

C 语言程序设计教程

——基础与提高

王正友 编著

上海大学出版社

内 容 提 要

本书是为高等院校的一门程序设计课程而编写的教材。全书分为十四章, 主要包括 C 语言概述、基本数据类型、控制结构、运算符和表达式、函数、数组、指针、结构体、共用体和枚举类型、位运算、指向结构体的指针与链表、存储类别、文件、Turbo C 2.0 图形处理、C 语言高级编程基础。

本教材在结构上突出了以程序设计为中心, 以语言知识为工具的思想。对 C 语言的语法规则进行了整理和提炼, 深入浅出地介绍了它们在程序设计中的应用; 在内容上注重知识的完整性, 以适合初学者的需要; 在写法上追求循序渐进, 通俗易懂。本书既可以作为高等院校非计算机专业的计算机语言教材, 也可以作为高等院校计算机专业本科、专科低年级学生学习计算机语言的入门和提高教材, 本书还可以作为科技人员自学 C 语言的参考书。

书 名: C语言程序设计教程——基础与提高

作 者: 王正友

出 版 社: 上海大学出版社

出版日期: 2003年1月

ISBN: 7-81058-347-6/TP312C

定 价: 38.00元

前 言

C语言是目前相当普遍且实用的编程工具,这主要是因为它功能强大,使用灵活,可移植性好,是介于高级与低级语言的一种中级编程语言,C语言已经成为广大计算机专业人士和非计算机专业人士的首选编程工具。

目前有关它的教材也是层出不穷,但在众多的C语言教材中,美中不足的是它们往往重蹈重理论轻实践的覆辙,读者在学完了C语言之后却不知如何使用这个强大的工具来开发软件。本教材将在这方面作重大改进,改变传统编写方法,在理论基础方面,介绍C语言的基本语法及结构化程序设计方法,内容主要包括C语言的顺序程序设计、分支程序设计、循环程序设计、数组、函数、指针、结构、文件等,并系统介绍了C语言的图形设计。

在理论联系实践方面则突出介绍了结构化程序设计的方法和技巧,以提高学生对学习程序设计的兴趣,让学生体会到学习程序设计不再是一件枯燥乏味的事情,而真正感到“乐在其中,用在其中”。根据作者多年从事C语言教学的经验,发现例题的选择尤其重要。因此,我们将书中的程序实例视为学生饭桌上的菜肴,力求将其做得不仅营养丰富,而且色、香、味齐全,符合学生胃口,这样,学生才会乐于去品尝,吃得饱、吃得香。本书对所配的大量例题、经典例题和习题均作过精心地挑选和设计,这些程序主要来源于生活,都是发生在我们身边最熟悉的现实问题,不但内容丰富,涉及面广,而且生动有趣,针对不同应用需求。为使学生了解一个整体的编程过程,提高学生的编程兴趣,编写了类似HdCOPY的程序、e的无限求解、24点的电脑求解等。本书的又一个特点是较系统地讲授了C语言在图形方面的基本应用、图形元素的绘制方法、静动画的制作和图像的传输,对从事广告传播、影视制作等行业人士将有所帮助,是开发图形视频软件的基础。

为方便授课和自学,书中不仅配有每章的教学目标,而且许多例题和习题都前后呼应,既有助于读者理解知识点,又可帮助读者系统学习、编写程序的方法。

本教材适用于本科院校、高等专科学校、民办高校、高等职业学校及计算机类专业,也适用于非计算机专业的C语言普及,还可以作为C语言的自学参考书。

胡郁老师参加了本书的习题编写和附录编写工作。

由于编者水平有限,书中错误在所难免,对于本书中的缺陷,恳请专家和读者批评指正,并请多多提出您的宝贵意见,以便再版时不断修正与完善。您的意见和建议可用E-mail: yin_he8888@sina.com 联系。

