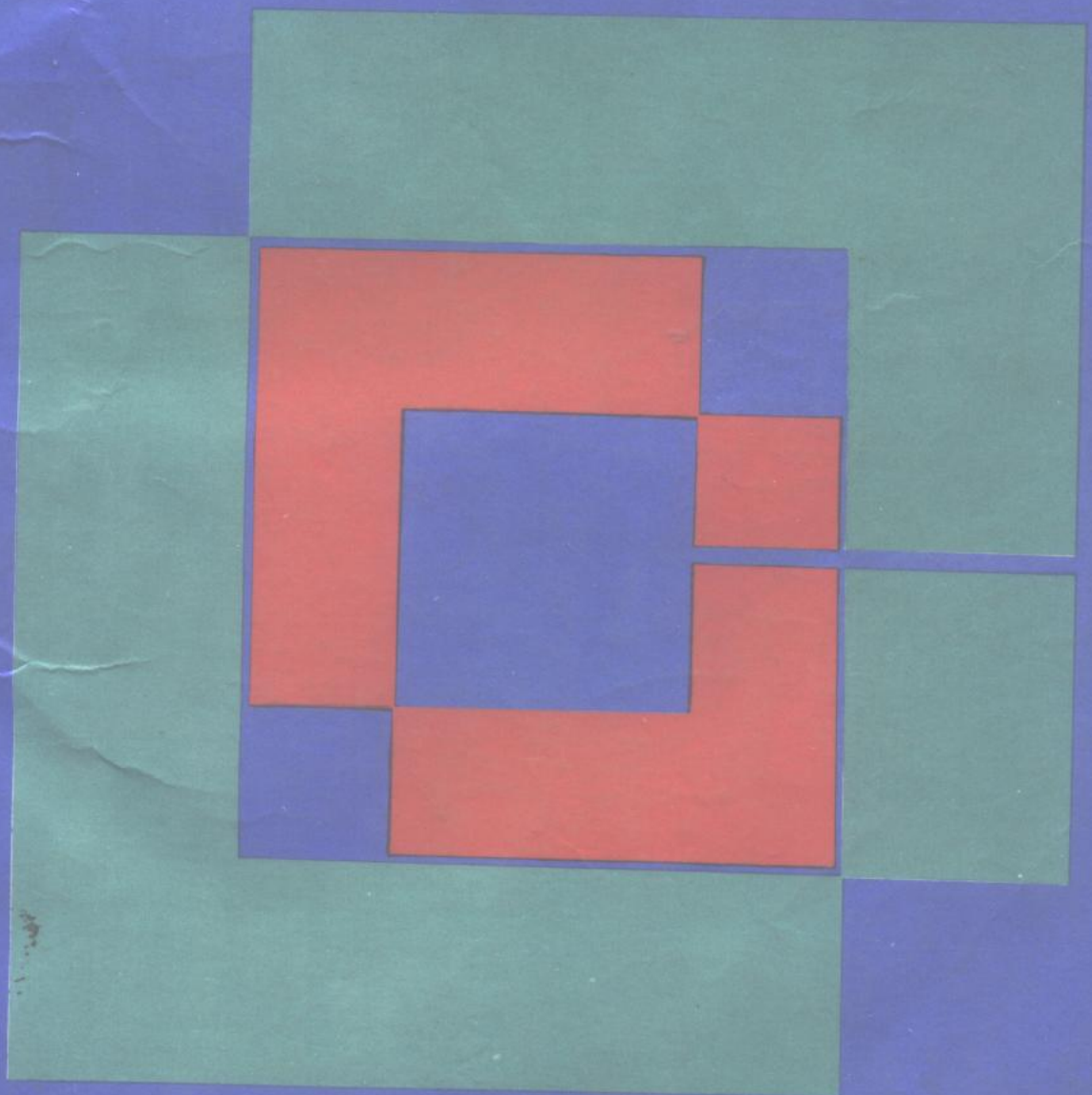
# C语言及其应用

孟庆昌　孙玉方　著



# C 语言及其应用

孟庆昌　孙玉方　编著

宇航出版社

## 内 容 简 介

C语言是最近几年来发展、推广最快的一种通用高级程序设计语言，它具有功能强、效率高、简洁灵活、可移植性好等特点，在软件工程领域里越来越受到人们的普遍重视，其应用日益广泛。

