的修饰符用 # 表示何意?	(148)

8.3 一般文件的读写操作	(149)
• 打开文件的方式有哪些?	(149)
• 常用的一般文件的读写函数有哪些?	(149)
• 使用 fopen() 函数打开一个文件时, 该文件的读写指针一定指向文件头吗?	(154)
• fgets() 函数有几个参数? 该函数有哪些结束条件?	(154)
• 使用 EOF 和函数 feof() 判断文件结束有何区别?	(154)
• 如何使用 fseek() 函数来定位文件读写指针? 如何确定文件读写指针的位置?	(155)
• 如何使用 fwrite() 和 fread() 函数对结构数组进行操作?	(156)
第九章 其他问题	(158)
• 什么是 const 指针? 它有什么用处?	(158)
• 什么是左值? 什么是右值? C 语言中表达式可以作右值吗?	(158)
• 两个动态分配内存空间函数: malloc() 和 calloc() 有何不同?	(159)
• far 指针与 near 指针有何区别?	(159)
• 栈(stack)和堆(heap)有何不同?	(160)
• 如何在指定的整数范围内产生一个随机数?	(160)
• C 语言中变量的寿命有哪几种类?	(161)
• 怎样在一个程序的执行期间去运行另一个程序?	(161)
• 怎样知道一些数字型变量的最大值和最小值?	(162)
• 使用 enum 关键字来定义常量比用 #define 来定义常量有何好处?	(163)
• 如何选择向屏幕上写数据的输出函数?	(163)
• C 语言程序中, 大括号的常用写法有哪几种?	(164)
• 什么是空循环? 什么是死循环? 二者有何区别?	(166)
• 指针能作为 if 语句的条件表达式吗?	(167)
• 如果一个函数的参数很多时, 采用什么方法更为合适?	(167)
• C 语言编译系统允许使用哪些种存储模式? 应该如何选用存储模式?	(168)
• 如何避免程序用尽内存?	(169)
• 在连接过程中出现“DGROUP, group exceeds 64 KB”信息该怎么处理?	(169)
• exit() 函数和 return 语句作用上有何不同?	(170)
• 选用什么方法来调试程序?	(170)
第十章 自我测试题	(172)
10.1 选择填空 50 题	(172)
10.2 分析程序输出结果 75 题	(180)
10.3 自我测试题答案	(209)

第一章 C 语言概述

C 语言是一种具有鲜明特点的结构化程序设计语言。由于这种语言具有许多优点,为编程提供了方便,因此人们比较喜欢使用这种语言。那么,C 语言到底有些什么特点?学习这种语言首先应该掌握哪些语法规则呢?请看下面的回答。

1.1 C 语言的特点

问题:为什么说 C 语言是一种不完全的结构化程序设计语言?

解答:首先要看到 C 语言是一种结构化程序设计语言,它具有结构化程序设计语言所必须具有的三种基本的结构模式:顺序结构、分支结构和循环结构。因此,可以使用 C 语言编写出结构化的程序。结构化程序设计方法具有易理解、易控制、易调试、易修改等优点,所以程序员都喜欢使用结构化程序设计方法进行程序设计。

有人说 C 语言是一种不完全的结构化程序设计语言是指:首先肯定它是一种结构化程序设计语言,同时又指出它与较严格较规范的结构化程序设计语言有不同之处。例如,它与 PASCAL 语言相比较,二者都是结构化程序设计语言,而 PASCAL 语言要比 C 语言规范得多,严格得多。正是由于 C 语言有些不够严格、不够规范的地方,因此,才说它是一种不完全的结构化程序设计语言。

C 语言对其结构化的不完全性主要表现在如下两个方面:一是严格的结构化程序设计语言中是不允许使用 goto 无条件转向语句的,因为 goto 语句会破坏结构化。例如,PASCAL 语句中就禁止使用 goto 语句。C 语言中允许使用 goto 语句,但是转向范围受到限制,即在 C 语言中规定限制使用 goto 语句。在使用 goto 语句时,转向范围只能在该函数体内。这就是说,不允许 goto 语句从一个函数体内转向到另一个函数体,只允许在一个函数体内转向。在实际编程中,应该尽量少用 goto 语句,最好不使用它,即使不会影响结构化,也会降低可读性。一般说来,goto 语句仅用在与 if 语句形成循环语句和在多重循环中从内重循环一次退出到最外重循环。二是严格的结构化程序设计语言中的结构化模块只允许有一个入口和一个出口。而 C 语言中允许一个函数(函数是 C 语言中最基本的结构化模块)有多个出口。即允许一个函数中可以使用多个返回语句 return,因此会出现多个出口。

上述两点是 C 语言的结构化不完全的表现,不难看出这也正是 C 语言灵活性的表现。C 语言为了使用起来灵活方便,而对于结构化中的某些规定有所“突破”。这也正是 C 语言的一个特点。了解了这一点有利于使用它进行灵活的编程。

问题:如何理解“C 语言是一种弱类型的语言”?

