



胡久清 主编

C语言程序设计

高等财经院校试用教材

GAODENG CAIJING YUANXIAO SHIYONG JIAOCAI

中国财政经济出版社

C YUYAN

CHENGXU SHEJI

12

11

tp312
HJQ/1

高等财经院校试用教材

C 语言程序设计

主 编 胡久清

副主编 朱兴德

中国财政经济出版社

051343

图书在版编目 (CIP) 数据

C 语言程序设计/胡久清主编. - 北京: 中国财政经济出版社, 1998.6
高等财经院校试用教材
ISBN 7-5005-3735-2

I. C… II. 胡… III. C 语言-程序设计-高等学校-教材 IV. TP312

中国版本图书馆 CIP 数据核字 (98) 第 05228 号

IS256/18

中国财政经济出版社出版

<http://www.ccfph.com>

(版权所有 翻印必究)

社址: 北京东城大佛寺东街 8 号 邮政编码: 100010

兴谷印刷厂印刷 各地新华书店经销

787×1092毫米 16开 14.25印张 343 000字

1998年5月第1版 1998年5月北京第1次印刷

印数: 1-2 050 定价: 16.00元

ISBN7-5005-3735-2/TP·0019

(图书出现印装问题, 本社负责调换)

编 审 说 明

本书是全国财经类通用教材。经审阅，我们同意作为高等财经院校试用教材出版。书中不足之处，请读者批评指正。

财政部教材编审委员会

1998年2月5日

前 言

高级程序设计语言是用户使用计算机解决实际业务问题的重要工具。对经济信息管理（或管理信息系统）专业的学生而言，牢固地掌握一种高级程序设计语言及相关的程序设计方法，并熟练地运用它们进行程序设计和上机训练，是一项至关重要的基本功。

由于过程型高级程序设计语言有多种，它们各有千秋，本课程采用当前世界上最流行而又极有影响的 C 语言作为工具，讲述相关的程序设计方法。C 语言支持结构化程序设计，描述能力强，可移植性好，目标程序效率高，应用广泛，它既有高级程序设计语言的优点，又有汇编语言的特色，对开发具有特色的管理信息系统是必不可少的工具。因此，经济信息管理（管理信息系统）专业学生掌握 C 语言及其程序设计方法，就能迅速适应其工作，及早发挥其作用，可望受到社会的欢迎。

本教材是财政部组织部属各高校教师编写的，它是根据财政部 1992 年 11 月普通高校教学大纲审定会审定通过的《C 语言程序设计》教学大纲而编写的。它适用于财经院校的经济信息管理、管理信息系统专业本科的教学需要，也可作为相关专业的教材和参考书。

根据学科体系及专业培养目标对本课程的要求，并吸取了教学经验和科研成果，经过编写组的认真研讨，在教材内容的选取和体系结构上，全面地反映了 C 语言的完整性及程序设计的基本方法，这包括它的数据结构及其类型、表达式及其运算、程序结构及控制语句、预处理器及库函数、系统调用，与汇编语言的接口等。结合其使用，系统地阐述基本程序设计方法及技术，这包括分支程序设计、循环程序设计、函数程序设计、结构化程序设计以及图形技术、汉字技术等。同时，按照认识论的方法，理论与实践相结合的原则以及学科的自身体系，合理地安排了教学内容。为了提高学生综合应用能力和编程能力，启发学生的智慧，还在最后一章讲述了两个有一定规模的综合应用程序。

教材编写组的分工如下：第一、十一章由胡久清执笔，第二、三章由张不同执笔，第四、五、六及十二章由朱兴德执笔，第七、八章第一节由勒中坚执笔，第九章第一节由胡乾顺执笔，第八章第二节、第九章第二节由安泉执笔，全书稿件由胡久清总纂、修改，最后由安泉及张建坤、卢斌、梁高峰、丁浩等将书稿输入计算机并排版。本书由潘省初、邱家武主审。在编写过程中，得到了财政部教材编审委员会及办公室、中南财经大学科研处各位领导的悉心指导和支持，评审专家们付出了辛勤的劳动，并提出了中肯的宝贵意见；中南财经大学信息系邱家武主任提出了很多宝贵意见，给予了大力支持，宋邦全、代炜、陆建平、贺棉桃等同志给予了热情的帮助，冯道海、彭军也作了一些工作，在此表示衷心的感谢！