本书较全面、系统地介绍了C语言及其程序设计方法和应用，共分十三章，内容包括C的数据类型、各种语句、程序结构以及程序设计的基本方法、技巧和实际应用，突出C的特点。书中的各类程序都具有实用意义。除第十三章外，各章都附有习题，书后附录列出了AT&T公司最新公布的“C程序员手册”（1985年）和其它实用资料。

本书深入浅出，通俗易懂，内容充实，实用性强。是一本较好的学习和掌握C语言及其应用的教科书。可供高等院校计算机系统、应用和管理等有关专业师生以及各部门的计算机工作者、科研、工程技术人员学习参考。


C语言及其应用

孟庆昌　孙玉方　编著

特约编辑　王荣

*

宇航出版社出版

北京和平里滨河路1号

邮政编码：100013

新华书店北京发行所发行

各地新华书店经售

顺义县印刷厂印刷

*

开本：787×1092 1/16　印张：26　字数：610千字

1988年4月第1版 1991年4月第3次印刷

印数：14001—19000

ISBN7-80034-062-7/TP·006

定价：13.50元

# 前　言

人们经常听到的计算机高级语言是FORTRAN、BASIC、COBOL、PASCAL。但是，随着UNIX操作系统在国际上的广泛流行，高级语言C也越来越为人们所关注。

UNIX系统和其主力语言C是70年代初在美国电话电报公司（AT & T）所属的贝尔实验室(Bell Labs.)诞生的。经过十多年的完善和发展，前者事实上已经成了多用户微型、小型和中型，甚至某些大型机的标准操作系统，而后者已成了所有计算机通用的高级语言。正如美国"Electronics"杂志向其开发者颁发1982年度最高成就奖时所讲的：UNIX系统与C语言的紧密组合对从微型机到大型机的广泛应用产生了重要影响。1983年，UNIX系统和C的开发者又获得了图灵奖。评选委员会的一致评语是："UNIX系统的成功是由于卓越地选用了一些有决定意义的观念，并且准确地加以实现。该系统作为一个样板，已经引导了一代软件设计者思考程序设计的新途径。"而它的成功和广泛流行又"主要是因为有了C。"很明显，要使用和掌握UNIX系统就必须掌握C语言。

不过，虽然C语言最初曾附属于UNIX，并且UNIX仍是其最好的运行环境之一，但是目前它已独立于UNIX而在不同操作系统上以及几乎所有的机器上运行了。

C语言是一种有效而又通用的高级语言，当然，它特别适用于编写应用软件和系统软件。对于广大软件开发者来说C是一个强有力的工具。它也适用于资料、文字处理，所以在事务处理、办公自动化方面能发挥出其特点。对于只有电子学或其它工程知识的人来说，C语言也将成为他们处理问题的得力工具。

我们在UNIX系统和C语言方面的研究、开发和教学工作已有七、八年时间，深感C语言的重要性。1983年为了满足教学需要，我们曾合作编写了一本《C语言程序设计教程》，这本书受到了国内广大读者的欢迎。但是它较多地强调了形式定义，而工程技术人员及一般的计算机工作者对形式定义不一定很感兴趣，他们主要强调其应用。所以许多同事及有关的用户希望我们再编写一本比较实用的侧重于应用方面的书。近两年，在开发系统和教学工作实践的基础上，研究了国内外若干有关书刊，特别是在宇航出版社的编辑同志们的关心和促进下，我们从下半年起开始了本书的编写。

