第 1 章　计算机和 C 语言

章节导学

（本章视频）

计算机改变了世界，参与者有用户、计算机、程序员和 C 语言。

用户借助软件使用计算机，而软件由程序员开发。用户通常是输入数据后查看输出结果，并不关注处理过程。程序员先与用户沟通获得需求；然后设计算法，即设计一系列解决问题的指令；最后实现算法，即把指令翻译成计算机能执行的命令。计算机执行命令，从而满足用户的需求。可用自然语言、伪代码、流程图等多种方式描述算法，但需要用编程语言实现算法。C 语言是经典的、面向过程的编程语言，直观地体现了计算机的特点，特别适合初学者学习。在商业开发中，需求分析师获得需求，算法工程师设计算法，程序员实现算法。

程序员眼中的计算机由五大部件组成，他们设计算法指挥计算机的组成部件来完成任务。计算机的五大组成部件分别是输入设备（如键盘）、存储器、运算器、输出储备（如显示器）和控制器。输入设备用于输入数据；存储器用于存储数据；运算器用于加工数据；输出设备用于输出信息；控制器用于执行指令，指挥各部件协同工作。

计算机中的组成部件有对应的 C 语言语法单位。输入设备对应 scanf 函数，输出设备对应 printf 函数，内存储器中的存储单元对应变量，运算器对应操作符命令，控制器分析、执行 C 语言命令。

本章用一个求两个整数的和的示例简单分析了算法的设计与实现。

C 语言语句由程序员编写并用于指挥计算机，其中的字符（串）多为 C 语言的标记，被称为标识符。常见的标识符有关键字、变量和函数。不是标识符的普通字符（串）需被写作'a'（"abc"）。语句中的命令有关键字、函数和操作符，而命令的操作数（命令的对象）为变量、有返回值的函数调用和形如 32 的字面量。

函数是完成特定功能的一系列指令的集合，由程序员定义，是 C 语言中的自定义命令。利用函数无须编码即可实现特定功能，可以极大地提高编程效率。C 语言程序从 main 函数开始执行，因此编程时需定义 main 函数。

分析 C 语言程序时，不要急于查看程序的运行结果，应找出程序中每条语句的命令与操作数，尝试人工执行语句，仔细分析每条语句的作用和语句间的联系，并得到程序最终的运行结果。多上机编程是学好 C 语言的必由之路，只有实践才能出真知，并且在理论指导下的实践最有效，因此一定要养成人工执行代码的习惯。

本章讨论

1．分析下面的程序。

```
#include <stdio.h>
int sum(int x, int y){
    int z;
```

```
    z = x + y;
    return z;
}
void main ( ){
    int a, b, c;
    scanf("%d%d", &a, &b);
    c = sum(a, b);
    printf("%d + %d = %d\n",a, b, c);
}
```

这个程序也可以求出用户输入的两个整数的和，但与例 1-1 中的程序相比，两者有何区别？

求两个整数的和时，sum 函数与例 1-1 中的程序有何不同？

有读者觉得没必要定义求两个整数和的 sum 函数，你的看法是什么呢？

2．有返回值的函数和无返回值的函数在使用上有何区别？

3．例 1-1 能求出两个任意大的整数的和吗？

4．C 语言中能定义一个求两个数的和的函数吗？

5．讨论 C 语言标准和 C 语言编译器对 C 语言初学者的影响。

下面以求两个整数的和为例，分析用户、计算机和程序员三者之间的关系。

1.1 用户、计算机和程序员

求两个整数的和的程序的运行过程如图 1-1 所示。

图 1-1 两个整数的和的程序的运行过程

程序运行时，计算机会先显示一行提示信息，得到用户输入的整数后，求出和，再把计算结果输出到显示器上。用户通过键盘输入两个整数，然后从显示器上“得到”这两个整数的和。计算机是只会执行命令的机器，没有程序，计算机完不成求和的任务。用户使用计算机时，通常只会运行程序，不会编写程序。程序员是联结用户与计算机的纽带。程序员的工作就是根据用户的需求给计算机设计加工、处理数据的步骤，并把这些步骤翻译成计算机能够“理解”并执行的命令。