编写组在编写中虽然尽了自己的努力，但由于水平所限，书中欠妥之处在所难免，诚望读者和专家批评指正。

编 者

1998 年 2 月

目 录

第一章 导论	(1)
1.1 程序设计语言与程序设计	(1)
1.2 C语言的发展及特点	(3)
1.3 C语言的词汇与程序结构	(5)
1.4 C语言程序上机操作步骤	(9)
习题一.....	(10)
第二章 基本数据类型及表达式	(12)
2.1 基本数据类型	(12)
2.2 常量及变量	(14)
2.3 运算符	(18)
2.4 表达式	(26)
习题二.....	(26)
第三章 程序控制语句	(28)
3.1 程序的三种基本控制结构	(28)
3.2 简单语句及其程序设计	(31)
3.3 条件语句及其程序设计	(41)
3.4 循环语句及其程序设计	(47)
3.5 goto 语句及标号	(52)
习题三.....	(54)
第四章 数组	(57)
4.1 数组的概念	(57)
4.2 一维数组	(59)
4.3 字符和字符串数组	(63)
4.4 多维数组	(73)
习题四.....	(75)
第五章 函数	(77)
5.1 程序和函数	(77)

5.2 函数类型、参数及返回值·····	(81)
5.3 函数与数组 ·····	(86)
5.4 变量的作用域、存储类·····	(88)
习题五·····	(95)
第六章 指针 ·····	(96)
6.1 指针的概念 ·····	(96)
6.2 指针的定义、初始化及空指针·····	(99)
6.3 指针运算 ·····	(101)
6.4 指针与数组 ·····	(106)
6.5 指针数组 ·····	(111)
6.6 指针空间的申请与释放 ·····	(112)
6.7 指针与函数 ·····	(114)
6.8 命令行参数 ·····	(117)
6.9 指针型函数 ·····	(121)
6.10 递归函数·····	(123)
习题六·····	(125)
第七章 结构与联合 ·····	(127)
7.1 结构的定义与说明 ·····	(127)
7.2 结构数组 ·····	(132)
7.3 结构指针 ·····	(133)
7.4 结构与函数 ·····	(136)
7.5 结构嵌套 ·····	(147)
7.6 位段 ·····	(150)
7.7 联合 ·····	(153)
7.8 枚举类型 ·····	(156)
习题七·····	(159)
第八章 预处理与系统功能调用 ·····	(160)
8.1 预处理 ·····	(160)
8.2 系统功能调用与汇编语言接口 ·····	(162)
习题八·····	(167)
第九章 文件 I/O 与图形函数 ·····	(168)
9.1 文件 I/O 函数 ·····	(168)
9.2 图形函数及应用 ·····	(176)
习题九·····	(193)

第十章 综合应用程序.....	(194)
10.1 综合应用程序之一.....	(194)
10.2 综合应用程序之二.....	(201)
第十一章 由 C 到 C++ 的发展	(206)
11.1 C 语言的发展	(206)
11.2 C++ 对 C 的扩展及特点	(207)
附录一: ASCII 字符集	(217)
附录二: 本书使用的 BIOS 及 DOS 系统调用的中断号及功能表示	(218)

第一章 导 论

自 1946 年第一台电子计算机问世以来,计算机科学技术发展日新月异,它的辉煌成就对人类社会产生着深远的影响。当今信息化社会的进程无不与之息息相关,计算机已广泛应用于社会生活的各个领域,且已形成计算机文化,计算机运用正逐步成为人们工作、交往乃至生活的重要组成部分。

