第 1 章　程序设计和 C 语言

1.1　计算机程序和计算机语言

许多人觉得计算机结构复杂，神秘莫测。其实计算机并不神秘，它的工作是由程序控制的。想要让计算机按照人们的愿望工作，人们必须事先编好一个程序(program)，把它输入计算机中，然后执行程序，计算机才能产生相应的操作。

人和计算机怎么沟通呢？计算机并不懂得人类的语言，它只能识别二进制的信息。在计算机产生的初期，人们为了让计算机工作，必须编写出由 0 和 1 所组成的一系列的指令，通过它指挥计算机工作。在研制一种计算机时，要事先设计好该型号计算机的指令系统，规定好一条由若干位 0 和 1 组成的指令使计算机产生哪种操作。例如，在某型号计算机中用“00000100 00000001”让计算机进行一次加法运算。

要使计算机执行一系列操作，就需要编写许多条类似这样的由 0 和 1 组成的指令，这是非常烦琐和费时的。这种计算机可以直接识别和接收的二进制代码称为机器指令(machine instruction)。机器指令的集合及其规则就是该计算机的机器语言(machine language)。在语言的规则中规定各种指令的表示形式以及它的含义。

显然，机器语言与人们习惯用的语言差别太大，难学、难写、难记、难检查、难修改、难推广使用，因此，初期只有极少数的计算机专业人员会编写计算机程序。

为了克服机器语言的上述缺点，研究人员创造出一种符号语言(symbol language)，用一些英文字母和数字表示一个指令。例如，用 ADD 代表“加”，SUB 代表“减”，LD 代表“传送”等。一般用一条符号指令对应转换为一条机器指令，如上面介绍的那条 0 和 1 组成的机器指令可以用下面一条符号指令代替：

```
ADD A, B      (执行 A+B → A,将寄存器 A 中的数与寄存器 B 中的数相加,送到寄存器 A 中)
```

但是，计算机还是不能直接识别和执行符号语言的指令，需要用一种称为汇编程序的软件把符号语言的指令转换为机器指令，因此，符号语言也称为汇编语言(assembler language)。虽然汇编语言比机器语言简单、好记一些，但仍然难以普及，只在专业人员中使用。

由于机器语言和汇编语言都依赖于具体机器，即在底层进行控制，所以被称为**低级语言**(low level language，意思是贴近计算机硬件的语言，而不是指“水平低级”)。用低级语言编写程序很不直观，烦琐枯燥，工作量大，无通用性。不同型号计算机所用的机器语言是不通用的。因此，要使用不同型号的计算机，就要学习不同的机器语言，这种情况使计算机难以

在非专业人员中推广应用。

为了克服低级语言的缺点，1954 年创造出了 FORTRAN 语言。它很接近人们习惯使用的自然语言和数学语言。程序中用到的语句和指令是用英文单词表示的，程序中所用的运算符和运算表达式与人们日常所用的数学式差不多，很容易理解。程序运行的结果用英文和数字输出，十分方便。例如，在 FORTRAN 语言程序中，想计算和输出 $3.5\times 6\sin(\pi/3)$，只需写出下面这样一个语句：

```
PRINT *, 3.5 * 6 * SIN(3.1415926/3)
```

即可得到计算结果。显然，这是很容易理解和使用的。这种语言功能很强，且不依赖于具体机器，用它写出的程序对任何型号的计算机都适用（或只做很少的修改），它与具体机器距离较远，故称为计算机**高级语言**（high level language）。FORTRAN 是第一个通用的高级语言。显然，用高级语言编写程序直观、易学、易理解、易修改、易维护、易推广、通用性强（不同型号计算机之间可以通用）。

程序是为实现特定目标而用计算机语言编写的命令序列的集合，它包括为实现预期目的而进行操作的一系列语句和指令。

当然，计算机也是不能直接识别高级语言程序的，必须事先把用高级语言编写的程序翻译成机器语言程序，计算机才能识别并执行，这个“翻译”工作不是由人工完成的，而是由一个称为编译程序（或称编译器，compiler）的软件来实现的，编译程序把用计算机高级语言写的程序（称为源程序，source program）转换为机器指令的程序（称为目标程序，object program），然后让计算机执行机器指令程序，最后得到结果。高级语言的一个语句往往对应多条机器指令。

