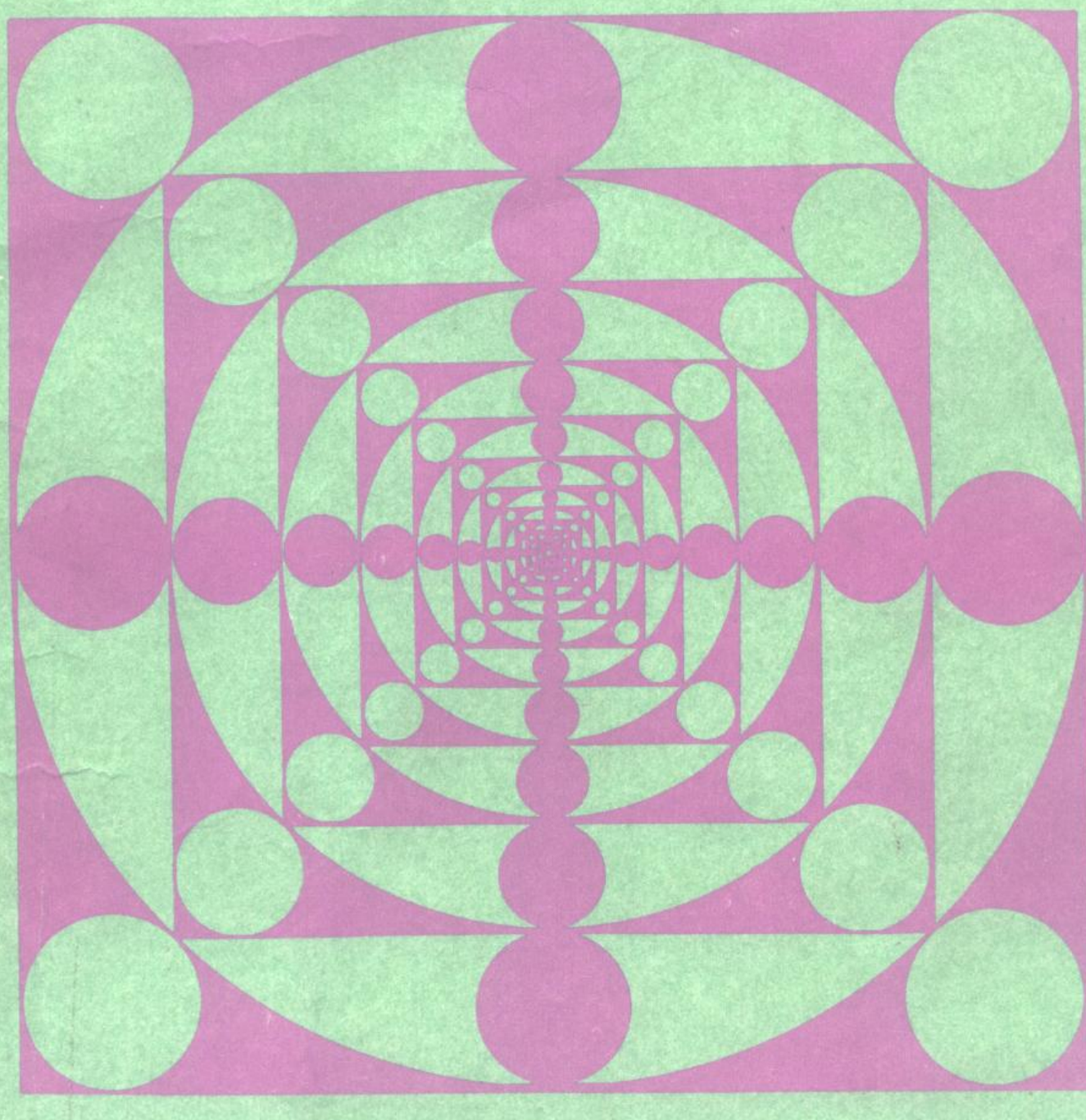
电子计算机应用系列教材

C语言程序设计基础

李成付　陈世鸿　刘良观　编著



科学出版社

电子计算机应用系列教材

C 语言程序设计基础

李成付　陈世鸿　刘良观　编著

科学出版社

1992

(京)新登字 092 号

内 容 简 介

本书为电子计算机应用系列教材之一. 本书系统地讲述用 C 语言进行程序设计的方法. 全书共十二章，第一章综述 C 语言的发展、基本概念与特征，C 程序的基本框架和书写方法. 第二章到第十章系统地介绍 C 语言程序设计的各种基础知识和方法，如数据、算符、表达式、函数和数组、指针、结构及输入/输出功能. 第十一章叙述 C 的编译功能以及它与操作系统的接口，如源文件的建立、编辑与修正、编译、运行等. 第十二章对一个稍大型的 C 程序实例进行综合分析，以建立整体设计概念. 本书可作为在职科技人员、大学生及其他人员学习 C 语言的教材；也可供从事计算机应用工作的科技人员参考.