编 者

2003年3月

目 录

第一章 C 语言概述	1
☞ 教学目标	1
1.1 C 语言发展简史	1
1.2 C 语言的特点	2
1.3 C 程序的基本结构	2
1.4 基本符号、标识符和关键字	6
1.5 程序设计中的几个基本概念	7
1.6 库函数 scanf()和 printf()的使用	8
1.7 Turbo C 2.0 的使用	12
1.7.1 Turbo C 2.0 简介	13
1.7.2 Turbo C 2.0 的安装和启动	14
1.7.3 运行环境的设置	15
1.7.4 输入和编辑 C 源程序	15
1.7.5 运行 C 程序、查看结果和错误处理	16
1.7.6 存盘和装入	17
1.7.7 退出 Turbo C 2.0 系统	17
1.8 经典例题	18
1.9 习题	19
第二章 基本数据类型	21
☞ 教学目标	21
2.1 C 语言的数据类型	21
2.2 数值数据的表示	21
2.2.1 整型	21
2.2.2 浮点型	22
2.3 常量的定义	23
2.3.1 字符常量	23
2.3.2 字符串常量	24
2.3.3 符号常量	24
2.4 变量的定义和赋值	24
2.4.1 变量的定义	25
2.4.2 变量的赋值	25

2.5	C 语言类型修饰符	27
2.6	数据类型转换	29
2.6.1	表达式类型转换	29
2.6.2	赋值类型转换	30
2.6.3	强制类型转换	31
2.7	计算机中数据的表示	31
2.7.1	二进制、八进制、十六进制	31
2.7.2	进制之间的转换	32
2.7.3	二进制数在计算机中的表示	33
2.7.4	字符编码	34
2.8	经典例题	34
2.9	习题	35
第三章 控制结构		37
☞ 教学目标		37
3.1	算法	37
3.1.1	算法的结构化描述	38
3.2	结构化程序设计	41
3.3	顺序结构	42
3.4	选择结构	43
3.4.1	if 语句	43
3.4.2	SWITCH 语句	50
3.5	循环控制语句	53
3.5.1	while 语句	53
3.5.2	do while 语句	55
3.5.3	for 语句	56
3.5.4	循环嵌套	58
3.5.5	break 语句与 continue 语句	59
3.5.6	goto 语句	60
3.6	经典例题	60
3.7	习题	65
第四章 运算符和表达式		67
☞ 教学目标		67
4.1	算术运算符和算术表达式	67
4.1.1	算术运算符	67
4.1.2	算术表达式	67
4.2	关系运算符和关系表达式	68
4.2.1	关系运算符	68

4.2.2	关系运算符优先级	69
4.2.3	关系表达式	69
4.3	逻辑运算符和逻辑表达式	70
4.3.1	逻辑运算符	70
4.3.2	逻辑运算符优先级	71
4.3.3	逻辑表达式	72
4.4	复合赋值运算符和表达式	72
4.5	自增和自减运算符	74
4.6	逗号运算符和逗号表达式	75
4.7	条件运算符和条件表达式	76
4.8	sizeof 运算符和表达式	77
4.9	运算符的优先级与结合性	77
4.10	经典例题	80
4.11	习题	82
第五章 函数		83
☞	教学目标	83
5.1	概述	83
5.2	函数的定义	83
5.3	函数的类型说明	85
5.4	函数的返回值	86
5.5	函数的调用与参数	88
5.5.1	函数调用形式	89
5.5.2	函数的嵌套调用	90
5.5.3	函数的递归调用	90
5.6	Turbo C 2.0 库函数介绍	92
5.6.1	数学计算函数	93
5.6.2	字符分类函数	94
5.6.3	字符串处理函数	94
5.6.4	图形处理函数	95
5.7	经典例题	96
5.8	习题	102
第六章 数组		104
☞	教学目标	104
6.1	一维数组	104
6.1.1	一维数组的定义	104
6.1.2	一维数组的引用	105
6.1.3	一维数组的初始化	106

