C语言程序开发实训教程

C YUYAN CHENGXU KAIFA SHIXUN JIAOCHENG

主编 谢圣献 王贤勇 杨华庆

国防工业出版社
National Defense Industry Press

内 容 简 介

本书根据高等院校C语言程序设计教学大纲，并参照全国计算机等级考试二级C语言程序设计考试大纲编写。主要内容包括：C语言简介、算法及其描述、基本数据类型及运算、顺序结构、选择结构、循环结构、数组、函数、指针、构造数据类型、编译预处理、位运算、文件。本书可作为各类高等院校C语言程序设计课程的教学用书，也可作为计算机等级考试和自学参考用书。

图书在版编目(CIP)数据

C语言程序开发实训教程/谢圣献主编.—北京：国防工业出版社，2012.4

ISBN 978-7-118-08019-3

Ⅰ.①C… Ⅱ.①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050749号

※

国防工业出版社出版发行

(北京市海淀区紫竹院南路23号 邮政编码100048)

北京奥鑫印刷厂印刷

新华书店经售

*

印张787×1092 1/16 印张16 字数368千字

2012年4月第1版第1次印刷 印数：1—4000册 定价：30.00元

(本书如有印装错误，我社负责调换)

国防书店：(010)8854077 发行邮购：(010)88540776

发行传真：(010)8854075 发行业务：(010)88540717

《C语言程序开发实训教程》

编 委 会

主　编：谢圣献　王贤勇　杨华庆

编　委：包　云　王　勇　张振领　亓民勇

于承敏　董金新　韩红燕　谢　光

许丽莉　贾保先

前　言

C 语言是一种通用的、过程式的编程语言，广泛用于系统与应用软件的开发。具有高效、灵活、功能丰富、表达力强和较高的移植性等特点，在程序员中备受青睐。

C 语言是由 UNIX 的研制者丹尼斯·里奇（Dennis Ritchie）和肯·汤普逊（Ken Thompson）于 1970 年研制出的 B 语言的基础上发展和完善起来的。目前，C 语言编译器普遍存在于各种不同的操作系统中，例如 UNIX、MS-DOS、Microsoft Windows 及 Linux 等。C 语言的设计影响了许多后来的编程语言，例如 C++、Objective-c、java、C#等。

作为一种程序语言，C 语言的设计目标是提供一种能以简易的方式编译、处理低级存储器、产生紧凑的机器码以及不需要任何运行环境支持便能运行的编程语言。C 语言也很适合搭配汇编语言来使用。尽管 C 语言提供了许多低级处理的功能，但仍然保持着良好跨平台的特性，以一个标准规格写出的 C 语言程序可在许多计算机平台上进行编译，甚至包含一些嵌入式处理器（单片机或称 MCU）以及超级计算机等工作平台。

由于其简洁、高效、可移植性，C 语言已经成为 IT 相关专业编程语言的标准。不仅如此，C 语言还是全国计算机等级考试二级程序设计语言的翘楚。一些非 IT 相关专业的理工科学生，在学习了计算机文化基础之后，也想在程序设计方面能够拓展知识、学以致用，学习 C 语言、使用 C 语言，也应是其首选。

为了更好地普及 C 语言知识，我们编写了本书。编写的指导思想是：

首先，学习语言是为了使用语言，而不是学术研究。本书摒弃了一些编程语言深层次的东西，忽略了一些哲学和方法论方面的讨论，而是将语言活泼、实用的一面展现给读者。

其次，学习的过程是循序渐进的，教材本身提供的就应该是一种指导、一种学习方法。百科全书、参考手册可以作为学习的工具，而非学习的教材。本书将各知识点较均匀地分布在各章节，而且辅以大量实例，降低了学习的难度。

再次，语言是工具，平台是环境，二者无依赖关系。本书介绍 C 语言知识时，重点在于那些与具体环境无关的、通用的、语言本身的部分；而举例应用时，则会特别注明不同平台上的不同效果。