电子计算机应用系列教材

C 语言程序设计基础

李成付 陈世鸿 刘良观 编著

责任编辑 那莉莉

科学出版社出版

北京东黄城根北街 16 号

邮政编码：100707

湖北省武汉市华中电子信息产业开发集团公司激光照排

天津市静一胶印厂印刷

新华书店北京发行所发行 各地新华书店经售

★

1992 年 2 月第 一 版 开本：787×1092 1/16

1992 年 2 月第一次印刷 印张：15 1/4

印数：0001—7 200 字数：344 000

ISBN7-03-001355-7/TP · 90

定价：9.70 元

电子计算机应用系列教材主持、组织编著单位

主持编著单位：

国务院电子信息系统推广应用办公室

组织编著单位(以笔划为序)：

广东、广西、上海、山东、山西、天津、云南、内蒙古、
四川、辽宁、北京、江苏、甘肃、宁夏、江西、安徽、
河北、河南、贵州、浙江、湖北、湖南、黑龙江、福建、
新疆、广州、大连、宁波、西安、沈阳、武汉、青岛、
重庆、哈尔滨、南京等35省、市、自治区、计划单列市

电子振兴
计 算 机 领导小组办公室
科技工作

电子计算机应用系列教材联合编审委员会名单

(以姓氏笔划为序)

主编审委员：

王长胤* 苏世生 何守才 陈有祺 陈莘萌* 邹海明* 郑天健
殷志鹤 童 颋 赖翔飞 (有"*"者为常务主编)

常务编审委员：

于占涛 王一良 冯锡祺 刘大昕 朱维华 陈火旺 陈洪陶 余 俊
李 祥 苏锦祥 佟震亚 张广华 张少润 张吉生 张志浩 张建荣
钟伯刚 胡秉光 高树森 徐洁盘 曹大铸 谢玉光 谢育先 韩兆轩
韩培尧 董继润 程慧霞

编审委员：

王升亮 王伦津 王树人 王振宇 王继青 王翰虎 毛培法 叶以丰
冯鉴生 刘开瑛 刘尚威 刘国靖 刘晓融 刘德镇 孙令举 孙其梅
孙耕田 朱泳岭 许震宇 何文兴 陈凤枝 陈兴业 陈启泉 陈时锦
邱玉辉 吴宇尧 吴意生 李克洪 李迪义 李忠民 迟忠先 沈林兴
肖金声 苏松基 杨润生 闵福德 张志弘 张银明 张 勤 张福源
张翼鹏 郑玉林 郑 重 郑桂林 孟昭光 林俊伯 林钧海 周俊林
赵振玉 赵惠溥 姚卿达 段银田 钟维明 袁玉馨 唐肖光 唐楷全
徐国平 徐拾义 康继昌 高登芳 黄友谦 黄 侃 程锦松 楼朝城
潘正运 潘庆荣

秘 书 组：

秘 书 长：胡茂生
副秘书长：何兴能 林茂荃 易 勤 黄雄才

序

当代新技术革命的蓬勃发展，带来社会生产力新的飞跃，引起整个社会的巨大变革．电子计算机技术是新技术革命中最活跃的核心技术，在工农业生产、流通领域、国防建设和科学研究方面得到越来越广泛的应用．

党的十一届三中全会以来，我国计算机应用事业的发展是相当迅速的．到目前为止，全国装机量已突破三十万台，十六位以下微型计算机开始形成产业和市场规模，全国从事计算机科研、开发、生产、应用、经营、服务和教学的科技人员已达十多万人，与1980年相比，增长了近八倍．他们在工业、农业、商业、城建、金融、科技、文教、卫生、公安等广阔的领域中积极开发应用计算机技术，取得了优异的成绩，创造了显著的经济效益和社会效益，为开拓计算机应用的新局面作出了重要贡献．实践证明，人才是计算机开发应用的中心环节．我们必须把计算机应用人才的开发与培养放在计算机应用事业的首位，要坚持不懈地抓住人才培养这个关键．