近半个世纪,计算机硬件的发展经历了四代(按所用的元器件划分为电子管、晶体管、集成电路、大规模集成电路和超大规模集成电路)。当今,新一代计算机正在孕育之中。以计算机系统软件(包括操作系统,程序设计语言的编译或解释系统,数据库管理系统等)为核心的软件技术,也与硬件技术一样有相应的发展,两者相互依存并相互促进。以程序设计语言为例,它也从最初的机器语言、汇编语言的较低级阶段,发展到品种较齐全的高级阶段。随着计算机与通信技术、多媒体技术、人工智能技术、决策分析方法等的紧密结合,计算机对人类的作用将愈加显著。

1.1 程序设计语言与程序设计

从计算机在社会生活各个领域中的广泛应用,足见其功能之强大,而这些功能都是通过运行存储在其中的程序来实现的。那么,什么是程序呢?这里所说的程序乃是指计算机赖以实现其功能的一组指令序列,即计算机程序,简称程序。或者说,程序是指对人们所要处理或求解问题的对象、处理规则或过程的描述。这就是说,将解决问题所涉及的对象、处理规则或过程,用计算机能识别的相关指令一一加以描述,从而形成相应的指令序列,这个指令序列的全体即成为求解该问题的程序。这里所说的指令(也可称作命令),一般由操作和操作对象两部分组成。

可以看出,这里涉及两个问题:一是用于描述的指令,二是描述任务的实现。前者便是关于程序设计语言的问题,后者则是关于程序设计的问题,下面分别进行讨论。

1.1.1 程序设计

程序设计是实现所要解决问题的描述任务的过程。这就是说,程序设计就是要完成对求解问题的描述,这种描述是一项活动(过程),这包括对问题的理解(给出完整的含义),并将这种理解加以描述,以形成求解的算法(或步骤),最后用机器所能识别的指令(或者程序设计语言)对该算法或过程进行描述,从而形成求解该问题的程序。

【例 1-1】编写核对进入某应用程序时的口令的程序（段）。

本问题的程序设计可描述如下：

首先理解问题，欲要使用某应用程序，用户必须知道该应用程序事先规定的口令。在进入时，它必须向用户提示“请输入口令”。当用户输入口令之后，它就核对输入的口令是否与规定的口令相符，如果符合，才正式进入该应用程序，让用户使用。可见本程序段应在应用程序之前执行。

其次设计算法。根据上述对问题的理解，这里所使用的数据之一是口令，一般而言，口令是字符串；另一数据是向用户的提示信息，即“请输入口令”。因此可设计算法如下：

1. 显示信息“请输入口令”；
2. 核对口令；
3. 若不相符，则显示信息“对不起，你不能使用！”而退出；
4. 若相符，则显示信息“欢迎你使用！”而进入。

最后用程序设计语言实现上述算法。在此，用 C 语言实现。假定本程序段放在应用程序的前面，最先执行这个程序段，为简便起见，还假定口令为整数。程序如下：

```
#include <stdio. h>          /* 标准库函数头文件 */
long password = 48857358; /* 说明 password 为长整型变量，口令为 48857358 */
main ()                    /* 这是应用程序的主函数 */
{ long    pass;           /* 从这里开始是本程序段，首先说明 pass 为长整型变量 */
  printf (" \nPlease input password: ");
  scanf ("%d", &pass);
  if (pass != password)
  { printf (" \nSorry, You can't use !"); goto exit; }
  else printf (" \nWelcome, You can use !");
/* 以下是应用程序部分 */
  .....
}
```

在这里，我们暂不对本程序段所使用 C 语言的各种语言成分作出说明，而仅了解到程序设计有编写程序这一工作步骤。

程序设计除了上述三个主要步骤之外，还应该测试程序的工作，这样才能保证程序的正确，因为不正确的程序是没有任何意义的；在程序测试完成以后，还有一项重要的工作，这就是整理、编写程序的文档（包括使用手册等），这对使用、维护该程序是必不可少的。

