语言程序设计

第1章 引　　论

1.1 基 础 知 识

1.1.1 计算机系统

任何一个计算机系统，无论是大型机、中型机、小型机、微型机、个人计算机或计算机网络，均由硬件和软件两部分组成。硬件部分是构成计算机系统的物理设备，是整个计算机系统的物质基础；软件部分包括使用和管理计算机所需的各种程序、有关的数据和文档，软件是计算机系统的灵魂。

1．硬件部分

计算机单机系统的硬件部分包括主机和外部设备。主机由中央处理单元（简称 CPU）和主存储器（简称内存）组成，外部设备包括输入设备、输出设备和外部存储设备。输入设备和输出设备通常简称为 I/O 设备。外部存储设备通常简称为外存，外存是内存的辅助存储设备。计算机系统的硬件组成如图 1.1 所示。

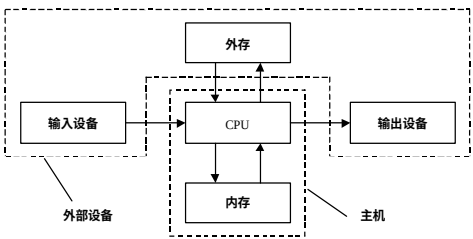


图 1.1　计算机系统的硬件组成

（1）CPU　CPU 是计算机系统的心脏，主要包括运算器和控制器两部分。运算器完成所有的算术运算和逻辑运算；控制器负责从内存取出指令和数据，并执行指令规定的动作以及将执行结果存入内存。

（2）内存　内存用于存储正在执行的程序和被程序处理的数据。内存由若干连续的存储单元组成，存储单元是一些仅具有 0 和 1 两种状态的电子元件。因此，每个这样的电子元件能够表示二进制数的一位（称为一个 bit），多个相邻的 bit（二进制位）就能够表示由 0 和 1 数字串组成的代码（称为二进制代码）。内存中的程序和数据就是以这种二进制代码形式存储的。

存储单元的编号称为存储单元的地址（或内存地址）。8 个相邻的二进制位称为一个字节，存储单元从 0 开始依次按字节编址。内存所包含的存储单元的总字节数称为内存的容量。CPU 数据总线的 bit 数称为计算机的机器字长，例如：若机器字长为 2 字节则称为 16 位机，若机器字长为 4 字节则称为 32 位机。内存中存储的程序和数据称为存储单元的内容，从存储单元取出内容称为读内存，将程序或数据存入内存称为写内存。存取或读写内存统称为访问内存，访问内存必须由 CPU 通过存储单元的地址进行。

在高级语言程序中，存取内存中的数据一般是通过程序中的变量进行的，每个变量对应于一定数目的存储单元（以字节为单位），存储单元的内容就是变量的值。在 C 程序中，还可以通过存储单元的地址（即指针）访问内存中的数据。

作为一个程序员，必须了解内存的以下重要性质：

- 任何时刻计算机一经加电则存储单元中必有内容，但在未通过程序存入内容之前，这些内容仅是随机的，无意义的信息。
- 任何时刻计算机一旦切断电源则存储单元中的一切内容立即消失，将不复存在。
- 读内存操作永远不改变内存中的内容，即如果仅仅执行读内存的操作，无论读多少次，存储单元中的内容保持不变。
- 写内存操作要破坏存储单元中原来的内容。任何时候一旦执行了写内存操作，则存储单元中原来的内容立即被新写入的内容所代替，原来的内容不复存在（通常称为被冲掉或被覆盖）。

（3）输入输出（I/O）设备　I/O 设备用于人和计算机通讯。输入设备用于将程序、各种形态的数据或文档送入计算机内存，输出设备用于将存储在计算机内存中的各种信息送到外部设备上显示、打印或保存。

输入设备分为字符设备和图形设备两类。最普通的字符输入设备是终端键盘，最普通的图形输入设备是鼠标。输出设备主要包括终端显示器、打印机和绘图仪，打印机和终端显示器可以兼作字符输出设备和图形输出设备，绘图仪只能用于输出图形。其中，终端键盘和终端显示器通常被系统约定为标准输入设备和标准输出设备。

（4）外存　外存既是信息的永久存储设备又起着输入输出设备的作用。主要的外部存储设备有磁盘驱动器、磁带机和光盘驱动器，内存的信息可以通过这些外部存储设备写到磁盘、磁带或光碟等存储媒体上。信息一经写入，只要存储媒体未被损坏或未重新写入新的信息，则信息将被永久保存（关掉电源以后信息也不会丢失）；另一方面，保存在外存上的任何程序必须从外存读入内存才能被执行，任何数据必须从外存读入内存才能被处理。换言之，当需要执行外存上的程序或需要处理外存上的数据时，必须首先将它们从外存输入到内存。

