

可下载教学资料

<http://www.tup.tsinghua.edu.cn>

21世纪普通高校计算机公共课程规划教材

C语言程序设计

陈明晰 谢蓉蓉 编著

清华大学出版社

21 世纪普

共课程规划教材

C语言程序设计

陈明晰 谢蓉蓉 编著

清华大学出版社
北京

内 容 简 介

本书以程序设计为主线,系统介绍C语言及其程序设计技术。全书共10章,主要包括算法及C语言程序设计的初步知识、数据类型与表达式、顺序结构程序设计、选择结构程序设计、循环结构程序设计、数组、函数和编译预处理、指针、结构体与共用体、文件、C语言上机实验等内容。

本书内容全面,章节安排由浅入深、注重实践,各章均安排了适量的习题,并将上机实验内容编入书中,适合作为高校“C语言程序设计”课程的教材,还可作为全国计算机等级考试的参考用书。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

C语言程序设计 / 陈明晰, 谢蓉蓉编著. —北京: 清华大学出版社, 2013.1

21世纪普通高校计算机公共课程规划教材

ISBN 978-7-302-30635-1

I. ①C… II. ①陈… ②谢… III. ①C语言-程序设计-高等学校-教材 IV. ①TP312

中国版本图书馆CIP数据核字(2012)第272498号

责任编辑: 郑寅堃

封面设计: 常雪影

责任校对: 白 蕾

责任印制: 沈 露

出版发行: 清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址: 北京清华大学学研大厦A座 邮 编: 100084

社总机: 010-62770175 邮 购: 010-62786544

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

课 件 下 载: <http://www.tup.com.cn>, 010-62795954

印 装 者: 三河市金元印装有限公司

经 销: 全国新华书店

开 本: 185mm×260mm 印 张: 19.25 字 数: 457千字

版 次: 2013年1月第1版 印 次: 2013年1月第1次印刷

印 数: 1~3000

定 价: 33.50元

产品编号: 049263-01

前 言

程序设计是高等院校重要的基础课程之一。根据教育部高等学校计算机基础课程教学指导委员会提出的《关于进一步加强高校计算机基础教学的意见》精神,“程序设计基础”课程一般定位为各专业大学生第二门计算机公共基础课,通过该课程的学习,一是让学生掌握一种高级程序设计语言;二是了解程序设计的思想和方法,培养程序设计的能力。

C 语言是国内外广泛使用的一种面向过程的程序设计语言,它不仅具有丰富的数据类型与运算符、灵活的控制结构、简洁高效的表达式、清晰的程序结构和良好的可移植性等优点,而且还具有直接对计算机硬件操作的强大功能;既具有高级语言的优点,又有低级语言的特点;生成目标代码质量高,执行速度快也是其特点之一。C 语言的卓越性能,使它流行于全世界,成为最受欢迎的程序设计语言之一。当今流行的面向对象语言 C++以及 Internet 上的通用语言 Java 就是在 C 语言基础上发展起来的。

本书以程序设计为主线,全面、系统地介绍 C 语言及程序设计的基础知识。全书共分 10 章,包括程序设计基础,数据类型和表达式,顺序、选择和循环结构程序设计,数组,函数和编译预处理,指针,结构体与共用体,文件,C 语言上机实验等内容,将 C 语言上机实验内容作为第 10 章,主要为了方便教学。本书在编写过程中参考了大量同类教材,结合了作者多年从事程序设计教学和研究的经验,内容编排由浅入深、循序渐进、通俗易懂,通过大量的例题介绍 C 语言程序设计的基本方法与技巧,通过习题加深对 C 语言的掌握,训练学生的程序设计技能,是一本适合初学程序设计的人员学习 C 语言的书籍,还可作为普通高等院校非计算机专业“C 语言程序设计”课程的教材。