从目前来看，我国计算机应用人才队伍虽然有了很大的发展，但是这支队伍的数量和质量还远不适应计算机应用事业发展的客观需要，复合型人才的培养与教育还没有走上规范化、制度化轨道，教材建设仍显薄弱，培训质量不高．因此，在国务院电子信息系统推广应用办公室领导、支持下，全国三十五个省、市、自治区、计划单列市计算机应用主管部门共同组织118所大学和科研单位的400多位专家、教授编写了全国第一部《电子计算机应用人才培训大纲》以及与之配套使用的电子计算机应用系列教材，在人才培训和开发方面做了一件很有意义的工作，对实现培训工作规范化、制度化将起到很好的推动作用．

《电子计算机应用人才培训大纲》和电子计算机应用系列教材贯穿了从应用出发、为应用服务，大力培养高质量、多层次、复合型 应用人才这样一条主线．大纲总结了近几年各地计算机技术培训正反两方面的经验，提出了计算机应用人才的层次结构、不同层次人才的素质要求和培养途径，制定了一套必须遵循的层次化培训办学规范，编制了适应办学规范的“课程教学大纲”．这部大纲为各地方、各部门、各单位制定人才培养规划和工作计划提供了原则依据，为科技人员、管理人员以及其他人员学习计算机技术指出了努力方向和步骤，为社会提供了考核计算机应用人才的客观尺度．“电子计算机应用系列教材”是培训大纲在教学内容上的展开与体现，是我国目前规模最大的一套计算机应用教材．教材的体系为树型结构，模块化与系统性、连贯性、完整性相兼容，教学内容注重实用性、工程性、科学性，并具有简明清晰、通俗易懂、方便教学、易于自学等特点，是一套很好的系列教材．

这部大纲和系列教材的诞生是各方面团结协作、群策群力的结果，它的公开出版和发行，对计算机应用人才的培训工作将起到积极的推动作用．希望全国各地区、各部门、各单位广泛运用这套系列教材，发挥它应有的作用，并在实践中检验、修改、补充和完善它．

通过培训教材的建设，把培训工作与贯彻国家既定的成人教育、函授教育、电视教育

和科技人员继续工程教育等制度相结合，逐步把计算机应用人才的培训工作引向规范化、制度化轨道，为培养和造就大批高素质、多层次、复合型计算机应用人才而努力奋斗，更好地推动计算机应用事业向深度和广度发展.

李祥林

1990年10月17日

前　言

本书是写给想用C语言去写程序的人们，即使完全不了解C语言的人也可以阅读，而具有一定C语言基础的读者又可从中获得更多的编制程序的技巧.

当今时代，计算机的应用特别是微型机的应用在不断发展，电力、机械、化工等领域的工业自动化控制，企事业的现代化管理，人工智能与知识处理乃至家庭的事务管理都要用到微机及各种知识信息，而用C语言作各种系统的程序设计是很方便、有效的.

C语言以其简洁、通用、易学和高效性，受到广泛关注，成为世界上有影响的语言之一. C是一种通用的程序设计语言，目前使用最多的是UNIX流的C和在MS DOS下运行的C，但C并不局限于某种具体的计算机，它以系统程序设计语言著称，可以方便地写操作系统、编译系统以及各种应用软件. 它提供了定义新的数据结构的功能，具有程序紧凑、结构清晰、便于模块化设计和移植等特点，且同其他高级语言一样，具备一些基本的功能，如丰富的算术运算符、实用的表达式、方便的控制流结构和数据类型，函数调用和函数递归等.

此外，C语言中备有函数库，如输入输出函数库、数学函数库、浮点库、标题库等等. 在程序的头部装入这些函数库，程序运行中即可随时使用库中的各种标准函数.

当然C并不是一个"大"语言，其本身并不提供一些处理复合对象的操作，如字符串、表和集合等，也没有直接的如READ和WRITE一类的输入输出语句，更谈不上多道程序和并行操作，但对这些不足，可通过函数调用解决.

初跨入C语言大门的读者首先要了解C语言的基本概念、语法、语句、程序结构和程序的书写方法. 本书正是以深入浅出的叙述、容易理解的形式和丰富的例子对此进行了说明.