外存容量比内存容量大得多，外存容量一般以 GB（1GB= 10^9B）为度量单位，而内存容量一般以 MB（1MB=10^6B）为度量单位。所以，外存是内存的辅助存储设备，可用于存放大量的程序、数据和文档。

2．软件部分

计算机系统的软件部分包括系统软件和应用软件。

（1）系统软件

系统软件由计算机厂家及软件公司提供，是用于使用和管理计算机的各种程序及相关数据和文档的总称。系统软件主要包括操作系统、语言处理程序、各种实用程序和数据库管理

系统等。

● 操作系统　操作系统是最基本的核心系统软件，用于管理计算机系统的各种资源并控制各种程序的正常执行，例如 MS-DOS、Windows、UNIX 等。

● 语言处理程序　语言处理程序是将程序设计语言转变为计算机能够直接识别的机器语言的翻译程序，例如 IBM 汇编程序、Basic 解释程序、C 编译程序等。

● 实用程序　实用程序是为用户提供某种服务功能的程序，例如 DOS 环境下的文本编辑程序 Edit，Windows 环境下的图文排版程序 Word，计算机病毒杀毒软件等。

● DBMS　DBMS（数据库管理系统）是能够集中存储和统一管理某应用领域用户的所有相关信息，并具有数据独立性、完整性、一致性和安全性的数据管理软件。

（2）应用软件

应用软件是为使用计算机处理各种实际问题而开发的具有专门用途的程序，由于计算机的应用极为广泛，因而应用软件多种多样，例如各种 MIS 系统、CAD 系统和过程控制系统等。

1.1.2　算法及其表示

1．算法的基本特性

人们通常把计算机为解决某一问题所需的方法与步骤称为算法。算法可分为两大类：一类是科学计算领域用于处理数值数据的算法，例如求定积分、解方程、求极限等；另一类是数据处理领域用于处理非数值数据的算法，例如分类排序、情报检索、绘图等。算法具有以下基本特性：

● 有穷性　一个算法包含有限个步骤，即算法经过有限步执行后必须终止。

● 确定性　算法的每一步所规定的动作不能有两种以上的理解，即算法每一步的动作不能有语义二义性。

● 有输入　一个算法有一个或多个输入，输入是执行算法时所需的信息，包括被算法处理的数据和执行的控制信息。

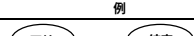
● 有输出　一个算法有一个或多个输出，输出是算法执行的结果。

● 有效性　算法每一步所规定的动作都能够执行。例如两个数相除，当除数为零时则不能执行除运算，即算法所规定的动作无法执行。

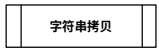
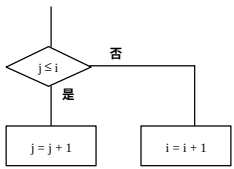
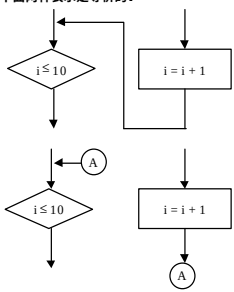
2．算法的表示

将算法用一种适当的方式描述出来称为算法的表示。算法的表示有多种方式，本书采用了两种最基本和最常用的方式——自然语言和传统流程图来表示算法。传统流程图通常简称流程图或框图。流程图用图形符号（框和线）表示算法的每一步及各步之间的联系，流程图常用符号及其含义如表 1-1 所示。

表 1-1　流程图常用符号

符　号	符号名称	意　义	例
	起止框	表示算法的开始或结束	

续表

符　　号	符号名称	意　　义	例
	判断框	表示判断选择：根据框中条件从两种可选动作中选一执行	
	处理框	表示按顺序执行的处理	
	调用框	表示调用函数	
	流程线	表示两个步骤相邻，且执行顺序由箭尾一方到箭头一方，对于自上而下和自左而右的顺序，箭头可省缺	
	连接点	连接点必须以相同的形式成对出现，用于表示一条流程线被断开后的两个端点	下面两种表示是等价的： 