数十年来，全世界涌现了几千种不同用途的高级语言，其中应用比较广泛的有 100 多种，影响较大、使用较广的有：FORTRAN 和 ALGOL（适合数值计算）、BASIC 和 QBASIC（适合初学者的小型会话语言）、COBOL（适合商业管理）、Pascal（适合教学的结构程序设计语言）、LISP 和 PROLOG（人工智能语言）、Visual Basic（支持面向对象程序设计的语言）、C（系统描述语言）、C++（支持面向对象程序设计的大型语言）、Java（适于网络使用的语言），以及近来流行的 Python 等。

自从有了高级语言，一般的科技人员、管理人员、大中学生以及广大计算机爱好者都能较容易地学会用高级语言编写程序，指挥计算机进行工作，而完全不用考虑机器指令；人们也可以不必深入懂得计算机的内部结构和工作原理，就能得心应手地利用计算机进行各种工作，为计算机的推广普及创造了良好的条件。有一些专家将高级语言的出现称作是计算机发展史上“惊人的成就”。

1.2 C 语言的发展过程

1972 年，美国贝尔实验室的 D.M.Ritchie 设计出了 C 语言。最初的 C 语言只是为描述和实现 UNIX 操作系统提供一种工作语言而设计的。C 语言问世后很受欢迎。1978 年以后，C 语言先后移植到大、中、小、微型计算机上，并很快风靡全世界，成为世界上应用最广泛

的一种程序设计高级语言。

C 语言问世以来经过多次发展和扩充。1989 年，美国国家标准化协会（简称 ANSI）公布了一个新的 C 语言标准——ANSI X3.159-1989（简称 C89）。1990 年，国际标准化组织（International Standard Organization，ISO）接受 C89 作为国际标准 ISO/IEC 9899：1990，通常简称 C90。ISO 的 C90 和 ANSI 的 C89 基本上是相同的。1999 年，ISO 又对 C 语言标准进行修订，命名为 ISO/IEC 9899：1999，简称 C99，C99 是 C89 的扩充。但目前有的软件公司提供的 C 语言编译系统并未完全实现 C99 建议的功能，而是以 C89 为基础进行开发的。不同的软件公司提供的 C 语言编译系统所实现的语言功能和语法规则又略有差别。本书的叙述基本上是以 C99 为基础的。

自 20 世纪 90 年代初 C 语言在我国开始推广以来，学习和使用 C 语言的人越来越多，成了学习和使用人数最多的一种计算机语言。C 语言的功能强大、使用灵活，既可用于编写应用软件，又能用于编写系统软件；它不仅有广泛应用的优势，又有较为系统的基础性质。学习了 C 语言以后，再学习和使用任何其他语言都不会发生困难。我国多数理工科大学和高职高专院校都开设了 C 语言程序设计课程。C 语言是计算机专业人员的一项基本功。

1.3　从最简单的 C 语言程序开始

下面先观察几个简单的 C 语言程序，然后从中分析 C 语言程序的特点。

【例 1.1】

任务要求：

在屏幕上显示出以下一行信息。

```
This is a C program.
```

解题思路：

用 C 语言提供的输出函数 printf()来输出所需的内容。

编写程序：

```
#include <stdio.h>
int main()
{
    printf ("This is a C program.\n");
    return 0;
}
```

运行结果：

```
This is a C program.
```

程序分析：

这是一个最简单的 C 语言程序。程序第 2～6 行是 C 语言程序中的主函数。C 语言要求：每一个函数要有**函数名**，也要有**函数体**（即函数的实体）。函数体由一对花括号“{}”括起来。

程序第 2 行指定了函数的名称和函数的类型。main 表示此函数为“**主函数**”，每一个 C 语言程序都必须有一个 main()函数。main 前面的 int 表示此主函数的值是整型的(int 是 integer 的缩写)，表示计算机在执行完此函数后，会使该函数获得一个整数的值，称为函数返回值。