本书由陈明晰、谢蓉蓉编写。刘天时教授对全书做了审阅,并提出了许多宝贵的意见,在此表示衷心的感谢。

本书在编写过程中参阅了大量的其他参考文献、资料,在此对它们的作者表示衷心的感谢。由于编者水平有限,加之时间仓促,书中不当之处在所难免,恳请读者批评指正。

编 者
2012 年 10 月

目 录

第 1 章 C 语言程序设计概述	1
1.1 算法与程序设计	1
1.1.1 算法	1
1.1.2 程序	3
1.1.3 程序设计语言	4
1.1.4 程序设计的一般过程	4
1.2 C 语言发展历史和特点	5
1.3 C 语言程序的结构	7
1.3.1 C 语言程序的一般形式	7
1.3.2 C 程序中的主要成分	8
1.4 C 语言上机环境及操作步骤	10
1.5 用流程图表示算法	14
1.6 结构化程序设计简介	15
1.6.1 结构化程序	15
1.6.2 结构化程序设计方法遵循的原则	15
习题 1	17
第 2 章 数据类型与表达式	20
2.1 C 语言的数据类型	20
2.2 常量和变量	21
2.2.1 常量和符号常量	21
2.2.2 标识符与变量	21
2.3 整型数据	23
2.3.1 整型常量	23
2.3.2 整型变量	24
2.4 实型数据	26
2.4.1 实型常量的表示方法	26
2.4.2 实型变量	27
2.4.3 双精度型数据	28
2.5 字符型数据	28
2.5.1 字符常量	28

2.5.2	字符变量	30
2.5.3	字符串常量	31
2.6	系统函数	32
2.6.1	简例	32
2.6.2	常用数学函数	33
2.7	C 运算符概述	34
2.8	算术运算符	34
2.8.1	基本的算术运算符	34
2.8.2	算术表达式和运算符的优先级与结合性	35
2.8.3	自加、自减运算符	36
2.8.4	类型转换运算符及类型转换	37
2.9	关系运算符和逻辑运算符	38
2.9.1	关系运算符	38
2.9.2	逻辑运算符	39
2.9.3	条件运算符	40
2.10	位运算符与长度运算符	41
2.10.1	原码、反码和补码	41
2.10.2	移位运算符	42
2.10.3	位逻辑运算符	43
2.10.4	求长度运算符	44
2.11	赋值运算符和赋值表达式	45
2.11.1	赋值运算符和赋值表达式	45
2.11.2	类型转换问题	45
2.11.3	算术复合赋值运算符	46
2.11.4	位复合赋值运算符	46
2.11.5	赋值运算符的优先级与结合性	46
2.12	逗号运算符和逗号表达式	47
2.13	运算符的优先级与表达式的分类	48
2.13.1	运算符的优先级	48
2.13.2	C 表达式的分类	48
习题 2		49
第 3 章	简单的 C 程序设计	54
3.1	C 语句概述	54
3.2	赋值语句和表达式语句	54
3.2.1	赋值语句	54
3.2.2	表达式语句	56
3.3	格式化输入/输出	56
3.3.1	printf() 函数	56

3.3.2	scanf()函数	60
3.4	字符数据的输入/输出	63
3.4.1	putchar()函数	63
3.4.2	getchar()函数和 getch()函数	64
3.5	顺序结构程序举例	65
习题 3		66
第 4 章	控制结构程序设计	71
4.1	if 语句和条件运算符	71
4.1.1	if 语句	71
4.1.2	if 语句的嵌套	73
4.1.3	条件运算符	75
4.2	switch-case 语句	76
4.3	while 循环语句	79
4.4	do...while 循环语句	83
4.5	for 循环语句	86
4.6	三种循环比较及循环嵌套	90
4.6.1	三种循环比较	90
4.6.2	循环嵌套	91
4.7	标号语句与 goto 语句、break 语句和 continue 语句	93
4.7.1	标号语句与无条件转移语句 goto	93
4.7.2	break 语句和 continue 语句	96
习题 4		97
第 5 章	数组	105
5.1	一维数组	105
5.1.1	一维数组的定义	105
5.1.2	一维数组元素的引用	106
5.1.3	一维数组的初始化	107
5.1.4	应用举例	107
5.2	二维数组和 multidimensional 数组	110
5.2.1	二维数组的定义	110
5.2.2	二维数组变量的存储	111
5.2.3	二维数组元素的引用	112
5.2.4	二维数组的初始化	113
5.2.5	多维数组	113
5.2.6	应用举例	114
5.3	字符数组与字符串	117
5.3.1	字符数组的定义	118