本书通过大量有实用价值的例子讲述C语言的基本成分。使读者不但明瞭C的结构，而且还能掌握它的编程技巧。书中例子都经过上机调试通过，拿来即可使用。

本书偏重于实用，对C的一些词法和语法条文也尽量通过实例来解释。几乎在每章后面都有应用举例，希望以此帮助读者综合该章的知识并能灵活选用，最后还专门用一章来讲述一些重要的应用程序。由于这两年C在某些方面又有了新的发展，在本书中尽量注意把它们反映出来，在附录中给出1985年由贝尔实验室编著、Prentice-Hall, Inc出版的《C程序员手册》供读者参考。考虑到许多用户使用的是PC-DOS操作系统，在附录中给出了在PC-DOS上运行DR/C的有关情况。

当然，由于我们囿于其环境及实践经验，因此可能在某些地方叙述不当，一些例子还可以编得更好一些。我们只想以此抛砖引玉，以利于国内形成更好的C语言方面的

教材。

第一章是引论，讲述C语言的历史演变，以此帮助读者了解其整体结构，也介绍了它的一些特点。并以一些程序为例，阐述它的一些基本成份，为以后各章的展开打下基础。为了使读者能边学边上机实习，还简单介绍了C程序的编辑、编译和运行。

第二章基本类型。引进C的一些常量和若干基本类型，而把在这些类型基础上组成的构造类型放到第八、九、十和十一章去介绍。

第三章运算符。C语言有丰富的运算符，本章分别介绍了其中算术、关系、逻辑、位逻辑、增减量、赋值、条件等运算符，最后给出了其优先级和结合性。对于一些有特点的运算符重点进行剖析。

第四章语句和控制流。着重介绍C的语句和控制结构，分别讲述了若干简单语句、选择、重复以及若干转向语句。

第五章函数。介绍了C程序基本单位及其构成原理。重点是其类型说明、调用方式、函数的递归算法、分别编译和程序连接。

第六章变量存储类。与其它语言相比，C的变量存储类有其特色。本章重点介绍C的四种存储类的含义、用法，并讲述与此有关的初始化问题。

第七章预处理功能。这也是C与别的语言的不同之处。主要介绍宏替换、文件包含、条件编译等预处理功能及其应用。

第八章数组。这是C中的一种构造类型，其中数组的表示与别的语言有不同之处。重点是讲述这些差异，C中有特点的字符数组、多维数组及其应用。

第九章指针。指针是C中很重要的、具有特色的数据类型，而又较难于理解。因此从指针的基本性质——说明和运算开始，逐渐展开讲述指针的应用及与C中其它成分的关系，特别介绍了命令行参数。

第十章结构和联合。重点讲述结构与其它各个成分间的关系和它的应用。与结构的外形有相似之处的联合也作了介绍。

第十一章位段、类型定义和枚举。讲述了C中其他一些类型的含义、说明和应用。

第十二章输入/输出和库函数使用。到第十一章为止，C的成分和功能都已介绍完毕。有关程序的输入/输出以及库函数的调用，这些不属于它的基本成分，是必须由C的环境(主要是操作系统)提供的。

第十三章综合应用举例。为了帮助读者进一步全面深入地掌握C语言，用若干例子介绍了一些编程过程以及综合应用C的各种成分的方法。

附录A、B、C、D分别介绍了C的一些编辑、编译方法，UNIX环境下完整的系统调用和库函数，最新的1985年出版的标准C的程序员手册以及IBM PC DOS环境下DR/C编译错误信息表。最后给出了编写本书时所用到的一些主要参考资料。

本书前四章由孙玉方执笔，其余各章及附录由孟庆昌执笔，全书并由孟庆昌统一整理。在编写过程中得到过很多同志的关怀和帮助，在此，一并表示感谢。

虽然本书的酝酿和材料收集已有一段时间了，但由于我们水平有限，时间又比较仓促，所以书中肯定会存在缺点和错误，敬请广大读者不吝指教。

编　者

1985年12月

# 目　录

# 第一章　C语言及其程序设计方法概述

C是一种通用计算机程序设计语言，整个语言的设计及编译程序的实现都是由D·里奇(Dennis Ritchie)一个人完成的。虽然后来又有可移植C编译程序及各种各样的C语言子集或“小”型C语言，但是大家一致公认D·里奇是C语言之“父”。

