基　础　篇

第 1 章　C 语言概况

本章主要介绍程序设计的基本概念、结构化程序设计方法、C 语言的起源与特点、初识 C 语言程序以及 C 语言程序的开发过程。通过本章的学习，读者可对 C 语言程序设计的基本概念和过程有一个大致的了解，并通过模仿编写求解简单问题的 C 语言程序。

1.1　程序设计的基本概念

1.1.1　程序和程序设计语言

什么是程序？怎样设计程序？这往往是计算机语言初学者首先遇到的问题。有人以为计算机是“万能”的，只要把任务告诉计算机，计算机就会自动完成一切，并给出正确结果。其实，这是一种误解，要让计算机按照人的意志来完成某项任务，首先要编制该项任务的解决方案，再将其分解成计算机能够识别并可以执行的基本操作指令，并把这些指令按一定的规则组织排列起来存放于计算机内存储器中，当给出执行命令后，计算机按照规定的流程依次执行存放在内存储器中的指令，最终完成所要实现的目标任务。人们把这种计算机能够识别并可以执行的指令序列称为程序。也就是说，程序是人与计算机进行“交流”的工具，它用我们常说的程序设计语言来描述。

程序设计语言是计算机能够理解和识别的语言。它通过一定的方式向计算机传送操作指令，从而使计算机能够按照人们的意愿进行各种操作处理。任何一种程序设计语言都有一定的使用规则，通常称之为语法规则。要学习程序设计语言，必须注意学习它的语法规则，就像学习汉语要学习汉语语法一样。而学习程序设计语言的目的就是为了设计计算机程序。

程序设计语言的种类很多，大体上经过了由低级语言到高级语言的发展过程，目前广泛使用的有 C、Pascal、C++、Java、Delphi 等高级语言，这些高级语言采用的都是接近于人们熟悉的数学语言和自然语言的表达形式，使人们的学习和使用更加容易和方便。我们把由高级语言编写的程序称为源程序。显而易见，用高级语言编写的源程序，计算机不能直接识别并执行，因为计算机只能识别和执行二进制形式指令或数据，因此，必须有一个工具先将源程序转换成计算机能够识别的二进制形式程序，我们把这种二进制形式表示的程序称为目标程序，而承担转换的工具称为语言处理程序。每种程序设计语言都有与它对应的语言处理程序，语言处理程序对源程序的处理方式有编译方式和解释方式两种，相应的转换工具分别称为编译程序和解释程序。编译方式是指将源程序输入计算机中后，用相应的编译程序将整个源程序转换成目标程序，然后再通过装配连接程序形成可执行程序，最后运行可执行程序得到结果。目标程序和可执行程序都是以文件的方式存放在磁盘中的，再次运行该程序，只需直接运行可执行程序，不必重新编译和连接。解释方式是指将源程序输入到计算机中后，用相应的解释程序将其逐条解释，边解释边执行，得到结果，而不保存解释后的机器代码，下次运行该程序时还要重

新解释执行。采用编译方式，程序的运行速度快，效率高。因此，目前常用的高级语言除 BASIC 语言采用解释方式外，大部分都采用编译方式。

1.1.2 程序设计的一般过程

对于初学者来说，往往把程序设计简单地理解为只是编写一个程序，这是不全面的。程序设计是指利用计算机解决问题的全过程，它包含多方面的内容，而编写程序只是其中的一部分。利用计算机解决实际问题，通常先要对问题的性质与要求进行深入分析并确定求解问题的数学模型或方法，然后考虑数据的组织方式和算法，并用某一种程序设计语言编写程序，最后调试程序，使之运行后能产生预期的结果，这个过程称为程序设计。程序设计的基本目标是实现算法和对初始数据进行处理，从而完成对问题的求解。

有些初学者，在没有把所要解决的问题分析清楚之前就急于编写程序，结果编程思路混乱，很难得到预期的效果。因此，为了用计算机解决一个实际问题，作为设计人员，从拿到任务到得出正确结果，往往要经过以下 6 个设计阶段。