6.2	二维数组	107
6.2.1	二维数组的定义	108
6.2.2	二维数组的引用	109
6.2.3	二维数组的初始化	110
6.3	多维数组	111
6.3.1	多维数组的定义	111
6.3.2	多维数组的引用	111
6.4	字符数组与字符串数组	112
6.4.1	字符数组	112
6.4.2	字符串数组	113
6.5	变长数组	116
6.5.1	一维变长数组的初始化	116
6.5.2	二维变长数组的初始化	117
6.6	字符串操作函数	117
6.6.1	单个字符的输入输出	117
6.6.2	字符串的输入和输出函数	119
6.7	数组作为函数参数	122
6.7.1	一维数组元素作为函数参数	122
6.7.2	二维字符串数组作为函数参数	122
6.7.3	数组名作为函数参数	123
6.7.4	多维数组作为函数参数	125
6.8	经典例题	126
6.9	习题	131
第七章 指针		134
☞ 教学目标		134
7.1	基本概念	134
7.1.1	内存地址	134
7.1.2	变量地址	134
7.1.3	变量值的存取	135
7.1.4	指针与指针变量	135
7.2	指针变量的定义与引用	135
7.2.1	指针变量的定义	135
7.2.2	指针变量的引用	136
7.2.3	指针变量作为函数参数	138
7.3	数组的指针和指向数组的指针变量	139
7.3.1	指向数组的指针变量	139
7.3.2	通过指针引用数组元素	140
7.3.3	用数组名作函数参数	141

7.3.4 用指向数组元素的指针变量作函数参数 142

7.3.5 动态数组的实现 143

7.4 二维数组的指针及其指针变量 144

7.4.1 指向二维数组元素的指针变量 145

7.4.2 指向二维数组中一维数组的指针变量 146

7.5 字符串的指针和指向字符串的指针变量 147

7.5.1 指向字符串的指针变量 147

7.5.2 字符串指针作函数参数 148

7.6 返回指针值的函数 150

7.7 函数的指针和指向函数的指针变量 151

7.7.1 函数的指针 151

7.7.2 指向函数的指针变量 151

7.7.3 指向函数的指针的引用 151

7.7.4 指向函数的指针变量作函数参数 152

7.8 指针数组 153

7.9 指向指针的指针 156

7.10 命令行参数 158

7.11 经典例题 159

7.12 习题 161

第八章 结构体、共用体和枚举类型 163

🔧 教学目标 163

8.1 结构体 163

8.1.1 结构体类型的定义 163

8.1.2 结构体变量的定义 164

8.1.3 结构体变量的初始化 165

8.1.4 结构体变量的引用 166

8.2 结构体数组 167

8.2.1 结构体数组的定义 168

8.2.2 结构体数组的初始化 169

8.2.3 结构体数组的引用 169

8.3 共用体 170

8.3.1 共用体类型的定义 171

8.3.2 共用体类型变量的定义 171

8.3.3 共用体变量的引用 172

8.4 枚举类型 174

8.4.1 枚举类型的定义 174

8.4.2 枚举类型变量的定义 174

8.4.3 枚举类型变量的基本操作 175

8.5	定义已有类型的别名	176
8.6	经典例题	177
8.7	习题	179
第九章 位运算		180
☞	教学目标	180
9.1	位运算	180
9.1.1	按位与(&)	181
9.1.2	按位或()	181
9.1.3	按位异或(^)	181
9.1.4	取反(~)	183
9.1.5	左移(<<)	183
9.1.6	右移(>>)	183
9.1.7	位运算赋值运算符	184
9.1.8	不同长度数据间的位运算	184
9.2	位段	185
9.3	习题	186
第十章 指向结构体的指针与链表		188
☞	教学目标	188
10.1	指向结构体变量的指针	188
10.1.1	结构体指针变量的定义	188
10.1.2	结构体指针对结构体成员引用	189
10.2	指向结构体数组的指针	190
10.3	结构体数据作函数参数	191
10.3.1	结构体变量的成员作函数参数	191
10.3.2	结构体变量作函数参数	191
10.3.3	指向结构体的指针作函数参数	192
10.4	链表的概念	193
10.4.1	链表结构	193
10.4.2	简单链表结点定义	193
10.5	链表的基本操作	194
10.5.1	创建链表	194
10.5.2	在链表中插入结点	197
10.5.3	对链表的删除操作	199
10.6	经典例题	200
10.7	习题	205
第十一章 存储类别		207
☞	教学目标	207