5.3.2	字符数组元素的引用	118
5.3.3	字符数组的初始化	118
5.3.4	字符串的输入/输出	118
5.3.5	字符串处理函数	120
5.3.6	应用举例	123
习题 5		126
第 6 章	函数和编译预处理	132
6.1	函数的定义和调用	132
6.1.1	函数定义的一般方式	132
6.1.2	函数调用的方式	133
6.1.3	形式参数与实际参数	134
6.2	函数返回值和函数类型说明	135
6.2.1	函数的返回值	135
6.2.2	函数的类型声明	136
6.3	数组或字符串作为函数参数	137
6.3.1	数组元素作为函数的实参	137
6.3.2	一维数组名作为函数参数	138
6.3.3	多维数组名作为函数参数	140
6.3.4	字符串作为函数参数	142
6.4	函数的嵌套调用和递归调用	143
6.4.1	函数的嵌套调用	143
6.4.2	递归调用的形式	145
6.4.3	递归函数的使用	145
6.4.4	消去递归	146
6.5	变量存储类型	147
6.5.1	局部变量与全局变量	147
6.5.2	自动变量	149
6.5.3	寄存器变量	149
6.5.4	外部变量	150
6.5.5	静态变量	151
6.6	内部函数与外部函数	154
6.6.1	内部函数	154
6.6.2	外部函数	154
6.7	编译预处理	155
6.7.1	宏定义	156
6.7.2	文件包含	161
6.7.3	条件编译	164
习题 6		166

第 7 章 指针	175
7.1 内存数据的指针与指针变量	175
7.2 指针变量的定义及指针运算	176
7.2.1 指针变量的定义	176
7.2.2 指针变量的运算	177
7.2.3 指针变量作为函数的参数	179
7.3 数组元素的指针与数组的指针	181
7.3.1 数组元素的指针	181
7.3.2 数组的指针	183
7.3.3 多维数组的指针	186
7.3.4 指向由 m 个元素组成的一维数组的指针变量	187
7.4 函数的指针和返回指针的函数	190
7.4.1 指向函数的指针变量	190
7.4.2 返回指针的函数	192
7.5 字符指针	193
7.5.1 字符串的指针	193
7.5.2 字符数组和字符指针变量的区别	195
7.6 指针数组与指向指针的指针	196
7.6.1 指针数组	196
7.6.2 指向指针的指针	197
7.6.3 命令行参数	200
7.7 指针类型小结及相关说明	202
7.7.1 指针类型小结	202
7.7.2 与指针相关的运算	202
7.7.3 使用指针的利与弊	203
习题 7	203
第 8 章 结构体与共用体	209
8.1 结构体类型与结构体类型的变量	209
8.1.1 结构体类型的定义	209
8.1.2 结构体类型变量的定义	211
8.1.3 结构体类型变量的引用	212
8.1.4 结构体类型变量的初始化	213
8.2 结构体数组	214
8.3 指向结构体类型数据的指针	217
8.3.1 指向结构体变量的指针变量	217
8.3.2 指向结构体数组的指针	219
8.3.3 用结构体变量（或数组）作为函数参数	221