程序设计涉及问题还很多，诸如程序设计的原则、方法、风格和工作环境等，这将在后面的章节中逐步加以讨论。

1.1.2 程序设计语言

程序设计语言是用来编写计算机程序的一种语言。或者说，为设计程序而专门研制的一种计算机所能识别的语言，以便用来编写程序。它与人们日常使用的自然语言相似，只不过是一种受到严格限制的极简单的自然语言罢了。该语言的基础仍然是一组记号（或符号）和一组规则，根据这些规则由记号组成记号串的总体就是语言。一种语言总是由若干词法单位

和语法规则所构成的各种语句集来体现的。因此，在考察程序设计时，也一定要注意它的各种语言成分的组成规则及含义，即程序设计语言的语法（syntax）和语义（semantics）。语法表示构成语言各记号之间的组合规则（即结构或形式）；语义表示按各规则形成各个记号的特定含义。

前已指出，程序设计语言经历了由低级阶段到高级阶段的发展。早期的程序设计语言是机器语言（由机器的指令系统构成）。机器的指令一般由操作码部分（用以表示操作或运算）和操作数部分（用以表示操作对象或操作对象所在的地址）组成。由于机器语言是二进制代码（由 0、1 数字组成），不便于记忆和修改，进而发展到汇编语言阶段。汇编语言其主要特点就是将机器指令用符号表示，同时增加了一些汇编控制语句（或命令）。尽管汇编语言弥补了机器语言的一些不足，但由于它对机器（语言）的依赖性太强，对于非计算机专业人员而言，他们的使用仍然十分困难，不便于计算机的广泛应用，由此又发展到高级阶段，即高级程序设计语言阶段。从 50 年代中期至今，高级程序设计语言已有上百种，其中著名的常用语言有 BASIC、FORTRAN、COBOL、C、PASCAL、MODULA-2 等等。尽管各种高级语言程序设计语言的类别不同，但一般都有如下四种成分：

1. 数据成分。用以描述程序中所涉及的数据；
2. 运算成分。用以描述程序中所包含的诸运算；
3. 控制成分。用以描述程序中的控制机制；
4. 传输成分。用以描述程序中的数据传输机制。

各种高级程序设计语言在使用中不断完善，每种语言都形成了诸多的版本，而且有些语言，国际化标准组织（ISO）为之制定了标准，使得对该语言的使用有一个共同遵守的标准，这有利于其推广应用。本教材采用的是 C 语言，而且以在微机上流行的 BORLAND C 2.0 版本为蓝本加以介绍。本章仅对 C 语言作简要的介绍。

1.2 C 语言的发展及特点

C 语言是一种通用的过程型高级程序设计语言，用它既可以写计算机操作系统之类的系统软件，也可以编写各种应用软件，它已成为当今广泛应用的高级程序设计语言。

1.2.1 C 语言的发展简史

C 语言是为描述 UNIX 操作系统而设计的。它的发展与 UNIX 操作系统一样，经历了产生、完善、广泛应用的过程。

C 语言的前身是 BCPL（Basic Combined Programming Language）语言。BPCL 是由英国剑桥大学 Martin Richards 在 1967 年设计的，具有结构化好，提供指针处理方式，能和内存地址打交道等特点。1970 年，美国电话与电报公司（AT&T）贝尔实验室的 Ken Thompson，以 BPCL 为基础设计了 B 语言，并于 1971 年在 PDP-11/20 计算机上实现了 B 语言，并使用 B 语言编写 UNIX 操作系统。但 B 语言过于简单，功能有限。1972 年，贝尔实验室的 D. M. Ritchie 在 B 语言的基础上设计出 C 语言，它保持了 BCPL 及 B 语言的优点（精

练,接近硬件),同时克服了它们的缺点(过于简单、数据无类型等)。1973年, Ken Thompson 和 D. M. Ritchie 把 UNIX 操作系统用 C 语言重新写了一遍(原来是用汇编语言编写的)。