C与UNIX操作系统紧密地联系在一起。事实上，当初D·里奇开发C主要就是为了更好地描述UNIX操作系统。自从有了C之后，UNIX核心(操作系统)、所有UNIX系统上的命令解释程序，各种实用程序(包括C语言编译程序)以及后来在UNIX上开发的所有应用系统(数据库管理系统、网络通信系统)等都是用C语言写的。UNIX系统的许多特点，特别是其易于裁剪易于移植等主要受益于C语言。另一方面，C之所以有今天，又因为它是植根于UNIX系统上，在UNIX系统上开发，受到UNIX系统的强有力的支持，在编写C语言程序时可以直接使用UNIX操作系统所提供的许多功能，特别是输入/输出库程序及系统调用等。虽然目前C已不限于在UNIX系统，也不限于在PDP-11机，而且在PC-DOS、CP/M、MP/M、VMS等多种操作系统及IBM-PC、Cromemco、各种16位、32位微机上运行，但是UNIX系统仍是C语言最合适的运行环境。

1983年UNIX和C语言的两个主要开发者K·汤普逊(K. Thompson)和D. 里奇(D. Ritchie)双双获得计算机科学和技术领域的最高荣誉奖——图灵奖。评选委员会对UNIX与C作了很高的评价，也明确指出了C语言对UNIX整个系统的贡献。

本章简单回顾一下C语言的发展历史，讨论其一些特点。刚开始读者还不一定对这些特点有很深的体会，待学到后面，不妨常常回味一下，加深对其理解。本章后面几节主要对C作一初步介绍，使读者对C有一概括的了解。这里也以UNIX为背景，讲述C程序的编译和运行，以便读者能在学习过程中按此方法上机实习，培养编程能力，从而有助于对C语言的理解。最后讲一下C语言的一些基本符号，包括所用的关键字等等。

## 1.1　C语言的由来和基本特点

### 1.1.1　由来

1969年贝尔实验室退出了MULTICS工程，K·汤普逊在一台废弃的PDP-7上开始为自己建造软件开发环境，不久D·里奇也参加了进来，不到一年，便完成了一个初步的系统。为了表示与MULTICS的复杂关系，同事们取名为UNIX，随即又把它搬到了PDP-9和PDP-11上。为了摆脱汇编语言的困扰，K·汤普逊决定开发一种高级语言来更有效地描述UNIX系统，从而诞生了一种新语言——B，它是以BCPL语言为基础构成的。然而，B语言没有流行起来，主要因为它缺乏丰富的数据类型，它以字长编址也不适应PDP-11机的字节编址方式。为此D·里奇决定改进B，加入了丰富的数据类型和强有力

的控制结构，从而形成了C。1973年，K·汤普逊和D·里奇用C重写了UNIX，并乘机扩充了UNIX系统的功能，从而开创了UNIX系统发展的新局面。

图1-1展示了C语言的由来。

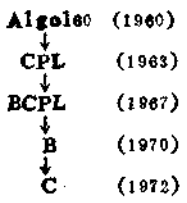



图1-1 C语言之由来

Algol 60是比较复杂的语言，由于它的结构非常严密，其设计者非常注重语法(BNF)、分程序结构，因此对于后来的程序设计语言，特别是PASCAL、PL/1等都有极重大的影响。不过，由于它是面向过程的语言，与计算机硬件相距甚远，所以不适宜编写软件，特别是系统软件。CPL(Combined Programming Language)是仿照Algol 60而设计出来的语言，但它仍然过于庞大和复杂，所以影响了它的实现和广泛使用。M·理查德(M·Richards)提出的BCPL(Basic CPL)对CPL进行了精简，并保持了它的基本特点，所以目前仍有其顽强的生命力。K·汤普逊对BCPL作了进一步简化，设计了非常简单而又接近于硬件的B语言，所以可以用来描述操作系统。但是BCPL和B在简化方面走得似乎太远了一些，语言能力有限，只能用于特殊用途而不能解决一般常见的许多问题。为此，D·里奇把一些缺少的功能加入了B并作了整体规划和整理，设计出了既能描述计算机硬件，又能适应数字计算、正文处理和数据处理需要的通用语言C。