程序员眼中的计算机由输入设备、存储器、运算器、输出设备和控制器五大部件组成，其各部分关系如图 1-2 所示。

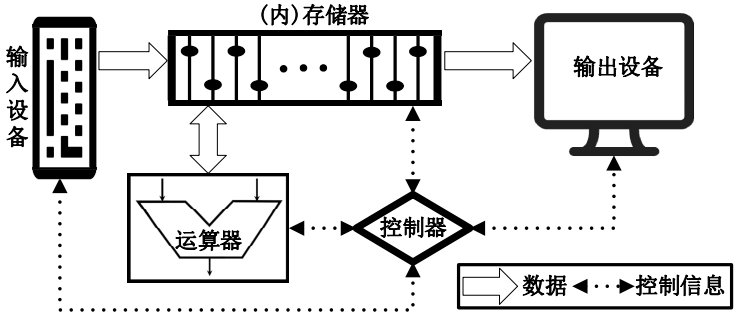

图 1-2 计算机的五大组成部件

输入设备（如键盘）用于向计算机中输入数据；存储器用于存储数据，它分为内存储器和外存储器两类，用户输入的数据在内存储器中存储；运算器用于加工、处理数据；输出设备（如显示器）用于显示信息；控制器用于执行指令，指挥各部件协同工作。计算机与工厂类似：输入设备类似于向工厂输送原料的运输设备，存储器类似于工厂的仓库（存放原料，即加工后的半成品、成品等），运算器类似于工厂的加工车间，输出设备类似于工厂的产品展示中心，控制器类似于工厂中操控生产流程的调度。

设计求和步骤时，程序员使用计算机的五大组成部件。解决问题的一系列命令又称算法。可用如下算法求两个整数的和。

第一步：在显示器上提示用户输入两个整数。

第二步：获得用户输入的数据，并将其存储到内存中。

第三步：运算器求和，并把计算结果转存到内存中。

第四步：在显示器上输出计算结果。

1.2 C 语言、计算机和程序员

C 语言是编程语言，计算机能够“理解”并执行 C 语言命令（语句）。程序员需要把 1.1 节用自然语言描述的求和算法翻译成 C 语言语句。

第一步：在显示器上输出图 1-3 所示的提示信息。

C 语言中用 printf 函数控制输出设备，使用该函数可以“命令”计算机在输出设备上显示指定的信息。用 C 语言语句 printf("请输入两个整数：\n");就可以在显示器上该程序的运行窗口中显示图 1-3 所示的信息。函数调用的语法为函数名加一对圆括号( )。printf 函数会在光标指示的位置原样输出一对双撇号中的数据，其中的\n 表示回车键，输出\n 时相当于按下回车键，光标会移动到下一行的第一列。

第二步：获得用户输入的数据，并将其存储到内存中。图 1-4 显示了用户可能输入的数据。

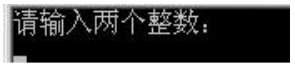

图 1-3　在显示器上输出提示信息

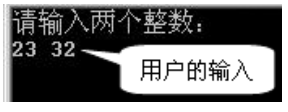

图 1-4　用户可能输入的数据

C 语言中用 scanf 函数控制输入设备，使用该函数可以让计算机获得用户通过键盘输入的数据。执行 scanf 函数时，程序通常会被暂停运行，等待用户输入。当用户以按下回车键的方式表示输入完成后，计算机就会获得用户输入的数据。

内存中的存储单元用于存储数据，使用 scanf 函数时，需明确存放输入数据的存储单元。计算机中使用地址标识存储单元。如果把存储单元比作房间，地址就是房间号。C 语言中用变量标识内存中的存储单元，变量的名字可以是简单易记的字符（串），如 x、flag 等。借助变量而非地址使用存储单元，极大地提高了程序的可读性。

存储单元分类型，常见的有存放整数的整型存储单元、存放小数的浮点型存储单元和存放字符的字符型存储单元。整型存储单元不能存放小数。C 语言中用关键字申请不同类型的存储单元。关键字是 C 语言中具有特定意义的字符串，也被称为保留字。C 语言关键字表参见附录一。关键字是 C 语言的命令。关键字 int 与整型存储单元或整数相关，关键字 float 与浮点型存储单元或小数相关，关键字 char 与字符型存储单元或字符相关。