① 分析问题。即分析问题需求，也就是弄清楚该问题有哪些已知数据，程序运行需要输入什么数据，需要输出什么结果，需要进行哪些处理，等等。

② 确定处理方案。如果是数学问题，就要根据该问题的数学解法，考虑所用到数学公式或相关函数；如果是工程问题，就要先建立该问题的数学模型，把工程问题转化成数学问题，以便用计算机解决。对同一个问题可以用不同的方案来处理，不同的方案决定了不同的处理步骤，效率也有所不同。

③ 确定操作步骤。根据选定的处理方案，具体列出让计算机如何进行操作的步骤。这种规定的操作步骤称为算法，而这些操作步骤之间的执行顺序就是控制结构。通常使用流程图来描述算法，把算法思想表达清楚，比较简单的问题可直接进入编写程序阶段。

④ 根据操作步骤编写源程序。用计算机语言编写的操作步骤就是计算机程序。

⑤ 运行调试程序。将计算机程序输入计算机中，经过编译、连接和运行，如果程序是正确的，应该能得到预期的结果。如果得不到正确的结果，应检查程序是否有错误，改正后再调试运行，直到得出正确的结果为止。

⑥ 整理输出结果，写出相关文档。

图 1.1 所示为程序设计的一般过程。

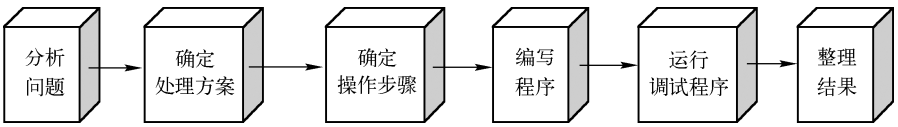

图 1.1 程序设计的一般过程

图 1.1 中，前两个步骤类似于人们解决问题的一般过程，即分析问题，然后确定一种处理方案。后 4 步则是程序设计的环节，其中最关键的是第③步“确定操作步骤”，或称算法设计。只要算法是正确的，编写程序就不会太困难。对于一个具体问题，编程者应该具备设计算法和正确使用已有算法的能力。

1.1.3 结构化程序设计方法

结构化程序设计是指为使程序具有一个合理的结构以保证程序正确性而规定的一套如何进行程序设计的原则。结构化程序设计的原则是：采用自顶向下、逐步求精的方法；程序结构模块化，每个模块只有一个入口和一个出口；使用 3 种基本控制结构描述程序流程。其中，模块化是结构化程序设计的重要原则。所谓模块化就是把一个大型的程序按照功能分解为若干相对独立的、较小的子程序（即

模块），并把这些模块按层次关系进行组织。按照结构化程序设计的原则，一个程序只能由顺序结构、选择结构和循环结构这 3 种基本结构组成。

人们解决复杂问题普遍采用自顶向下、逐步求精和模块化的方法，在这种设计方法的指导下开发出来的程序，具有清晰的层次结构，容易阅读和维护，软件开发的成功率和生产率可极大提高。因此，使用结构化方法设计出的程序等于数据结构加算法。

已经证明，任何复杂的算法都可以用顺序、选择、循环这 3 种结构组合而成。所以，这 3 种控制结构称为程序的 3 种基本控制结构。

1．顺序结构

顺序结构的 N-S 流程图如图 1.2 所示。其中 A 和 B 是顺序执行的关系，即先执行模块 A 操作，再执行模块 B 操作。

【例 1.1】　求两个整数 *m* 与 *n* 的和。

用自然语言描述的求解该问题的算法步骤如下：

步骤 1：输入整数 *m* 和 *n*。

步骤 2：求和 sum=*m*+*n*。

步骤 3：输出两数之和 sum。

图 1.3 所示为【例 1.1】的 N-S 流程图算法。

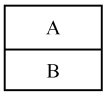

图 1.2　顺序结构

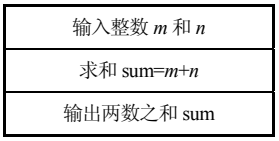

图 1.3　例 1.1 算法

2．选择结构

选择结构又称为分支结构，其 N-S 流程图如图 1.4 所示。其中，P 代表一个条件，当条件 P 成立时（或称为“真”时），执行模块A，否则执行模块 B。注意，只能执行 A 或B之一，两条路径汇合在一起结束该选择结构。