整个70年代，C语言和UNIX系统一样都还只是在贝尔实验室内部和大学科研机构中使用；今日随着UNIX系统被越来越多的人们所认识，C语言也已成了人们进行程序设计特别是软件开发、资料准备、正文处理中得心应手的工具。C语言的流行当然与UNIX系统的普及有密切的关系，但是C语言的成功主要是因为它具有众多的优点。

### 1.1.2 特点

C语言的特点中优点是很明显的，主要有：

1. 简洁 C有功能很强的运算符，其中许多成分代表了它的设计者的嗜好和原始环境中所提供的东西。比如，增量算符++在PDP-11的机器语言中可以直接模拟；间接地址和地址运算可写入表达式用以构成一个完整的语句或表达式，而如果用其它语言描述可能要写好几个语句。对软件生产率的研究表明，平均来说，在一个工作日里，一个程序员只能编制少量工作程序。采用简洁的语言可以明显地提高程序设计者的基本生产效率。

2. 小型 “小”在程序设计中常常意味着“漂亮”。C的关键字比PASCAL少(PASCAL中关键字称作保留字)。许多I/O功能也不是由C实现而是由其运行环境(比如UNIX、PC-DOS等)提供的。这样，语言编译的实现就比较简略，用户进行I/O时可以直接调用相应的库函数而不必过多地考虑其细节。C的“小”还表现在它细心地选取了正确的控制结构和数据类型，而不是毫无目的地引入各种控制结构和数据类型、致使语

言搞得十分庞杂。

3．表达能力强　它可以直接处理字符、数字、地址，可以完成通常由硬件实现的普通的算术、逻辑运算。它反映了当前计算机的性能，所以有效到足以取代汇编语言来编写各种系统软件和应用软件。最明显的例证是UNIX系统，整个系统除了核心内部的1000行左右(整个核心的10%以下)，因为效率、机器硬件表达的需要而用汇编语言书写之外，其余的都是用C语言描述的。当然它同样可以表达数值计算、正文处理和数据处理，所以又是一种通用语言。

4．模块化结构　C支持一种子程序结构作为外部函数。这种函数通过传值来调用。C语言不允许函数嵌套，它通过在文件中使用静态存储类来允许某种形式的专用权。这些措施连同典型的UNIX环境一起，很容易支持用户定义的模块程序设计。

5．可移植性好　虽然C与许多计算机的环境相适应，但它独立于具体计算机的结构体系。因此，只要稍加小心，便很容易地编写出“可移植”的程序。所谓可移植，即不作或稍加改变就可在各种硬件上运行。可移植的程度通常取决于实现依赖性。在一般情况下，为了达到一致性，要进行非常精心和困难的转换工作。可移植程序很容易加以剪裁以适合一台新的机器。C编译程序的代码量不到10000条C语句，用几个人月就可搬到一个新系统上。系统实用程序和预处理程序允许程序员尽可能地把与机器有关的部分从程序中分离出来。这样就便于从一个C系统开始重新定义另一个C系统。可以说C的可移植性是C最重要的一个特点。

当然C并不是没有缺点。比如，它不如现在一些程序设计语言(如PASCAL、ADA)的类型强；它允许编译程序在表达中重新安排计算顺序和参数表，从而产生副作用；它没有动态数组界的检查；类型转换比较随便。

然而这些缺点与它众多的优点相比，仅仅是很次要的部分。C是一个杰出的语言，在某种角度，它对程序员使用的机器没有限制，它的不拘形式的特点使它比一个更严格更完善而有更多限制的语言更易存在。C语言正以它迷人的风貌吸引着人们。按我们的观点来说，C是各种计算机(包括微型机和大型机在内)上很好用的通用交互式程序设计语言。语言系统和机器构成一个动态实体，这个实体可通过上机试验很好地得到理解。

