第 1 章 C 语言程序设计概述

计算机是 20 世纪人类最伟大的科学技术发明之一，对人类的生产活动、社会活动、科学研究等诸多方面都产生了非常重要的影响，同时也带动了全球范围内的技术进步。计算机是一种具有内部存储能力的、自动的、高效的电子设备。计算机之所以能为我们自动完成各种各样的工作，是因为它执行了编程者事先编制好的相应程序，这些程序都是应用程序设计语言编制的。

在众多的程序设计语言中，C 语言是当今国际上最流行的计算机程序设计语言之一。它是一种高级程序设计语言，因方便、灵活、通用、直接操作计算机硬件等特点，深受编程者的欢迎。无论是开发系统软件，还是设计应用软件，都可以看到 C 语言的身影。

本章主要介绍程序设计语言的基本概念，C 语言程序的组成、结构特点、算法，以及 C 语言的开发环境等。

1.1 程序与程序设计语言

1.1.1 程序的基本概念

有人认为计算机无所不能，会自动为我们进行工作。其实，计算机的每一个操作都是根据编程者事先设定的指令进行的。例如，用一条指令要求计算机执行一次加法运算，用另一条指令要求计算机将此运算结果输出到显示屏等。为了让计算机为我们完成一定的工作，就必须编制一系列操作指令，并把这些指令存放在计算机中，需要时就向计算机发出一个简单的命令（触发指令），计算机就会自动地逐条执行这些指令，全部指令执行完就得到了预期的结果。

所谓程序，就是一组计算机能识别和执行的指令的集合。每一条指令都会让计算机执行特定的操作。计算机执行一个程序，其实就是执行这个程序中的一条条指令来完成一定的工作。为了使计算机系统能实现各种功能，就需要各种各样的程序。这些程序大多数是由计算机软件（程序）设计人员根据需要设计好，作为计算机的软件系统的一部分提供给

用户使用。此外，用户还可以根据自己的实际需要设计一些应用程序，例如学生成绩统计程序、财务管理程序等。

总之，计算机的一切操作都是由程序控制的，离开程序，计算机将无法工作。计算机的本质是程序的执行机器，程序和指令是计算机系统中最基本的概念。

1.1.2 程序设计语言

人和人之间的交流需要通过语言来完成。人和计算机之间的交流也需要一种语言，这就是程序设计语言。人们需要借助计算机能够理解的语言，完成一系列指令的编写和程序的设计，告诉计算机要“做什么”“怎么做”，计算机才能按人们的要求完成特定的操作并给出预期的结果。

程序设计语言经历了以下几个发展阶段。

1. 机器语言

我们知道，计算机工作是基于二进制的，计算机只能识别和接受 0 和 1 组成的序列所表示的数据和指令。我们把计算机能直接识别和接受的二进制代码称为机器指令。机器指令的集合称为机器语言。在机器语言中，每条指令都用 0 和 1 组成的序列来表示。例如，1011011000000000 是某计算机的一条加法指令。

不同类型的计算机，使用的机器语言不相同。用机器语言编写的程序，计算机可以直接识别和执行，执行效率高。但机器语言的指令不直观，难认、难记、难理解、易错。用机器语言编写程序时，要求编程者必须相当熟悉计算机结构，而且编写的程序不能通用，因而，目前很少用机器语言直接编程。

2. 汇编语言

为了克服机器语言的缺点，人们采用“助记符号”来表示机器语言中的机器指令，这样便形成了汇编语言。汇编语言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”等。上面的加法指令“1011011000000000”用汇编语言描述为

ADD A，B（将寄存器 A 中的数据与寄存器 B 中的数据相加，放到寄存器 A 中）

显然，计算机不能直接识别和执行用汇编语言编写的程序，需要用一种称为汇编程序的软件把汇编语言编写的程序转换为计算机能识别的机器指令。人们通常把汇编语言编写的程序称为源程序（source program），而把计算机能识别的机器指令称为目标程序（object program）。

虽然汇编语言比机器语言简单好记一些，但仍然难以普及，只有专业人员使用。一般情况下，汇编语言指令和机器语言指令之间具有一一对应的关系。因而，不同的计算机其汇编语言也不尽相同，不能相互通用。

不论是机器语言还是汇编语言，都与计算机硬件直接相关，它们对机器过分依赖，要求使用者必须对硬件结构及其工作原理都十分熟悉，非计算机专业人员是难以做到的。

3. 高级语言

为了克服机器语言、汇编语言的缺点，20 世纪 50 年代中期开始，人们创造了与人类自然语言（特别是数学语言）很接近的程序设计语言。这类语言功能强大，且不依赖于具体机器，用它们编写的程序对任何型号的计算机都适用（或只需做很少的修改），程序与具体机器距离较“远”，故称为高级语言。自 1954 年第一个完全脱离机器硬件的高级语言 FORTRAN 问世以来，共有几百种高级语言出现。目前被广泛使用的高级语言有 Pascal、C、FORTRAN、VC、VB、C++、Java 等。

在这种程序设计语言中，所用到的语句和指令是用英文单词表示的，所用到的运算符号和运算表达式与人们日常所用数学公式很类似，很容易理解。程序运行的结果也用英文和数字输出，十分方便。

例如，在 C 语言程序中，想计算并输出 4.5×7sin(π/7)，只需写出下面这样一条语句即可。

```
printf("%f",4.5*7*sin(3.1415926/7));
```

当然，用高级语言编写的程序，计算机也是不能直接识别的，也要进行“翻译”。人们用一种称为编译程序的软件把用高级语言写的程序（称为源程序）转换为机器指令的程序（称为目标程序），然后让计算机执行机器指令程序，最后得到运算结果。

高级语言的一个语句往往对应多条机器指令，表达能力很强。在使用高级语言时，编程者不需要熟悉计算机的指令系统，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。

高级语言经历了以下几个发展阶段。

（1）非结构化语言。初期的高级语言属于非结构化语言，编程风格比较随意，只要符合语法规则即可，没有严格的规范要求，程序中的流程可以随意跳转。人们往往追求程序执行的效率而采用了许多“小技巧”，使程序变得难以阅读和维护。早期的 BASIC、FORTRAN 和 ALGOL 等都属于非结构化的语言。

（2）结构化语言。为了解决以上问题，迪科斯彻（E. W. Dijkstra）于 1969 年提出了“结构化程序设计方法”，规定程序必须由具有良好特性的基本结构（顺序结构、选择结构、循环结构）构成，程序中的流程不允许随意跳转，程序总是按由上而下顺序执行各个基本结构。这种程序结构清晰，易于编写、阅读和维护。QBASIC、FORTRAN 77 和 C 语言等属于结构化语言，这些语言的特点是支持结构化程序设计方法。