后来, C 语言多次作了改进,但主要还是在贝尔实验室内部使用,直到 1975 年, UNIX 在 6 版公布以后, C 语言的突出优点才引起了人们的普遍注意。1977 年出现了不依赖于具体机器的 C 语言编译系统,使 C 移植到其他机器上时所需做的工作大为简化,这也促使 UNIX 操作系统迅速在各种机器上实现。随着 UNIX 操作系统的日益广泛使用, C 语言也迅速得到推广。1978 年以后, C 语言已先后移植到大、中、小、微型机上,已独立于 UNIX 操作系统和 PDP 系列计算机了,且已成为世界上应用最广泛的计算机高级程序设计语言之一。

以 1978 年发表的 UNIX 第 7 版中的 C 编译系统为基础, B. W. Kernighan 和 D. M. Ritchie (合称 K&R) 合著了《The C Programming Language》一书,其中介绍的 C 语言便成了后来广泛使用的 C 语言版本之基础。1983 年美国国家标准协会(ANSI)为了使 C 语言得到更快更好的发展,制定了 C 语言标准,称为 ANSI C。1987 年又公布了新标准: 87 ANSI C。现在流行的各种版本都是以这个标准为基础的。直到 1990 年才有国际标准: ISO/IEC 9899。现在微机上使用的 Microsoft C, Turbo C, Quick C 等,它们的基本部分是相同的,但也有些不同,同时,它们中的不同版本又略有差别。因此,在使用 C 语言时,要了解所用的计算机上 C 编译系统的特点和规定(可参考相关的使用手册)。

1.2.2 C 语言的特点

C 语言能够得到快速的发展,受到用户的欢迎,这是和它的优越特点分不开的。这主要是:

1. C 是结构化语言

C 语言提供了支持结构化程序设计的设施:结构化的控制语句(如分支结构,循环结构等);以花括号对“{”、“}”为分隔的代码块化(或复合语句)的程序结构;以函数调用为主的程序设计风格(即以函数为模块来编写程序的模块化设计方法)。这保证了 C 语言所编写的程序具有良好的结构,易于阅读和维护。此外, C 语言中有些存储类型(如静态 static 类型, extern 类型)在功能上有助于数据隐藏的模块化程序设计。

2. 语言表达能力较强

C 语言除了有高级语言的表达能力外,还特有汇编语言的功能。提供了位操作,地址运算等丰富的运算符,可直接访问物理地址;此外,其表达式类型多样化,还可实现在其他高级语言中难以实现的运算。因此,用 C 语言可编写系统软件(操作系统,编译系统,数据库管理系统等)和一般应用程序。

3. 数据类型较丰富

C 语言除了基本数据类型(如整型,实型,字符型等)以外,还有构造类型(如数组,结构,联合,位段等)和指针类型等。这样,可进行各种复杂的数据结构运算,比如链表,栈,树等的运算。指针类型的使用是 C 语言的显著特点,使参数传递简单,迅速,节省内存。这也是其他高级语言(除 PASCAL 外)所少有的。

4. 语言简洁

C 语言的基本语句少,书写简单,自由,使初学者容易掌握。它一共才有 32 个关键字

(保留字) 和 45 个标准的运算符 (也称为操作符)。除了在表示方法上尽可能简洁外, C 语言中的许多成分都是通过显式的函数调用来完成, 因而所写的 C 语言程序非常简洁而功能很强。

5. 可移植性好

在 C 语言的语句中，没有依赖于硬件的输入/输出语句，需要和硬件打交道，则是通过调用编译系统提供的标准库例程序（或标准库函数）来实现的，而这些标准库例程序不是 C 语言的组成部分。从而使得 C 语言程序容易从一种计算机环境上不加修改或少加修改，就可以移植到与该计算机环境不同的计算机系统上。这也是 C 语言广泛应用的主要原因之一。

6. 生成的代码质量较高

许多实验表明,针对同一问题,用 C 语言描述,其代码效率只比用汇编语言描述低 10%~20%。由于用 C 语言描述问题比用汇编语言描述问题(编写程序)迅速,可读性好,再加上 C 语言的可移植性又较好,因此, C 语言就成为人们描述系统软件和应用软件比较理想的工具。在代码效率方面, C 语言的确可以和汇编语言相差无几。