【例 1.2】　求 *a*、*b* 两个整数中较小的数。

用自然语言描述的求解该问题的算法步骤如下：

步骤 1：输入整数 *a* 和 *b*。

步骤 2：进行判断，如果 *a*<*b*，则 min=*a*，否则 min=*b*。

步骤 3：输出两个数中较小的数 min。

图 1.5 所示为【例 1.2】的 N-S 流程图算法。

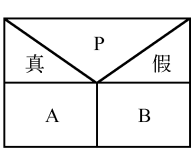

图 1.4　选择结构

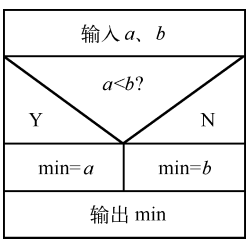

图 1.5　例 1.2 算法

3．循环结构

循环结构又称为重复结构，有两种循环形式。一种是当型循环结构，其 N-S 流程图如图 1.6 所示。其中，P 代表一个条件，当条件 P 成立（为“真”）时，反复执行模块 A 操作，直到 P 为“假”时才停止循环。另一种是直到型循环结构，如图 1.7 所示。先执行模块 A 操作，再判断条件 P 是否为“假”，若 P 为“假”，再执行 A，如此反复，直到 P 为“真”为止。

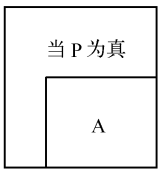

图 1.6　当型循环结构

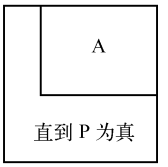

图 1.7　直到型循环结构

【例 1.3】　计算 1+2+3+4+…+100。

用自然语言描述的求解该问题的算法步骤如下。

步骤 1：定义变量 sum 用来存放和值，并将初值 0 赋给 sum，使 sum 的值为 0；定义变量 *k*，用来存放每项的值，并将 1 赋给 *k*。

步骤 2：判断 *k* 的值是否小于或等于 100，如果是，则继续执行步骤 3，否则转到步骤 5，退出循环。

步骤 3：将 sum 与 *k* 的和赋给 sum。

步骤 4：将 *k* 的值增 1，返回步骤 2 重复执行。

步骤 5：输出和值 sum。

图 1.8 所示为【例 1.3】的 N-S 流程图算法。

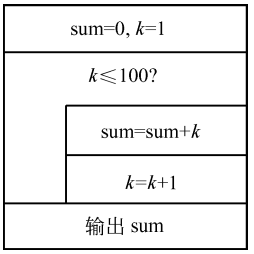

图 1.8　例 1.3 算法

可以看到，3 种基本控制结构共有的特点是：有一个入口，有一个出口；结构中每一部分都有被执行到的机会，也就是说，每一部分都有一条从入口到出口的路径通过它（至少通过一次）；没有死循环（无终止的循环）。

结构化程序要求每个基本控制结构具有单入口和单出口的性质是非常重要的，这是为了便于保证和验证程序的正确性。在设计程序时，一个结构一个结构顺序地写下来，整个程序结构如同砌墙一样顺序清楚，层次分明；在需要修改程序时，可以将某个基本控制结构单独取出来进行修改，由于其具有单入口单出口的性质，不会影响到其他的基本控制结构。可以把每个基本控制结构看作一个算法单位，整个算法则由若干个算法单位组合而成。这样的算法称为结构化算法。而这样设计出的程序清晰易读，可理解性好，容易设计，容易验证其正确性，也容易维护。同时，由于采用了“自顶向下、逐步细化”的实施方法，能有效地组织人们的思路，有利于软件的工程化开发，提高编程工作的效率，降低软件的开发成本。

1.2　C 语言的初步知识

1.2.1　C 语言的起源与特点

C 语言诞生于 1972 年，由美国电话电报公司（AT&T）贝尔实验室的 D.M.Ritchie 设计，并首先在一台使用 UNIX 操作系统的 DEC PDP-11 计算机中实现。