3．算法表示举例

问题：求 $s=\sum_{n=1}^{10} n^n$

下面分别用自然语言和流程图表示该级数求和的解题算法。

- **用自然语言表示**

① 用变量 s 存放各项的累加和，置初值为 0；用变量 i 作项数计数器，置初值为 1。

② 如果 i≤10 则计算 s，否则转步骤③。

计算 s：

②-1　用变量 j 作每一项的累乘次数计数器，置初值为 1；

②-2　用变量 ai 存放第 i 项的累乘积，置初值为 1；

②-3　如果 j≤i 则计算第 i 项，否则转步骤②-4；

计算第 i 项：

②-3-1　ai=ai*i；

②-3-2　j=j+1，转步骤②-3；

②-4　将第 i 项加到累加和中去：

②-4-1　s=s+ai；

②-4-2　i=i+1，转步骤②；

③ 输出 s，结束。

- 用流程图表示

其具体流程图见图 1.2。

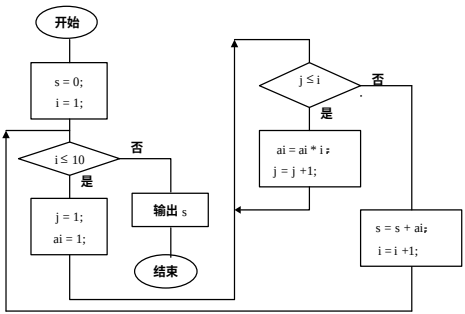


图 1.2　计算 $s=\sum_{n=1}^{10} n^n$ 的算法流程

显然，用流程图表示算法比用自然语言表示算法更为简明、直观。

1.1.3　程序设计及程序设计语言

程序是用程序设计语言表示的计算机解题算法或计算机解题任务。程序设计是将解题任务转变成程序的过程，一般包括分析问题，确定算法（对复杂算法需画出流程图），用选定的程序设计语言编写源程序，最后上机调试、运行程序等基本步骤。

程序设计语言是计算机能够理解的、用于人和计算机通讯之间的语言，程序设计语言由低级到高级可分为三类：低级语言、高级语言和专用语言。

1．低级语言

低级语言又分为机器语言、符号语言和汇编语言。机器语言用二进制代码表示机器指令和数据，机器语言程序能够直接被机器理解和执行，因而程序效率高；但编程繁琐，且不便于记忆和阅读，因而程序维护困难。符号语言用符号代替二进制代码表示机器指令，汇编语言进而用符号来表示指令和数据的内存地址。现在人们用低级语言编程通常指用汇编语言编程。

用汇编语言编写的程序称为汇编语言源程序（简称汇编语言程序）。汇编语言程序必须被转换为机器语言程序才能被计算机理解和执行，完成这种转换任务的系统软件称为汇编程序；

这种转换过程称为汇编。低级语言是面向机器的，用低级语言写的程序效率高，但没有可移植性，即不能从一个机器系统上移到另一个机器系统上运行。此外，用机器语言编写程序要求程序员必须懂得具体机器系统的硬件结构（指令系统）。

2．高级语言

高级语言是类似于人类自然语言，英语和数学语言的程序设计语言，例如 C 语言和 Pascal 语言。用高级语言编写的程序称为高级语言源程序，简称源程序或程序。例如 C 源程序或 C 程序，Pascal 源程序或 Pascal 程序。

与汇编语言源程序类似，高级语言源程序也必须被转换成机器语言程序才能够被机器理解和执行，完成这种转换任务的系统软件称为编译程序。例如，C 编译程序、Pascal 编译程序。将高级语言源程序转换成机器语言程序的过程称为编译。

高级语言是面向解题过程的，语言本身与具体机器系统无关，因而用高级语言编写的应用程序可移植性好。编译程序是一种语言的具体实现，编译程序与具体机器系统有关，人们常将一个编译程序称为某种语言的一个版本。同一种语言的不同版本不完全相同，在使用一种具体的高级语言及其编译程序开发软件时，必须参考与编译程序配套的有关资料。本书阐述的内容遵从 C 标准，对于大多数 C 编译程序具有通用性。考虑到上机环境，尽量简单以便于学习和练习，书中所有例题均在 Turbo C 2.0 上运行通过。

3．专用语言

专用语言是专门为某个应用领域而设计的计算机程序设计语言。例如，数据库系统中的数据查询语言 SQL、军事应用领域的 ADA 语言、商业应用领域的 COBOL 语言等。专用语言是面向问题的语言，它比高级语言更抽象，描述能力比高级语言更强。用专用语言编程不需要指出“如何做”，只需说明“做什么”。

1.2 C 语言的发展过程及特点

1.2.1 C语言的发展过程

C 语言是迄今为止世界上应用最广泛的通用程序设计语言之一。C 语言的发展过程可粗略地分为诞生、发展和成熟三个阶段。

1．C 语言的诞生（1970 年~1973 年）

C 语言是由于编写 UNIX 操作系统的需要而诞生的，作者是美国 AT&T 公司贝尔实验室的 Dennis. M. Ritchie。C 语言的前生是 B 语言（1970 年，作者：贝尔实验室的 Ken Thompson），B 语言的前生是 BCPL 语言（1967 年，作者：英国剑桥大学的 Martin Richards）。

B 语言被用在 PDP_7 计算机上实现了第一个 UNIX 操作系统。1973 年至 1975 年 K.Tompson 和 Dennis. M.Ritchie 用 C 语言重写了 UNIX 操作系统，先后推出了 UNIX V5，UNIX V6。此时的 C 语言是附属于 UNIX 操作系统的。