注意：执行一个函数后会得到一个函数值，例如，正弦函数 sin(x)有一个确定的值。函数的值是提供给函数的调用者的。main()函数是由操作系统调用的，执行完 main()函数也有一个值，返回给操作系统，操作系统依此判定 main()函数是否正常结束。那么 main()函数的值是什么呢？程序最后一行“return 0;”表示“返回 0”，即把 0 作为函数的值。如果程序没有正常结束，就不会执行此 return 语句，不返回 0，系统会使 main()函数的值为一个非 0 值(一般为 1)。操作系统就可以知道程序未正常结束，并采取相应的措施(如输出一个信息)。

本例中主函数内只有两个语句。其中“return 0;”是把 0 作为函数的返回值，这个 return 语句是每一个 C 程序都需要的。因此，本程序实际上只有第 4 行 printf 语句是用户用来实现所需功能的。printf 是 C 编译系统提供的标准函数库中的输出函数(详见第 2 章)。printf 语句中圆括号内双撇号内的字符串按原样输出。“\n”是换行符，在执行程序时，输出“This is a C program.”，然后执行回车换行。

注意：所有语句最后都应有一个分号(;)。

在使用标准函数库中的输入/输出函数时，编译系统要求程序提供有关输入/输出函数的信息(例如对这些输入/输出函数的声明)，程序第 1 行“#include <stdio.h>”的作用就是用来提供这些信息的。“stdio.h”是 C 编译系统提供的一个文件名，stdio 是“standard input & output”的缩写，即有关“标准输入/输出”的信息。在开始时对此可不必深究，以后会有详细的介绍。在此只需记住：在程序中用到系统提供的标准函数库中的输入/输出函数时，应在程序的开头写这样一行：

```
#include <stdio.h>
```

【例 1.2】

任务要求：

求两个整数之和。

解题思路：

声明两个变量 a 和 b，用 C 语言提供的赋值语句进行 a+b 的运算，结果保存在变量 sum 中，然后用 printf()函数输出结果。

编写程序：

```
#include <stdio.h>
int main()                          //主函数
{
  int a,b,sum;                      //这是声明部分，定义 a、b、sum 为整型变量
  a=123;                            //以下 5 行是 C 语言的语句
  b=456;
  sum=a+b;                          //将 a 和 b 相加，得到的和送到变量 sum 中保存
  printf("sum is %d\n",sum);        //输出 sum 的值
  return 0;
}
```

运行结果：

```
sum is 579
```

程序分析：

本程序的作用是求两个整数 a 和 b 之和 sum。各行右侧的“//……”表示这是**注释部分**。注释可以用英文；如果使用的是汉字 C 语言系统，则在注释中可以用汉字。注释主要是给其他人看的，对编译和运行程序不起作用，也就是说，注释部分不影响程序运行的结果。注释可以出现在一行中的最右侧；也可以单独成为一行，根据需要写在程序中任何一行的右侧。如果注释内容多，一行容纳不下，可以连续用几个注释行（或用/*……*/），例如：

```
//如果一行写不下，
//可以在下一行接着写
```

第 4 行是声明部分，用来定义变量 a、b 和 sum 为整型变量，int 代表“整型”。

第 5、6 行是两个赋值语句，使 a 和 b 的值分别为 123 和 456。

第 7 行执行 a+b 的运算，然后把运算的结果赋予变量 sum。

第 8 行是输出语句，双撇号中的“%d”是输入/输出的**“格式字符串”**，用来指定输入/输出时的数据类型和格式（详见第 2 章）。“%d”表示输入/输出时用“十进制整数”形式表示。在执行输出时，双撇号中的字符“sum is”按原样输出，而在格式字符串“%d”的位置上代入一个十进制整数值。printf()函数中括号内的双撇号和逗号右面的 sum 是要输出的变量，现在 sum 的值为 579（123 与 456 之和），在输出结果时它应代替“%d”，出现在“%d”原来的位置上，如图 1.1 所示。“\n”是换行符，实现回车换行。