C 语言是在 B 语言的基础上发展起来的，早在 1970 年，美国贝尔实验室的 K.Thompson 就以 BCPL 语言（Basic Combined Promgramming Language）为基础，设计出一种既简单又接近于硬件的 B 语言，并用它编写了第一个 UNIX 操作系统。而 BCPL 语言是英国剑桥大学的 Matin Richards 于 1967 年基于 CPL 语言（Combined Programming Language）提出的一种改进语言。CPL 语言又是英国剑桥大学于 1963 年根据 ALGOL 60 推出的一种接近硬件的语言。由此可见，C 语言的根源可以追溯到 ALGOL 60，其演变过程如下：

ALGOL 60（1960 年）→CPL（1963 年）→BCPL（1967 年）→B（1970 年）→C（1972 年）

C 语言问世以后，在应用中多次进行了改进。1973 年贝尔实验室的 K.Thompson 和 D.M.Rithie 用 C 语言将 UNIX 系统（即 UNIX 第 5 版）重写了一遍，增加了多道程序设计功能，使整个系统，包括 C 语言的编译程序都建立在 C 语言的基础上。第 5 版 UNIX 系统（UNIX V5）奠定了 UNIX 系统的基础。到 1975 年，UNIX 第 6 版问世。随着 UNIX 的巨大成功和被广泛移植到各种机器上，C 语言也被人们所接受，并移植到大型、中型、小型和微型机上，它很快风靡全世界，成为世界上应用最广泛的计算机程序设计语言之一。

1978 年又推出了 UNIX 第 7 版，以该版本中的 C 语言编译程序为基础，B.W.Kernighan 和 D.M.Ritchie 合作（被称为 K&R）出版了“The C Programming Language”一书。该书介绍的 C 语言被称为标准 C，这本书成为后来被广泛使用的 C 语言的基础。1983 年，美国国家标准化协会（ANSI）对 C 语言的各种版本做了扩充和完善，推出了新的标准，被称为 ANSI C，它比原来的标准 C 有了很大的改进和发展。1987 年，ANSI 又公布了 87 ANSI C 新版本。当前流行的各种 C 语言编译系统都是以 87 ANSI C 为基础的。在微机上使用的 C 语言编译系统多为 Microsoft C、Turbo C、Borland C 和 Quick C 等，它们略有差异，但按标准 C 语言书写的程序，基本上都可运行。读者若要了解不同版本的编译系统的特点和规定可参阅有关手册。目前，全国计算机等级（二级 C 语言）考试中采用的 C 语言编译系统是 Visual C++ 6.0（简称为 VC）。Visual C++ 6.0 是一个集程序编辑、编译、连接和运行为一体的可视化集成开发环境，是计算机界公认的最优秀的应用开发工具之一。

C 语言之所以成为世界上应用最广泛、最受人们喜爱的计算机高级语言之一，与它本身所具有的突出的优点是分不开的。C 语言的主要特点概括如下。

① 语言简洁、紧凑，使用方便、灵活。C 语言一共只有 32 个关键字，9 种控制语句。程序书写形式自由，主要用小写字母表示。

② 支持结构化程序设计。C 语言具有结构化的控制语句，以函数作为程序的模块单位，这使得它可以很方便地实现这种构造。

③ 运算符丰富。C 语言除了拥有一般高级语言都有的运算符外，还拥有不少独特的运算符，可以实现其他高级语言不能实现的运算，增强了 C 语言的运算功能。

④ 数据类型丰富。C 语言能方便地实现各种复杂的数据结构，具有很强的数据处理能力。

⑤ 较强的编译预处理功能。C 语言的编译预处理功能，为开发规模较大的程序提供了方便，极大地提高了程序开发的效率。

⑥ C 语言的可移植性好。C 语言本身只需稍加修改便可用于各种型号的计算机和各类操作系统，因此，用 C 语言编写的程序也可以很方便地用于不同的系统，这也是 C 语言得以广泛应用的原因之一。

⑦ C 语言本身既有一般高级语言的优点，又有低级（汇编）语言的特点。C 语言可以允许直接访问内存地址，可以进行位（bit）运算，也可以直接对机器硬件进行操作。因此，既可以用它来编写大型系统软件，也可以用它编写各种应用软件，许多以前只能用汇编语言处理的问题，现在可以改用 C 语言处理了。