（3）面向任务的程序设计语言。上述两类语言都是面向过程的，利用这类语言求解一个复杂的问题，必须先要分析解决问题的过程，描述问题如何求解，然后才能用算法语言进行程序设计来实现。面向任务的程序设计语言是非过程化的语言，也就是说，不需要知道问题是如何求解的，只需描述需求解的问题是什么，然后便可用程序设计语言来实现。

数据库操纵语言便是一种面向任务的程序设计语言。例如，设在某数据库应用系统

中，有一个学生情况表 STUDENT，若要从表中查找学生的信息，可以使用数据库查询语言（SQL）。采用 SELECT 语句便可完成这个任务，SELECT 语句描述如下：

```
SELECT  NO,NAME,AGE,SEX  FROM  STUDENT
```

该语句从 STUDENT 表中查找到学生的 NO、NAME、AGE、SEX 等方面的信息。至于 SELECT 语句是如何进行查询的，这个过程用户就不必了解了。

面向任务的程序设计语言可以提高应用程序的开发速度和质量，也可使应用程序能容易迅速被修改。同时，非计算机专业人员也能很方便地使用面向任务的程序设计语言开发自己的程序。

这类语言在应用系统的开发中使用较为广泛，尤其是一些管理信息系统应用程序的开发。

（4）面向对象的程序设计语言。面向对象的程序设计语言在 20 世纪 90 年代开始流行。现在，面向对象的程序设计语言已成为程序设计的主流语言之一。由 C 语言发展而来的C++就是一种非常优秀的面向对象的程序设计语言。

面向对象的方法是分析方法、设计方法和思维方法的综合。其所追求的基本目标是使人们分析、设计和实现一个系统的方法尽可能与人们认识一个系统的方法相接近。

在面向对象编程中，程序被看作是相互协作的对象集合，每个对象都是某个类的实例，所有的类构成一个通过继承关系相联系的层次结构。面向对象的程序设计语言具有对象生成功能、消息传递机制以及类和继承机制。

对象是对客观事物的抽象，面向对象的编程，就是针对客观事物设计的程序。因此，面向对象的编程是非常直观的。面向对象的程序设计方法比面向过程的程序设计方法更清晰，更适合用于开发大型复杂的软件。

综上所述，每一种语言都有它的优势和劣势，对于不同的问题，要根据实际情况来选择，以便更高效、更优质地解决相关的问题。

1.2 算法及其描述

在程序设计过程中，对稍微复杂的问题，我们必须先通过分析问题来设计、描述解决问题的方法和基本步骤，这就不可避免地需要涉及算法。有人曾说“计算机科学就是研究算法的科学”，足见算法在程序设计中的重要性了。下面从算法的概念和描述方法两方面对算法的问题进行讨论。

1.2.1 算法的概念

著名的瑞士计算机科学家、Pascal 语言发明者沃思（Niklaus Wirth）教授提出一个公式：

程序=算法+数据结构

这个公式的重要性在于它说明了程序与算法的关系；同时，也说明了数据结构的选择也是十分重要的。其实一个程序的设计内容主要包括两方面。一方面是对数据的描述。在程序中要指定用到哪些数据，以及这些数据的类型和数据的组织形式，这就是数据结构。另一方面是对操作的描述。要求计算机进行操作的步骤，也就是算法。

数据是操作的对象，操作的目的是对数据进行加工处理，以得到期望的结果。

算法是解决“做什么”和“怎么做”的问题。程序中的操作语句，实际上就是算法的体现。显然，不了解算法就谈不上程序设计。通常认为，算法是对特定问题求解步骤的一种描述，是由一套规则组成的一个过程，是指令的有限序列，其中每一条指令表示一个或多个操作。

算法应当具备以下特点。

（1）有穷性。一个算法应包含有限的操作步骤，而不能是无限的。

（2）确定性。算法的每一个步骤都应当是确定的，不应当是含糊的、模棱两可的。

（3）有零个或多个输入。所谓输入是指在执行算法时需要从外界取得必要的信息。

（4）有一个或多个输出。算法的目的是求解，“解”就是输出。

（5）有效性。算法中的每一个步骤都应当能有效地执行，并得到确定的结果。例如，若 b=0，则 a/b 是不能有效执行的。

1.2.2 算法的描述方法

为了表示一个算法，可以用不同的方法。常用的方法有自然语言、流程图、N-S 流程图、伪代码、计算机语言等。我们这里只介绍流程图方法。

流程图是用一些图框来表示各种操作，用线表示这些操作的执行顺序，用图形表示算法，直观形象，易于理解。美国国家标准化协会（ANSI）规定了一些常用的流程图符号（如图 1.1 所示），已为世界各国程序工作者普遍采用。具体介绍如下。