2．C 语言的发展（1973 年~1978 年）

1977 年 C 语言作者发表了不依赖于具体机器系统的 C 语言编译文本《可移植 C 语言编译程序》，1978 年又发表了对上述编译文本的补充《C 语言修订报告》，从而使 UNIX 操作系统推广到各种计算机上并推动了 UNIX 操作系统的不断发展：1978 年推出了 UNIX V7，1983 年推出了 UNIX System Ⅴ。UNIX 操作系统的巨大成功和广泛使用展示了 C 语言的突出优点，从而又促进了 C 语言的迅速推广。同时，C 语言也伴随着 UNIX 操作系统的发展而不断发展。

1978 年 Brian W. Kernighan 和 Dennis. M. Ritchie 以 UNIX V7 中的 C 编译程序为基础写了经典著作《The C Programming Language》，这本书是以后介绍 C 语言的各种书籍的蓝本。由《可移植 C 语言编译程序》和《C 语言修订报告》定义的 C 语言被称为传统 C。1978 年以后，C 语言先后移植到各种计算机系统上，且不再依赖于 UNIX 操作系统而独立存在。

3．C 语言的成熟（1988 年以后）

C语言的不断发展产生了各种C语言版本，不同的C语言版本对传统C都有所扩充。1988 年，美国国家标准协会（ANSI）在综合各种C语言版本的基础上制定了C语言文本标准，称为**ANSI C**标准。ANSI C实现了C语言的规范化和统一化，B. W. Kernighan和Dennis. M. Ritchie按照ANSI C标准重写了《The C Programming Language》一书，于 1990 年正式发表了《**The C Programming Language Second Edition**》。1990 年国际标准化组织（ISO）公布了以ANSI C为基础制定的C语言的国际标准**ISO C**，人们通常称之为标准C。C语言标准的制定标志着C语言的成熟，1988 年以后推出的各种C语言版本对标准C是兼容的。

1.2.2 C语言的特点

C 语言之所以成为目前世界上使用最广泛的程序设计语言，并被选作适应近代软件工程需要而发展起来的面向对象的程序设计语言 C++的基础语言，是由于 C 语言的诸多突出优点所决定的。下面从语言本身和应用角度两个方面概括了 C 语言的主要特点。

C 语言的突出优点是表达能力强、语言简洁、使用灵活、效率高、可移植性好。

- 表达能力强　数据类型和运算符丰富；可以直接访问内存物理地址和硬件寄存器；能够表示直接由硬件实现的针对二进制位的运算。
- 语言简洁　只有 6 种基本语句（见第 3 章），且运算符、语句等语言成份的表示简明扼要，使 C 源程序显得精练。
- 使用灵活　C 语言是弱类型语言，基本类型的变量及其值在数据类型上不要求严格对应；表达式可以作为语句使用；用非零及整数零表示具有真假含义的逻辑值；数组元素和结构成员既可以用其名称访问也可以用指针访问，即同一数据可以有多种等价的表示形式。
- 效率高　C 语言在代码质量上几乎可以与汇编语言媲美，即 C 源程序经编译后生成的目标程序运行速度快且占用的存储空间小。
- 可移植性好　由于 C 语言的标准化，以及 C 程序的输入输出、内存管理等操作采用 C 库函数实现，不仅使得 C 编译程序很容易在不同的机器系统上实现，而且使得用户 C 程序可以不作修改或作少量修改就能在不同的机器系统上运行。

注意：C 语言使用灵活是 C 语言深受程序员喜爱的原因之一。但如果用法不正确，会使写出的程序不能得到正确的结果，或不能正常运行，甚至死机。用法上的错误不属于语法错，

因此编译是检查不出来的。初学者往往误认为没有语法错的程序就是正确的程序，编写正确、可靠的 C 程序要靠程序员正确使用 C 语言来保证。

例如最常见的用法错误有：写表达式时运算符的优先级与结合性同所要表示的逻辑不一致，或未注意表达式求值的顺序、表达式求值的副作用、后缀式自增自减的计算延迟、运算过程中的类型转换等对计算结果的影响；写循环语句时循环变量未赋初值，或用作循环条件的表达式的值永为真（非 0），或组成循环体的多个语句未写成复合语句；写格式输入输出语句时用于说明数据格式的转换字符与数据的类型不一致；特别是使用指针时没有向指针赋值之前就向指针所指对象赋值，或引用指针所指对象的值，或写地址表达式时不注意指针的类型等都是严重的用法错误。

1.3 C 程序的基本结构

一个计算机高级语言程序均由一个主程序和若干个（包括 0 个）子程序组成，程序的运行从主程序开始，子程序由主程序或其他子程序调用执行。在 C 语言中，主程序和子程序都称之为函数，规定主函数必须以 main 命名。因此，一个 C 程序必须由一个名为 main 的主函数和若干个（包括 0 个）子函数（简称函数）组成，程序的运行从 main 函数开始，其他函数由 main 函数或其他函数调用执行。