⑧ 语法限制不太严格，程序设计自由度大。C 语言由于放宽了语法检查，使程序员有较大的自由度。例如，对数组下标越界不做检查，要由程序设计人员自己保证其正确性。因此，用 C 语言编写程序，对程序设计人员的要求相应地要高一些。

上述 C 语言的特点，在初学时也许还不能深刻理解，边学边体会，待学完 C 语言之后就会有比较深的体会和感受。

1.2.2 初识 C 语言程序

下面通过两个简单的 C 语言程序，介绍 C 语言程序的一些基本构成和格式，并对程序中的语句做了比较详细的注释，使大家能够对 C 语言程序有一个最基本的认识。

【例 1.4】 求两个整数 *m* 与 *n* 的和。

程序中，*m* 和 *n* 分别表示两个整数，sum 表示两个整数的和。

程序代码如下：

```
#include  "stdio.h"                /* 文件包含命令，将头文件 stdio.h 包含进来 */
main( )                            /* 主函数 main( ) */
{  int  m,n,sum;                   /* 定义变量 m, n, sum*/
   m=5;                            /* 给变量 m 赋值 5 */
   n=3;                            /* 给变量 n 赋值 3 */
   sum=m+n;                        /* 求 m+n 的值,并赋给变量 sum */
   printf("sum is %d \n",sum );    /* 输出 sum 的值 */
}
```

程序的运行结果如下：

```
sum is 8
```

【例 1.5】 求两个整数中的较小者。

程序中，*x*、*y* 分别表示两个整数，min 表示两个整数之中的较小值。

程序代码如下：

```
#include "stdio.h"
main( )                          /* 主函数 main( ) */
{  int  x,y,min;                 /* 定义变量 x,y,min */
   int  fun(int a,int b);        /* 当定义的函数在主函数之后时,要进行函数的声明*/
   printf("input x,y:");         /* 提示输入数据 */
   scanf("%d,%d", &x,&y);        /* 通过键盘给变量 x 和 y 赋值 */
   min=fun(x,y);                 /* 调用 fun( )函数,将 fun( )函数的返回值赋给 min */
   printf("min=%d\n",min);       /* 输出 min 的值 */
}
int fun(int a,int b)             /* 定义 fun 函数,值为整型,a 和 b 为该函数的形式参数*/
{  int c;                        /* 函数中用到的变量 c 也要定义 */
  if(a<b)  c=a;                  /* 比较 a 和 b 的大小，将较小者赋给 c */
  else  c=b;
  return(c);                     /* 将 c 的值返回至调用处 */
}
```

程序的运行结果如下：

```
input x,y:10,2<Enter>
min=2
```

程序运行时，屏幕显示： input x,y:，然后从键盘输入 10，2，再敲击回车。

说明：<Enter>表示单击键盘上的 Enter 键。

1.2.3　C 语言字符集与标识符

从【例 1.4】和【例 1.5】可以看出，一个 C 语言程序由一些字符、字符组合以及一些有意义的符号按照一定的规则组成。其实任何一种程序设计语言都有自己的语法、句法规则，C 语言有自己的特定字符集和标识符，下面简要介绍有关内容。

1．C 语言字符集

字符（Character）是组成程序设计语言最基本的元素。C 语言的字符集由字母、数字、空白符、标点和特殊字符组成。

① 字母：26 个英文字母，包括大小写。

② 数字：0～9 共 10 个。

③ 空白符：空格符、制表符、换行符统称为空白符，共 3 个。

④ 标点和特殊字符：如+ – * / % _ . = < > & | ( ) [ ] { } ; ? : ' " ! # 等共 25 个。

2．标识符

标识符（Identifier）是一个由有限个有效字符组成的序列，在 C 语言中只起标识作用，可用作符号常量名、变量名、函数名、数组名、文件名等。

（1）标识符的构成规则

C 语言允许用作标识符的有效字符包括：

- 26 个英文字母，包括大小写
- 数字 0，1，…，9
- 下画线

合法的标识符必须由字母（A～Z，a～z）或下画线（_）开头，后面可以跟随任意的字母、数字或下画线。C 语言标识符的长度（即一个标识符允许包含的字符个数）受 C 语言编译系统的限制，例如，某 C 编译系统规定标识符的有效长度是 31，若超过 31 个字符，则后面的字符无效。不同的 C 语言编译系统规定的标识符的长度可能会不同，学习者在使用标识符时应当了解所用编译系统的规定。