以上所述是 C 语言的主要特点。当然, 和其他高级语言一样, C 语言也有其不足, 比如, 它不如一些高级程序设计语言 (如 PASCAL, ADA) 的类型性强, 类型转换比较随便, 不够安全; 它没有动态数组界的检查; 运算符优先级太多, 容易混淆等。这些缺点和它的优点相比是次要的, 并不影响它为一个富有生命力的描述工具。

1.3 C 语言的词汇与程序结构

前已指出，程序设计语言总是由若干词法单位和语法规则所构成的语句集来体现的。C语言也是一样，有它的词法单位和语法规则。在这里，我们讨论它的基本词法单位——词汇（或单词）和程序的一般结构。

1.3.1 词汇集

C 语言有若干基本词法单位,称为 C 语言的词汇集。该词汇集由标识符,关键字,常数,字符串,运算符及其他分隔符号等组成,而这些词汇都由字符构成。

1. 字符集 (Characters Set)

字符是构成词汇的基本元素，不同机器的字符集可能是不同的。目前国际上常用的有 ASCII 码（美国信息交换标准码：American Standard Code for Information Interchange）和 ISO 码（ISO 是国际标准化组织 International Standards Organization 的缩写）。C 语言中使用合法字符集（ASCII 码）：

小写字母: $a, b, c, \dots, x, y, z$

大写字母: A, B, C, ..., X, Y, Z

数字: 0, 1, 2, ..., 9

特殊字符: +, =, -, _, (,), *, &, %, \$, #, !, |, ., ,, ;, :, ", ', /, ?, <, >, \, ^, ~, `

{,}, ~, \, [,], -

不可打印字符: 空白符 (包括空格符, 换行符, 制表符)

2. 标识符 (Identifiers)

标识符是用来标识常量、变量、类型、函数及程序的名字。在 C 语言中构成标识符必须符合下列语法规则:

- (1) 以字母或下划线 (可作为字母对待) 作为标识符的第一个字符;
- (2) 在第一个字符之后, 可以是任意的字母、数字、下划线组成的字符序列, 这个序列可以是空串。例如 A2 _ah Sm _ep beta 等都是标识符。

在 C 语言中, 标识符的长度可以任意, 而 ANSI C 标准则为 31 个字符。

3. 关键字 (Keywords)

关键字是一类特殊的标识符 (也称作系统标识符或保留字), 它们是 C 语言中具有特定严格含义的一些单词, 不得赋予其他含义, 因此不允许在 C 语言程序中另作它用。标准 C 语言规定的关键字为 (共 32 个):

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

在 Turbo C 中还规定了一些非标准关键字 (共 27 个):

asm	_ax	_ah	_al
cdecl	_bx	_bh	_bl
near	_cx	_ch	_cl
far	_dx	_dh	_dl
huge	_bp	_sp	_di
interrupt	_si	_cs	_ds
pascal	_es	_ss	

以上所有关键字和非标准关键字必须按上述给定形式书写。此外, 在 C 语言中还规定了具有特定含义的特定字, 虽然不是关键字, 建议也不要把它们当作一般标识符使用, 以免造成混乱。特定字为:

define include undef ifdef ifndef endif line

这些特定字主要用于 C 语言的预处理程序中。

C 语言的词法单位还有常数, 字符串, 运算符及其他分隔符等, 这将在下一章叙述。

1.3.2 语法描述

语法是语言的结构规则, 所有的语言成分都必须合乎语法。如果语言的结构规则描述不严密, 就会导致歧义。常用的语法描述工具有扩充的巴科斯—劳尔形式体系 (Extended

Backus - Naur Formalism, 简称为 EBNF)*。用 EBNF 描述程序设计语言的语法, EBNF 本身也可看作是一种语言, 相对它所描述的程序设计语言来说, 称它为元语言, 它所描的程序设计语言称作对象语言。对象语言的语法规则 (或称产生式) 表现为一组元语言公式, 每个元语言公式有元语言变量 (通常用尖括号 “ $\langle$ ”, “ $\rangle$ ” 括起), 元语言符号和对象语言由基本词汇集中的词汇组成。