变量只有在被定义之后才能使用。语句 int i;定义了一个整型变量 i，其中关键字 int 命令计算机在内存中准备一块整型的存储单元与变量 i 关联。程序中借助变量 i 就可以使用该整型存储单元。语句 i = 5;可以让计算机把整数 5 存储到变量 i 的存储单元中。存储单元存储整数 5 后，整型变量 i 的值就变成了 5。语句中的=在 C 语言中不是等号而是赋值号，是 C 语言的操作符命令。这条语句可读作“变量 i 赋值为 5”。C 语言操作符表参见附录五。

需向计算机申请两个整型存储单元来存储用户输入的两个整数，即定义两个整型变量。首先用语句 int a, b;定义两个整型变量 a 和 b，然后翻译成 scanf("%d%d", &a, &b);。一对双撇号中的%d 表示一个整数，两个%d 表示两个整数。变量 a 前面的操作符&是取地址命令，&a 的值是变量 a 的存储单元地址。这条语句命令计算机获得用户输入的两个整数，并将其存储到变量 a 和 b 中。如图 1-4 所示，当用户按下回车键确认输入完成后，23 和 32 就被存储到变量 a 和 b 中，即变量 a 的值变成了 23，变量 b 的值变成了 32。使用 scanf 函数时，需明确输入数据的类型、个数及存储输入数据的变量的地址。

第三步：运算器求和，并把计算结果转存到内存中。

用户输入的整数存储在整型变量 a 和 b 中，求和就变成了求变量 a 与 b 的和。C 语言中用操作符命令指挥运算器处理数据，+操作符用于命令求和，a + b 就可以让运算器求出变量 a 与 b 的和。用语句 int c;定义一个整型变量 c 来存储计算结果，这一步可翻译成 c = a + b;，读作“变量 c 赋值为变量 a 与变量 b 的和”。其中+和=（赋值号）是操作符命令，a、b 和 c 是变量。+使运算器求出变量 a 与 b 的和，=命令计算机把计算结果转存到变量 c 中。

第四步：在显示器上输出计算结果。

printf 函数控制输出设备，利用语句 printf("和为%d", c);可以把变量 c 的值输出到显示器上该程序的运行窗口中，如图 1-5 中最后一行所示。与 scanf 函数中的%d 相同，这条语句中的%d 也表示一个整数。遇到%d 时，printf 函数会用逗号后面对应位置上的整数替换%d。利用语句 printf("和为 c ");可以输出“和为 c”。一对双撇号中的 c 是普通字符而非变量。

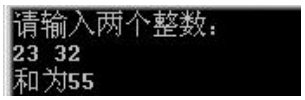

图 1-5　输出结果

综上所述，依次执行下面的 C 语言语句，计算机就可以求出用户输入的两个整数的和。

（1）printf("请输入两个整数：\n");。

（2）scanf("%d%d", &a, &b);。

（3）c = a + b;。

（4）printf("和为%d", c);。

求和算法的实现过程可以得到计算机与 C 语言的对应关系，如图 1-6 所示。

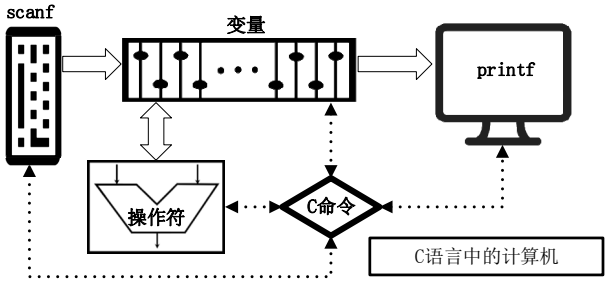

图 1-6　计算机与 C 语言的对应关系

1.3 C 语言自定义命令——函数

1.3.1 使用函数命令

编程求一个整数的绝对值，一些具体的输入和输出如表 1-1 所示。

表 1-1 一些具体的输入和输出

输入和输出	第一次	第二次	第三次	变量
用户可能输入的数据	3	0	−3	n
程序预期输出的数据	3	0	3	n 或−n

可设计如下求整数的绝对值的算法。

第一步：提示用户输入一个整数。

第二步：获得用户输入的数据，并将其存储到内存中。

第三步：求出绝对值，并把它转存到内存中。

第四步：在显示器上输出绝对值。

想要实现算法，需要把算法中的每一步翻译成 C 语言语句。先用语句 int m, n;定义两个整型变量。第一步可翻译成：printf("请输入一个整数：\n");。第二步可翻译成：scanf("%d", &n);。第三步中求绝对值就是求整型变量 n 的绝对值，用什么命令求变量 n 的绝对值呢？