合法的标识符：student，a10，sf，_5n，x_sum。

不合法的标识符：30d　　错在以数字开头

a$n　　错在出现“$”

n abc　　错在中间有空格

（2）C 语言标识符的分类

标识符是形成 C 语言代码的基础。C 语言中的标识符由 3 种类型组成：关键字、预定义标识符和用户标识符，每种标识符都有自己的要求。

① 关键字

C 语言中有一些标识符被称为关键字或保留字，在系统中具有特殊用途，只能以特定的方式用在特定的地方，如果试图将关键字用于其他用途，编译程序将产生一个编译错误。例如，标识符 int 是整型数据类型关键字。

表 1.1 列出了 C 语言完整的关键字列表，随着教材内容的深入，读者将理解在什么地方、为什么和如何使用这些关键字。

表 1.1 C 语言关键字

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

② 预定义标识符

C 语言中有些标识符虽然不是关键字，但总是以固定的形式用于专门的地方，使用较多的预定义标识符是 C 语言标准函数（参见附录 B）。例如，printf 是 C 语言提供的标准函数名，define 是 C 语言提供的编译预处理命令等。因此，用户也不要把它们当作一般标识符使用，以免造成混乱。

③ 用户标识符

用户标识符是由用户根据需要定义的标识符。一般用于给变量、符号常量、数组、函数、指针、文件等命名。在程序中使用用户标识符时除了要遵守标识符的构成规则外，还应注意以下的问题。

- 大小写字母有不同的含义，例如，sum、Sum 和 SUM 是 3 个不同的标识符。习惯上，变量名用小写字母表示，符号常量名用大写字母表示。
- 在构造用户标识符时，应注意做到“见名知意”，即选用有含义的字符组合（如英文单词或汉语拼音）作为标识符，以增加程序的可读性。例如，表示较小的数可用 min，表示长度可用 length，表示和可用 sum 等。

1.2.4 C 语言程序的基本构成

从【例 1.4】和【例 1.5】还可以看出一个 C 语言程序的基本架构，它由多个彼此独立的被称为函数的模块构成。

（1）C 语言程序由函数构成

一个 C 语言程序至少应包含一个 main( )函数（如例 1.4），或者包含一个 main( )函数和若干个用户自定义函数（如例 1.5）。因此，函数是 C 语言程序的基本单位，相当于其他语言的子程序或过程。

C 语言的函数可以分为系统提供的标准函数（库函数，如例 1.5 中的 printf( )和 scanf( )函数）和用户自定义函数（如例 1.5 中的 fun( )函数）。各种 C 语言的编译系统所提供的标准函数的数量和功能以及函数名都不完全相同，标准 C 提供了 100 多个标准函数。通过定义函数以及函数调用使 C 语言很容易实现程序的模块化。

（2）一个 C 语言函数通常由两部分组成：函数的首部和函数体

函数的首部包括函数类型、函数名、一对圆括号、函数参数（形参）名和参数类型的说明。比如，例 1.5 中 fun( )函数的首部如图 1.9 所示。

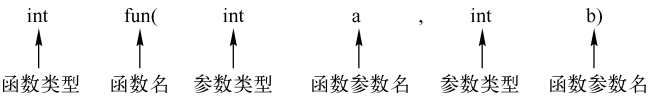

图 1.9 例 1.5 中 fun( )函数的首部

注意：C 语言规定，一个函数名后面必须紧跟一对圆括号，函数的所有形参都必须写在括号内。如果函数没有形参，则括号内可以是空的，但不能省略圆括号，例如，主函数 main( )经常不带参数。

函数体就是函数首部后面一对大括号{}内的部分。如果一个函数内有多对大括号，则最外面的一对大括号{}所包含的内容就是该函数的函数体。函数体一般包括说明部分和执行部分。

① 说明部分

说明部分由若干条变量定义语句或函数声明语句组成。C 语言规定，函数中使用的所有变量（或数组）必须在使用前进行定义，否则会在编译时出错。如例 1.5 中 main( )函数的变量定义语句“int x,y,min;”。当然也可以没有说明部分。例如下面的程序代码：

```
main( )
{  printf("It is fine today!" );}
```