下面是几个具有代表性的简单 C 程序。

【例 1.1】输出字符串“The c programming language”，并在输出后换行。

程序 1.1

```
/* 1*/ /*example1-1.c*/
/* 2*/ #include<stdio.h>
/* 3*/
/* 4*/ void main（void）
/* 5*/ {
/* 6*/       printf（"The c programming language\n"）;
/* 7*/ }
```

程序运行时输出下面一行信息，显示器屏幕光标停在下一行的行首：

```
The c programming language
_
```

程序 1.1 仅由一个主函数（4~7 行）组成。

第 1 行：/*examp1e1.1.c */是注释，/*是注释的开始符号，*/是注释的结束符号，注释符号之间的文字是注释内容。/*和*/必须成对出现且不能嵌套，例如/*.../*...*/...*/是非法的。注释是供程序员看的，编译对注释不作翻译。注释内容任意，例如 examp1e1-1.c 为该程序的磁盘文件名，也可以是对程序功能、被处理数据或处理方法的说明。注释可以出现在程序中分隔符（见 1.4.3 节）可以出现的任何位置。

第 2 行：#include 是编译程序的预处理指令（见 5.6.2 节），它不是 C 语言的语句，指令末尾不能加分号。stdio.h 是 C 编译程序提供的系统头文件（或称为包含文件）之一，程序中凡是调用了标准输入输出函数则必须在调用之前写上#indude<stdio.h>（注意，只有在 DOS 操作系统环境下可以缺省#indude<stdio.h>）。预处理指令必须独占一行，一般写在一个源程序文

件的开始部分。

第 4 行：main 是主函数的函数名，void 是一个类型名。main 前面的 void 是函数返回值的类型，用 void 说明函数的返回值类型表明该函数没有返回值（void 不能缺省）；main 后面用圆括号（）括起来的部分是函数参数表，（）中的 void 说明该函数的参数表为空，即该函数没有参数（void 可以缺省）。函数返回值类型、函数名和参数表三部分合起来称为函数的头部。

第 5~7 行：用花括号｛｝括起来的部分称为函数体，函数体中可以有说明部分和语句部分。说明部分由说明语句组成，语句部分由可执行语句组成。本例中主函数的函数体无说明语句，仅有一个可执行语句（见第 6 行），该语句是对 C 的标准输出函数 printf 的调用，习惯上称为输出语句。该语句执行的结果是输出一行英文文字——“The c programming language”（称为字符串），其中\n 是换行字符，表示输出字符串中\n 左边的文字之后要回车换行。分号是一个 C 语句的结束符，每一个 C 语句都必须以分号结束。

【例 1.2】输入两个整数，输出其中较大的一个值。

程序 1.2

```
/* 1*/ /*example1-2.c：找两个数中的较大值并输出结果*/
/* 2*/ #include<stdio.h>
/* 3*/
/* 4*/ void main（void）
/* 5*/ {
/* 6*/      int a, b, c;
/* 7*/
/* 8*/      scanf（"%d%d", &a, &b）;
/* 9*/      if （a > b）      c = a;
/*10*/      else  c = b;
/*11*/      printf（"max=%d\n", c）;
/*12*/ }
```

程序运行时输入 100 200，则输出：

```
max=200
_
```

程序 1.2 也是仅由一个 main 函数组成，但函数体包括两部分：说明部分和语句部分。说明部分必须位于语句部分之前。说明部分包含一个说明语句 int a,b,c;（见第 6 行），该说明语句定义了 a，b，c 三个整型变量，分别用于存放用户输入的两个整数及比较结果。语句部分包含三个可执行语句（见 8~11 行）。

第 8 行：是对标准输入函数 scanf 的调用，习惯上称为输入语句。该语句的作用是从标准输入设备（键盘）输入两个整数分别赋予变量 a 和 b。参数%d%d 是格式字符串（见 3.2.3 节），由两个%d（转换说明）组成，用于指出两个输入数据均为十进制整数；&a 和&b 分别表示变量 a 和变量 b 的地址。scanf 函数要求格式字符串后面的参数必须是变量的地址（指针参数）。