运算器会算加法，但不会求一个整数的绝对值。C 语言中既没有用于求绝对值的操作符命令，也没有与绝对值相关的关键字命令。若找不到求绝对值的命令，则上面求绝对值的算法不可行。

C 语言中有一个 abs 函数，可用于求整数的绝对值，利用函数调用 abs(−3)就能求出整数−3 的绝对值。计算机执行 abs(−3)的结果是整数 3，即−3 的绝对值。因此，第三步可翻译成：m＝abs(n);。这条语句用函数 abs 求出整型变量 n 的绝对值，并将绝对值赋值给整型变量 m。第四步可翻译成：printf("%d 的绝对值为%d", n, m);。

abs 函数是由程序员定义的。程序员先设计算法，细化求一个整数的绝对值的步骤，使每个步骤都能被翻译成计算机可以直接执行的关键字命令或操作符命令。C 语言函数由一系列完成特定功能的 C 语言语句组成，执行 C 语言函数时，计算机依次执行函数中的语句，从而得到结果。函数是 C 语言中的自定义命令。

1.3.2 函数定义

求一个整数的绝对值的算法有点复杂，为了介绍定义函数的语法，把求两个整数的和的算法定义成一个名为 sum 的函数。

函数的功能体现为由输入得到输出，函数的输入被称为参数，函数的输出被称为返回值或函数值。使用 sum 函数求和时，需提供两个整数作为参数，即待加工数据。将这两个参数放在函数调用的一对圆括号中，并用逗号分隔，如用 sum 函数求整数 5 与 3 的和时，调用函数的形式为 sum(5, 3)，该函数调用的执行结果应为整数 8，即 5+3 的和。

sum 函数有两个整型的参数，有一个整型的返回值，其第一行的定义为 int sum(int x, int y)。第一个关键字 int 定义了一个匿名的整型存储单元，用于存储函数的返回值。函数执行完毕后，函数的使用者会在这个约定的匿名整型存储单元中得到函数的执行结果。标识符 sum 是函数

名。一对圆括号是函数的标志，圆括号中用逗号分隔的 int x 和 int y 定义了两个整型变量 x 和 y，也就是函数的参数，用于存储两个加数。

函数定义的第一行又称函数的首部，格式为：函数返回值类型　函数名(函数的参数)。函数首部中的参数是形式从参数，简称形参。有几个输入数据，函数就要定义几个形参，形参之间用逗号分隔。函数的首部清晰地表明了函数返回值（处理结果）的类型、函数的名称、函数形参（输入值）的个数及其类型。

函数的定义由首部和函数体两部分组成。函数体用一对花括号界定。函数体中由实现算法的语句来完成从输入到输出的处理。sum 函数的函数体中需求出输入数据（形参 x 和形参 y）的和，sum 函数的完整定义如下。

```
int sum(int x, int y)
{
     int z;
     z = x + y;
     return z;
}
```

函数体中语句 int z;定义了一个整型变量 z，用于存储和。变量定义的一般形式如下。

```
数据类型变量列表;
```

数据类型可以是关键字 int、float 或 char，即整型、浮点型或字符型。变量列表由一个变量名称或由逗号分隔的多个变量名称构成，如变量名 1，变量名 2，…，变量名 n。变量名是 C 语言标识符。标识符是一个由大写或小写（英文）字母、数字或下画线组成的字符串，且不能以数字开头。在 a、X2、_sum、2x、a#s 中只有前三个是合法的标识符。

语句 z＝x＋y;中，+使运算器求出形参 x 与形参 y 的和，=（赋值号）使计算机把和存储到整型变量 z 中，即整型变量 z 的值为 x 与 y 的和。

语句 return z;中的 return 是 C 语言关键字，用于结束函数的执行并返回函数值。执行这条语句时，无论后面是否还有 C 语言语句，函数都将立即结束执行，并把整型变量 z 的值存储到 sum 函数首部中定义的用于存储函数返回值的匿名变量中。