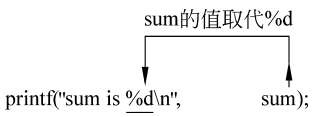

图　1.1

因此，程序运行时输出“sum is 579”。

【例 1.3】

任务要求：

输入两个数，要求输出其中的较大者。

解题思路：

（1）从键盘输入两个数。

（2）用一个函数来实现求两个整数中的较大者。

（3）在主函数中调用此函数并输出结果。

编写程序：

```
#include <stdio.h>
int main()                          //主函数
{
  int max(int x,int y);             //对被调用的 max()函数进行声明
  int a, b, c;                      //定义整型变量 a、b、c
  scanf("%d,%d",&a,&b);             //从键盘输入变量 a 和 b 的值
  c=max(a,b);                       //调用 max()函数，将得到的值赋给整型变量 c
  printf("max=%d\n",c);             //输出变量 c 的值
  return 0;
}
```

```
//下面是求两个整数中的较大数的函数
int max(int x,int y)          //定义 max()函数,函数值为整型,形式参数 x 和 y 为整型
{
  int z;                      //max()函数中的声明部分,定义本函数中用到的变量 z 为整型
  if (x>y) z=x;               //如果 x>y,则将变量 x 的值赋给变量 z
  else z=y;                   //否则,将变量 y 的值赋给变量 z
  return(z);                  //将变量 z 的值返回到主函数中调用函数的位置
}
```

运行结果：

```
8,5↙                          (输入 8 和 5,赋给 a 和 b)
max=8                         (输出两个数中的较大者,即 c 的值)
```

说明：

为了使读者分析运行情况时能区别输入和输出的信息，本书对向程序输入的信息加了下画线(在屏幕上并无此下画线)。如上面运行结果中的第 1 行表示：从键盘输入“8,5”，然后按 Enter 键。第 2 行是从计算机输出的信息，显示在屏幕上。

程序分析：

本程序包括两个函数——主函数 main()和被调用的函数 max()。max()函数的作用是将 x 和 y 中的较大数的值赋给变量 z。return 语句将 z 的值作为 max()函数值带回给主函数 main()。返回值是通过函数名 max 带回到 main()函数中调用 max()函数的位置。

程序第 4 行是在主函数中对被调用函数 max()的声明。由于在主函数中要调用 max()函数，而 max()函数的定义却在 main()函数之后，为了使编译系统能够正确识别和调用 max()函数，必须在调用 max()函数之前对 max()函数进行声明，以通知编译系统：“在 main()函数中，max()是一个已定义的函数”。对函数的声明会在第 6 章详细介绍，在此只要初步了解即可。

main()函数中的 scanf 是“输入函数”的名字(scanf()和 printf()都是 C 语言的标准输入/输出函数)。程序中 scanf()函数的作用是：在程序运行时要求用户输入 a 和 b 的值。&a 和 &b 中“&”的含义是数据的地址符，&a 表示“变量 a 的地址”，&b 表示“变量 b 的地址”。本例中 scanf()函数的作用是：将从键盘输入的两个数值分别送到变量 a 和 b 的地址所标识的单元中，也就是把 8 和 5 分别输入给变量 a 和 b。scanf()函数中双撇号括起来的“%d,%d”的含义与前相同，只是现在用于“输入”，它指定用户应当按十进制整数形式输入 a 和 b 的值。

在程序第 7 行中调用 max()函数，在调用时将**实际参数** a 和 b 的值分别传送给 max()函数中的参数 x 和 y(称为**形式参数**)。经过执行 max()函数得到 x 和 y 二者中的较大者，存放在 z 中。程序把 z 作为函数的返回值，通过 return 语句把这个值返回到调用 max()函数的位置，即第 7 行“=”的右侧，代替了原来的 max(a,b)，然后把这个值赋给变量 c。

程序第 8 行输出变量 c 的值。

在执行 printf()函数时，对双撇号括起来的"max=%d\n"是这样处理的：①将字符串“max =”原样输出；②“%d”由 c 的值取代之；③“\n”是回车换行。请对照运行结果分析。

本例用到了函数调用、实际参数和形式参数等概念，在此只作了很简单的解释，读者如对此不太理解，可以先不深究，在学到第 6 章(函数)时自然会迎刃而解。现在介绍此例子，无非是使读者对 C 语言程序的组成和形式有一个初步的了解。