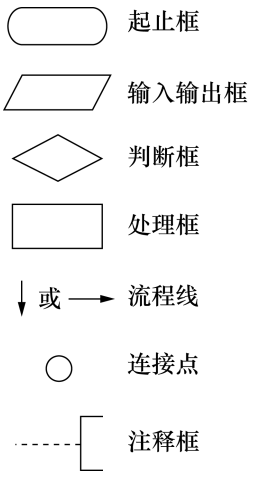

图 1.1　流程图符号

（1）起止框：扁圆表示一个算法的开始或结束（转向外部环境或外部环境转入的端点符）。

（2）输入输出框：平行四边形表示数据的输入输出，其中可注明数据名称、来源、用途或其他文字说明。

（3）判断框：菱形表示判断。菱形内可注明判断的条件。它只有一个入口，但可以有若干个可供选择的出口。

（4）处理框：矩形表示各种处理功能。矩形内可注明处理名称或其简要功能。

（5）流程线：带箭头的实线，表示程序执行的方向。

（6）连接点：小圆圈表示将画在不同地方的流程线连接起来。它表示这两个点是连接

在一起的，实际上它们是同一个点，只是画不下才分开来画。用连接点可以避免流程线交叉或过长，使流程图清晰。

（7）注解框：注解是程序的编写者向阅读者提供的说明。它用虚线连接到被注解的符号或符号组上。

例 1.1　键盘输入 a、b、c，计算方程 $ax^2+bx+c=0$ 的实数根。

流程图如图 1.2 所示。

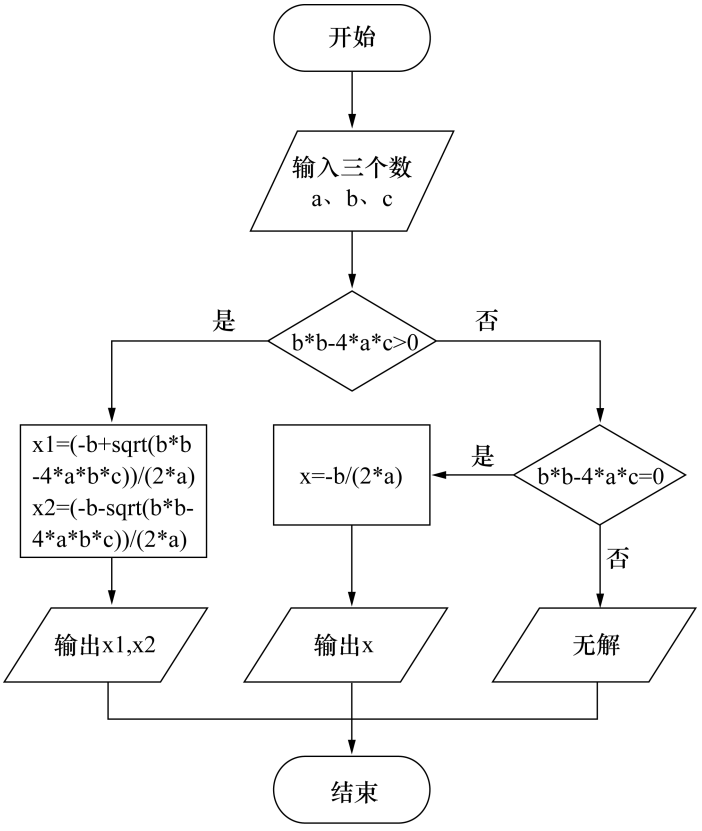

图 1.2　求二次方程根的流程图

1.3　C 语言的发展及特点

1.3.1　C 语言的发展

1972 年，美国贝尔实验室的丹尼斯·里奇（D. M. Ritchie）在 B 语言的基础上设计了 C 语言。C 语言的设计初衷只是为描述和实现 UNIX 操作系统提供一种工作语言。1973 年，肯·汤普逊（Ken Thompson）和里奇合作用 C 语言改写了 UNIX 操作系统，其中 90%以上的

代码是用 C 语言改写的，即 UNIX 5.0。随着 UNIX 的日益广泛使用，C 语言也迅速得到推广。

1978 年，布赖恩·克尼汉（Brian W. Kernighan）和里奇合著了影响深远的名著《C 语言设计语言》（*The C Programming Language*），这本书中介绍的 C 语言成为后来广泛使用的 C 语言版本的基础，可以说它是第一个 C 语言标准。1983 年，ANSI 根据 C 语言问世以来各种版本对 C 语言的发展和扩充，制定了第一个 C 语言标准草案（'83 ANSI C）。1989 年，ANSI 公布了一个完整的 C 语言标准——ANSI X3.159-1989（常称为 ANSI C 或 C 89）。1990 年，国际标准化组织（ISO）接受 C 89 作为国际标准，发布 ISO/IEC 9899：1990，它和 ANSI 的 C 89 基本上是相同的。

C 语言是一种用途广泛、功能强大、使用灵活的结构化程序设计语言，既可用于编写应用软件，又可用于编写系统软件，因此在问世以后得到迅速推广。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，C 语言成为学习和使用人数最多的一种计算机语言，绝大多数理工科大学都开设了 C 语言程序设计课程。

1.3.2 C 语言的特点

C 语言主要有以下特点。

（1）C 语言是一种结构化语言。C 语言的主要成分是函数，函数是 C 语言程序的基本结构模块。同时，C 语言具有结构化的控制语句（如 if…else 语句、while 语句、do…while 语句、switch 语句和 for 语句）。

（2）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，略去了一切不必要的成分，源程序短。

（3）运算符丰富。C 语言的运算符包含的范围很广泛，它把括号、赋值和强制类型转换等都作为运算符处理，同时还增加了类似++、--等运算符。

（4）数据类型丰富。C 语言提供的数据类型包括整型、浮点型、字符型、数组类型、指针类型、结构体类型和共用体类型等。尤其是指针类型数据，使用十分灵活和多样化，能用来实现各种复杂的数据结构（如链表、树、栈等）的运算。

（5）C 语言允许直接访问物理地址，能进行位（bit）操作，能实现汇编语言的大部分功能，可以直接对硬件进行操作。因此 C 语言既具有高级语言的功能，又具有低级语言的许多功能，可用来编写系统软件。C 语言的这种双重性，使它既是成功的系统描述语言，又是通用的程序设计语言。

（6）用 C 语言编写的程序可移植性好。C 语言将与硬件有关的因素从语言主体中分离出来，通过库函数或其他实用程序实现它们。这主要体现在输入输出操作上，C 语言不把输入输出作为语言的一部分，而是作为库函数由具体实用程序实现，大大提高了程序的可移植性。

（7）生成的目标代码质量高，程序执行效率高。由于 C 语言的可移植性好，硬件控制、表达和运算能力强，经常用来编写系统软件和许多大的应用软件。以前只能用汇编语

言处理的问题，后来也可以用 C 语言来处理了。目前 C 语言的主要用途之一是编写嵌入式系统程序。

当然，C 语言也有一些不足之处，比如对数据类型检查不严格，表达式出现二义性，不能自动检查数据越界，初学者较难掌握运算符的优先级和结合性等。

1.4 C 语言程序的基本结构

1.4.1 简单程序举例

在学习 C 语言的语法之前，先通过几个例子，初步了解 C 语言程序的基本结构。

例 1.2 在屏幕上输出“This is a C program.”一行信息。

源程序代码：

```
#include<stdio.h>                          /* 这是编译预处理指令 */
main()                                     /* 主函数 */
{                                          /* 函数开始的标志 */
    printf("This is a C program.\n");      /* 输出所指定的一行信息 */
}                                          //函数结束的标志
```

【运行结果】

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

【程序说明】

（1）main()函数。C 语言程序由函数组成，其中有且只有一个 main()函数，称为主函数。这里 main()为函数名。main()后面由花括号对“{}”括起来的部分是 main()函数的函数体。

（2）注释。注释符“/*”与“*/”之间的内容构成 C 语言程序的注释部分。“/*”与“*/”之间的内容可以是一行，也可以是多行。另外，程序中的单行注释，除可以用上面的注释符外，还可以用“//”开始，“//”后面为注释内容，如上述程序中第 5 行。注释部分不参与程序的编译和执行，只是起说明作用，增加程序的可读性。

（3）预处理指令。程序中用到语句“printf("This is a C program.\n");”，它是调用了系统为我们提供的库函数 printf（后面会详细讲解），其功能是输出括号中的字符串“This is a C program.”。\n 是换行符，即在输出“This is a C program.”后，显示器上的光标移动到下一行的开头。这个光标位置称为输出的当前位置，即下一个输出的字符出现在此位置上。

在使用函数库中的输入输出函数时，编译系统要求程序提供有关此函数的信息（例如对输入输出函数的声明、宏的定义、全局量的定义等，这些后文会介绍），程序第 1 行“#include <stdio.h>”的作用就是用来提供这些信息的。stdio.h 是系统提供的一个文件名，