8.4	内存的动态分配与单链表	224
8.4.1	存储空间的动态分配	224
8.4.2	单链表的存储	225
8.4.3	单链表的基本操作	226
8.4.4	单链表上的其他操作	232
8.5	共用体	234
8.5.1	共用体类型及共用体类型变量的定义	234
8.5.2	共用体变量的引用	235
8.5.3	使用共用体应注意的问题	235
8.6	位段	236
8.6.1	位段的概念	236
8.6.2	使用位段应注意的问题	237
8.7	枚举类型	238
8.7.1	枚举类型的定义和枚举变量的定义	238
8.7.2	枚举类型在使用中应注意的问题	239
8.8	typedef 语句	240
8.8.1	typedef 语句的一般形式及使用方法	240
8.8.2	使用 typedef 语句应注意的问题	241
	习题 8	243
第 9 章	文件	247
9.1	C 语言文件概述	247
9.2	文件类型指针	249
9.3	文件的打开与关闭	249
9.3.1	文件的打开	249
9.3.2	文件的关闭	251
9.4	文件的读写	251
9.4.1	文件的字符读写函数	251
9.4.2	文件的字符串读写函数	253
9.4.3	文件的数据块读写函数	255
9.4.4	文件的格式化读写函数	257
9.4.5	文件的其他读写函数	258
9.5	文件的定位	259
9.5.1	rewind() 函数	259
9.5.2	ftell() 函数	260
9.5.3	fseek() 函数	260
9.6	文件操作中的错误检测	264
9.6.1	ferror() 函数	264
9.6.2	clearerr() 函数	264

9.6.3	feof()函数	265
9.6.4	常用文件操作函数表	265
习题 9		265
第 10 章 C 语言上机实验		271
10.1	C 语言上机环境	271
10.1.1	Visual C++ 6.0 集成开发环境	271
10.1.2	利用 Turbo C 运行 C 语言程序	277
10.2	上机实验内容	280
实验一	顺序结构 (数据类型、输入与输出)	280
实验二	选择结构	281
实验三	循环控制	282
实验四	数组	282
实验五	函数	283
实验六	编译预处理	284
实验七	指针	285
实验八	结构体与共用体	285
实验九	位运算	286
实验十	文件	286
附录 A C 语言的字符集		288
附录 B C 语言的关键字		289
附录 C C 语言的库函数		290
参考文献		296

第 1 章

C 语言程序设计概述

C 语言是一种目前比较流行的、过程式的程序设计语言，广泛用于系统与应用程序的开发。本章首先介绍算法和程序设计的基本概念，然后介绍 C 语言的发展历史和特点、C 语言程序的结构以及在 Turbo C 2.0 集成开发环境下的上机操作过程，最后介绍用流程图表示算法以及结构化程序设计的思想。学习本章的目的是使读者对 C 语言和程序设计有一个大概的了解，并掌握上机运行简单程序的操作步骤。

1.1 算法与程序设计

1.1.1 算法

1. 算法的概念

做任何事情都有一定的步骤。为解决一个问题而采取的方法和步骤，就称为算法。在计算机领域，算法就是用计算机解决数值计算或非数值计算问题的方法。

解决同一个问题有时可以采取不同的方法与步骤，即存在不同的算法，但要选择质量更高、更合理的算法。比如要从上海到北京，应先买火车票，然后到车站检票上车，火车到达后再坐公交车抵达目的地，这是一种算法。但不同的人还可以根据自己的情况选择更快速、更合理的交通工具，比如乘飞机、乘客车或自驾到达。再看下面两个例子。

【例 1-1】 计算 $1+2+3+\cdots+100$ 。可采取以下两种算法中的一种。

算法 1

可以设两个变量（变量是指其值可以改变的量），一个变量代表和（s），一个变量代表加数（i），用循环算法表示如下：

第一步： $0 \Rightarrow s, 1 \Rightarrow i$ 。

第二步： $s+i \Rightarrow s$ 。

第三步： $i+1 \Rightarrow i$ 。

第四步：如果 $i \leq 100$ ，转第二步；否则，转第五步。

第五步：输出结果 s，结束。

在算法描述中，形如 $e \Rightarrow v$ 表示计算 e 的值，存放到 v 所代表的变量中。例如， $0 \Rightarrow s$ 表示将数值 0 赋给变量 s； $s+i \Rightarrow s$ 表示将变量 s 和变量 i 所代表的值相加，把结果赋给变量 s。算法中第二步到第四步组成一个循环，实现算法时多次执行该循环，只有当第四步经过判断不满足要求时才不返回第二步，而执行第五步输出结果。C 语言有实现循环功能的语句，加上计算机的高速运算，实现这一算法是轻而易举的。