第 9~10 行：是 if-else 语句，用于判断 a 和 b 中哪个值较大，并将较大的一个值赋予变量 c。

第 11 行：用于输出变量 c 的值，即求解结果。其中%d 的含义同 scanf 中的%d，它指出

c 的值按十进制整数格式输出。输出时在%d 的位置上被替换成 c 的值，max=是普通字符，照样输出。

【例 1.3】修改程序 1.2，将其中找较大值的任务定义成一个函数。

程序 1.3

```
/* 1*/ /*example1-3.c*/
/* 2*/ #include<stdio.h>
/* 3*/
/* 4*/ int max(int x, int y)
/* 5*/ {
/* 6*/     int z;
/* 7*/
/* 8*/     if (x > y)      z = x;
/* 9*/     else  z = y;
/*10*/     return (z);
/*11*/ }
/*12*/
/*13*/ void main(void)
/*14*/ {
/*15*/     int a, b, c;
/*16*/
/*17*/     scanf("%d%d", &a, &b);
/*18*/     c = max(a, b);
/*19*/     printf("max=%d\n", c);
/*20*/ }
```

程序 1.3 的功能与程序 1.2 完全相同，只是程序的结构不同。程序 1.3 由两个函数组成：main 函数和一个名为 max 的用户定义函数（见 4~11 行），max 函数由主函数 main 调用执行（第 18 行：c=max（a,b)；，该语句的作用是调用 max 函数并将函数的返回值赋予变量 c)。

用户定义函数 max 用于代替程序 1.2 中第 9~10 行的 if-elie 语句找出两个整数中较大的一个值，并由 max 函数体中的 return 语句将该值返回给 main 函数（由 max（a,b）表示该值)。函数 max 的语法形式与主函数 main 相同，只是各组成部分的具体内容不同。max 函数有 x 和 y 两个整型参数，参数表（int x，int y）是对参数的定义（称为说明），参数表中说明的参数称为形式参数，简称形参。函数返回值类型为整型，返回值为形参 x 和 y 中较大的一个值。形参 x 和 y 的值来源于调用时的参数（称为实际参数，简称实参）的值，即 main 中变量 a 和 b 的值。a 的值被赋予形参 x，b 的值被赋予形参 y。

C 程序基本结构小结

（1）C 程序的组成

一个 C 程序可以由若干个函数构成，其中必须有且只能有一个以 main 命名的主函数，可以没有其他函数。每个函数完成一定的功能，参数是被函数处理的数据，参数能够在函数与函数之间传递数据。

main 函数可以位于源程序文件中任何位置，但程序的运行总是从 main 函数的第一个可

执行语句开始，当遇到一个函数调用时，执行的控制转入被调用函数；从被调用函数返回到调用函数后继续执行调用点之后的可执行语句。

（2）函数的组成

函数是一个独立的程序块，相互不能嵌套。main 函数以外的其他任何函数只能由 main 函数或其他函数调用，自己不能单独运行。

一个函数由两部分组成：函数头部和函数体。函数头部包括函数返回值的类型、函数名和参数表（函数头部的末尾不能加分号）。参数表可以为空，参数表为空时用类型名 void 表示（void 可以缺省）。函数体包括说明部分（局部说明）和语句部分，可以没有说明部分，也可以没有语句部分。说明部分和语句部分都为空的函数称为哑函数，例如 int max（int x, int y）{ }，max 是一个哑函数。哑函数是一个最小合法函数，调用一个哑函数在功能上不执行任何操作，但在调试由多个函数组成的大程序方面很有用处。

（3）C 标准函数

C 函数分为两类：标准函数和用户定义函数。用户定义函数是由程序员在自己的源程序中编写的函数，例如程序 1.3 中的 max 函数。标准函数是由 C 编译程序提供的一些通用函数，这些函数以编译后的目标代码形式集中存放在称为 C 标准函数库（简称 C 库）的文件中，C 标准函数又称为 C 库函数。例如，scanf 和 printf 函数都是 C 库函数。

用户程序需要使用标准函数时，只需在使用前用#include 包含该标准函数所需的系统头文件，例如 scanf 和 printf 函数的头文件为 stdio.h，然后按规定的格式调用所需标准函数即可。系统头文件中包含了标准函数的相应说明（函数原型）、有关的类型定义及常量定义。

（4）书写格式

书写 C 程序时，一个语句可以写成多行，但不能在一个单词（见 1.4 节）内部换行；也可以在一行上写多个语句，但最好一行只写一个语句（有利于程序调试）。

为了使程序清晰、美观，易于阅读，易于在调试程序时检查错误，对具有嵌套结构的语句应写成层层缩进对齐的格式。即将处于同一层次的语句在列上对齐，处于下一层次的语句用制表符（按 Tab 键）使其向右缩进相同的空白。此外，程序中还应加必要的注释（英文或汉字均可），对程序员自己起备忘录的作用，对他人则起帮助理解程序功能和算法的作用。

1.4 C 语言的基本语法单位

C 语言的各种语法对象，例如，表达式、语句和函数，都是由基本语法单位按照一定的语法规则构成的。基本语法单位是指具有独立语法意义的最小语法成分，在 C 语言中，基本语法单位被称为单词，组成单词的基本符号是字符，标准 C 及大多数 C 编译程序使用的字符集是 ASCII 字符集（见附录 A）。