最后，语言有最新标准，教材也必须与时俱进。作为一种充满生机和活力、一直在使用着的实际的编程语言，C 语言也在不断演化。本书对 C 语言语法的介绍，完全依照最新的 C99 标准；某些环境中不符合该标准的实现，出现时也会特别注明，并且不建议使用。

鉴于水平有限，上述指导思想未必完全正确；编写过程中的疏漏、甚至错误在所难免，望广大读者批评指正。

目 录

第 1 章　C 语言简介

C 语言是当前国际上教学及应用中较为流行的面向过程的计算机高级语言。它适合作为系统描述语言，既可以用来编写系统软件，也可用来编写各种应用软件。

1.1　计算机与程序、程序设计语言

1946 年 2 月，世界上第一台电子计算机 ENIAC 在美国宾夕法尼亚大学诞生，它的出现具有划时代的伟大意义。从第一台计算机的诞生到现在，计算机被应用到了社会的各个环节，改变了世界，改变了人们的生活。

计算机并不能天生“自动”地工作。它的运行，完全是由人们赋予它的程序所控制。人们事先编制好程序，输入到计算机中，计算机执行程序就会产生相应的结果。

计算机内部并不懂人类的自然语言，它只能识别二进制的信息。人和计算机如何沟通呢？也就是说程序如何编制呢？人们通过程序设计语言，按照某种语法规则来编制具有一定封装结构的程序，让计算机能够理解和执行。这里所说的理解和执行也可能经过某种中间环节的翻译，也就是说可能与人类所使用的自然语言有较大差别。

程序设计语言目前经历了三个阶段，简要介绍如下。

1. 机器语言（Machine Language）

计算机能够直接识别和执行的只有二进制编码的指令，这种编码形式称作机器语言。机器语言程序是由一条一条的指令构成，每条指令可能还有操作码（干什么）、操作数（对谁进行操作）、转移地址（到哪里或者在哪里）等，一律都是二进制形式。

计算机的语言就是机器语言，机器语言程序有最高的执行效率。但是对于用户，机器语言则是最难懂、难记、易出错的编程工具。即使用十六进制表示，仍然难以普及。下面是一段十六进制表示的机器语言的程序，可以在 MCS-51 单片机上运行。

```
74H 34H 24H 45H F5H 09H 74H 12H 34H 23H F5H 08H
```

如果不对照指令系统的机器码，很难看出这段程序要实现什么功能。

2. 汇编语言（Assembly Language）

为了解决上述困难，人们建立了一种助记机制，用助记符代替指令的操作码，操作数使用有意义的符号和符号地址表示，转移的目标地址也采用标号的形式。这些符号地址和标号代表的地址，可以是写程序前固定好的，也可以是在写程序的过程中才确定的。这种使用助记符、符号地址、标号等符号来编写程序的系统称为汇编语言。以下代码段就是与上述机器语言等价的汇编语言版本。分析可知，程序段的作用是将两个 16 位数 1234H 和 2345H 相加，结果存入 RES_HIGH 和 RES_LOW 符号代表的两个内部 RAM 单元中。

```
RES_LOW     DATA        09H     ; 定义存储结果的低一半空间
```

```
RES_HIGH    DATA        08H      ; 定义存储结果的高一半空间
            MOV         A, #34H
            ADD         A, #45H
            MOV         RES_LOW, A      ; 实现 34H 和 45H 相加，存入 RES_LOW
            MOV         A, #12H
            ADDC        A, #23H
            MOV         RES_HIGH, A     ; 实现 12H 和 23H 及前面加法指令的进位
                                        ; 相加，结果存入 RES_HIGH 中
```

计算机是无法直接理解汇编语言程序的，所以还必须将汇编语言程序翻译成机器语言，这个翻译过程叫做汇编。汇编可以人工完成，也可以使用一个计算机程序(不管原来是用什么语言编写的)完成这个翻译过程，这个程序叫做汇编程序(汇编器，Assembler)。