算法 2

第一步： $100 \times 101 / 2 \Rightarrow s$ 。

第二步：输出 s，结束。

算法一更为灵活，若将题目改为计算 $1+2+3+\cdots+1000$ ，只须将第四步改为 $i \leq 1000$ ；若将题目改为计算 $1+3+5+\cdots+101$ ，只须将第三步改为 $i+2 \Rightarrow i$ ，第四步改为 $i \leq 101$ 。

【例 1-2】 判断一个大于等于 3 的正整数是不是素数。

所谓素数是指除了 1 和该数本身之外，不能被其他任何整数整除的数。例如 23 是素数，因为它不能被 2、3、4、 $\cdots$ 、21、22 整除。

判断素数的方法很简单，例如判断 n ($n \geq 3$) 是不是素数，只须将 n 作为被除数，将 $2 \sim n-1$ 各个整数轮流作除数，作除法运算，如果都不能被整除（余数不为 0），则 n 是素数。算法表示如下：

第一步：输入 n 的值。

第二步： i 作除数， $2 \Rightarrow i$ 。

第三步： n 除以 i ，得余数 r 。

第四步：如果 $r=0$ ，表示 n 能被 i 整除，则打印 n 不是素数，转第七步；否则执行第五步。

第五步： $i+1 \Rightarrow i$ 。

第六步：如果 $i \leq n-1$ ，返回第三步；否则打印 n 是素数，转第七步。

第七步：结束。

实际上，除数只要为 $2 \sim \sqrt{n}$ 之间的整数即可。把第六步的条件改变一下，程序执行时间会大大缩短。

2. 算法的属性

从上述算法设计的例题中，不难体会算法具有以下属性：

(1) 有穷性。有穷性是指一个算法的操作步骤必须是有限的、合理的，即在合理的范围之内结束算法。例如，求整数累加和的算法，由于整数本身是个无限集合，如果不限定其范围，会导致求解步骤是无限的。又例如，计算机执行某个算法需要几千年，虽然是有限的，但却是不合理的。当然，究竟什么算“合理”，并没有严格标准，由人们的常识和需要而定。

(2) 确定性。算法中每个操作步骤都应当是明确的，而不应是含糊或模棱两可的。在计算机算法中最忌讳的是歧义性，所谓“歧义性”是指可以被理解为两种或多种可能的含义。因为计算机至今还没有主动思维的能力，如果给定的条件不确定，计算机就无法执行。例如，“计算 3 月 1 日是一年中的第几天”，这个问题是不确定的，因为没有指明是哪一年，不知道是不是闰年，闰年和平年 2 月份的天数不一样，所以无法执行。

(3) 有零个或多个输入。执行算法时需要从外界获得必要信息的操作称为输入。输入的数据个数根据算法确定。例如，计算 $1 \sim 100$ 累加和的算法不需要输入；计算 $n!$ 的算法需要输入 n 的值；计算 m 和 n 的最大公约数和最小公倍数则需要输入 m 和 n 两个数的值。

(4) 有一个或多个输出。执行算法得到的结果就是算法的输出，没有输出的算法是没有意义的。最常见的输出形式是屏幕显示或打印机输出，但并非唯一的形式。执行算法的目的就是为了求解，“解”就是输出。

(5) 有效性。算法中的每一个步骤都应当有效地执行，并得到确定的结果。例如，当 $b=0$ 时， a/b 是不能有效执行的。又例如，在 C 语言中， $a \% b$ 中的 a 和 b 都必须是整型数据，

否则也不能有效执行。

算法有优劣之分,一般希望用简单和运算步骤少的算法。因此,为了有效地进行解题,不仅要保证算法正确,还要考虑算法的质量,选择合适的算法。

算法的描述方式没有统一的规定,同一个算法可采用不同的方式描述。常用的算法描述方式有自然语言方式、流程图方式(见 1.5 节)、计算机语言方式、伪代码方式(将某种计算机语言进行适当修改以描述算法)等。上述两例中算法的描述采取的是自然语言加数学公式的方式。