C 语言的单词分为六类：标识符、关键字、常量、字符串、运算符及分隔符。常量、字符串及运算符在下章介绍，本节介绍标识符、关键符及分隔符。

1.4.1 标识符

1．标识符的含义

C 语言中，标识符是指程序中的常量、变量、数据类型和函数的名字。例如程序 1.1~程序 1.3 中的变量名 a，b，c，z，形参 x，y 以及函数名 max 都是标识符，主函数名 main 和标准函数的名字（例如 scanf，printf）也是标识符。数据类型名 int 和 void 不是标识符而是关键字（见 1.4.2 节）。

2．标识符的组成规则

① 确良标识符由字母（A~Z，a~z）和数字（0~9）组成，必须以字母开头，随后可跟 0 个或多个字母或数字（即标识符是以字母开头的字母、数字串）；

② 字母要区分大小写，例如 A 和 a，Ab 和 ab 等都是不同的标识符；

③ 下划线（_）被作为一个字母看待。

在使用标识符时，习惯上变量名和函数名用小写，常量名（符号常量 2.2.2 节）和用 typedef 定义的数据类型名（见 8.7 节）用大写。此外，为便于阅读和记忆，应选用能够表达含义的英文单词、英文单词的一部分或缩写作为标识符，例如 day，date，year，sum，flag，lines，count，HEIGHT_START，HEIGHT_END 等。

下面是一些合法的标识符例子：

x　x1　i　name　_ab　a_b　ab_　A　PI

下面的表示均不是标识符：

4ab	不是以字母开头（非法表示）
order.no	小数点（.）不是字母也不是数字
up-to	减号（-）不是字母也不是数字（非法表示）
name[i]	[]不是字母也不是数字
p->x	->不是字母也不是数字

3．标识符的有效长度

在组成标识符的字符中，能够被编译程序识别的那一部分字符的数目称为标识符的有效长度。这就是说，程序员可以写一个很长的标识符，但在有效长度以内的字符才是有意义的字符。标准 C 规定标识符的有效长度为前 31 个字符。实际应用中为便于记忆和书写，在能够区别于其他标识符及能够表达一定含义的前提下，标识符应尽可能简短一些。

注意：标识符不能与关键字同名。

1.4.2 关键字

关键字由固定的小写字母组成，是由系统预先定义好的名字，用于表示 C 语言的语句、数据类型、存储类型或运算符。用户不能用它们来作为自己定义的常量、变量、数据类型或函数的名字。关键字又称为保留字，即被系统保留作为专门用途的名字。

标准 C 定义的 32 个关键字如下：

char　int　short　long　signed　unsigned　float　double

const void volatile enum struct union typedef
auto extern static register
if else switch case default while
do for break continue goto return
sizeof

以上关键字将在各章内容中逐一涉及，程序员应熟记这些关键字。

1.4.3 分隔符

分隔符是一类字符，包括空格符、制表符、换行符、换页符及注释符。分隔符统称为空白字符，空白字符在语法上仅起分隔单词的作用。程序中两个相邻的标识符、关键字或常量之间必须用分隔符隔开（通常用空格符）。换句话说，当两个单词之间如果不用分隔符就不能将两者区分开始时，则必须加分隔符。

例如：程序 1.1 至程序 1.3 中主函数的头部 void main （void）不能写成

voidmain（void）

因为 voidmain（void）表示函数名为 voidmain，而函数返回值类型为可以缺省的 int。但可以写成

void main（void）

因为 main 和参数表中的 void 之间有“(”隔开，所以可以加或不加空格。

此外，任何单词之间都可以适当加空白字符使程序更加清晰，更易于阅读。例如将变量说明 int a,b,c；写成 int a, b, c；较好，后者在逗号（,）后面加了一个空格符。

1.5 运行 C 程序的一般步骤

运行一个 C 程序，是指从建立源程序文件直到执行该程序并输出正确结果的全过程。在不同的操作系统和编译环境下运行一个 C 程序，其具体操作和命令形式可能有所不同，但基本过程是相同的，即必须经过如图 1.3 所示的四个步骤。

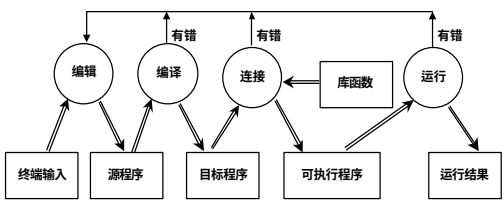


图 1.3 运行 C 程序的一般步骤

图中每一个园框表示一个处理步骤，方框表示处理的输入数据或输出数据，双线箭头表示数据的流向，单线箭头表示各处理步骤之间的关系。

1．建立源程序文件（编辑）

在文本编辑环境下，例如 UNIX 的 vi，DOS 的 edit，或 C 编译程序提供的集成开发环境中的编辑窗口，建立以.c 为扩展名的 C 源程序文件。

2．编译

将 C 源程序翻译为机器代码形式的目标程序，在 UNIX 环境下得到扩展名为.o 的目标程序文件，在 DOS 环境下得到扩展名为.obj 的目标程序文件。如果有语法错误，则回到步骤 1，在编辑环境下修改源程序，然后重新执行步骤 2。重复此过程，直到没有语法错误为止。