本书并非包揽C的一切，特别是对在不同操作系统下运行的C及其不同的高层次功能只一般触及. 但学习了本书并通过相应的教学实习，写出有相当规模的C程序应当是不困难的，想用C语言作大规模系统开发的读者，必须结合具体的计算机进一步了解C的有关功能.

本书共十二章，每章后均附有丰富的习题，供读者编写程序和上机实习. 第一章综述C语言的发展、基本概念与特征，C程序的基本框架和书写方法，以建立初步概念. 第二章到第十章系统介绍C语言程序设计的各种基础知识和程序设计方法与技巧，如数据、算符、表达式、函数及各种宏功能，数组、指针、结构以及输入输出功能等. 第十一章叙述C的编译功能以及它与操作系统的接口，如源文件的建立、编辑与修正、编译、运行、结果获取等. 第十二章介绍C作为开发工具的应用价值，本书还给出一个稍大型的C程序实例并对其进行综合分析，以建立整体设计概念.

在本书的编写中，得到了陈莘萌教授的指导，易勤参加了大纲的编写和讨论，潘正运副教授审阅了全稿，在此一并表示感谢.

目 录

第一章　C语言概述及书写方法

C是1972年由美国BELL实验室研制开发的通用程序设计语言. 设计师是D. M. Ritchie. 其后不久,Ritchie和Ken Thompson一起用C进行了UNIX操作系统的开发. 由于C的简洁、通用和高效性,C很快受到广泛的关注和重视.

进入70年代中期,C被推广到很多大学,深受社会和用户的欢迎. 到了80年代,好几家计算机公司相继发表了C的编译系统,并很快从UNIX系统推广到其他系统.

C属于传统的语言体系,继承了程序设计语言的传统特性——可靠性、规则性、简洁性和容易使用性. 这种体系称为结构化语言.

结构程序设计在70年代是程序设计方法的主流,其中心语言是C以外的语言,其色彩最浓的是堪称C姐妹的Pascal. 而C不仅将各种功能,而且将可靠性、方便性和通用性放在重点,其优良的结构特性可以在结构程序设计中担当重任,并且很可能成为今后软件开发的主力语言.

C经历了以下的发展过程:

ALGOL　60
(1960年　国际委员会设计)
↓
CPL　〈Combined Programming Language〉
(1963年　伦敦大学设计)
↓
BCPL
(1967年　剑桥大学设计)
↓
B
(1970年　BELL实验室Ken Thompson设计)
↓
C
(1972年　BELL实验室D. M. Ritchie设计)

ALGOL尽管比FORTRAN晚几年问世,但它是非常精炼的语言,对其后的程序设计语言的发展有很大影响,但由于太抽象,以致在美国没有形成主流. 为了更容易、更有效地使用ALGOL,不久又出现了CPL语言. 但CPL与ALGOL一样仍然难记难用,不能充分地发挥语言的效率. 为了克服这些问题,产生了BCPL.

其次是B,是由Ken Thompson在UNIX上开发的语言. B很适应当时的硬件. 但是B和BCPL一样,功能有限,应用范围也有限,仅对特定问题的解决比较方便. 为了解决通用

性问题和数据类型过少的问题，1972 年，D. M. Ritchie 等设计了 C 语言. C 有容易理解的规则，用 C 的局部小模块或函数可以组合成复杂的整体模块，进而形成 C 的优良结构. 如同用十几种元素可以组合成有机化合物，用 12 个音阶可以组成交响乐一样.

对于 C 语言来说，用简单的元素构成复杂的程序的能力是很强的. 由于 D. M. Ritchie 的研究领域是系统软件，所以在实现计算机语言和操作系统等工具时，C 最能充分发挥效率. 另外，也可以用 C 语言去写一般软件和图形以及游戏软件等.

C 基本上是属于面向问题一级的语言. 用 C 所写的程序，执行速度比较快，且只使用很少的存贮空间. 用 C 语言写的编译程序目前广为流传，类型较多，有在 8 位字长的计算机的 CP/M 操作系统上运行的 BDSC 编译系统，有在 16 位字长计算机的 MS DOS 上运行的 Lattiec C 编译系统，还有在 UNIX OS 上运行的 C 编译系统等等.

1.1 简单程序及其要领

学习 C 语言的目的是学会用 C 语言去写程序. 我们将遵循这一目标，从典型例子入手，一步一步地去了解 C 程序的结构.