定义 sum 函数之后，就可以使用 sum 函数命令求两个整数的和了。sum 函数的直观表示如图 1-7 所示。

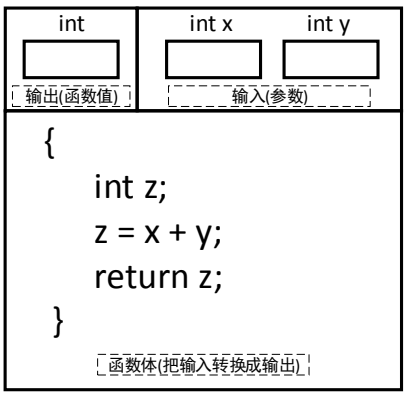

图 1-7　sum 函数的直观表示

C 语言函数可以没有返回值，此时函数首部的返回值类型用关键字 void 表示。没有返回值

的函数在函数体中不能使用关键字 return 返回数据，如语句 return 3;，但可以使用语句 return;立即结束函数的执行。没有 return 语句时，函数体中的每条语句会自上而下依次执行，在界定函数体的封闭花括号}处，函数结束执行。

C 语言函数也可以没有输入值。没有输入值的函数不需要定义形参。函数首部的一对圆括号中为空或用关键字 void 表示没有形参。没有输入值和返回值的 C 语言函数的首部为 void f( )或 void f(void)。

1.3.3 函数调用

使用函数又称调用函数。调用函数时需向函数提供具体的输入值，即待加工数据。用 abs 函数求−3 的绝对值时，−3 就是具体的输入值，函数调用 abs(−3)命令计算机求出−3 的绝对值。函数的输入值被称为参数，它与函数首部用于存储具体输入值的形参不同，函数调用中的具体的输入值被称为实际参数，简称实参。函数调用 sum(5, 3)中的两个实参为 5 和 3。

函数被调用执行时，实参先向形参赋值，函数体中的语句再依次执行。函数调用 sum(5, 3)执行时，将实参 5 赋值给形参 x，实参 3 赋值给形参 y，然后 sum 函数体的语句依次执行。使用语句 z = x + y;先求出形参 x 与形参 y 的和为 8，再把 8 存储到变量 z 中。语句 return z;执行时，把变量 z 的值存储到匿名的整型存储单元中，结束函数的执行。函数调用 sum(5, 3)的执行结果就是匿名整型存储单元中的整数 8。函数调用执行完毕后，函数的使用者会在约定的匿名整型存储单元中得到函数的返回值 8。sum(5, 3)+3 就是 8+3，printf("%d", sum(5, 3))就是 printf("%d", 8)。sum(5, 3)的执行过程如图 1-8 所示。