汇编语言虽然比机器语言易懂、易读，但它仍然是与具体的机器密切相关的，不同指令系统（一个计算机系统所能识别和执行的所有指令的集合）的汇编语言也不相同。另外，由于汇编语言中的基本单位是指令，所以稍复杂些的运算通常要用一个较长的子程序实现，而无法用一条简单的语句描述。

3. 高级语言(High-level Language)

高级语言是面向过程和问题并能独立于机器的通用程序设计语言，是一种接近人类自然语言和常用数学表达式的计算机语言。在使用高级语言编写程序时，可以不必顾及计算机内部操作细节，而把主要精力集中在解决问题的实现方法上。高级语言编写的程序，计算机是不能直接识别和执行的，也需要翻译成机器语言才能在计算机上运行，这个翻译过程通常由编译程序（编译器，Compiler）实现；也有一边翻译一边执行的，称为解释程序（解释器，Interpreter）。

高级语言的出现被认为是计算机发展史上的惊人成就，为计算机的推广普及提供了可能。常见的高级语言很多，如 BASIC 语言、PL/M 语言、C 语言等。以下是用 C 语言编写的一段代码。

```
int     a, b, c;
a = 0x1234;
b = 0x2345;
c = a + b;
```

这段程序实现的功能仍是两个 16 位数相加。使用 C 编译器可将它编译成与前面的汇编语言、机器语言版本功能相同的机器码。欲在不同 CPU 的计算机上实现同样的功能，只需用相应的编译器重新编译，程序不必修改。高级语言程序的这个特点称作可移植性。

4. 三种语言的比较

高级语言语法规则同自然语言相近，易学、易懂、方便、通用，源程序代码较短，便于推广和交流。如果有高效的编译器，高级语言是最好的软件开发工具。但是，编译器掩盖了机器内部的细节，接管了对部分资源的分配方式。如果要学习机器的体系结构，或希望自己管理整个计算机系统的所有资源，高级语言是不合适的。

机器语言编写的程序无冗余，执行速度快，但是不好记忆、不易理解、使用麻烦。除非没有其他工具，一般很少直接用机器语言编写程序。

汇编语言比机器语言好理解、便于记忆和使用，但不能独立于机器。

总之，在科学计算、信息处理等方面采用高级语言比较合适；而在实时控制中，通

常使用汇编语言。即使在高级语言开发的软件中，对执行速度要求严格的程序段也常用汇编语言编写。

综上所述，计算机和程序密不可分，程序是计算机的灵魂；而程序又是借助于程序设计语言来编制，借助计算机的能力来实现人们预想的各项任务。因此，计算机程序和程序设计语言既有面向用户的一面，同自然语言和日常思维有着各种联系，同时又有面向计算机系统的一面，有着独特的、适应计算机硬件系统及处理过程的程序设计思想。

1.2 C 语言的发展过程

C 语言的原型是 ALGOL 60 语言（也称为 A 语言）。ALGOL 60 是 1960 年出现的一种面向问题的高级语言，离计算机硬件比较远，不宜用来编写系统程序。

1963 年，剑桥大学将 ALGOL 60 语言发展成为 CPL（Combined Programming Language）语言。CPL 语言在 ALGOL 60 的基础上更接近硬件，但是规模比较大，难以实现。

1967 年，剑桥大学的 Martin Richards 对 CPL 语言进行了简化，于是产生了 BCPL（Basic CPL）语言。1970 年，美国贝尔实验室的 Ken Thompson 将 BCPL 进行了进一步简化修改，提炼出 BCPL 的精华，设计出了很简单而且很接近硬件的一种语言，并为它起了一个有趣的名字“B 语言”（BCPL 的第一个字母），并且用 B 语言写了第一个 UNIX 操作系统。