解答:我们说 C 语言是一种弱类型的语言,是指它与 PASCAL 语言相比较在数据类型方

面不如 PASCAL 语言那样严格,而有些类型可以自动转换,特别是在有些情况下,类型不一致也不报错,因此我们说 C 语言中类型比较弱,故称为弱类型语言。

C 语言中数据类型较弱表现在有些类型可以自动转换。例如, int 型和 char 型之间可以自动转换,这就是说,在一定条件下, int 型可以转换成 char 型, char 型可以转换成 int 型。这就为在 C 语言中可以实现一个字符常量加上或减去一个整数,或者两个字符常量可以相减等运算。下列表达式在 C 语言中是合法的:

```
'a'+2;  
'x'-5;  
'a'-'A';等等。
```

这样一些规定为某些操作带来了方便。例如,将小写字母转换为大写字母,可用如下表达式,设 c 是一个字符变量,将它转换为大写字母:

```
c-'a'+'A';
```

又例如, C 语言中, float 型在机器中一律转换成 double 型,这无疑将提高精确度。还有 short int 型自动转换成 int 型; unsigned char 型和 unsigned short 型自动转换成 unsigned int 型等等。

这些自动转换一方面给 C 语言编程带来方便,可以实现某些在其他语言中难以实现的操作,另一方面也将带来一些麻烦。由于 C 语言允许某些类型之间自动转换,因此在许多情况下, C 语言编译系统将不作类型检查,一旦出现了类型错也不报错,这必将给编程带来一定的麻烦。下面的一个例子,你将会从中看到不进行严格的类型检查所带来的麻烦。

```
main()  
{  
    int a=5;  
    float b=5.5;  
    printf("%d, %.2f\n", b, a);  
}
```

执行该程序输出结果为

```
0,0.00
```

造成莫名其妙的结果的原因在于类型的不一致性所致。因为 printf() 函数的控制串中两个格式符分别是 %d 和 %.2f, 它要求对应的表达式的类型是 int 型和 float 型,而实际上程序中给出的却是 float 型和 int 型。因此,类型不一致。而编译时又不作类型检查,所以导致上述的错误。

可见 C 语言的弱类型给编程带来了方便性和灵活性,同时也带来了一些麻烦。因此,在用 C 语言编程时,要注意类型的一致性问题。例如,在函数调用时,要求形参和实参的类型一致,一般情况下,出现了不一致,也不报错,只是在结果上出问题。如何克服 C 语言弱类型带来的麻烦,一是要注意类型的一致性问题,二是要在必要时采用强制类型的措施,即使用强制类型运算符,三是选用一些进行类型检查的方法,例如,用原型进行函数说明时,调用函数则对形参和实参进行类型检查,用类型定义 (typedef 语句) 定义的类型,使用时进行类型检查等等。

问题:有人说,“C 语言是一种中级语言”,这是为什么?

解答:“C 语言是一种中级语言”一语关键在于搞清楚这里所指的“中级语言”的含义。C 语言是一种高级语言,这一点是毫无疑问的,C 语言又是一种低级语言,这一点下面进行解释,在这个意义上来讲,有人说 C 语言是中级语言,因为它具有高级语言和低级语言两方面的特点。实际上是说中级语言比高级语言更高级。

C 语言到底具有哪些低级语言的特点呢?第一,C 语言具有二进制位运算的功能,即可对二进制数进行逻辑位操作(包括按位与、按位或、按位求反、按位异或)和移位操作(包含左移和右移)。位操作功能使得 C 语言扩大了应用范围。第二,C 语言具有寄存器操作功能,通过 C 语言编程可实现用 CPU 通用寄存器存取数据,进而大大的提高执行速度。C 语言提供了一种寄存器型的存储类,定义为寄存器类的变量,运行时有可能被存放到 CPU 的通用寄存器中,一旦被放到寄存器中,存取的速度会比存放在内存中提高很多。因此,人们在编程中常常将一些使用频度较高的变量定义为寄存器类型的,以求提高执行速度。第三,C 语言具有地址操作的功能。地址操作是指对内存的地址进行操作。由于 C 语言引进了指针这一概念,指针是一种用来存放某种变量地址值的变量。因此,对指针的运算实际上就是对地址值的运算,但是两者还是有不同之处的。指针的使用给构造的数据类型的变量的操作带来了很大的方便。上述三点是属于低级语言中的功能,而 C 语言都具备。

由于 C 语言既具有高级语言的特点又具有部分低级语言的功能,而使 C 语言不仅可用来编写应用软件,又可用于编写系统软件。例如,大家所熟悉的 UNIX 操作系统的大部分程序是用 C 语言编写的;又例如,当前流行的很多数据库系统的源程序和图形系统的源程序都是用 C 语言编写的。

问题:C 语言语句简练表现在哪些方面?