例如，用 C 语言打印一个早晨见面的礼貌用语“早上好!”：

```
main( )
{
  printf("Good morning!. \n");
}
```

这四行简单得不能再简单的程序是完全合法的，执行此程序，在终端屏幕上显示出下面的字样：

```
Good  morning!
```

程序虽小，但体现了 C 的基本组成：main( )是 C 程序的主部，其自身是一个函数. 任何 C 程序，无论大小，都必须具有一个 main( )函数，它指明程序运行的起始位置. 由花括号括起来的是执行语句，main( )的后面必须跟这样一对花括号. 本例中只有一个执行语句，每一语句末尾用分号结束，否则将出现语法错误. 语句 printf 实际上是系统中的一个函数，它输出信息到终端. 其中的双引号“ ”是 printf 的输出格式，例中输出的是字符串 Good morning!. 符号\n 是换行符，即输出字符串后自动跳到下一行. 如果省略\n，即

```
printf("Good morning!");
```

也可以执行，印出相同的字符串，但不再转行.

在 C 中，源程序必须经编译，被确认完全无误后方能运行. 为此首先要在 OS F 建立源程序文件，例如 file-1-1. C 这样的 C 文件，经过 C 编译程序的语法检查，生成可执行文件，最后运行程序得到所求的结果. 重写程序如下：

file-1-1. C：

```
main( ) / * print string"Good morning!" * /
{
  printf("Good morning!. \n");
}
```

印出字样：

Good morning!

当改成以下形式：

```
main( )
{
  printf("Good morning!. \n");
  printf("Good evening!. \n");
}
```

印出字样为：

Good morning!

Good evening!

当略去前面一个 printf 中的换行符\n 时，印出字符变为：

Good morning! Good evening!

这是最简单的情形，C 的实际情形远非如此. 下面给出的是关于数字转换的稍复杂的例子.

1949 年 10 月 1 日是中华人民共和国成立纪念日. 试用 C 语言将数字 1949 转换成字符显示在终端上. 我们用在 MS DOS 上运行的 C 编译系统，首先建立名为 file-1-2. C 的源程序文件，并将它保存在磁盘内供显示、修正和编译. 源程序如下：

file-1-2. C：

```
main( ) / * convert 1949 to character * /
{
  int a=1949 ;
  a=romanize(a,1000,'m');
  a=romanize(a,500,'d');
  a=romanize(a,100,'c');
  a=romanize(a,50,'l');
  a=romanize(a,10,'x');
  a=romanize(a,5,'v');
  romanize(a,1,'i');
  putchar('\n');
}
romanize(i,j,c) / * output c for i,j and return i-j * /
char c;
int i,j;
{
  while(i>j)
  {
    putchar(c);
    i=i-j;
  }
    return(i);
```

}

本程序是将 1949 转换成由 m,a,d,l,x,v,i 等字母组成字符串输出. 程序从 main 开始运行,该程序 7 次调用用户自编函数 romanize. romanize 函数从当前 a 中减掉第二个参数(如 1000,500,…)的值后作为新值返回,并输出相应的字母(如 m,d,c,…). 主程序中的第三行 int a=1949 说明变量 a 是一个整型变量,并赋初值 1949,当说明为

```
int a;
a=1949;
```

其效果是一样的. 函数 romanize 中的 while(i>j)是循环语句头部,圆括号中为条件表达式,{…}括起来的部分是其循环体,体中的 putchar(c)为输出一个字符,它是标准函数.

这一程序反映了 C 程序的基本结构和要领. 由这一程序可以看出,C 程序由一主程序和一个(或多个)函数构成. 由一个个小函数构成大的程序正是 C 的一个特征. 另外,从上例还可以看出,C 的程序中还有注释,即符号/ * 和 * /括起来的部分. 此外,C 语言同其他高级语言一样,程序中所有用到的变量都要预先加以说明. 例中 romaniza 函数的说明语句

```
char c;
int i,j;
```

说明了 c 是字符类型, i,j 是整型变量. 有关程序中的其他成分,在后面各章中还要详细讨论.

1.2 程序操作和编译

从广义而言,上述程序是将阿拉伯数字转换成字符的例子. 我们主要是想通过说明此程序的运行过程来了解 C 程序的一般特征.

file-1-2. C 程序的执行过程乃是反复的函数调用过程,是作若干次的减法运算. 从 1949 减去 1000 开始,第一次函数调用,显然输出字符 m,由于 1949−1000=949<1000,故 m 只能输出一个. 接着作第二次函数调用,作减法 949−500=449,输出 d 而且仅有一个 d. 以下的执行完全相同,从剩余数中减去 100,50,5 直到 1,各得到相应的输出. 由此可见,C 程序的运行,和其他高级语言有许多类同之处,但也有其自身的特征.