当时的 B 语言虽然有着精炼、接近硬件的优点，但是又过于简单，数据没有类型。1972 年，美国贝尔实验室的 D. M. Ritchie 在 B 语言的基础上最终设计出了一种新的语言，他取了 BCPL 的第二个字母作为这种语言的名字，这就是 C 语言。1975 年 UNIX 第 6 版发布之后，多次改进后的 C 语言的突出优点引起人们的普遍关注。1977 年 D. M. Ritchie 发表了不依赖于具体机器系统的 C 语言编译文本《可移植的 C 语言编译程序》，使得 C 语言移植到其他机器时所需做的工作大大简化，推动了 UNIX 操作系统的推广。伴随着 UNIX 的日益广泛使用，C 语言也走出了贝尔实验室内部，在全社会得到了迅速推广，很快风靡全世界，成为世界上应用最广泛的几种计算机语言之一。

1978 年由美国电话电报公司(AT&T)贝尔实验室正式发表了 C 语言，同时由 B. W. Kernighan 和 D. M. Ritchie 合著了著名的《The C Programming Language》一书。该书通常简称为《K&R》，也有人称之为《K&R》标准。但是，在《K&R》中并没有定义一个完整的标准 C 语言，后来由美国国家标准化协会（American National Standards Institute）在此基础上制定了一个 C 语言标准，于 1983 年发表，通常称之为 ANSI C。

ANSI C 第一版在很多语言细节上也不够精确，甚至没有很好表达它所要描述的语言，在 1987 年公布了 87 ANSI C。1989 年末又公布一个新的标准 89 ANSI C（简称 C89），1990 年，国际标准化组织 ISO（International Organization for Standards）接受了 89 ANSI C 为国际标准 ISO/IEC 9899:1990，通常称为 C90。

1995 年，ISO 对 C90 进行了修订，称为 C95。1999 年，ISO 又在保留原来的 C 语言特征的基础上，增加了一些面向对象的特征，命名为 ISO/IEC 9899:1999，简称为 C99。

目前流行的 C 语言编译系统大多是以 ANSI C(C89)为基础进行开发的，常用的 C 语言集成开发环境都能够较好实现支持 C，但不同版本的 C 编译系统所实现的语言功能和语法规则有略有差别。本书论述以 C99 为基础。

1.3 C语言的特点

C语言在当今时代仍然有着较强生命力和广泛的使用，主要同其自身的显著特点分不开。

1. 语言结构简洁紧凑、使用灵活方便

C语言一共只有37个关键字（其中有5个是C99标准新增的，参见附录），9种控制语句，程序书写形式自由，区分大小写，压缩了一切不必要的成分。相较于其他许多高级程序语言，C语言简练，源程序短，输入程序工作量少。同时，C语言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，可以像汇编语言一样对位、字节和地址进行操作，而这三者是计算机最基本的工作单元。

2. 运算符丰富

C语言的运算符包含的范围很广泛，共有34种运算符（参见附录）。C语言把括号、赋值、强制类型转换等都作为运算符处理，使C语言的运算类型极其丰富，表达式类型多样化。灵活地使用各种运算符，可以实现其他高级语言中难以实现的运算。

3. 数据类型丰富

C语言的常用数据类型有：整型、实型、字符型、数组、指针、结构体、共用体类型等。能用来实现对链表、树、图等各种复杂数据结构的运算。C语言引入了指针概念，使程序效率更高。另外C语言具有强大的图形功能，支持多种显示器和驱动器，计算功能、逻辑判断功能强大。

4. 完全模块化和结构化的编程语言

模块化、结构化语言的显著特点是代码及数据的分隔，即程序的各个部分除了必要的信息交流外彼此独立。这种结构化方式可使程序层次清晰，便于使用、维护及调试。C语言是以函数形式提供给用户的，这些函数可方便地调用，并具有多种循环、条件语句控制程序流向，从而使程序完全结构化。