11 1	存储类别的概念	207
11 2	变量的存在性和有效性	207
11 3	变量存储类别	208
11 3 1	自动变量	208
11 3 2	寄存器变量	210
11 3 3	静态变量	211
11 3 4	外部变量	212
11 4	函数的存储类别	213
11 5	习题	215
第十二章 文件		217
☞	教学目标	217
12 1	文件概述	217
12 2	文件的打开与关闭	218
12 2 1	文件类型指针	218
12 2 2	文件的打开	218
12 2 3	文件的关闭	220
12 3	文件的读写操作	221
12 3 1	字符读写操作	222
12 3 2	字符串读写操作	223
12 3 3	整数读写操作	224
12 3 4	按格式读写操作	226
12 3 5	数据块读写操作	229
12 3 6	读写其他类型数据	232
12 4	读写函数的选用原则	232
12 5	文件的定位	232
12 6	设备文件	236
12 7	检测函数	237
12 8	非缓冲区型文件	238
12 9	经典例题	242
12 10	习题	252
第十三章 Turbo C 2.0 图形处理		254
☞	教学目标	254
13 1	显示器的工作模式	254
13 1 1	显示卡(显示适配器)	254
13 1 2	文本模式	254
13 1 3	图形模式	254
13 2	文本窗口和图形窗口	255

13 2 1	文本窗口	255
13 2 2	图形窗口	256
13 3	文本模式下的程序设计	256
13 3 1	文本模式下程序设计的基本结构	256
13 3 2	文本输出与窗口管理	257
13 3 3	窗口及模式控制	258
13 3 4	属性控制	258
13 3 5	状态查询	260
13 4	图形模式下的程序设计	261
13 4 1	图形模式下程序设计的基本结构	261
13 4 2	图形系统控制	267
13 4 3	基本图形函数	267
13 4 4	管理屏幕及图形窗口	275
13 4 5	图形模式下的文本输出	277
13 4 6	颜色控制	281
13 5	动画技术	282
13 5 1	动态开辟图形窗口技术	282
13 5 2	显示页和编辑页交替变化技术	283
13 5 3	画面存储再重放技术	283
13 5 4	图像动态存储器操作技术	285
13 6	经典例题	285
13 7	习题	294
第十四章 C 语言高级编程基础		296
☞ 教学目标		296
14 1	8086 芯片的简介	296
14 1 1	80×86 的内部寄存器	296
14 1 2	地址计算	297
14 2	Turbo 编译方式简介	298
14 2 1	Turbo C 的六种存储方式	298
14 2 2	混合模式编程	299
14 3	系统功能的调用	301
14 3 1	BIOS 功能的调用	301
14 3 2	DOS 功能的调用	308
14 4	C 语言与汇编语言的混合编程	310
14 5	习题	312
附录 A 常用字符与 ASCII 码对照表		313
附录 B 常用 Turbo C 2.0 库函数		314
参考文献		338

第一章 C 语言概述

🔑 教学目标

本章主要介绍 C 语言特点、C 语言程序格式、main 函数作用、基本输入输出语句,以及 C 程序的编辑、编译和运行方法。要求了解 C 语言特点,掌握 C 程序基本结构,掌握输入输出函数 scanf()和 printf()的基本使用方法,熟悉 C 程序编辑、编译、调试和运行过程。

1.1 C 语言发展简史

1. 语言的选择

我们在为计算机编制程序时究竟使用哪种语言好呢?这要根据实际情况来确定,也就是根据资源条件(包括技术力量)和要实现的功能来确定,即因地制宜。

C 语言是一种很灵活的语言,它把许多程序设计技巧交给了编程人员,受到人们普遍喜爱,作为系统描述语言,它成功地编写出 UNIX、FoxBASE 等系统软件。作为通用程序设计语言,越来越多的人选择使用它以替代其他高级语言。目前,C 语言已成为发达国家软件开发的主流语言,无论是设计系统软件,还是图形处理、数据分析、数据计算等应用软件,都可以看见 C 语言的“身影”。

掌握了 C 语言后,再学 C++、Java 等语言就比较容易了。

2. C 语言的诞生与发展

(1) C 语言诞生以前,系统软件(如操作系统)主要用汇编语言编写的,如当时的 UNIX 版本是用汇编语言编写的。由于汇编语言依赖于计算机硬件,其可读性和移植性都很差;但一般高级语言又难以实现对计算机硬件的直接操作(这正是汇编语言的优势),于是人们盼望有一种兼有汇编语言和高级语言特性的新语言。C 语言就是在这种背景下应运而生的。