称为头文件，输入输出函数的相关信息已事先放在 stdio. h 文件中。现在用#include 指令把这些信息调入供使用。如果没有此#include 指令，就不能执行 printf 函数。关于编译预处理指令#include 后文会详细介绍。这里要记住：在程序中如要用到标准函数库中的输入输出函数，应该在本程序文件模块的开头加上下面一行代码：

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

（4）语句。程序中要求计算机完成的操作是由函数中的 C 语句完成的。在每条语句的最后必须有一个分号，分号是 C 语句的必要组成部分。

例 1.3 键盘输入两个整数，求较大者。

源程序代码：

```
#include<stdio.h>          /* 编译预处理指令* /
//主函数
main()
{
    int max(int x,int y);  /* 对用户自定义函数 max()的声明 */
    int a,b,m;             /* 定义整型变量 a、b、m */
    scanf("%d%d",&a,&b);/* 从键盘输入两个整数，分别存入变量 a、b 中 */
    m=max(a,b);            /* 调用 max()函数，将返回的值赋给 m */
    printf("max=%d\n",m);/* 输出 m 的值 */
}
//求两个整数中较大者的 max()函数
int max(int x,int y);      //定义 max()函数，函数值为整型，形式参数 x、y 为整型
{                          //函数开始标志
    int z;                 //定义本函数中用到的局部变量 z
    if(x>y)                //若 x>y 成立，将 x 的值赋给 z
        z=x;
    else                   //否则(即 x>y 不成立)，将 y 的值赋给 z
        z=y;
    return z;              //将 z 的值作为函数 max()的值返回到调用 max 的位置
}                          //函数结束标志
```

【程序说明】

（1）本程序用到两个函数：一个是主函数 main( )，另一个是用户自定义函数 max( )。max( )函数的功能是返回两个整数中较大者。因为 max( )是用户自定义函数，所以程序中必须定义，即编制 max( )实现功能的代码。

（2）变量、函数等的声明。程序中用到的变量、常量、用户自定义函数等，都必须提前声明或定义。例如，“int max(int x, int y);” 是对用户自定义函数 max( )的声明，“int

a,b,m;”是定义整型变量 a、b、m。

1.4.2 C 语言程序的结构

1. 一般 C 语言程序包含的内容

一般地，一个 C 语言程序由一个源程序文件组成，其主要内容包括：

（1）头文件（一组#include<*.h>语句，也称包含文件）；

（2）用户自定义函数说明部分；

（3）全局变量定义；

（4）主函数；

（5）用户自定义函数。

在主函数和用户自定义函数中，一般又包含了局部变量、若干个库函数、控制流程语句、用户函数的调用语句等。设 fl( )、fn( )代表用户自定义的函数，则 C 语言程序的一般形式可表达为

```
头文件# include…预处理命令
函数声明
全局变量定义
main()
{
    局部变量定义
    程序段
}
fl()
{
    局部变量
    程序段
}
…
fn()
{
    局部变量
    程序段
}
```

2. C 语言程序结构及其特点

结合例 1.2 和例 1.3 及上述一般 C 语言程序包含的内容，可以看到一个 C 语言程序结构主要有以下特点。

（1）一个程序由一个或多个函数组成，其中有且只有一个主函数——main( )。

每一个 C 语言程序都必须有一个 main( ) 函数作为主控函数，称为主函数。主函数的函数体由一对花括号“{}”括起来，函数体内由一条条的语句组成，每条语句完成一定的任务（当然也可以有空语句）。程序从 main( ) 函数的第一条语句开始执行，最后一条语句执行后程序结束。

【注意】一个规模较小的程序，一般只包括一个源程序文件，而规模较大的程序一般包含多个源程序文件，但只能是其中一个源程序文件中包含一个主函数。

（2）预处理指令。如#include<stdio. h>，还有一些其他预处理指令，如#define 等。

C 编译系统在对源程序进行“翻译”之前，先由一个预处理器（也称预处理程序、预编译器）对预处理指令进行预处理。比如，对于#include <stdio. h>指令来说，就是将 stdio. h 头文件的内容读进来，取代#include<stdio. h>。由预处理得到的结果与程序其他部分一起，组成一个完整的、可以用来编译的最终源程序，然后由编译程序对该源程序正式进行编译，才得到目标程序。

（3）用户自定义函数。用户自定义函数是完成特定功能的一段程序代码，有了函数的概念，所以 C 语言很容易实现模块化设计。有关内容在第 7 章详述。

（4）函数体。函数首部后的花括号内的部分称为函数体。如果在一个函数中包括有多层花括号，则最外层的一对花括号是函数体的范围。函数体内的内容主要包括以下部分。

①声明部分。声明部分包括：定义在本函数中所用到的变量，如例 1. 3 中在 main( ) 函数中定义变量“int a,b,c;”；对本函数所调用函数进行声明，如例 1. 3 中在 main( ) 函数中对 max( ) 函数的声明“int max(int x,int y);”。

②执行部分。执行部分由若干个语句组成，指定在函数中所进行的操作。

（5）语句。程序中要求计算机完成的操作是由函数中的 C 语句完成的。如赋值、输入输出数据的操作都是由相应的 C 语句实现的。在每条语句的最后必须有一个分号，分号是 C 语句的必要组成部分。

（6）注释。一个良好的编程习惯是在源程序中加上必要的注释，以增加程序的可读性。C 语言的注释有两种方式：一种是由“//”开头（用于单行注释），另一种是由“/*”和“*/”包含起来（用于多行或单行注释）。

1. 5　C 语言字符集、标识符与关键字

1. 5. 1　C 语言字符集

任何一个计算机系统所能使用的字符都是固定的、有限的，它要受硬件设备的限制。要使用某种计算机语言来编写程序，就必须使用符合该语言规定的，并且计算机系统能够使用的字符。

C 语言和其他语言一样，它的基本字符集包括英文字母、阿拉伯数字以及其他一些符号，具体归纳如下。

（1）英文字母：大小写各 26 个，共计 52 个。
（2）阿拉伯数字：0～9，共计 10 个。
（3）下划线：_ 。
（4）其他一些特殊符号：

```
+     -     *     /     %     ++    -     <     >     =
>=    <=    ==    ! =   |     &     !     ||    &&    ^
~     &=    (     )     [     ]     {     }     \
?     :     .     ,     ;     #     "     '     -
```

1.5.2 标识符

C 语言中标识符是指对变量、符号常量、函数等命名的有效字符序列。简单地说，标识符就是程序中用到的变量、符号常量、函数等的一个名字，比如 PI、printf、x、y 等。