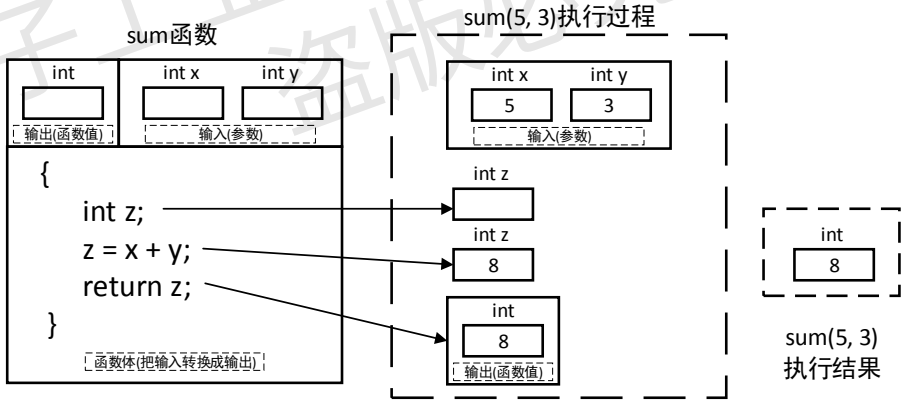

图 1-8 sum(5, 3)的执行过程

函数调用的形式为：函数名(实参列表)。实参列表由逗号分隔的多个实参构成，实参的个数与类型必须和函数定义中的形参相匹配。函数没有形参时，实参列表为空，且一对圆括号不能省略。若实参中含有 C 语言命令，则实参向形参赋值之前，相关命令会被执行，实参最终是一个具体的值。函数调用 sum(3 + 2,3)的实参 3 + 2 中有加号命令，其执行时，计算机会先求出 3 + 2 的和为 5，再执行函数调用 sum(5, 3)。函数调用的执行过程为：首先对实参求值；然后用实参给形参赋值；最后自上而下依次执行被调用函数的函数体，直到遇到 return 语句或界定函数体的封闭花括号}处才结束。函数调用 sum(sum(2, 3), 5)中，一个实参为 sum(2, 3)，另一个实参为 5。实参 sum(2, 3)中有函数命令 sum，计算机会执行函数调用 sum(2, 3)，它的返回值为整数

5，故原函数调用相当于 sum(5, 5)。函数调用 sum(sum(2, 3), 5)的返回值为整数 10。

1.3.4 main 函数

C 语言程序由函数构成，函数是 C 语言程序的基本组成单位。C 语言规定，程序中必须有且仅有一个函数名为 main 的函数。main 函数即主函数，C 语言程序从 main 函数开始运行，main 函数执行完毕，程序运行结束。从形式上看，编写程序就是定义一个函数名为 main 的函数。为方便初学者快速入门，main 函数的首部可定义为 void main()。将求和的算法放到 main 函数的函数体中，就可以得到求两个整数的和的 C 语言程序。

例 1-1 求两个整数的和的程序。

```
void main( )
{
      int a, b, c;
      printf("请输入两个整数：\n");
      scanf("%d%d", &a, &b);
      c = a + b;
      printf("和为%d", c);
}
```

这个程序由 3 个函数组成，分别是 main 函数、printf 函数和 scanf 函数。程序运行时，main 函数被调用执行，函数体中的每条语句自上而下依次执行。首先定义 3 个整型变量，然后调用 printf 函数输出提示信息，接着调用 scanf 函数获得用户输入的两个加数，并求和，最后调用 printf 函数输出和，在界定函数体的封闭花括号}处结束执行。main 函数执行完毕，程序也就运行结束了。

程序（main 函数）的使用者是用户，用户借助键盘输入两个加数，通过显示器查看输出结果。自定义命令 sum 函数的使用者是程序员，调用函数时直接给出两个加数，使用约定存储单元中的函数返回值。

1.4 “懂”C 语言的计算机

1.4.1 虚拟的 C 语言计算机

计算机是机器，只“懂”机器语言，不“懂”C 语言。机器语言使用二进制代码表示指令和数据。计算机能直接执行用机器语言编写的程序，但机器语言程序难以编写，可读性也非常差。C 语言是高级语言，表达方式接近于人类的自然语言，如用 3 + 2 命令计算机求 3 与 2 的和，但计算机不能识别这样的命令。用高级语言编写的程序被称为源程序。C 语言源程序只有在被翻译成由机器语言组成的可执行程序之后，计算机才能执行，而这个“翻译”的任务通常由编译程序（编译器）完成。

如果把编译器及其依赖的操作系统也看作计算机的组成部分，就可以说计算机能够“理解”和运行 C 语言程序了。一个装有 C 语言编译器和相关操作系统的计算机就是一个虚拟的 C 语言计算机，它可以执行用 C 语言编写的源程序，能“听懂”C 语言命令。借助于其他编译器，计算机还可以变成“懂”其他语言的虚拟计算机。虚拟计算机的层次结构如图 1-9 所示。