该程序的作用是在屏幕上显示以下文字：

```
It is fine today!
```

这里没有用到任何变量，所以不需要变量的定义。

② 执行部分

执行部分由若干条执行语句组成。程序的功能就是通过执行部分实现的。C 语言的任何语句都必须以分号“;”结束，分号是 C 语句的必要组成部分。例如：

```
y=x+b;
```

C 语言的语句可以从任意位置开始书写，书写格式自由，一行内可以写多条语句，语句中多个空格与一个空格等效。

一个函数甚至可以既无说明部分，也无执行部分，例如：

```
comp( )
{}
```

这个函数是一个空函数，什么作用也没有，但却是合法的。

（3）main( )函数

一个 C 语言的程序只能有一个 main( )函数，C 程序总是从 main( )函数开始执行，并终止于 main( )函数，而不论 main( )函数在整个程序中处于什么样的位置（main( )函数可以放在程序的开头、最后，或某两个函数之间）。但是，为便于理解，通常将 main( )函数放在程序的开头或最后。

（4）为了增加程序的可读性，可以用“/*……*/”在任何位置上对 C 语言程序的任何部分进行注释，一般在一个程序或函数的开始或某些程序的难点之处加上必要的注释。在 Visual C++ 6.0 环境下也可使用符号“//……”引出注释。

1.2.5　简单的屏幕输出

一个程序必须要有数据信息的输出部分，以便告诉编程者或使用者程序的运行结果是什么。C 语言没有专门的输出语句，printf( )是 C 语言的标准输出函数，可实现在屏幕上输出一个字符串，或者按指定格式输出若干变量的值，这里给出 printf( )的基本用法，详细内容详见第 3 章。

1．输出一个字符串

例如：printf(“Enter number：”);

运行后将双引号中的所有内容原样输出，结果为：Enter number：

【例 1.6】　用“*”号输出字母 C 的图案。

程序分析：可先用“*”号在纸上写出字母 C，再分行输出。

程序代码如下：

```
#include "stdio.h"
main( )
```

```
{
  printf(" ****\n");
  printf(" *\n");
  printf(" * \n");
  printf(" ****\n");
}
```

说明：\n 是转义字符，作用是换行，也就是说前面字符输出完之后将光标移动到下一行行首。

2．按指定格式输出若干变量的值

例如：printf("sum is %d \n",sum); /* 假设变量 sum 的值是 8 */

其中，双引号中的%d 是格式控制符，表示在此位置以十进制整数类型输出变量 sum 的值，双引号中其他内容将作为字符串原样显示在用户屏幕上，所有信息输出后换行。因此，该语句运行后的结果如下：

```
sum is 8
```

【例 1.7】 分析程序的运行结果，理解 printf( )函数的屏幕输出功能。

```
#include "stdio.h"
main( )
{
    int a,b;
    a=10;
    b=8;
    printf("%d,",a);     /* 将 a 的值输出后，再输出逗号，注意没有换行 */
    printf("%d\n",b);
    printf("a=%d\n",a);
    printf("b=%d\n",b);
    printf("a=%d,b=%d\n",a, b);
}
```

程序的运行结果如下：

```
10,8
a=10
b=8
a=10,b=8
```

1.2.6 C 语言程序的上机调试过程

如何调试一个 C 语言程序呢？在不同的环境下调试的方法稍有差异，但归纳起来一般是依照图 1.10 所示的过程调试的。

从图 1.10 看出，C 语言程序的调试过程基本上可以分为以下 4 步。

1．编辑

编辑就是用 C 语言写出源程序。其方法有两种：一种是使用文本编辑程序将源程序输入计算机，经修改认为无误后，以.c 为后缀（C 源程序的后缀一般定为“.c”）存入文件系统中；另一种是使用 C 语言编译系统提供的编辑器将源程序输入计算机，并且存入文件系统中。例如在图 1.10 用户将源程序文件定名为 file.c。