在 C 语言中，一个标识符由字母、数字和下划线组成，其中第一个字符必须是字母或下划线。

下面给出一些合法与不合法命名的标识符。

合法标识符：age、average、sum、i、x、day、student、arr、_high。

不合法标识符：a-b、$238、#au、a * b、6abc。

1.5.3 关键字

关键字是指在程序设计语言中事先定义好的，具有特定含义的标识符。所以，关键字不能作为变量或函数名来使用，用户只能根据系统的规定使用它们。根据 ANSI 标准，C 语言可使用以下 32 个关键字：

```
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
```

1.6 用 C 语言求解问题的过程

通过前面讲述我们知道，用 C 语言编写的程序称为源程序，计算机不能直接识别和执行使用高级语言编写的指令（程序），必须经过编译程序将 C 语言源程序翻译成二进制形式的目标程序，然后再将目标程序与系统的函数库以及其他目标程序连接起来，形成可执行程序。下面结合一个例子来说明运行 C 语言程序的具体步骤和方法。

例 1.4　求 1～100 所有奇数的和。

1.6.1 问题分析与算法设计

本问题求在一定范围内（1～100）、满足一定条件（奇数）的若干整数的和，是一个求累加和的问题。

这类问题的基本求解方法是：设置一个变量（如 sum），将其初值置为 0，再在指定范围（1～100）内寻找满足条件（奇数）的整数，将它们一个一个累加到 sum 中。为了处理方便，将正在查找的整数也用一个变量表示（如 i）。所以，一次累加过程的 C 语言语句为

```
sum=sum+i;
```

它表示把 sum 的值加上 i 后再重新赋给 sum。

这个累加过程要反复做，就要用程序设计语言的循环控制语句来实现。在循环过程中，需解决以下两个问题。

（1）要判别 i 是否满足问题要求的条件（奇数）。可以用选择结构语句实现只把满足条件的整数累加到 sum 中。

（2）需要对循环次数进行控制。这可通过 i 值的变化进行控制，即 i 的初值设为 1，每循环一次加 1，一直加到 100 为止。

基于上述解决问题的思路，就可以逐步明确解决问题的步骤，即确定解决问题的算法。

根据前面的分析，可以用流程图来描述解决步骤（算法），如图 1.3 所示。

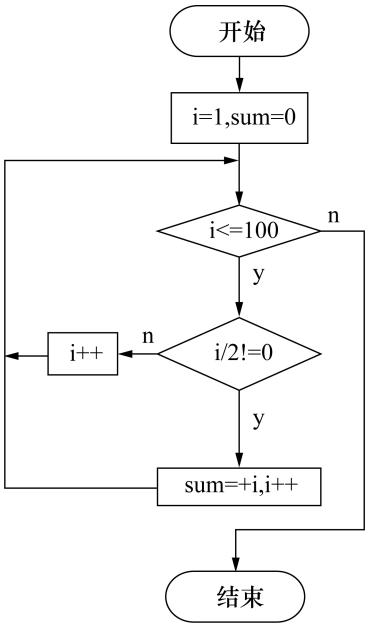

图 1.3　“求 1～100 所有奇数和”的流程图

1.6.2 编辑程序

当确定解决问题的步骤后，就可以开始编写程序了。一般是在编程环境中应用其中的编辑功能直接编写程序，最后将此源程序以文件形式存放在自己指定的文件夹内。C 语言源程序扩展名一般为 .c。

源程序代码：

```
#include<stdio.h>
int main()
{
    int i,sum=0;                    /* 定义整型变量 i、sum */
    for(i=1;i<=100;i++)
        if(i%2!=0)
            sum=sum+i;
    printf("sum=%d\n",sum);
}
```

【程序说明】

（1）“int i,sum=0;”定义了两个整数类型（int）的变量 i 和 sum，同时把 sum 初值赋为 0。

（2）“for(i=1;i<=100;i++)”是一个循环，它表示从 i 等于 1 的时候开始循环，每循环一次 i 加上 1（i++），只要 i 的值小于等于 100，这个循环就一直进行（也就是说 i 大于 100 时就退出循环）。每次循环执行的内容就是它后面的 if 语句。

（3）“if(i%2!=0){sum=sum+i};”表示如果 i 是奇数，就将 i 累加到 sum 中。%是一个针对整数的运算，代表求余，如果 i 被 2 除后的余数不为 0，则说明它是一个奇数。在 C 语言中，相等的判断用两个等号“==”来表示，不相等的判断用“!=”来表示。

（4）当循环退出时，就调用 C 编译系统事先定义好的 printf( ) 函数将所要的结果值（sum）输出。输出时要求按十进制整数方式（由%d 来说明）输出。

包括 C 语言在内的许多程序设计语言都有一个原则：若想使用一个对象，需要先说明（或定义）该对象，除非有某种默认规则。所以，一般变量的定义放在具体语句的前面，在调用一个函数前先声明或定义这个函数。由于编程时会经常使用编译系统事先定义好的一些函数，所以编译系统会事先把各种类型的函数声明组织在一个文件中，如所有输入输出函数就组织在一个头文件 stdio.h 中，编程者只要在程序的开头写上语句

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

就可以直接调用该文件中声明的所有函数。

1.6.3 编译

当编辑好程序后，下一步工作就是应用该语言的编译程序对其进行编译，以生成二进制代码表示的目标程序（一个二进制文件，文件后缀为 .obj）。

对源程序进行编译，先用 C 编译系统提供的“预处理器”（又称“预处理程序”）对程序中的预处理命令进行编译预处理。例如，对于#include <stdio.h>命令来说，就是将 stdio.h 头文件的内容读进来，取代#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，然后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，则发出“错误信息”，告诉编程者认真检查改正。修改程序后重新进行编译，如果还有错，再发出“错误信息”。如此反复进行，直到没有语法错误为止。这时，编译程序自动把源程序转换为二进制形式的目标程序（在 Dev C++中后缀为 .obj，如 f.obj）。如果不特别指定，此目标程序一般也存放在用户当前目录下，此时源文件没有消失。

当然，如果程序有语法错误，编译程序就会指出该语法错误所在，而不生成二进制代码。

1.6.4 连接

实际上，通过编译所产生的二进制目标文件还不能直接运行，它需要与编程环境提供的库函数进行连接（link），形成可执行的程序（文件后缀为 .exe）后才能被执行。

这是因为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），它只是程序的一部分。必须把所有编译后得到的目标模块连接装配起来，再与函数库相连接形成一个整体，生成一个可供计算机执行的目标程序，即可执行程序，在 Dev C++中后缀为 .exe，如 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，因为要与函数库进行连接，才能生成可执行程序。

1.6.5 运行与调试

当程序通过了语法检查、编译生成可执行文件后，就可以在编程环境或操作系统环境中运行（run）该程序。

当然，如果程序没有语法错误，但存在逻辑错误，程序运行所得到的结果也可能不是我们想要的。比如，在上述程序中，如果把“if(i%2!=0)sum=sum+i;”中的“!=”（不等于）错写成“==”（等于），这个程序虽然通过了语法检测，但运行结果是 1～100 所