1.4　C 语言程序的结构

通过以上几个例子，可以看到 C 语言程序的结构和特点如下。

(1) C 语言程序主要是由函数构成的。一个 C 语言源程序必须包含一个 main()函数，也可以包含一个 main()函数和若干个其他函数，因此，函数是 C 语言程序的基本单位。被调用的函数可以是系统提供的库函数(如 printf()和 scanf()函数)，也可以是用户根据需要自己编制设计的函数(如例 1.3 中的 max()函数)。C 语言的函数相当于其他语言中的子程序，用来实现特定的功能。程序全部工作都是由各个函数分别完成的，编写 C 语言程序就是编写一个个的函数。C 语言的函数库十分丰富，ANSI C 建议包括 100 多个库函数，不同的 C 语言编译系统提供的库函数一般都多于 ANSI C 所建议的数量，如 Turbo C 提供 300 多个库函数。C 语言的这种特点使得程序容易实现模块化。

(2) 一个函数由两部分组成。

① 函数首部。即函数的第 1 行，包括函数名、函数类型、函数参数(形式参数)名和参数类型。

例如，例 1.3 中的 max()函数的首部为

```
   int      max       (int           x,          int          y )
    ↓        ↓          ↓            ↓           ↓           ↓
函数类型   函数名   函数参数类型   函数参数名   函数参数类型   函数参数名
```

一个函数名后面必须跟一对圆括号，括号内写函数的参数名及其类型。函数可以没有参数，例如：

```
int main( )
```

② 函数体。即函数首部下面的花括号内的部分。如果一个函数内有多个花括号，则最外层的一对花括号为函数体的范围。

函数体一般包括以下两部分。

a. 声明部分。这一部分中包括对有关的变量和函数进行声明(declare)，将有关的信息告诉编译系统。例如，例 1.2 程序中第 4 行“int a，b，sum；”的作用是告诉编译系统：“本函数中用到的变量 a、b、sum 是整型变量”。这样，编译系统就会对这些变量按整型数据进行存储。例 1.3 程序 main()函数中的“int a，b，c”的作用类似。以上都是对变量的声明。

例 1.3 程序第 4 行 “int max(int x，int y)；”是对 max()函数的声明。

声明部分可以包括若干个声明行，它们不是 C 语句，在程序运行期间不产生任何操作，而只在程序编译时起作用，影响数据存储，而不会生成目标代码。

b. 执行部分。该部分由若干个语句组成。C 语句是可执行语句，经编译生成目标代码，在程序运行期间执行相应的操作。

当然，在某些情况下也可以没有声明部分(如例 1.1)，甚至可以既无声明部分也无执行部分，例如：

```
void dump ( )
```

```
{ }
```

这是一个空函数，什么也不做，但这是合法的。

（3）一个 C 程序总是从 main()函数开始执行的，而不论 main()函数在整个程序中的位置如何（main()函数可以放在程序最前面，也可以放在程序最后；可以放在一些函数之前，也可以在另一些函数之后）。

（4）C 程序书写格式自由，一行内可以写几个语句，一个语句也可以分写在多行上，程序的各行没有行号（有的语言要求在每一行的开头要有行号，以便识别）。

（5）每个语句和数据声明的最后必须有一个分号，分号是 C 语句的必要组成部分。例如：

```
c=a+b;
```

分号是不可缺少的，即使是程序中最后一个语句也应包含分号，见以上各例。

（6）C 语言本身没有输入/输出语句。输入/输出的操作是由库函数 scanf()和 printf()等函数来完成的。C 语言对输入/输出实行“函数化”的方式。由于输入/输出操作牵涉具体的计算机设备，把输入/输出操作放在函数中处理，就可以使 C 语言本身的规模较小，编译程序简单，很容易在各种机器上实现，程序具有可移植性。

（7）可以用“//”对 C 程序中的任何一行或数行作注释。一个好的、有使用价值的源程序都应当加上必要的注释，以增加程序的可读性。

1.5 运行 C 语言程序的步骤与方法

在本章 1.3 节中看到的程序是用 C 语言写的**源程序**。前已说明，计算机是不能直接识别和执行用高级语言编写的指令的。为了使计算机能执行高级语言源程序，必须先用一种称为“编译程序”的软件把源程序翻译成二进制形式的**目标程序**（object program），然后将该目标程序与系统的函数库以及其他目标程序连接起来，形成**可执行的目标程序**。