(2) C 语言是贝尔实验室(Bell Laboratories)于 20 世纪 70 年代初为描述 UNIX 操作系统和 C 编译程序而开发的一种系统描述语言,即 C 语言的早期版本。

C 语言开始附属于 UNIX,运行在 PDP-11 机上,并随着 UNIX 操作系统的日益广泛使用,迅速得到推广。到 1978 年以后,C 语言就移植到各种微机上,并能在各种操作系统下运行,这时出现了各种 C 编译系统,这些 C 编译系统的规格是不尽相同的,因此,用户编辑的源程序要与所使用的编译系统相对应,即各种编译系统下所编的源程序不具有互换性。

为提高互换性,1983 年美国国家标准化协会(ANSI)制定了 C 语言标准,即 ANSI C。目前流行的各种版本的 C 语言都是以它为基础的。

(3) 目前,微机上广泛使用的 C 语言编译系统有 Turbo C(简称 TC)、Quick C(简称

QC)、Microsoft C(简称为 MSC)、Borland C(简称为 BC) 等。这些版本的 C 语言虽然基本部分相同,但也有些差异。欲知 Turbo C 使用的详细情况,可参阅《Turbo C 2.0 用户手册》。

1.2 C 语言的特点

概括地说,C 语言同时具有汇编语言和高级语言的双重特性。C 语言的主要特点如下:

1. 语言简洁、使用方便、灵活

C 语言一共只有 32 个关键字,9 条控制语句,源程序书写格式自由。

2. 丰富的运算符、支持对硬件的操作

C 语言的运算符数量多达 40 余种,超过其他高级语言。例如,自增(+ +)、自减(-)运算符,复合赋值运算符,位运算符,条件运算符等等。特别是提供了可以进行低级处理的位操作和地址运算符,使 C 语言的处理能力大大增强。

3. 丰富的数据类型

C 语言具有整型、浮点型、字符型、数组类型、指针类型、结构体类型、共用体类型等多种数据类型,可以实现各种复杂的数据结构(如链表、栈、树等)的运算。特别是 C 的指针类型,功能强大、灵活方便,恰当地使用不仅可以简化程序结构,而且可以节省存储空间、提高运算速度,这些数据类型可满足开发各类系统软件和应用软件的需要。

4. 结构化的控制语句

C 语言的控制结构语句符合结构化程序设计要求,并且用函数作为程序模块,使得程序结构清晰、可读性好、易于调试。

5. 高效率的目标代码

C 语言允许直接访问物理地址、直接对硬件操作,提供对字节、位、内存和寄存器操作,可以调用或嵌入汇编语言代码,并且经过 C 编译程序生成的目标代码质量好、程序执行效率高。C 语言目标代码的执行效率,一般只比汇编语言的低 10%~20%,因此 C 既具有高级语言功能,也具有低级语言的功能。

6. 通用性、可移植性强

用 C 语言写的程序,基本上可以不加修改地用于各种计算机和操作系统上。

7. 丰富的库函数

C 语言系统为方便用户,提供了十分丰富的库函数。例如,1987 年 Borland 公司推出的 Turbo C 2.0 提供了十几类 300 多个库函数。

1.3 C 程序的基本结构

一个完整的 C 语言程序,是由一个、且只能有一个 main()函数(又称 C 主函数)和若干个其他函数结合而成的,或仅由一个 main()函数构成。

我们通过下面几个例子,分析一下 C 程序的基本结构。

例 1.1 在屏幕显示一句话:This is my first c program !

```
#include<stdio.h>                                /* 包含输入输出头文件 */
void main( )                                       /* 定义主函数 */
{
    printf(" This is my first c program ! \n");
    /* 在屏幕上输出一句话后,光标移到下一行 */
}
```

运行结果:

 This is my first c program !

分析:

- (1) 程序由主函数 main()组成。
- (2) 主函数内容用大括号{ }括起来。
- (3) “/ * …… */”之间的内容为注释信息,方便程序阅读,在程序的编译、运行中无作用。
- (4) 只要程序用到输入输出函数,如 printf(),需要在程序中包含 stdio.h 头文件, #include <stdio.h>。其中:“ \n”表示输出后换行。
- (5) 语句结束加“ ;”。

举一反三:

 在屏幕上显示一句话“ This is a c book !”