有偶数的和。

如果程序有逻辑错误，就需要对程序进行调试。调试是指在程序中查找错误的过程。调试最主要的工作是找出错误发生的地方。调试手段我们在这里就不赘述。

目前，编程者大多使用集成开发工具来开发 C 语言程序，适合 C 语言的集成开发工具有多种，常用的有 Turbo C、Microsoft C、Visual C++、Borland C++、Dev C++、C++ Builder 等，它们各具特色，分别适合于不同的操作系统环境。Visual C++、Borland C++、Dev C++、C++ Builder 既适合开发 C++语言程序，也适合开发 C 语言程序。

1.7 Dev C++开发环境简介

Dev C++是一个 Windows 操作系统环境下的适合初学者使用的轻量级 C/C++ 集成开发环境，可以实现 C/C++程序的编辑、编译、连接和执行。下面介绍 Dev C++常用的一些基本操作。

1.7.1 Dev C++工作界面

首次启动 Dev C++时，选择中文方式或启动后点击主菜单“工具”→“环境选项”，在弹出的对话框中选择界面页，在“Language”下拉列表中选择“Chinese”即可。

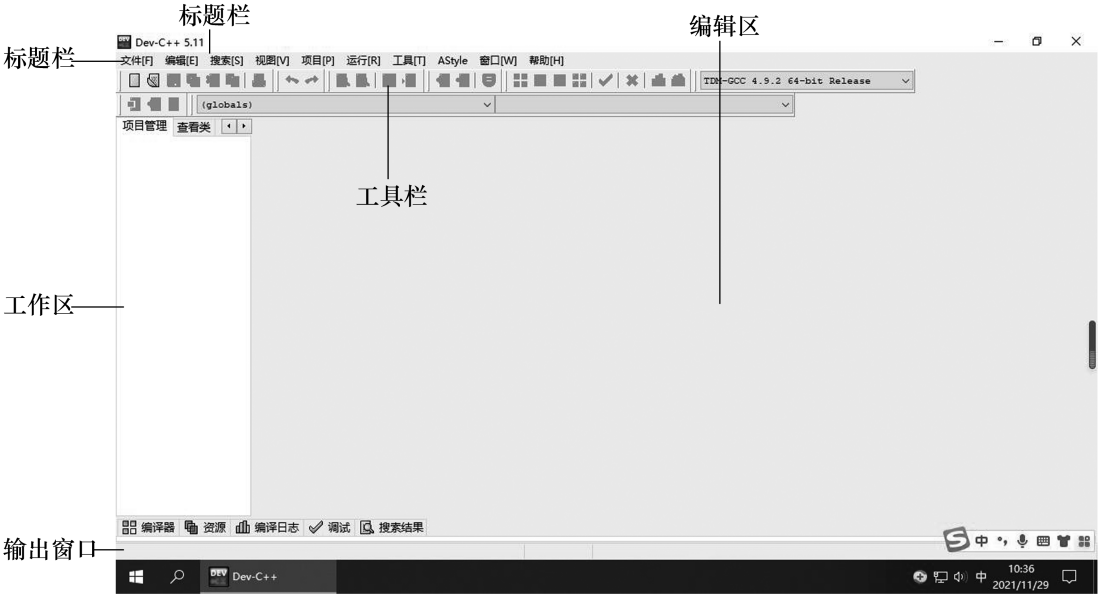

图 1.4 Dev C++工作界面

工作界面包括标题栏、菜单栏、工具栏、工作区、编辑区和输出区。

1. 标题栏

Dev C++主窗口的第一行就是标题栏。标题栏的左边显示当前文件的文件名以及版本信息，默认文件名是“未命名 1”。

2. 菜单栏

标题栏下面依次是菜单栏和工具栏。各菜单的主要功能如下。

文件：用来新建、打开、保存项目文件，也是必须要做的第一步。

编辑：用来编辑文件。

视图：用来查看代码，显示其他窗口，打开/关闭工具栏等。

项目：用来新建、添加、移去项和设置启动项等。应当注意的是，该项只有在新建项目或打开项目时才会出现)。

运行：编译、运行、调试等。

工具：用来对工具栏、菜单以及集成开发环境进行定制。

AStyle：只要点击菜单“AStyle”→“格式化当前文件”，就可以把当前窗口中的源代码按一定的风格迅速整理好排版格式。

窗口：用来新建/拆分窗口和窗口布局。

帮助：给出相关的帮助。

3. 工具栏

工具栏和菜单栏的作用是一样的，只不过工具栏是把菜单栏中经常用到的编辑功能挑选出来并用图形表示，方便操作。

4. 工作区

界面中的左窗口为解决方案资源管理器窗口，窗口中又包含外部依赖项、头文件、源文件和资源文件。

5. 输出窗口

主界面最下侧的窗口为输出窗口，显示程序运行状态，其作用是在编译、链接时显示编译、链接信息。

1.7.2 利用开发环境编写程序

Dev C++编辑流程通常包括新建项目、编写代码和资源文件、编译和调试，直到完成所需要的应用程序。下面结合一个简单的程序进行详细说明。

1. 新建源程序

Dev C++ 支持单个源文件的编译，如果你的程序只有一个源文件（初学者基本都是在单个源文件下编写代码)，那么不用创建项目，直接运行就可以；如果有多个源文件，才需要创建项目。