在编好一个 C 语言源程序后，怎样上机运行呢？一般要经过以下几个步骤。

1. 上机输入和编辑源程序

先进入 C 语言编译系统（一般是集成环境 IDE，如 Visual C++ 6.0，简称 VC++ 6.0）。建立一个文件，文件名自己指定，后缀为“c”（如 test.c 或 f.c）。通过键盘向此文件输入程序，并且认真检查有无错误，如发现有错误，要及时改正。这一工作称为“对源程序的编辑”。完成编辑后，将此源程序存放在自己指定的文件夹内（如果自己不专门指定，系统一般会自动把它存放在用户当前的目录下）。

2. 对源程序进行编译

用户进行编译操作后，C 语言编译系统先调出“预处理器”（又称“预处理程序”或“预编译器”），对程序中的预处理指令进行编译预处理。例如，对 #include <stdio.h> 指令进行“预处理”就是将 stdio.h 头文件的内容读进来，取代程序中的 #include <stdio.h> 行。由预处理得到的信息与程序其他部分一起，组成一个完整的、可以用来进行正式编译的源程序，再由编译系统对该源程序进行编译。

编译的作用首先是对源程序进行检查，判定它有无语法方面的错误，如有错误，则发出“出错信息”，告诉编程人员认真检查改正。在修改程序后重新进行编译，如还有错，再发出“出错信息”。如此反复进行，直到没有语法错误为止。此时，编译程序把源程序转换为二进制形式的目标程序（在 VC++ 中后缀为.obj，如 test.obj 或 f.obj）。如果不特别指定，此目标程序一般也存放在用户当前的目录下。此时源文件仍然存在，不会自动消失。

在用编译系统对源程序进行编译时，包括了如上的预编译和正式编译两个阶段，自动一气呵成。用户不必分别发出二次指令。

3. 进行连接处理

经过编译所得到的二进制目标文件（后缀为.obj）还不能供计算机直接执行。前已说明：一个程序可能包含若干个源程序文件，而编译是以源程序文件为对象的，一次编译只能得到与一个源程序文件相对应的目标文件（也称目标模块），它只是整个程序的一部分。必须把所有的编译后得到的目标模块连接装配起来。此外，程序中往往会用到系统提供的标准函数（如 printf()函数），因此目标文件还需要和系统的函数库等系统资源相连接成一个整体，生成一个可供计算机执行的目标程序，称为可执行程序（executive program），其后缀一般为.exe，如 test.exe、f.exe。

即使一个程序只包含一个源程序文件，编译后得到的目标程序也不能直接运行，也要经过连接阶段，因为还要与函数库进行连接，才能生成可执行程序。

以上连接的工作是由一个称为**“连接编辑程序”**（linkage editor）的软件来实现的。

4. 运行可执行程序

运行可执行程序后，会得到相应的运行结果。

以上过程如图 1.2 所示。其中实线表示操作流程，虚线表示文件的输入/输出。例如，编辑后得到一个源程序文件 f.c；接着在进行编译时再将源程序文件 f.c 输入，经过编译得到目标程序文件 f.obj；再将所有目标模块输入计算机，与系统提供的库函数等进行连接，得到可执行的目标程序 f.exe；最后把 f.exe 输入计算机，并使之运行，得到结果。