程序文件 file-1-2. C 有没有语法错误呢? 或者是否符合 C 的编译程序所能接受的那种 C 的规则呢? 要回答这些问题,必须对源程序作语法检查,即编译. 其编译过程可表示如下:

file-1-2. C→LC1→file-1-2. Q→LC2→file-1-2. OBJ

LC1,LC2 是 MS-DOS 下运行的一个 Lattice C 编译程序. LC1 的作用是对源程序作预处理,还作词法分析与语法检查,所有语法错误都在这一步检出. LC1 生成一个中间文件 file-1-2. Q. LC2 的作用是接收中间文件,并生成 Intel 8086 的格式目标文件 file-1-2. OBJ. 至此,程序的错误全部排除,并且已经格式化了. 最后,通过连接处理生成一个可执行文件 file-1-2. EXE (. EXE 是可执行文件的扩展名),运行这一文件就能得出所要求的结果. 上述过程在机器上的操作过程如下:

```
A>LC1 file-1-2. C    〈CR〉    第一遍编译
A>LC2 file-1-2. Q    〈CR〉    第二遍编译
```

A>Link C file-1-2 〈CR〉 连接

A>file-1-2. EXE 〈CR〉 执行

mdccccxxxxviiii 结果

其中 CR 是回车键，Link 后面的 C 即是 C. obj，是 C 程序的入/出口模块. A>是系统提示符.

1.3 C 的函数及其定义和名称

程序 file-1-2. C 的动作是根据两个函数进行的. 函数是 C 程序的基本组成部分和基本动作机构. C 的一般程序都有一个或多个函数，不同的函数完成各自不同的任务. 上例程序中的函数是 main 和 romanize. main( )表示 C 的主体程序的开始. 用户函数 romanize 一面执行 C 的库函数 putchar 在终端输出字符，一面进行减法操作. 各个函数互相独立，彼此平等. 但对某一函数来说，它的优先级也可能比其他的高一些，如函数 main( )，标明 C 程序中的主体位置，常常作为特殊的函数最先被执行. 在一个实际的大型 C 程序中包含着许多的定义函数和库函数调用，它们和 C 的控制机能一起构成了 C 程序的优良层次结构.

在 C 中像 Putchar 这样方便的标准库函数虽然比较多，但我们最感兴趣的是用户自定义函数，自定义函数由头部和体部组成，头部放在体部前面，给出函数的名称和自变量；而体部定义了该函数的全部操作. 我们详细地考察一下源程序文件 file-1-2. C 中的定义函数 romanize. 它虽然短，却包含了函数操作的重要部分.

```
romanize(i,j,c)
char c;
int i,j;
{
  while(i>=j)
  { putchar(c);
    i=i-j;
  }
  return(i);
}
```

前 3 行是函数头部，后面部分是函数体. 第一行是函数名，括号中的文字是形式变量(形参)，在函数调用时，首先给形参代入实际数据. 第二、三行是对形参的类型作详细说明，以便告知编译. 由此可见，定义函数头部由三部分组成：函数名、形参和说明部分.

值得注意的是函数 main( )中括号内为空，没有形参. 这没有关系，C 中的函数可以不带形参，主程序通常是不带参数的特殊函数.

定义的函数体包含着可执行语句，它进行实际的操作. romanize 的体部如下：

```
{ while(i>=j)
  { putchar(c);
    i=i-j;
  }
```

```
    return(i);
}
```

由分号结束的行称为一个语句，由花括号括起来的部分称为复合语句. 复合语句一般都含两个或两个以上的语句. 凡用花括号括起来的部分，编译时把它当作一个综合部分来考虑. 上述本体的意义是：当 while 的条件 i>=j 为真时，内花括号括着的两个语句反复执行，否则，执行 return 语句返回到函数调用.

通常程序启动执行时必须输入数据. 有几种数据，如 1949，1000，'m' 这样的常数，如 a 那样的变量. 一般地说，变量在使用之前必须先赋值，就像要从杯中喝水，必须先将水倒入杯中一样. 而 romanize 的 i，j，c 稍有不同，它们是一种特殊数据，它们须从函数调用的实参数中取得数据.