例 1.2 求 12 和 34 的和与积。

```
#include<stdio.h>                                /* 包含输入输出头文件 */
void main( )                                       /* 定义主函数 */
{
    int a, b, sum, product;                        /* 定义整数变量 a,b,sum,product */
    a = 12;                                         /* 给 a 赋值 */
    b = 34;                                         /* 给 b 赋值 */
    sum = a + b;                                    /* 令 sum = a + b */
    product = a * b;                                /* 令 product = a * b */
    printf(" sum = %d product = %d\n", sum, product); /* 输出 sum 和 product */
}
```

运行结果:

 sum = 46 product = 408

分析:

- (1) 包含定义输入输出函数的头文件。
- (2) 程序由主函数 main()组成。
- (3) 主函数内容由大括号括起来,包括:定义所有要使用的变量(int 说明 a、b、sum、product 都是整数),变量赋值(a、b 直接赋值常数),运算(sum 等于 a + b 的和,product 等于 a * b 的积),输出(整数 sum 和 product 的值的位 置用 %d 代替)。
- (4) 每一个语句结束时加“ ;”。

举一反三:

 求 456 和 123 的差。

例 1 3 输入 a 和 b 两个数,输出其中最大一个数的平方。

```
#include<stdio.h>                                     /* 包含输入输出头文件 */
float max(float x,float y)                             /* 定义求 x、y 的最大值的函数 max */
{ float z;                                             /* 定义实数变量 z */
  if(x>y) z=x;                                         /* 如果 x 大于 y,则 z 等于 x */
  else z=y;                                           /* 否则 z 等于 y */
  return(z);                                          /* 返回 z,则 max 函数值为 z 的值 */
}
void main( )                                           /* 定义主函数 */
{ float a, b, c;                                       /* 定义实数变量 a、b、c */
  printf(" please input(a,b) = ");
  scanf(" %f,%f", &a, &b);                             /* 输入变量 a、b */
  c=max(a,b);                                         /* c 等于 a、b 的最大值 */
  c=c*c;
  printf(" a = %f,b = %f,the max double = %f \n",a,b,c); /* 输出 c 的值 */
}
```

运行结果：

please input(a,b) = 12,25
a = 12.000000,b = 25.000000, the max double = 625.000000

分析：

- (1) 包含定义输入输出函数的头文件 `stdio.h`。
- (2) 程序有两个函数,主函数 `main()` 和用户自定义函数 `max(x,y)`。
- (3) 用户自定义函数 `max(x,y)` 用于求出给定两个实数 `x、y` 的最大数。
- (4) 用户自定义函数 `max(x,y)` 内容在大括号内,包括:定义放最大数的变量 `z`;进行运算(用 `if-else` 语句实现 `z` 等于 `x、y` 的最大值);最后 `return(z)` 使得函数 `max(x,y)` 的结果等于 `z`。
- (5) 主函数内容同样在大括号内,包括:定义所有要使用的变量,然后用 `scanf()` 输入函数等待用户通过键盘输入两个数 `12,25` 给 `a、b`(注意两个数之间用逗号间隔),接着 `c` 等于 `max(a,b)`(即 `a、b` 的最大数),最后输出 `c` 的值。
- (6) 每一个语句结束时加“`;`”。

举一反三：

输入 a、b 两个数,输出其中最小数的平方。

以上几个例子的目的是为了阐述 C 程序结构,其他问题无需深究。C 程序的基本结构如下：

```
预处理指令
声明函数
声明全局变量
void main( )
{
    局部变量声明语句
    可执行语句序列
}
```

```
f1( )
{
    局部变量声明语句
    可执行语句序列
}
f2( )
{
    局部变量声明语句
    可执行语句序列
}
...
fn( )
{
    局部变量声明语句
    可执行语句序列
}
```

对于 C 程序基本结构的几点说明如下：

(1) 预处理指令: 指示编译系统进行有关处理时, 要使用预处理指令。例如, 预处理指令 `# include < stdio .h >` 的作用是让编译系统把 `stdio .h` 文件内容插入到该命令所在位置。有些程序会用到多个预处理指令。