5. 语法限制不太严格，程序设计自由度大

虽然C语言也是强类型语言，但它的语法比较灵活，允许程序编写者有较大的自由度。一般的高级语言的语法检查比较严格，能检查出几乎所有的语法错误。但C语言放宽了语法检查，提供了编程人员较大的自由度。在C语言中，对变量的类型约束不严格，使用比较灵活（如整型与字符型及逻辑类型数据通用），影响程序的安全性。对数组下标越界不作检查，可能产生非法的地址访问。“限制”和“灵活”是一对矛盾，严格限制，失去了灵活性；而强调灵活性，就必然放松限制。因此，从应用的角度来看，C语言要比其他高级语言较难掌握，要求编程人员对程序设计更熟练一些。

6. 允许直接访问物理地址，对硬件进行操作

C语言允许进行二进制的位（Bit）操作，可直接访问物理地址，直接对硬件进行操作，因此它不仅具有高级语言的功能，同时又具有低级语言的许多功能，能够像汇编语言一样对位、字节和地址进行操作，可用来编写系统软件。

7. 生成目标代码质量高，程序执行效率高

C语言由于同计算机硬件有着较好结合，其程序比其他高级语言执行效率高，一般

只比汇编程序生成的目标代码效率低 10～20%。

8. 适用范围大，可移植性好

C 语言有一个突出的优点就是适合于多种操作系统，如 DOS、UNIX、Windows 系列；也适用于多种机型。C 语言具有强大的绘图能力，可移植性好，并具备很强的数据处理能力，因此适于编写系统软件、三维、二维图形和动画程序，也可进行科学数值计算。

1.4 C 语言程序的结构

1.4.1 C 程序结构

一个 C 语言程序可由下面不同的部分组合而成：文件包含部分；预处理部分；变量（全局/外部）说明部分；函数原型声明部分；主函数部分“；”函数定义部分。

对于程序的结构做如下几点说明：

（1）并不是所有的 C 程序都必须包含上面的 6 个部分。一个简单的 C 程序可以只包括文件包含部分及主函数部分。

（2）一个 C 程序源文件的后缀名为.C，C 程序是由函数构成的，并且一个 C 程序有且仅有一个主函数 main。

（3）函数是由语句构成的，一般 C 的语句都是由分号“；”来结束。

1.4.2 C 程序示例

为了说明 C 语言程序结构，下面通过几个程序分别加以说明。例程中有些内容会在后面相关章节详细讲解，这里只需掌握 C 程序的基本结构组成和书写格式特点。

例 1.1

```
#include "stdio.h"        /*  也可用#include <stdio.h>  */
int main()                /*  主函数定义  */
{
    printf("Hello,world'/n");        /*  输出信息字符串 Hello,world'  */
    return 0;             /*  返回函数值  */
}
```

本程序的功能是在屏幕上输出信息“Hello，world!”。对程序说明如下：

（1）include 称为文件包含命令，扩展名为.h 的文件称为头文件，这里包含的文件是 stdio.h 头文件。文件包含命令和头文件中间最好以空白符（空格或者 Tab 键）分隔开。头文件应以双引号“”或者尖括号<>来界定。一般的集成开发环境中，双引号的检索范围比尖括号大。

（2）文件包含命令是预处理命令中的一个，所有预处理命令都是以“#”来开头，预处理命令结束不需要分号。

（3）main 是一个特定函数的名字，即主函数。

（4）printf 也是一个函数名字，这个函数是一个标准库函数，是由系统定义好了的，说明在 stdio.h 中，可以在程序中直接调用。

（5）函数的定义内容必须放在一对花括号“{}”中。

（6）/* … */是 C 语言程序中的注释符号，表明其中的信息是对程序功能等进行的说明，对程序本身没有任何影响。将注释全部删除，对程序功能没有影响，但程序的可读性降低了。一个好的程序员应在程序中加注适当的注释。在 C99 标准中，又引入了C++中的单行注释符“//”，从“//”开始到一行结束的回车符号都是作为注释的。