2．编译

调用 C 语言编译程序对源文件进行编译，即检查其词法、语法、语义方面是否存在错误。

通俗点说，就是检查是否有拼错的关键词，是否有不符合 C 语言语法规则的表达式及语义方面的矛盾和含混。这些错误在编译阶段都可以查出。

如果有错误，则系统将显示“错误信息”。用户根据指出的错误信息，对源程序进行编辑修改，修改后再重新进行编译，直到编译无误为止。编译后生成的机器指令程序，被称为目标程序，此目标程序名与相应的源程序同名，但其后缀为.obj 。上述源程序文件 file.c 经编译后得到目标程序 file.obj。

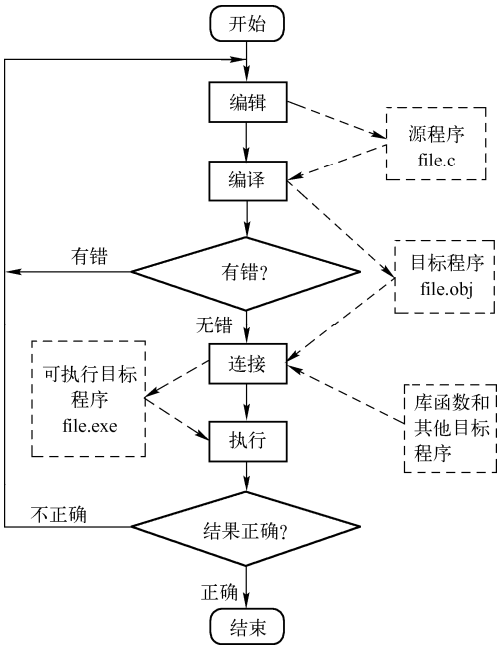

图 1.10 程序调试过程

3．连接

编译产生的目标文件一般不能直接运行，必须把它所引用的标准函数及源程序指定的目标文件一起装配连接起来，形成完整的可执行文件。一般可执行文件名与源程序文件名同名，后缀为.exe。例如，上述源程序文件 file.c 经编译连接后得到可执行文件 file.exe。

4．执行程序

当程序编译连接后，生成可执行程序便可以运行了，以后用户只需输入可执行目标文件名即可，例如：

```
file<Enter>
```

本 章 小 结

1．程序是人与计算机进行“交流”的工具，用“程序设计语言”来描述，程序设计语言是计算机能够理解和识别的语言。

2．程序设计是指利用计算机解决问题的全过程。程序设计的基本目标是实现算法和对初始数据进行处理，从而完成对问题的求解。

3．C 语言是伴随 UNIX 操作系统产生和发展起来的，它既具有高级语言的优点，又具有汇编语言的特点，有人称它是“高级汇编语言”。C 语言的语法紧凑、简洁，数据类型丰富，运算功能强大，高效率的代码和高度的可移植性使它成为编写大型工具软件和硬件控制程序不可替代的一门高级语言。目前，C 语言已经广泛应用于各个领域中。

4．C 语言程序由一个或多个函数组成，这些函数中必须包含一个名为 main( )的主函数，整个程序由它开始执行，也以该函数的结束而结束整个程序。

5．printf( )是最常用的 C 语言的标准输出函数，可实现在屏幕上输出任何信息。

6．C 语言程序的调试基本上可以分为 4 步：即编辑、编译、连接和运行。

习 题 1

1.1 思考题

1．什么是结构化程序设计？其基本结构有哪几种？

2．输入三个值 a、b、c，用它们作为三角形的 3 条边输出三角形的面积，画出实现该算法的 N-S 图。

3．标识符的含义是什么？C 语言的标识符构成规则是什么？有几种类型的标识符？请举例说明。

4．C 语言程序的基本单位什么？

5．一个 C 语言函数由哪两部分组成？

6．在 C 语言的源程序中如何引出注释内容？

7．C 语言程序的开发过程是什么？每个阶段生成文件有怎样的性质？

1.2 编程题

1．试参照本章例题编写计算梯形面积的 C 语言程序，梯形的上底、下底和高分别用 *a*、*b*、*h* 表示，并用 *a*=10，*b*=20，*h*=5 测试所编写的程序。

2．编写程序显示如图 1.11 所示的信息。

```
**************************
*        Hello World     *
**************************
```

图 1.11 显示信息