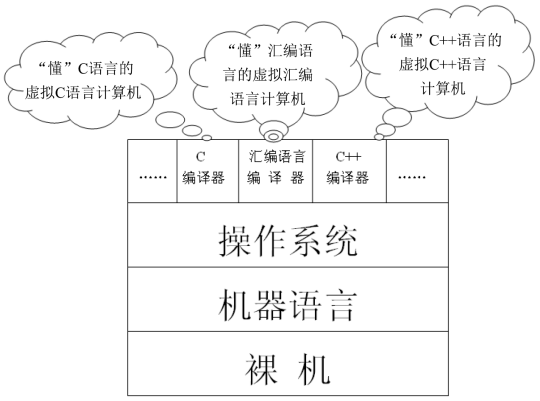

图 1-9 “懂”各种语言的虚拟计算机

1.4.2 VC6.0 编译程序

本书采用 VC6.0 编译 C 语言程序。VC6.0 是 Visual C++ 6.0 的简称，是在 Windows 操作系统中进行应用程序开发的 C/C++编译系统。VC6.0 是一个集成开发环境，包含许多独立的组件，如编辑器、编译器、调试器，以及各种各样为开发 Windows 下的 C/C++程序而设计的工具。编译系统简称编译器。VC6.0 是一个典型的“Windows 风格”的程序，图 1-10 显示了 VC6.0 集成开发环境界面。

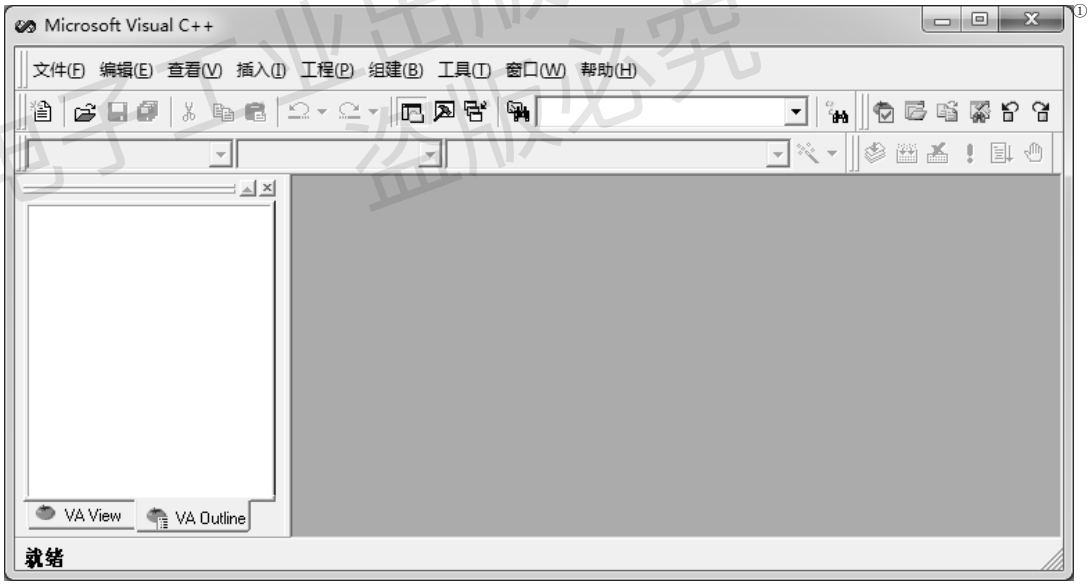

图 1-10 VC6.0 集成开发环境界面

用 VC6.0 编译程序的步骤如下：

（1）运行 VC6.0，单击“文件”（File）菜单中的“新建”（New）命令或按 Ctrl+N 组合键，会弹出“新建”（New）对话框，其中的“工程”（Project，也可称为项目）选项卡默认被选定，如图 1-11 所示。