1.1.2 程序

用计算机语言描述的算法称为计算机程序,或简称程序。只有用计算机语言描述的算法才能在计算机上执行。换言之,只有计算机程序才能在计算机上执行。人们编写程序之前,为了直观或符合人类的思维方式,常常先用其他方式描述算法,然后再翻译成计算机程序。

算法的描述有粗细之分,如果用其他方式描述的算法太粗略,则可能难以直接翻译成计算机程序语言。

【例 1-3】 输入任意 20 个整数,求出其中最大者。可采用以下算法:

第一步:输入一个整数赋给 big。

第二步: $1 \Rightarrow i$ 。

第三步:如果 $i \leq 19$,输入一个整数赋给 x,转第四步;否则,转第六步。

第四步:如果 x 大于 big, $x \Rightarrow \text{big}$,然后转第五步;否则,直接转第五步。

第五步: $i+1 \Rightarrow i$,转第三步。

第六步:输出结果 big,结束。

针对上述算法,用 C 语言可描述为:

```
#include<stdio.h>           /*#include 是预编译命令,它将 stdio.h 这个标准输入/输出  
                               头文件包含,使之成为源程序的一部分*/  
void main()                  /*main 是主函数。一个 C 程序必须有一个主函数,void 表示 main  
                               的返回值类型为 void 类型*/  
{ int i,x,big;               /*说明 i、x、big 是存放整数的变量*/  
  scanf("%d",&big);           /*输入一个数给 big */  
  for(i=1;i<=19;i++)         /*i 从 1~19 (每次加 1) 进行循环*/  
  { scanf("%d",&x);           /*每循环 1 次,输入一个数给 x*/  
    if(x>big) big=x;         /*如果 x 大于 big,将 x 的值赋给 big*/  
  }  
  printf("%d",big);          /*输出结果 big */  
}
```

程序中“/*”和“*/”括起来的内容只起注释作用,程序运行时不起作用。

假如将例 1-3 求最大数的算法简化为:

第一步:输入 20 个数,求其中的最大值并赋给 big。

第二步:输出 big,结束。

则要直接翻译成 C 语言程序就很困难, 因为算法中第一步太粗略, 应该加以细化, 应详细描述如何求 20 个数中的最大者。关于算法的粗细把握, 可以先粗, 再细。但细到什么程度, 这与编程者采用的计算机语言有关。只有全面学习了一种编程语言, 才能对这一点有比较深刻的体会。

1.1.3 程序设计语言

人类社会中有汉语、英语、法语、日语、俄语等语言交流工具, 每种语言又都有它的语法规则。人和计算机通信要通过计算机语言。计算机语言是面向计算机的人造语言, 是进行程序设计的工具, 因此也称程序设计语言。程序设计语言可以分为机器语言、汇编语言、高级语言。高级语言种类繁多(据统计有上千种), 曾经引起广泛关注和使用的语言有 FORTRAN、Basic、Pascal 和 C 等命令式语言(或称过程式语言); 有 LISP、PROLOG 等陈述式语言(两者又分别称为函数式语言和逻辑式语言); 还有面向对象的程序设计语言, 如 C++、Java、Visual C++、Visual Basic、Delphi、PowerBuilder 等。

计算机硬件能直接执行的是机器语言程序。汇编语言也称符号语言, 用汇编语言编写的程序称汇编语言程序。计算机硬件不能识别和直接运行汇编语言程序, 必须由“汇编程序”将其翻译成机器语言程序后才能识别和运行。同样, 高级语言程序也不能被计算机硬件直接识别和执行, 必须把高级语言程序翻译成机器语言程序才能执行。语言处理程序就是完成这个翻译过程的。按照处理方式的不同, 可以分为解释型程序和编译型程序两大类。C 语言采用编译型程序, 即把用 C 语言写的“源程序”编译成“目标程序”, 再通过连接程序的连接, 生成“可执行程序”才能运行。具体过程将在 1.4 节中详细说明。