（1）如图 1.5 所示，从主菜单选择“文件”→“新建”→“源代码”即可。

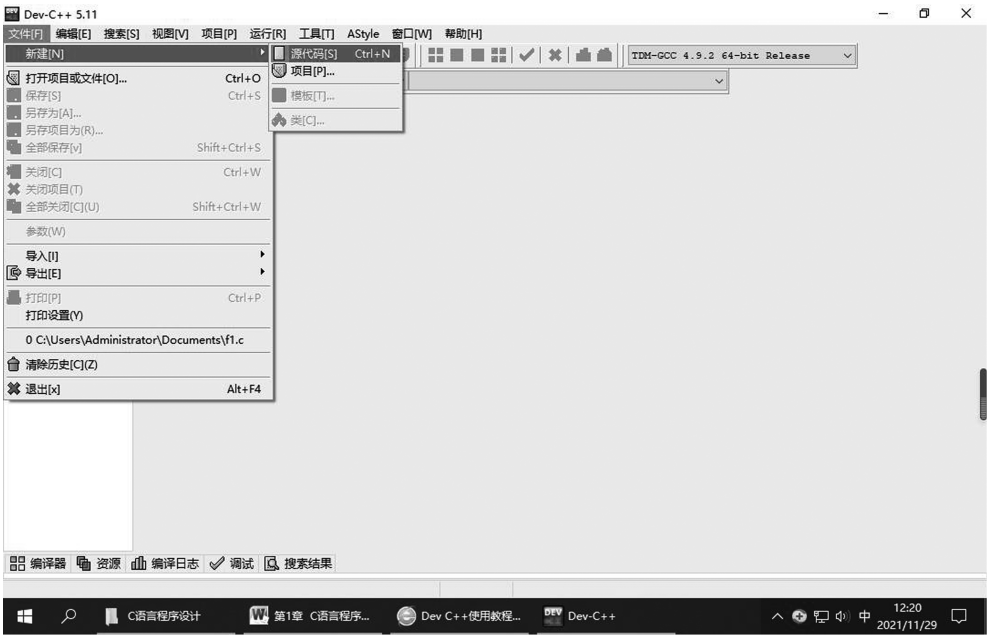

图 1.5　新建源程序

（2）输入源程序代码（如图 1.6 所示）。Dev C++编辑功能与一般编辑器类似，这里不再详述。

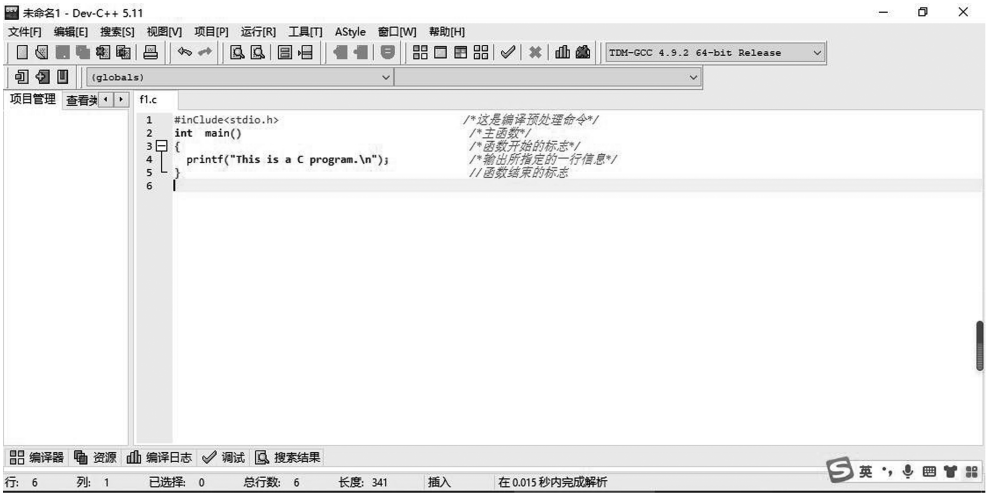

图 1.6　输入源程序代码

【说明】必须在英文输入环境下编辑程序。

（3）保存源程序文件。从主菜单选择“文件”→“保存”，输入文件名，选择存放文件目录、文件类型等，如图 1.7 所示。

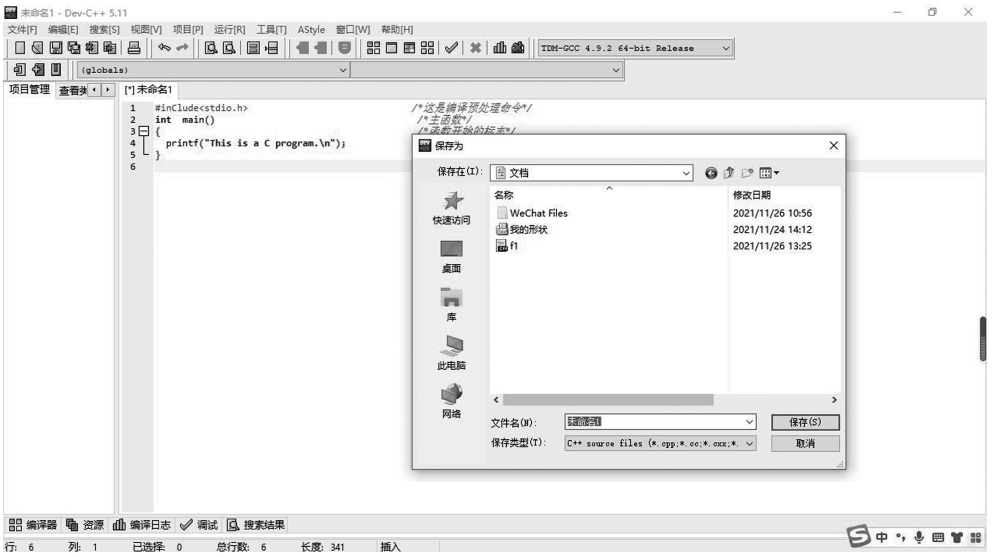

图 1.7　保存源程序文件

1.7.3　预处理、编译、链接程序

从主菜单选“运行”→“编译”或直接按快捷键“Ctrl+F9”，可以一次性完成程序的预处理、编译和链接过程。如果程序中存在词法、语法等错误，编译过程会失败，编译器将会在屏幕右下角的“Compile Log”标签页中显示错误信息，如图 1.8 所示。