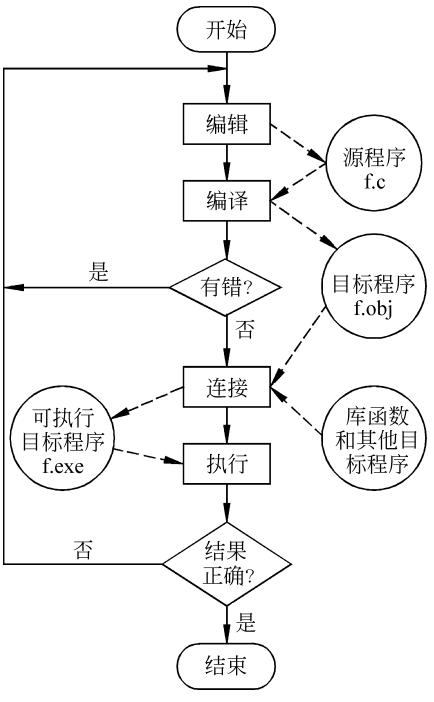

图　1.2

一个程序从编写到运行成功，往往不是一次成功的，而是需要多次反复地编写。编写好的程序并不一定能保证它正确无误，除了用人工方式检查有无错误外，还要借助编译系统来检查有无语法错误。从图 1.2 中可以看到：如果在编译过程中发现错误，应当重新检查源程序，找出问题，修改源程序，并重新编译，直到无错为止。有时编译过程未发现错误，能生成可执行程序，但是运行的结果不正确。一般情况下，这不是语法方面的错误，而是程序逻辑方面的错误，例如计算公式不正确、赋值不正确等，应当返回检查源程序，并改正错误。

为了编译、连接和运行 C 语言程序，必须要有

相应的编译系统。目前使用的编译系统大多是集成环境(IDE)。Visual C++是在 PC 上使用较广泛的集成环境，它把程序的编辑、编译、连接和运行等操作全部集中在一个界面上进行。集成环境功能丰富，使用方便，直观易用。由于 C++与 C 基本上是兼容的，因此用 Visual C++既可以对 C++程序进行编译，也可以对 C 语言程序进行编译。熟悉它以后也会有利于今后进一步学习 C++语言。在《C 语言程序设计教程学习辅导》中的第三部分介绍了怎样使用 Visual Studio 2010 对 C 语言程序进行编辑、编译和运行，另外还介绍了如何在线编译程序。

请读者上机运行本章介绍的 3 个 C 语言程序，初步掌握上机的方法。

1.6 算法是程序的灵魂

1.6.1 什么是算法

做任何事情都有一定的内容和步骤。例如，你想从北京去天津开会，首先要去买火车票，然后按时乘坐地铁到北京南站，登上火车，到天津站后坐公交车到会场参加会议。这些步骤都是按一定的顺序进行的，缺一不可，次序错了也不行。我们从事各种工作和活动，都必须事先想好步骤，然后按部就班地进行，才能避免产生错乱。实际上，在日常生活中，由于已养成习惯，所以人们意识不到每件事都需要事先设计出“行动步骤”。例如吃饭、上学、打球、做作业等，事实上都是按照一定的规律进行的，只是人们不必每次都重复考虑它而已。

算法是解决“做什么”和“怎么做”的问题。在写程序之前，必须想清楚“做什么”和“怎么做”。“做什么”往往是从题目或任务中可以看出来或整理出来的(例如，求三角形的面积、求一元二次方程等)，而“怎么做”则是由程序设计者去思考和设计的。“怎么做”包括两方面的内容：一是要做哪些事情才能达到解决问题的目的；二是决定做这些事情的先后次序。概括地说，算法是指解决特定问题的步骤和方法。

不要认为只有“计算”的问题才有算法。广义地说，为解决一个问题而采取的方法和步骤都称为“算法”。例如，描述太极拳动作的图解就是“太极拳的算法”。一首歌曲的乐谱也可以称为该歌曲的算法，因为它指定了演奏该歌曲的每一个步骤，按照它的规定就能演奏出预定的曲子。菜谱中的烹调方法就是做菜的算法。

本书所关心的当然只限于计算机算法，即计算机能执行的算法。例如，让计算机计算 $1\times2\times3\times4\times5$，或将 100 个学生的成绩按高低分数的次序排列，是可以做到的，而让计算机去执行“替我理发”或“做一碗红烧肉”，目前还不能实现。

对同一个问题，可以有不同的解题方法和步骤。例如，求 $1+2+3+\cdots+100$，即 $\sum_{n=1}^{100} n$。有人可能先进行 $1+2$，再加 3，再加 4，一直加到 100；而有人采取方法为：$100+(1+99)+(2+98)+\cdots+(49+51)+50=100+49\times100+50=5050$。还可以有其他的方法。当然，方法有优劣之分。有的方法只需进行很少的步骤，而有的方法则需要较多的步骤。一般来说，应尽量采用相对简单、运算步骤少的方法。因此，为了有效地进行解题，不仅需要保证算法正确，还要考虑算法的质量，应选择合适的算法。