1.1.4 程序设计的一般过程

从实际问题的描述入手, 经过对解题算法的分析, 设计、编写程序并调试和运行等一系列过程, 最终得到能够解决问题的计算机应用程序, 此过程称为程序设计。下面介绍程序设计的一般步骤。

1. 建立数学模型

对于一个简单问题, 编程者根据经验, 也许能直接写出正确的计算机程序, 如例 1-3 中所举的求 20 个数中最大者的例子。但就客观世界中的一般问题而言, 在程序设计之初, 首先应将实际问题用数学语言描述出来, 形成一个抽象的、具有一般性的数学问题, 从而给出问题的抽象数学模型。以计算个人收入所得税程序设计为例, 问题描述为: 当月收入超过 3500 元的, 超过部分纳税款 20%。输入月收入 income, 计算应交税款 tax。该问题可用数学方式描述为:

$$\text{tax} = \begin{cases} 0 & \text{income} \leq 3500 \\ (\text{income} - 3500) \times 0.2 & \text{income} > 3500 \end{cases}$$

对于任何一个人的月收入 income, 由上述分段函数可计算出应交税款 tax。

数学模型是进一步确定计算机算法的基础。数学模型和算法的结合将给出问题的解决方案。当然, 实际问题是各种各样的, 数学模型也是千变万化的, 这里只是举一个简单的例子。

2. 算法描述

数学模型建立以后, 需要采用一定的算法进行描述, 也可以比较几种算法的优劣, 选择较理想的算法。例如, 上述计算个人所得税的算法可描述为:

第一步: 输入月收入给 `income`。

第二步: 若 $\text{income} > 3500$, $(\text{income} - 3500) \times 0.2 \Rightarrow \text{tax}$; 否则 $0 \Rightarrow \text{tax}$ 。

第三步: 输出 `tax`。

算法的初步描述可以采用自然语言方式, 然后逐步将其转化为程序流程图或其他直观方式。对某些特殊问题, 其数学模型本身可能已是一种算法描述, 此时可省略由数学模型到算法描述这一步。

3. 编写程序

使用计算机系统提供的某种程序设计语言, 将已设计好的算法表达出来, 使得用其他形式表达的算法转变为由程序设计语言表达的算法, 这个过程称为程序编制(或编码)。程序的编写过程需要反复调试才能得到可以运行且结果正确的程序。

4. 程序测试

程序编写完成后必须经过科学的、严格的测试, 才能最大限度地保证程序的正确性。同时, 通过测试可以对程序的性能作出评估。

对于非数值计算问题, 如图书检索、人事管理、科研项目管理等, 在描述算法之前往往先要考虑数据结构, 即描述事物的数据元素和数据元素之间的关系的总称。例如, 要设计将考生的成绩存入计算机并提供查询功能的程序, 可以考虑以下形式的数据元素: 考号、姓名、数学、语文、英语、综合、总分, 每个考生对应一个数据元素。可以按考号从小到大对数据元素进行排序。这样, 对于任意两个考生来说, 考号小的必然在前, 考号大的必然在后, 这便是数据元素之间的关系。由于本教材不涉及复杂的数据结构, 所以这里不详细讨论有关概念。有兴趣的读者可参阅《计算机软件基础》或《数据结构》等参考书。

对于大型的复杂问题, 如何从问题描述入手构成解决问题的算法, 如何快速合理地设计出结构和风格良好的高效程序, 这些将涉及多方面的理论和技术, 因此形成了计算机科学的一个重要分支——程序设计方法学。

如果问题规模大、功能复杂, 则有必要将问题分解成功能相对单一的小模块分别实现。这时, 程序组织结构和层次设计越来越显示出其重要性, 程序设计方法将起到重要作用。程序设计过程实际上成为算法、数据结构以及程序设计方法学三个方面相统一的过程, 这三个方面又称为程序设计三要素。常见的程序设计方法有: 结构化方法、函数式方法、面向对象方法、事件驱动的程序设计方法、逻辑式程序设计方法。根据 C 语言的特点, 本书在 1.6 节中将介绍结构化程序设计的思想。