元语言变量是用尖括号括起来的一个元素, 它表示对象语言的一个语法单位。下列符号是 EBNF 的元语言符号, 但它们不是对象语言的符号:

$:: =$ 表示 “定义为”

$|$ 表示 “或者”

$\{ \}$ 表示花括号内的符号可以重复 0 次或任意有限多次

$[\]$ 表示方括号内的符号可以重复 0 次或 1 次, 也即可选或不选

$()$ 表示一种组合, 即 $(X | Y)$ 指不是 X 便是 Y, 也即必选其一

例如, 在 C 语言中:

$\langle \text{字母} \rangle :: = \langle \text{大写字母} \rangle | \langle \text{小写字母} \rangle | _$

$\langle \text{大写字母} \rangle :: = A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S$
 $| T | U | V | W | X | Y | Z$

$\langle \text{小写字母} \rangle :: = a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u |$
 $v | w | x | y | z$

$\langle \text{数字} \rangle :: = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$

其中, $\langle \text{字母} \rangle$, $\langle \text{大写字母} \rangle$, $\langle \text{小写字母} \rangle$, $\langle \text{数字} \rangle$ 都是元语言变量。就字母而言 (在这里把下划线当作字母看待), 它由三个元语言公式表示。

在 C 语言中, 使用了与 EBNF 中的元语言符号相同的符号。为了区别起见, 将元语言符号作些修改:

用 $|$ 代替 $|$

用 $\dots$ 代替 $\{ \}$

用 $[]$ 代替 $[]$

用 $\backslash \backslash$ 代替 $()$

例如标识符语法公式则写为:

$\langle \text{标识符} \rangle :: = \langle \text{字母} \rangle [\langle \text{字母} \rangle | \langle \text{数字} \rangle] \dots$

本书将采用代替符号的 EBNF 来描述 C 语言的语法公式。

* 注: BNF 也可用作 Backus Normal Form 的缩写, 中文为 “巴科斯范式”。

1.3.3 C 语言的程序结构

在 C 语言中, 函数是构成程序的基本单位。函数由函数头, 函数体两部分组成, 函数头包括函数类型说明, 函数名, 函数标志和参数说明等, 函数体由函数语句组成 (用花括号括起来)。C 语言程序结构主要由一个或多个函数构成, 但其中有且仅有一个名字为 main 的函数 (称为主函数), 程序的执行是从主函数开始的。其余的函数名字可以任意取。由于 C 语言的输入/输出及有关控制功能, 数学标准函数等均为编译程序提供的库函数来支持, 因此在程序中还要给出这些库函数的文件名称, 这便是预处理部分。可见一个 C 语言程序一般

由预处理部分，外部（或全局）变量说明部分，主函数部分和其他自定义函数部分组成。除了主函数部分外，其他部分可以缺省（即不需要这些部分）。现用例子来对 C 语言程序给予说明。

【例 1-1】的程序。

这个程序结构由预处理，全局变量说明，主函数三部分组成。其中，

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

这是预处理命令，它告诉编译程序将标准输入输出文件 `stdio. h` 嵌在本程序中，因为本程序中要用到这个文件中的有关库函数。在这里，预处理部分只有这一条命令。接下来是全局变量说明部分，这里也只一条语句：

```
long password = 48857358;
```

它说明 `password` 是一个全局变量，其类型为长整型，且具有初值：48857358。它用于存放应用程序的口令。再往下就是主函数部分。`main ()` 是函数头（这里没有给出参数说明）；接下来就是主函数的函数体，由左花括号“{”开始，最后由右花括号“}”结束。函数体中，其第一个语句：

```
long pass;
```

它说明 `pass` 是长整型变量。其第二个语句：

```
printf ( "\ nPlease input password : ");
```

它的作用是在屏幕上显示信息：‘Please input password:’，告诉用户输入口令。其第三个语句：

```
scanf ( "%d", &pass);
```