例 1.2

```
#include "stdio.h"
#include "math.h"
int main(void)

    double x,value;                    /* 变量定义声明 */
    /************注释的一种用法，分隔程序的不同部分******************/
    printf("input a number:? ");     // 执行部分,这里的注释用了"//"
    scanf("%lf",&x);
    value=sin(x);
    printf("sin(%lf) is %lf/n",x,value);
    return 0;
```

本程序功能是根据用户输入的浮点双精度数 x 求出 sin(x)的值并输出结果。说明如下：

（1）程序中包含了两个头文件：stdio.h、math.h。

预处理命令一行指明一个命令，所以文件包含命令一次只能指定一个文件进行包含。文件包含命令的作用是将指定的文件包含到本程序中，成为本程序的一部分。C 语言的头文件中包括了各种标准库函数的函数原型声明。库函数有很多，根据功能作用及类型分别将其在不同头文件中进行声明。因此，调用一个库函数时，要包含该函数的原型声明所在的头文件。在本例程序中调用的库函数 sin()，原型声明在 math.h 中，必须要包含进来，否则程序无法编译通过。其他如 sin()、cos()、tan()、exp()等常用数学函数也在 math.h 中进行了原型声明。stdio.h 中声明常用输入/输出函数，如输出函数 printf()及输入函数 scanf()。

（2）一个函数由两部分构成：

① 函数首部。即函数的第一行，各部分依次为函数返回值类型、函数名、(形式参数列表)。

int	main	(	void	)
函数返回值类型	函数名	函数括号	形式参数列表	函数括号

- ANSI C 中对函数返回值类型（也称函数类型）有个缺省设置，即缺省返回值类型默认为 int 类型。C99 中取消了隐含式的函数声明，不再支持隐含式的 int 规则。
- 形式参数列表可以为空，即 main()，或者 main(void)。
- main 主函数是程序的入口地址，由系统调用。主函数执行结束，程序也就结束，需通知系统程序执行状态，故这里用 int 作为函数返回值类型。函数结束时以 return 0 返回整型 0 值通知系统程序正常结束。

② 函数体。即函数首部下面的花括号内的部分。函数体又由（变量）定义声明部分和执行部分构成。

- 程序中使用的各种机制（变量、自定义函数等）都要求先定义（或者声明），然

后才能使用。如程序中首先在定义声明部分通过“double x,value;”语句定义出了两个变量 x,value，在程序执行部分才能使用。

● 函数体内由于复合语句的原因，有可能形成多个花括号嵌套，以最外层的花括号作为函数体的界定。

● 为了明晰层次关系，要注意养成良好的书写习惯。严格遵守层次缩进书写。如函数主体中的语句相对于界定的花括弧就缩进了一层。注意，缩进要用 Tab 键完成，一定不要用连续的空格。

例 1.3

```
#include "stdio.h"
int max(int x,int y);//max ( ) 的函数外原型声明，从声明开始一直到程序结束有效
int main(void)
{
    int a,b,big;
    // int max(int x,int y);
    /*
    上面的这个原型声明被注释了，否则它的位置是在函数内部声明的，本函数定有效。
    其他函数中不能调用这个 max 函数，除非也进行内部声明。
    */
    printf("please input value of a and b: "); //输出函数，提示输入数据
    scanf("%d%d",&a,&b);    //输入
    big=max(a,b);
    printf("max=%d",big):
}

int max (int x ,int y)
{
    int z;
    if(x>y) z=x;
    else    z=y;
    return(z);
}
```

程序要求用户输入两个整数，并返回输出其中的大值。说明如下：

（1）函数原型声明就是函数首部加上分号。注意函数原型声明不等同于函数定义。声明可以出现多次，以便在任何想调用这个函数的程序中引入这个函数功能。函数定义只能出现一次。

（2）按照先定义后使用的原则，函数也应该先定义出来。本程序中先在 main 函数中出现了 max 函数的调用，即先使用了 max 函数，而 max 函数的定义是出现在主函数定义之后。要想使用 max（），main 函数中必须声明一下其原型。