## 1.2 示范程序

用一些例子简要地介绍C语言程序的主要组成成分。

〔例1-1〕　印出一句话：it works.

```
main(  )
{
        Printf("it works.\n");
}
```

按规定，所有的C程序有且仅有一个称为main的函数，程序的执行就从这里开始。例中，main(  )是定义的唯一的函数。

在一般情况下，一个C程序可以放在若干文件中(至少在UNIX系统环境下是如此)，这些文件均要以.C结尾，比如example1.C、example2.C……。文件以.C结尾表示这是C语言源文件。在一个文件中又可以包含若干函数。C语言不象Algol型语言那样有过程和函数的区别。在C语言程序中，不论有否返回值，均称之为函数。在一个程序众多的文件和函数中，除了有一个且只有一个函数的名字取为main以外，别的函数可以取任何有意义的名字。最简单的情况是一个C程序只有一个文件，而这个文件中只有一个函数。显然，这个函数必须是main( )，本例即是。

例中，圆括号( )表明main是一个函数，通常在( )中可能有参数，这里暂时没有。main的正文括在花括号{ }之间，称为函数体。这里的{ }相当于Algol型语言中的语句括号begin和end，这里的函数体中只有一个由分号；终结的语句，它调用库函数Printf对输出进行形式加工。注意字符串要括在双引号之间。其中，\n代表换行符，在输出时可使光标移到下一行的开始位置。若这里不带\n，也没有任何语法和语义上的错误。

要注意的是，C语言每一个语句之后必须要带分号。分号在C语言中是语句的终止符，而不是一般的分隔符，所以不可省略。

函数体中可以没有语句，这样就构成了一个空函数，比如，

```
main( )
{
}
```

但它仍是一个合法的函数。

〔例1-2〕 计算三数之和。

```
/* sum.c, the sum of a, b, c */
main( )

{
        int a, b, c, sum;

        a = 1;
        b = 2;
        c = 3;
        sum = a + b + c;
        printf("sum is%d", sum);

}
```

其中，三个赋值语句只要写得下，并且看得清，可以合写成一行：

```
a = 1; b = 2; c = 3;
```

第4行为说明语句，它说明a、b、c、sum四个变量都是整数。

第10行Printf是一个很复杂的能产生格式输出的函数，或者称为通用格式转换函数。

Printf的第一个参数是格式信息，在"%"之前是要直接印出的字符串，每个"%"

都指示其它(第二个、第三个…)参数要被替换，并指出用什么格式来印出它。所以，其它参数即是要输出的变量。例1-1是使用Printf的各种形式中最简单的一种。也可以把它变成如下形式：

```
main( )
{
    printf("it");
    printf("works. ");
    printf("\n");
}
```

因为其中只有最后一行中有"\n"，前面的输出都在一行上，从而产生与例1-1中相同的结果。

但在例1-2中，如果sum是6，则输出将是　　sum is 6

这是一个比例1-1复杂的例子。因为Printf("sum is %d \n", sum); 中，第一个参数里含有"%d"，这意味着参数表中第一个参数(即变量sum)将作为一个基数为10的数被打印出来，而"%d"之前的"sum is"作为字符串原封不动地打印出来，从而形成上面的输出形式。

表1-1 列出常用的打印格式。

表1-1　常用打印格式

| 表示 | 功　　能 | 例　　子 | 输　　出 |
|---|---|---|---|
| %d | 十进制整数 | printf("…%d\n", sum); | …6(设sum=6) |
| %f | 十进制浮点数 | printf("…%f\n", sum); | …145.7(设sum=145.7) |
| %c | 单个字符 | printf("…%c\n", c); | …A(设c="A") |
| %s | 整个字符串 | printf("…%s\n", save); | …good bye! (设save="good bye! ") |
| %o | 八进制数 | printf("…%o\n", oct); | …1576(设oct=01576) |
| %x | 十六进制数 | printf("…%x\n", hex); | …16AF(设hex=0x16AF) |
| %% | %本身 | printf("…%%\n"); | …%<br>%在打印格式中有特殊含义，%%就取消了%的特殊含义，从而打出了一个%。 |