(2) 声明函数: 当 `main( )` 要调用自定义函数时, 被调用的函数需要声明。

注意: 声明函数和定义函数是两回事。实际上我们使用的系统库函数也是要声明的, 其声明在相应的头文件中, 例如 `printf( )` 和 `scanf( )` 函数的声明在 `stdio .h` 中。

(3) 声明全局变量: 全局变量是在程序中的各个函数中均可使用的变量, 其声明的位置在所有函数之外。比如可以在 `main( )` 函数定义之前声明全局变量, 以代替宏定义。

(4) 声明局部变量: 局部变量只在声明它的函数中有效, 在其他函数中无效。例如, 在例 1.3 中, `main( )` 函数使用的局部变量声明语句为:

```
int a,b,c;           / * 变量声明语句 */
```

(5) 函数体由一系列语句组成, 包括各种声明语句和可执行语句。

(6) 一个完整的 C 语言程序, 是由一个、且只能有一个 `main( )` 函数(又称主函数)和若干个其他函数组成, 或仅有一个 `main( )` 函数组成。

使用函数不但可以简化程序, 而且还可以将程序设计模块化。

C 语言的函数可以分为三类:

第一类: 主函数, 名为 `main( )`, 相当于主程序。C 程序总是从主函数开始执行, 而不论主函数在什么位置, 主函数执行完毕, 即程序执行完毕。

第二类: 其他函数的作用, 相当于子程序, 例如, `f1( )` 到 `fn( )` 和例 1.3 函数 `max( )` 代表若干个用户定义的函数可有可无, 数目不限。

第三类: C 语言提供的库函数, 例如输出函数 `printf( )` 和输入函数 `scanf( )`。C 程序中应该加入包含定义库函数的头文件(某些 C 环境默认 `# include < stdio .h >`, 此时可以缺省)。

- (7) 每个函数的内容用大括号“ { } ”括起来。
- (8) 每个语句后加“ ; ”。

1.4 基本符号、标识符和关键字

这一节介绍 C 语言的基本符号、标识符和关键字。

1. 基本符号

在 C 语言中,要用到以下一些基本符号:

- (1) 英文字母:26 个大写字母 A~Z 和 26 个小写字母 a~z。
- (2) 十进制字符:0,1,2,3,4,5,6,7,8,9 共十个。
- (3) 下划线: _。
- (4) 其他特殊符号:主要是分隔符、运算符等,如: +、-、*、/、%、<、<=、&&、~、||、<<、()、>、>=、>>、[]、{ }、. ;、,、? 等。

2. 标识符

标识符用于标识程序中的常量、变量、数据类型和函数名字。标识符有两类:一类由编程人员自己定义和使用,例如,在例 1.3 中使用的变量名 a, b, c 及函数名 max。另一类是关键字,是 C 语言系统已经使用的,具有特定作用的标识符,例如,主函数的名字 main。通常所说的标识符一般指自定义标识符。

C 的标识符必须满足以下规则:

- (1) 所有标识符必须由字母(a~z, A~Z)或下划线(_)开头,标识符的其他部分可以是字母、下划线或数字(0~9)。
- (2) 需要特别注意的是,C 语言的标识符对大小写敏感,例如: add, ADD, Add, aDD 就代表互不相同的标识符。
- (3) 从理论上说,一个标识符可以由多个字符组成,但有的 C 系统只识别前 8 个字符,而 Turbo C 识别前 32 个字符,即两个标识符能根据前 32 个字符相互区别,32 个以后的字符无效。但一般情况下,标识符不会这样长。
- (4) 用户定义的标识符不能与关键字拼法相同。
- (5) C 程序文件名的组成,服从操作系统对文件管理的规则。

下面是一些合法的标识符:

if, MAX, _cnt, CA_to_CB, buffer, input_data

下面是一些不合法的标识符:

4line(数字开头), Key-char(使用了-), MSC4\$ (使用了\$), float(与关键字拼法相同)

3. 关键字

关键字是 C 语言系统中已经使用,具有固定含义的标识符。关键字在语言系统中常作为语句或标准函数的组成部分。

由 ANSI 标准定义的关键字共有以下 32 个。

auto	double	int	struct	break	else	long	switch
case	enum	register	typedef	char	extern	return	union
const	float	short	unsigned	continue	for	signed	void