（3）一个程序的规模可能很大，包含了多个函数，相互之间的调用关系可能很复杂，为了简化声明过程，将所有的函数（除了主函数）的原型声明集中起来，统一在程序预处理命令之后进行声明。声明后，任何函数定义时都可自由按意愿调用其他函数来完成任务，不用关心先后的问题。从这个意义上讲，文件包含就是对库函数的声明。

（4）C 语言的输入/输出都是用库函数完成的。printf 函数完成指定格式的数据输出，scanf 函数完成指定格式的数据输入。一般在输入函数之前，都要有适当的输出，提示即将要进行的输入工作的特征。

1.5 C 语言设计程序的步骤

1.5.1 C 语言程序过程

用程序语言编写的程序文件称为源程序（Source Program），必须经过翻译转换成机器语言才能被计算机所识别并执行。以 C 语言来讲，其翻译过程采用的是编译方法，一个 C 程序的运行，通常经历了如下阶段：

（1）使用编辑软件，编写源程序，产生源（程序）文件.C。

（2）使用编译器对源文件进行编译，产生目标（程序）文件.OBJ。

（3）使用链接器对目标文件及其他相关文件进行链接，产生可执行文件.EXE。

（4）运行可执行文件。

这四个过程可简单总结为编辑→编译→链接→运行。可借助不同的软件工具来实现这几个阶段。假设要设计一个程序，名字定为 example，则整个过程的流程图如图 1-1 所示。目前，大多数的 C 编译系统都是集成开发环境（IDE）方式的，即把对程序的编辑、编译、链接、运行等操作过程全部集中在一个界面下完成，使用方便，功能丰富，直观易用。常见 C 语言的 IDE 有 Turbo C 2.0、Turbo C++ 3.0、Visual C++ 6.0 等。

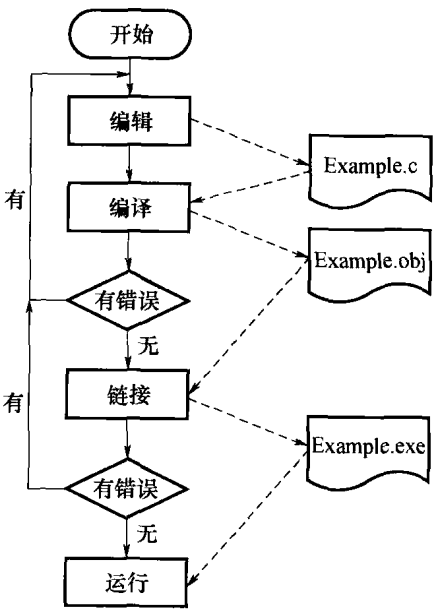

图 1-1 C 语言编程流程图

1.5.2 编译过程

适合 C 语言的编译器不止一种。一般 C 源程序的编译器由 3 个部分构成：

- 词法分析器（Lexical Analyzer）
- 语法分析器（Parser Analyzer）
- 代码生成器（Code Generator）

词法分析器按照从左到右逐个字符地读入源程序，即对构成源程序的字符流进行扫描然后根据构词规则识别单词(也称单词符号或符号)，并对单词进行分类。词法分析器通常不会关心单词之间的关系（属于语法分析的范畴）。例如词法分析器能够将括号识别为单词，但并不保证括号是否匹配。同时，词法分析会忽略所有注释内容。

词法分析器将分析结果保存在一个结构单元中，称之为 Token。

针对 C 语言表达式：sum=3+2，将其单词化后可以得到表 1-1 所列的 Token。

表 1-1　token

语　素	单 词 类 型	语　素	单 词 类 型
sum	标识符	+	加法操作符
=	赋值操作符	2	数字
3	数字	;	语句结束

语法分析器对 Token 进行分析，识别每个成分的角色，将单词序列组合成各类语法短语，如“程序”、“语句”、“表达式”等。语法分析器会判断源程序在结构上是否正确。