解答:语句简练是 C 语言的一大特点。C 语言语句简练表现在如下几个方面:

(1) C 语言中类型说明符使用简写字符,例如,整型说明用 `int`,字符型说明用 `char` 等,这与 PASCAL 语言相比更简单些。

(2) C 语言中语句简练的重要表现在有许多功能较强的运算符,使得语句简练。例如,计算前十个自然数的和,可用如下程序:

```
main()
{
    int i=1,s=0;
    while(i<=10)
        s+=i++;
    printf("%d\n",s);
}
```

该程序中使用 `s+=i++`;表达式求和值,这里用到 `++` 运算符,该运算符具有改变变量值的功能,使 `i` 每次增 1,这种功能较强的运算符使得语句简练。

又例如,已知两个数,求其最大者,可使用下述程序:

```

main()
{
    int a,b,max;
    a=5;
    b=8;
    max=a>b?a:b;
    printf("%d\n",max);
}

```

该程序中使用了三目运算符组成的条件表达式 $a > b ? a : b$ 来求两个数中最大者,该条件表达式相当于一个 if-else 语句;

```

if (a>b)
    max=a;
else
    max=b;

```

可见使用 C 语言所提供的功能较强的运算符会使语句简练,程序短小。

(3) C 语言还提供预处理功能使得书写简练,预处理功能主要包含有宏定义、文件包含和条件编译。适当的使用预处理命令会使 C 语言的程序语句简练。例如,下面程序中使用了宏定义使得程序简练。

```

#define PRD(x) printf("%d\n",x)
main()
{
    int a,b,c;
    a=5;
    b=a+3;
    c=a*b;
    PRD(a);
    PRD(b);
    PRD(c);
}

```

该程序中使用了宏定义,将一个较复杂的语句定义成一个简洁的标识符(宏名),这样在书写时就变得简单明了,在编译前将被替换。

问题:为什么说 C 语言的移植性强?

解答:移植性强又是 C 语言的一大特点。C 语言的移植性强是指 C 语言的编译系统可装载在各种大型、中型、小型和微型计算机上和用 C 语言编写的源程序可运行在不同系统下,有时只需稍加修改。

C 语言系统的移植性强是因为 C 语言的编译系统用少于 1 万条代码写成,因此便于修改。

因此,它可以比较方便地移植到各类机器上去,据统计资料表明,不同机器上的C编译程序80%的代码是公共的。

用C语言编写的源程序可在不同系统下运行,一般只是稍加修改就可以了,其原因有以下几点:

(1) C语言提供预处理功能可将与机器相关的因素与主代码分开,在用于不同机器时只修改与机器相关部分的代码即可,因此为移植提供了方便。

(2) C语言编译系统本身不提供输入输出功能,而输入输出库函数是由支持它的操作系统提供的,这也为移植带来方便。

(3) C语言的语义性不强,一些解释留给具体实现去规定。

问题: C语言本身有哪些不足?在实际编程中如何克服这些不足?

解答: C语言具有功能性强、语句简练、编程方便灵活、移植性好等特点,是人们喜欢使用的一种结构化程序设计语言。但在使用C语言的过程中,人们也会发现该语言还有一些不足,不认识这些不足往往在编程中会出现一些莫名其妙的问题。C语言本身存在的不足有如下几点:

(1) C语言运算符多,使用灵活。但是,存在有难记、难用的不足。标准C语言具有40多种运算符,分为15个优先级,两类结合性。同一种运算符在不同的条件下具有完全不同的功能。例如,星号(*),它作为单目运算符,表示取内容,即取其后面跟的地址值所指向的变量的值;它作为双目运算符,表示乘法操作,即求两个操作数之积。另外,星号与斜线符还可组成注释符(/*, */)等。有的符号既可作为运算符又可作其他符号用。例如,星号(*),前面讲过,它可作运算符使用,又可作说明符使用,在说明语句中,出现在变量名前面的星号表示该变量是指针。又例如,逗号(,)可以是运算符,用它将若干个表达式连成一个逗号表达式,还可以是分隔符,用来作多个变量或多个参数的分隔符。因此,初学C语言时首先要在运算符上下功夫,要记住它们的功能,又要分清它们的区别。

(2) C语言是一种弱类型语言,这为C语言编程提供了方便性和灵活性。但是,C语言对类型检查不够严格,在一些要求类型一致的情况下,出现类型不一致时不予报错,而进行“自动”转换,造成数据的混乱。因此,在编程中特别要注意使数据类型保持一致。另外,尽量采用检查类型的措施,保证数据的安全性。有关细节请参见问题:如何理解“C语言是一种弱类型语言”?

(3) C语言与其他高级语言一样都具有数组这一数据类型。C语言在定义数组和使用数组时都很方便。但在给数组赋值时要注意C语言不作越界检查,即使超出了数组的上界后,仍然可以赋值,这样一来便造成数据的混乱,有些有用的数据可能被冲掉。这一点在编程时应该注意,在定义数组时要开辟足够大的空间以防越界。另外,提醒注意,在给数组赋初值时不会出现越界现象,因此,建议对数组多用赋初值的方法。

(4) C语言中引用一些具有副作用的运算符,其目的是为了增强运算符的功能,使得语句简练,但是这种有副作用的运算符在有些情况下会产生二义性。

由于C语言编译系统允许改变表达式中各个操作数的计算顺序和参数表中各个参数的计算顺序。这样规定有利于编译中实现优化。这对于没有副作用的运算符来讲,改变计算顺序将不会影响表达式的值。例如,在表达式 $a+b$ 中,先计算 a ,后计算 b ,与先计算 b ,再计算 a ,其