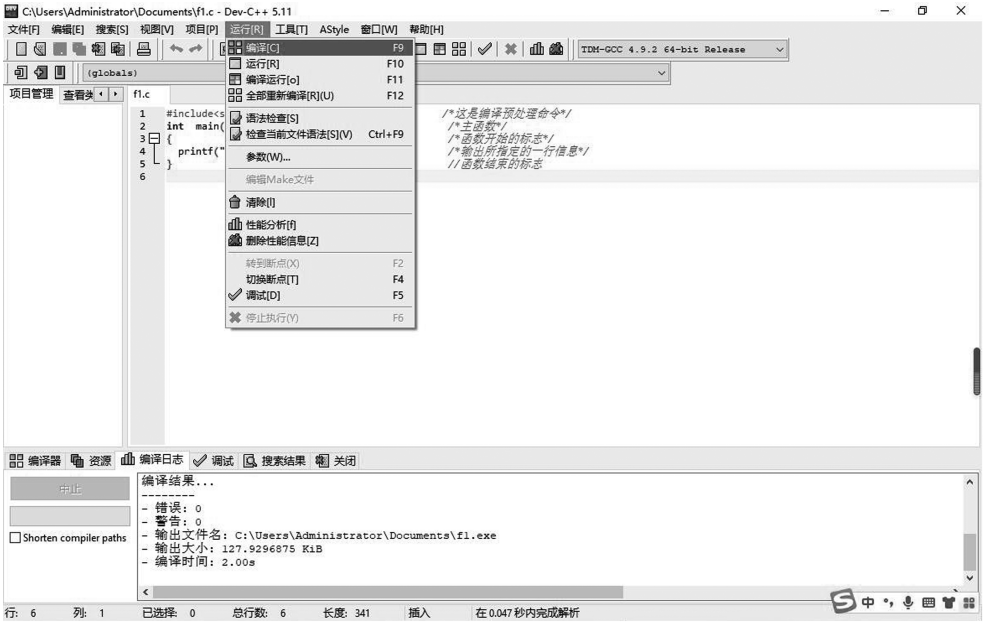

图 1.8　显示错误信息界面

1.7.4 运行程序

直接在 Dev C++环境下从主菜单选“运行”→“运行”或按快捷键“Ctrl+F10”运行程序，如图 1.9 和图 1.10 所示。

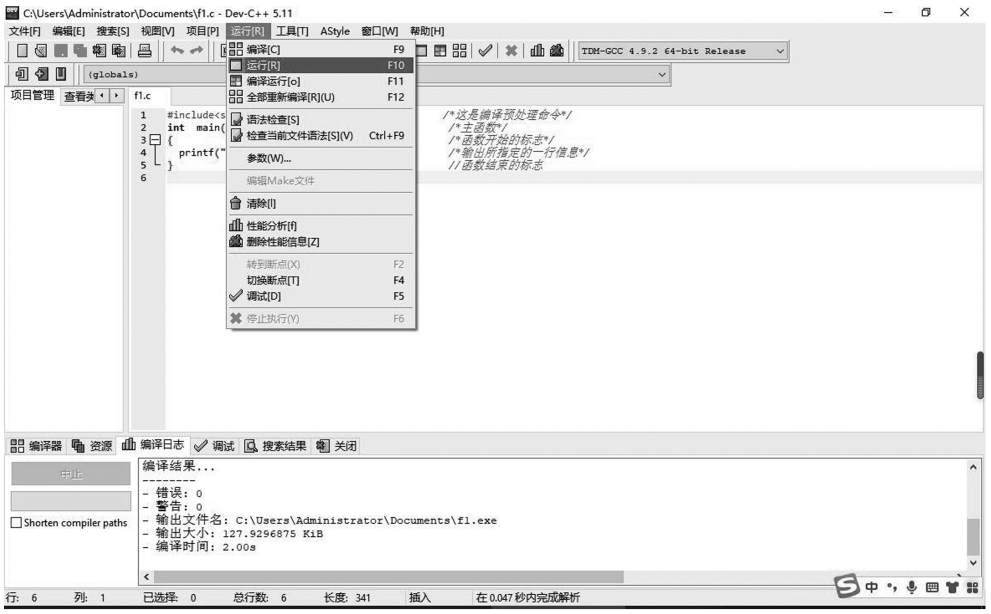

图 1.9　运行程序 1

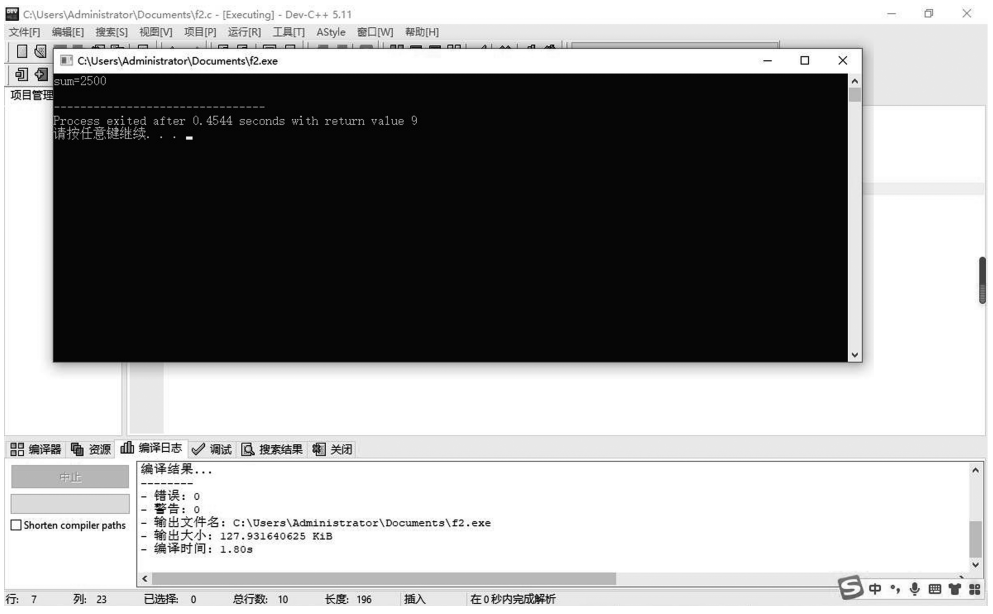

图 1.10　运行程序 2

在 Dev C++环境下，为了查看程序运行结果，需要在 main( ) 函数结束前增加一条语句：“system("PAUSE");” 或 “system("pause");”，这样程序运行到该语句时，结果显示页面将会停留，让大家有时间看程序的输出结果。否则结果显示面页将会一闪而过，当然需要在程序开头加上预处理指令#include<windows. h>。

习 题 1

一、选择题

1. 一个 C 语言程序的执行是（　　）。
   A. 从程序的 main( ) 函数开始，到 main( ) 函数结束
   B. 从程序的第一个函数开始，到本程序的最后一个函数结束
   C. 从程序的 main( ) 函数开始，到本程序的最后一个函数结束
   D. 从程序的第一个函数开始，到本程序的 main( ) 函数结束
2. 以下叙述正确的是（　　）。
   A. 在 C 语言程序中，main( ) 函数必须位于程序的最前面
   B. C 语言程序的每行中只能写一条语句
   C. C 语句以 “;” 结束
   D. 在对一个 C 语言程序进行编译的过程中，可发现注释中的拼写错误
3. 以下叙述不正确的是（　　）。
   A. 一个 C 语言程序可由一个或多个函数组成
   B. 一个 C 语言程序必须包含一个 main( ) 函数
   C. C 程序的基本组成单位是函数
   D. 在 C 语言程序中，注释说明只能位于一条语句的后面
4. C 语言规定，在一个 C 语言程序中，main( ) 函数的位置（　　）。
   A. 必须在最开始　　B. 必须在系统调用的库函数的后面
   C. 可以任意　　D. 必须在最后
5. 用 C 语言编写的代码程序（　　）。
   A. 可立即执行　　B. 是一个源程序
   C. 经过编译即可执行　　D. 经过编译解释才能执行
6. 下列叙述中正确的是（　　）。
   A. C 语言源程序不必通过编译就可以直接运行
   B. C 语言中的每条可执行语句最终都将被转换成二进制的机器指令
   C. C 语言源程序经编译形成的二进制代码可以直接运行

D. C 语言中的函数不可以单独进行编译

二、编程题

1. 编写一个 C 语言程序，运行时输出一行语句：

   Hello World!

2. 模仿例 1.4，编写一个 C 语言程序，计算并输出 1+2+3+…+100 的值。