注意：%d，前面可以有数字，比如%4d，表示打印出的十进制数至少占4位。%f前面也可以有特别标志，如%6f，表示印出的数要占六个字符的位置，%.2f表示小数点后面数占两位，而%6.2f则表示数共占六个字符的位置，其中小数点后占两个位置。

上述几种输出格式可以混合在一起使用。

〔例1-3〕

```
main( )
{
    int n;

    n=511;
    Printf("what is the value of %d in octal?", n);
```

```
    Printf("%s %d decimal is %o octal\n", "right", n, n);
}
```

将打印出：

```
what is the value of 511 in octal? right 511 decimal is 777 octal
```

因为第一个Printf后中没有换行标志"\n"，所以第二个Printf的输出与第一个Printf连接在一起了。第二个Printf中顺序有%s、%d和%o三种输出格式，正好对应于后面的三个参数"right"，n和n，按要求分别输出字符串right，n的十进制数表示511和n的八进制数表示777。

〔例1-4〕 把数1776转换成罗马数字表示。

```
/*
**Program to Print 1776 in Roman numerals
*/
main( )
{
    int a=1776;

    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');          /*end with a newline*/
}
/*
**Print the character c as many times as there are
**j's in the number i, then return i minus the j's
*/
romanize(i, j, c)
char c;
int i, j;
{
    while(i>=j)
      {
      Putchar(c);
      i=i-j;
      }
    return(i);
```

```
}
```

这个程序分成两个函数：main( )和romanize( )。main( )的工作主要是安排减法运算的次序以及程序的起始和终止。而romanize( )完成实际的减法运算，并利用一个库函数Putchar( )将罗马数字印出。

这里，/*和*/之间的是注释，主要是编程者作说明或提示之用。注释的行数并无限制，但注释不能嵌套。在编译时，C编译程序并不为它产生代码。

在例1-4中main( )中的

```
int a=1776;
```

实际是一种变量说明，并加了初始化。它说明a是整型量int，为了省写把初值直接写在其后。可以改写成：

```
int a;
a=1776;
```

效果完全一样，要特别强调的是，在C中变量都要先说明才能引用。变量说明要给出其类型和存储类。

紧接下面的7个语句，调用了7次函数romanize，其中前六次把返回值赋给了a，最后一次未赋值给a。这是因为前六次都需要把返回值用作下一次函数调用的实参值。现在解释一下romanize的工作：

```
romanize(i, j, c)
```

表示它有3个参数i，j，c。因为此时其具体值为多少尚不可知，所以称之为形式参数，简称形参。

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

说明形参c为字符型(char)量，i和j是整型(int)量。其作用是告诉编译程序有关引用量的存储空间大小。有时这两行也称为参数区分。

上述三行总称为函数之首部，针对此例，它包括函数名romanize，参数表(i, j, c)和参数说明char c; int i, j, c;

其实每一个函数的第一个开花括号{前面部称为函数首部，如函数main( )中，main( )即是首部，而函数romanize的首部占三行。

romanize( )里的函数体部分是一些可执行的代码。这里包含两个语句：while和return。而while语句中又包括了两个语句，为此要用花括号括起，构成一个复合语句。只要条件(i>=j)成立，这个复合语句中的两个语句就不断执行。while语句称为循环语句，是C语言中的一类重要的控制语句，与它有关的还有for和do-while语句。return(i)是一个返回语句，它向调用者返回一个值，并把控制也返回调用者。例中调用者为函数main。

下面我们具体说明这两个函数的调用关系：在main中，第一次是以a=1776,1000,'m'来调用romanize的，这里的1776，1000，'m'分别对应于romanize中的形参i，j，c，称之为实在参数，或简称为实参。当控制转到romanize之后，由于(i>=j)(因为1776>=1000)条件成立，所以执行while循环体中的两个语句；由Putchar(c);印出字符m。经过第二句运算之后得到i为776，再回上去时(i>=j)不成立，所以退出while