它的作用是接受用户输入的口令，并将其保存到变量 `pass` 中。其第四个语句是一个选择语句，其格式是：

```
if ( <条件>) <语句 1> else <语句 2>
```

它的作用是：如果条件成立，则执行语句 1，否则，执行语句 2。在这里条件是：

```
pass! = password
```

如果条件成立，表示输入的口令不相符，则执行复合语句：

```
printf ( "\ nSorry, You can't use !");
```

```
goto exit;
```

即在屏幕上显示信息：“Sorry, You can't use!”，并转移到标号为 `exit` 语句去执行。注意，这个语句在程序的尾部：‘exit:’，接下去是主函数结束的右花括号‘}’，从而退出应用程序。如果条件不成立，即输入的口令相符，则执行语句：

```
printf ( "\ nWelcome, You can use !");
```

在屏幕上显示信息：“Welcome, You can use!”，接下去是执行应用程序的后继语句，在程序中用‘……’表示。再者程序中还有以‘/*’开始，以‘*/’结束的注释语句，其中的内容用以对程序的相关部分作注释，它是不执行的。注释不允许嵌套。

【例 1-2】设有如下程序：

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

```
main ()
```

```
{ int w=0, x=30, y=60;
```

```

int  z=10;

z=max (x, y);          /* 调用函数 max () */
w=max (z, w);          /* 第二次调用函数 max () */
printf ( "max=%d\n", w); /* 输出最大数 */

}

int  max (int a, int b)    /* 函数 max (), 其形式参数 a, b 为整型变量 */
{   if (a>b) return (a); else return (b); }

```

这个程序由预处理, 主函数及函数三部分组成。函数 max () 是求两个整数的最大者, 其中选择语句中的 return (a) 是返回语句。在主函数 main () 中, 第一个语句是说明变量 w, x, y 为整型变量, 它们分别具有初值 10, 30 和 60。第三个语句是赋值语句, 它调用函数 max (x, y), 找出 x, y 中的大者作为该函数的值并赋给 z。第四个语句仍是赋值语句, 它调用函数 max (w, z), 找出 w, z 中的大者作为函数的值赋给 w。最后用 printf () 语句输出 w 的值 (即为 x, y, w 中的最大者)。

1.4 C 语言程序上机操作步骤

用高级程序设计语言编写的程序称为源程序, 它不能直接在计算机上运行, 还必须用相应的翻译程序将其转换成机器语言程序后, 才能在机器上运行 (执行), 被翻译后的机器语言程序称为目标程序。翻译程序一般有两种工作方式, 其一是解释方式, 其翻译程序称为解释程序, 它是对源程序边翻译 (或解释) 边执行 (不直接保留目标代码)。其二是编译方式, 其翻译程序成为编译程序, 它是将源程序翻译成目标程序。这种目标程序可以保存起来, 暂不执行, 也可保存并在下一步 (运行阶段) 执行。因此, 高级语言程序的执行, 一般要经过以下几步 (也称为上机操作步骤):

1. 建立源程序

这一步是将源程序建立起来, 通常是利用编辑程序, 将编好的源程序语句输入到计算机中, 建立一个源程序文件。如发现有错误可进行改正。

2. 编译

这一步是利用该语言的编译程序将源程序翻译成目标程序, 建立一个目标程序文件。如发现有语法错误或其他错误, 则将源程序进行修改, 即返回上一步用编辑程序对源程序中的错误给予改正, 再次回到第二步, 用编译程序对修改后的源程序进行编译。

3. 连接

这一步是利用连接程序, 将多个分别编译的目标程序连接成一个完整的可执行的目标程序。

4. 运行

这一步是将可执行的目标程序在机器上运行, 取得该程序的运行结果。如运行中有错, 则可利用排错程序进行排错, 或者下机后, 对输出的错误信息进行分析, 乃至修改源程序、重新编译、连接、运行。