3．连接

将步骤 2 得到的目标程序与 C 库函数装配成可执行的程序，在 DOS 环境下得到扩展名为.exe 的执行程序文件，在 UNIX 环境下得到扩展名为.out 的执行程序文件。如果连接不成功，根据错误情况可能要重复步骤 1 至 3，直到连接成功为止。

4．运行

执行扩展名为.exe 或.out 文件的程序。如果程序不能正常运行，或输出结果不正确，则需要重复步骤 1 至 4，直到正常运行并输出正确结果为止。

程序运行时出现的错误称为动态错，通常是由于循环控制不当，或算法逻辑有错，或输入输出方面有错而引起的。每个步骤的命令形式视具体编译程序而定，上机操作需参考所用操作系统和编译程序的有关资料。附录 D 给出了在 Turbo C 2.0 集成环境下运行 C 程序的方法。

小　结

本章介绍了 C 语言的发展史、特点及与程序设计密切相关的一些基础知识，应重点掌握以下几点：

（1）掌握流程图的画法。

（2）弄清 C 语言标识符的含义，掌握标识符的组成规则、对标识符的有效长度和大小写的规定。

（4）了解关键字的含义，逐步熟记标准 C 定义的 32 个关键字。

（5）弄清 C 程序的基本结构。

习　题　一

1．简述计算机系统的组成及各部分的作用。

2．什么是读内存、写内存和访问内存？对程序员来说应了解内存的哪些重要性质？

3．什么是系统软件和应用软件？主要的系统软件是哪些？各自的功能如何？

4．什么是算法？算法具有哪些基本特性？如何用流程图描述算法？

5．什么是程序、程序设计和程序设计语言？什么是源程序和编译程序？

6．C 语言的主要特点是什么？

7．简述 C 程序的组成和 C 函数的结构。

8．C 语言包括哪些基本语法单位？C 语言中标识符的含义是什么？对标识符的语法（组成规则、有效长度和大小写）有哪些规定？

9．指出下面程序中的错误：

```
/*sum.c：/*计算 s=1+2+3+…+10*/*/
#include <stdio.h>;
int sum（）;
{
    int s, n;

    s=0; n=1;
    while （n<=10） {
        s=s+n;
        n=n+1
    }
    return（n）
};

voidmain（）
{
    int to-tat;
    to-tal=sum（）;
    printf（“sum=%d\ n”,to-tal）;
}
```

10．运行 C 程序包括哪些基本步骤？各个步骤的作用是什么？

第 2 章　基本数据类型和运算

本章内容是用 C 语言编写程序的重要基础，包括基本数据类型、常量和变量、运算符和表达式以及运算中的数据类型转换等内容。

2.1　基本数据类型

数据是程序处理的对象。在高级程序设计语言中，数据的取值范围是由该类型数据的存储长度（即存储单元字节数）及数据在机器内的表示方法决定的。为了数据存储和处理的需要，编译程序将数据划分为不同的类型，并为每一种类型的数据规定了数据在内存存放的存储单元字节数（称为类型的长度）和对该类型数据所允许的运算。由于数据类型的长度总是有限的，所以任何一种类型的数据其值均被限制在一定的范围，称为数据类型的值域。简言之，数据类型决定一种数据值的范围和对该种数据所允许的运算。

2.1.1　C的数据类型分类

C 有丰富的数据类型。C 的数据类型（见表 2-1）分为基本类型和导出类型。基本类型包括整型和浮点型（即实数类型），整数和浮点数都是算术量，所以基本类型又称为算术类型。整型包括各种整型、字符型和枚举型；浮点型分为不同精度的浮点型。从数据的结构上分，基本类型和指针类型是简单类型，即一个数据仅由一个成员组成；除指针以外的导出类型都是构造类型，即一个数据由多个成员组成。

本章介绍基本类型，导出类型在第 6 至 8 章介绍。

表 2-1　C 的数据类型

类　别		数 据 类 型
基本类型	整型	char（字符）
		int（整数）
		short（短整数）
		long（长整数）
		signed（有符号整数）
		unsigned（无符号整数）
		enum（枚举）
	浮点型	float（单精度浮点数）
		double（双精度浮点数）
		long double（长双精度浮点数）
导出类型		指针
		数组
		结构（struct 结构名）
		联合（union 联合名）