循环，从而执行return语句，把i(=776)返回给了调用者。当控制回到main( )时，第一个语句

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

a得到返回值776。之后又第二次，第三次，直到第六次调用函数romanize，此时a得到的返回值是1。最后一次调用函数romanize，只是把控制返回，而没有返回值，因为调用者不需要。最后main调用Putchar执行回车换行工作，此时程序就结束了。

〔例1-5〕 写一个程序，计算从标准输入(通常指终端键盘)上输入的字符中每个数字、空白格(空格、制表和换行)及其它所有字符出现的次数。

输入字符共计有十二类，即十个数字、空白符和其它字符，所以使用数组来保存每个数字出现的次数比用十个单独的变量要方便得多。这里是程序的一种形式：

```
main( )    /*count digits, white space, others*/
{
    int c, i, nwhite, nother;
    int ndigit[10];

    nwhite=nother=0;
    for(i=0; i<10; ++i)
        ndigit[i]=0;    /*array initializers*/
    while((c=getchar( ))!=EOF)
        if(c>='0' && c<='9')
            ++ndigit[c-'0'];
        else if(c==' '||c=='\n' ||c=='\t')
            ++nwhite;
        else
            ++nother;
    printf("digits=");
    for(i=0; i<10; ++i)
        Printf("%d", ndigit[i]);
    printf("\n white space=%d,other=%d\n",
        nwhite, nother);
}
```

数组是变量的一个有序集合，其中所有变量具有相同的类型。在C语言中数组表示形式为

数组名[ ]

其中，数组名是标识符，ndigit[ ]就是数组，而ndigit[0]、ndigit[1]、…、ndigit[n]就是数组ndigit的元素。其中[ ]中的值是其下标，而只有一个下标的数组就称为一维数组。

上面这个程序依赖于数字的字符表示，如测试语句

```
if(c>='0' && c<='9')…
```

用来确定c中的字符是否是数字，如果是数字则此数字的数值是

c－'0'，

这必须满足下述条件才正确：'0'，'1'，…等都是正整数且按升序排列，在'0'和'9'之间没有数字之外的其它字符。对ASCII字符集说来，这些条件总是成立的。

根据定义，在进行运算之前将包含char和int的算术运算都转换成int，所以在算术运算的过程中char变量和常数实质上等价于int。例如，c－'0'是一个整型表达式，它依据c中存储的字符是'0'到'9'，而对应地取得0到9之间的值，因此它是数组ndigit的合法下标。

在C语言程序中，函数放的位置先后是无关紧要的，但它总是从函数main开始执行。另外，一个名字之后紧跟圆括号（ ）是函数的标志。

用C语言进行程序设计时，鼓励人们把一个大问题划分成若干小问题，为每一个小问题构造一个小函数，然后把这个小"零件"（函数）拼装起来构成一个完整的运行良好的大"机器"（程序）。

原则上C是无格式语言，在格式上没有任何特殊的要求，但是为了便于阅读和修改，建议读者在编写时注意缩进和上下对齐。

## 1.3 C语言程序的编辑、编译和运行

程序的编辑可以借用宿主系统（UNIX、PC-DOS、VMS等）所提供的编辑工作。编辑好的程序应放在某（些）个以·C结尾的文件中，比如〔例1-1〕放在文件example1·C中；〔例1-2〕放在文件example2·C中。

程序编辑好之后就可以编译，在UNIX环境下如果是编译文件example1·c中的程序，则打入如下命令：

```
cc  example1·c
```

如果是编译文件example2·c中的程序，则打入命令：

```
cc    example2·c
```

可以看到一般的编译命令是

```
cc    文件名
```

打入这个命令后，系统把c编译程序调入内存对要编译的程序进行词法分析、语法分析、代码生成等多种处理。如果没有错，就按缺省原则，把最后的可执行目标程序放在文件a.out中。为了运行这个可执行目标程序并获得最后结果，只要打入命令

```
a.out
```

如果要保留可执行目标程序，此时在编译时要用任选项－o，它把可执行程序放在紧跟在－o之后的文件中。比如，

```
cc    -o    example1 example1·c
```

或

```
cc    -o    example2  example2·c
```

那么执行时只要打入