代码生成器是将经过语法分析后没有语法错误的程序指令转换成机器语言指令。

1.5.3　链接过程

通过编译生成的目标程序代码虽然是机器语言的形式，但却不是机器可以执行的方式。这是因为为了支持软件的模块化，允许程序语言将不同时期开发的具有独立功能的软件模块作为一个单元，一个可执行的程序中可能含有一个或者多个这样的程序单元。因此，目标程序只是一些松散的机器语言，要获得可执行的程序，还需将它们链接起来。

程序的链接工作由链接器（Linker）来完成。这种可执行的程序是一种可存储在磁盘上的文件。

1. 单个源程序文件

假设一个独立的源程序文件 mywork.c，经过编译生成 mywork.obj，链接目标文件后生成 mywork.exe 可执行文件。

2. 多个源程序文件

假设一个程序由三个程序员编制，有三个不同的文件 file1.c、file2.c、file3.c，每个源程序都具有不同的函数功能。则三个源程序可分别编译成 file1.obj、file2.obj、file3.obj。然后由链接程序将三个目标文件链接成一个完整的可执行文件。

注意，并不是任何目标程序都可以链接成可执行程序。被链接的多个目标程序文件中，只允许一个程序中带有唯一的 main 函数，即有且仅有一个可被加载的程序入口。

习　题

1. 机器语言、汇编语言、高级语言有什么区别？计算机可以直接理解的是哪类语言？人们编程最方便的是哪类语言？

2. 怎样在程序中使用系统提供的函数？

3. C 语言程序由哪些部分构成？

4. C 语言程序运行分为哪几个步骤？

第 2 章　算法及其描述

设计一个程序，首先要考虑的有两个方面。

1. 程序中要处理的数据

要认真思考如何对要处理的数据进行组织和使用，从而使程序对数据的处理速度更快、效率更高。这个问题涉及了数据的类型及数据组织方式，也就是数据结构的应用。

2. 数据处理的步骤

程序中对数据的处理过程要有明确可行的操作步骤，也就是对数据处理的方法及过程，我们称为算法。

一个程序设计人员，必须认真考虑并设计程序中对应相应的数据结构和算法，才能最终得到高效的程序。著名科学家沃思（Nikiklaus Wirth）提出了如下的公式：

程序＝数据结构＋算法

自从沃思将程序设计形象地用上面的公式表示出来后，这条“黄金定律”便成为了人们学习程序设计、进行程序开发的准则。要想成为一名真正专业的程序设计人员，基本的数据结构基础和常用的算法知识是必须掌握的。脱离了这两点，编写出来的程序一定不是健壮的好程序。

当然单纯地掌握了一些数据结构基础和常用的算法知识也是远远不够的。空洞地掌握所谓的数据结构和算法等理论知识只是纸上谈兵，这些知识必须依托于一门程序设计语言才具有真正的生命力，才能够转化为真实的程序代码，才能真正地解决实际问题。

伴随计算机技术的发展，将沃思的公式扩展为：

程序＝算法＋数据结构＋程序设计方法＋语言工具和环境

四个方面中，算法是灵魂，数据结构是加工对象，语言是工具，编程要采用适当的方法。算法解决了做什么和怎么做的问题。本书并非专门介绍算法，借助本章，单介绍算法的特点及对程序处理的作用。

2.1　数据和算法

1. 数据（Data）

数据是程序中最为基本的概念。自然中的一个数或者一个独立的事物都可用计算机程序中的一个数据来表征。程序设计的一个主要问题就是如何将要处理的数据有机组织起来（数据结构），这里仅考虑一个数据的表达问题。

对于一个数据或者一个量，在程序中主要考虑 4 个特征：

1）值或者内容

一个数据的值是计算机要处理的或者产生的。在程序设计中一定要注意任何一个数据在第一次使用的时候一定要有一个确定的值。