1.2 C 语言发展历史和特点

1972~1973 年, 美国贝尔实验室的 Dennis Ritchie 发明了 C 语言。C 语言是在一种称为 B 语言的基础上发展起来的, 并首先在配备 UNIX 操作系统的 DEC PDP-11 计算机上实现。在很长的一段时期内, UNIX V 操作系统上配备的 C 语言一直被认为是 C 语言公认的标准。1978 年以后, 随着微型机的普及, 出现了一大批 C 语言系统。绝大多数 C 语言系

统所接受的 C 语言源程序具有很高的兼容性。然而由于没有统一的标准,必然存在差异。1983 年,美国国家标准化协会(American National Standard Institute, ANSI)公布了一个标准,称为 ANSI C。随着 C 语言的发展,1987 年 ANSI 又公布了一个新的标准,称为 87 ANSI C。1990 年,国际标准化组织 ISO 接受 87 ANSI C 为 ISO C 的标准,目前流行的 C 编译系统虽然略有差异,但都是以它为基础的。

目前 C 语言编译系统有多种版本,在微机上常用的有 Microsoft C、Turbo C、Quick C、Borland C++ 5.0、C-Free 3.5 和 Visual C++ 6.0。

1994 年 ANSI 又制定了 ANSI C++ 标准草案。C++ 在 ANSI C 的基础上,扩充了类的部分,因此 ANSI C 是 C++ 的子集。就是说,按照 ANSI C 标准编写的程序可以运行在 C++ 的语言环境中。

C 语言是一种高级语言,和其他高级语言相比,具有以下特点。

1. 兼有低级语言的功能

C 语言虽然属于高级语言,但它兼有机器语言和汇编语言的某些功能。C 语言允许直接访问物理地址,能进行位(bit)操作,可以直接对硬件进行操作。它把高级语言的基本结构和语句与低级语言的实用性结合起来,是处于汇编语言和高级语言之间的一种程序设计语言,因此也可称其为“中级语言”。C 语言的这一特性使其既可以用来设计应用软件,也可以用来设计系统软件。

如果对 C 语言的地址操作这一功能使用不当,则容易造成极其隐蔽、难以排除的故障。所以必须一分为二地去看 C 语言的这一特点。如果只是设计一般的应用程序,可以不使用 C 语言的这一功能。

2. 可移植性好

可移植性表示可将为某种计算机编写的程序改编到另一种机器上去。举例来说,如果为苹果机写的一个程序 A 能够方便地改为可以在 IBM PC 上运行的程序 B,则称 A 为可移植的。与汇编语言相比,C 程序基本上不作修改就可以运行于各种型号的计算机和操作系统上。

3. 结构化的程序设计语言

结构化程序是指整个程序可分解为不同功能的模块,每一个模块又由不同的子模块组成。最小的模块是一个最基本的结构(见 1.3 节)。C 语言程序采用函数结构,便于把整体程序分割成若干相对独立的功能模块,为程序模块间的相互调用以及数据传递提供了便利。C 语言还有多种选择、循环的控制语句以控制程序的流向,使程序完全结构化。

4. 语言比较简洁、紧凑

C 语言一共只有 32 个关键字,而且关键字比较简洁。与 IBM Basic 相比,后者的关键字达 159 个之多。C 语言区分大小写,其关键字都是小写英文单词。例如 else 是关键字,ELSE 则不是。所有关键字构成 C 语言的命令。与其他语言相比,C 语言的控制语句和表达式也比较紧凑。控制语句中去掉了一些不必要的成分,表达式也采取了一些简略的书写方式。

5. 运算符丰富

C 语言的运算符非常丰富,有 34 个。FORTRAN、Basic 等语言的运算符只有 20 个左右。这一特点使得 C 语言的功能强大,使用灵活,可以实现在其他高级语言中难以实现的