花括号的位置影响着执行语句的顺序，也可以给编译过程以帮助. 请看下面两段程序，它们执行的结果是完全不同的：

```
(1)  while(i>=j)
     { putchar(c);
     }
       i=i-j ;
(2)  while(i>=j)
     { putchar(c);
       i=i-j;
     }
```

所以，作 C 的函数定义时，花括号的使用和程序行从何处开始都是要注意的问题，因此用不用或如何用花括号对程序正确运行都有很大意义.

程序文件 file-1-2. C 的源代码形式原则上也可以用如下形式出现：

```
main( ) { int a=1949;a=romanize(a,1000,'m'); a=romanize(a,500,'d');a=romanize
(a,100,'c'); a=romanize(a,50,'L');a=romanize(a,10,'x'); a=romanize(a,5,'v');ro-
manize (a, 1,'i'); putchar ('\n') ;} romanize (i, j, c) char c; int i, j; { while (i>=j) {
putcharcc);i=i-j;} return(i);}
```

它和原来表示形式的编译及所得结果是完全相同的. 然而，很容易明白，即使刚刚开始学语言第一次动手写程序的人也断然不会采用此种形式，而采用原来的表示形式. 原因不是别的，而是可读性太差.

在 C 程序中被使用的名字有常数名、变量名和函数名. 它们都遵循同样规则，那就是名字由字符和数字组成，且一定要由字母开头. 下线(_)也和字符一样使用. 一个名字中间不允许存在空格，所以当名字较长时，为了区分各个部分可以使用下线. 如变量名

```
interest_to_date
```

是合法的. 另外，C 中大写字母与小写字母具有不同的含义，需注意加以区分. 如

```
VELOCITY
Velocity
velocity
```

是三个完全独立的名字. 一个名字的字符长度多少无关紧要，C 编译时只取前 8 个字符为

有效字符.就编译程序来说,名字

honorific 和 honorificcabilitudinata

interest_paid 和 interest_paydable

是完全相同的.

在程序中使用的名字应尽量用易读易写的形式,并能精炼地表示其含义,如函数完成何种操作,变量表示何种意义等,使之一目了然,容易明白.

关于变量名还有一个必须注意的问题,那就是用户指定的名字不准使用下面列出的保留字:

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

因为所有保留字在编译时都有特殊的含义,这将在后面各章加以叙述.现在只要记住在用户程序的函数名和变量名中不能使用它们就行了.

函数是彼此独立的,这在前面已经说过了.然而在它们之间如何通讯呢?显而易见,在main( )和 romanize 之间,是用变量与返回值的形式传递数据.我们通过例子说明这一点.追踪C程序的执行,当然要从 main( )开始,而 main( )中最前面的语句是

```
int a=1949;
```

把 a 说明为整型变量,同时赋以初值 1949.第二句是

```
a=romanize(a,1000,'m');
```

其中 a 是变量名,1000 是数值常数,'m'是字符常数.这些实参数和函数 romanize(i,j,c)中的形式参数 i,j,c 一一对应,函数调用时,a 的内容传递给 i,1000 传递给 j,'m' 传给 c.执行 romanize(a,1000,'m')就称为一次函数调用.程序运行到达函数调用处,程序流就从 main( )转换到 romanize.同时,实参值 1949,1000,'m' 就分别复制到形式变量 i,j 和 c,于是 romanize 的本体部的语句就可以利用这些数据开始运行,并首先输出 m,然后作减法 1949−1000,并将 949 返回,如此一直执行下去.最后,执行语句 return,返回到 main( )跳到 romanize 的位置.但是这个返回是指函数内计算了的如 949 这样的值的返回,此值和原来的函数调用作置换.总之,main( )中原来语句为 a=romanize(a,1000,'m');经过调用和取得返回值后,a=949,即由原来的 1949 变成了 949.从而下一语句 a=romanize(a,500,'d'):执行时,传递给 romanize 的值就是 949 了.随着 romanize 的不断调用,a 就由 1949 变到 949,一直变下去,最后到 1.函数 romanize 的作用是根据第一、二变量 i 和 j 重复第三变量 c 的输出,并将 i 的新值返送 main( ).而 main( )函数的作用是执行对函数 romanize 的连续调用,而 romanize 连续调用 putchar 输出特殊字符\n 到终端以实现换行操作.