① 为了学习方便，此处用中文界面说明。

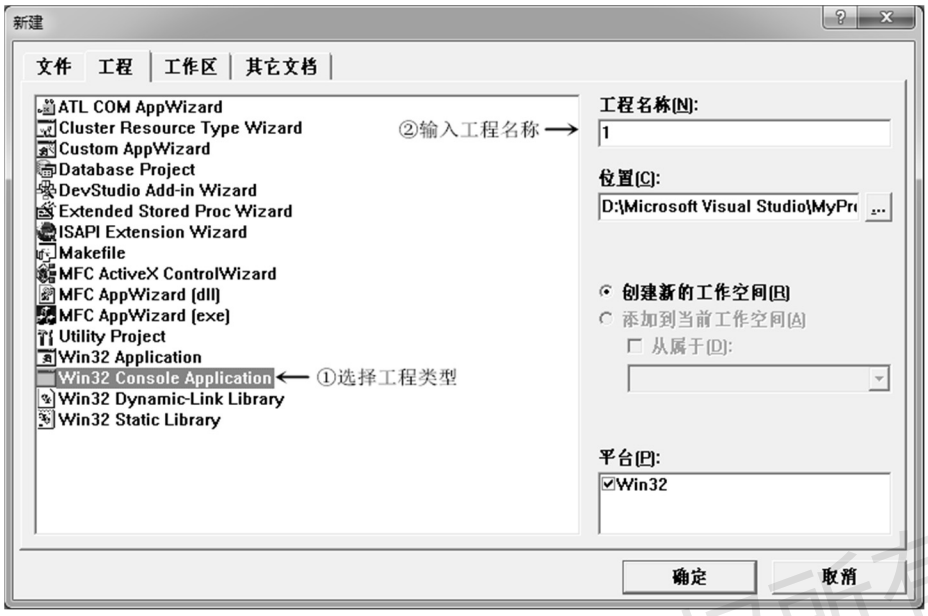

图 1-11　“新建”（New）对话框中的“工程”（Project）选项卡

在 VC6.0 中，源程序作为工程的一部分，编程时要先建一个工程。

工程类型选择 Win32 Console Application（Win32 控制台应用程序）。控制台应用程序使用字符用户界面 CUI（Character User Interface），其特点是以键盘输入命令的方式实现用户与计算机的交互。早期的操作系统（如 DOS）多采用这种模式。现在流行的图形用户界面 GUI（Graphical User Interface）的特点是利用鼠标借助窗口、菜单等方便、快捷地进行人机交互。支持 GUI 的程序较复杂，初学者常常先学习编写控制台应用程序。

在接下来弹出的图 1-12 和图 1-13 所示的对话框中，分别单击“完成”和“确定”按钮，VC6.0 会呈现出图 1-14 所示的界面。

图 1-12　“Win32 Console Application-步骤 1 共 1 步”对话框

图 1-13　“新建工程信息”对话框

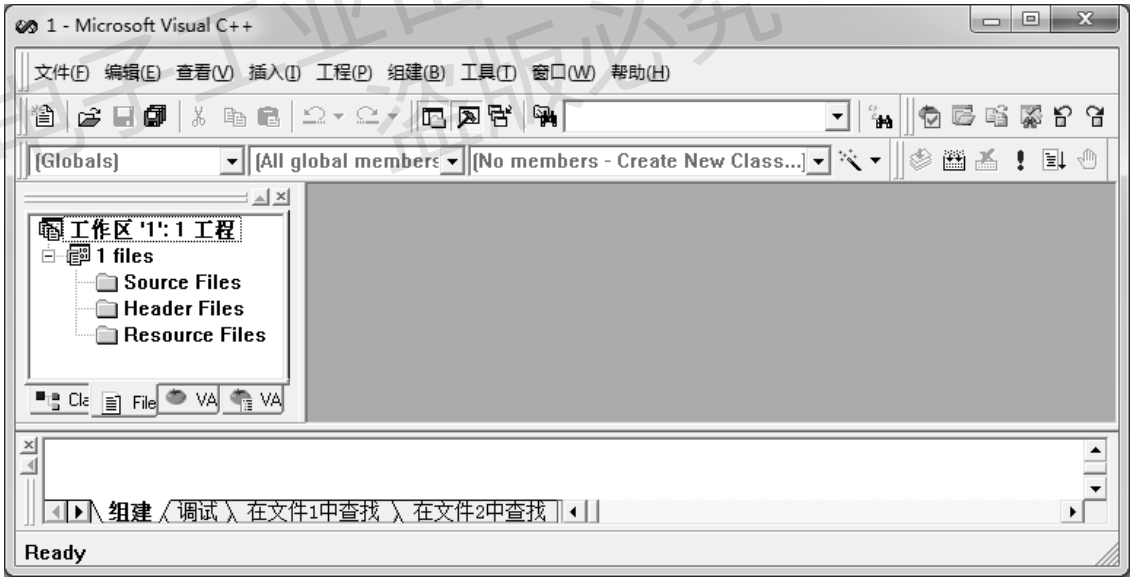

图 1-14　新建一个名为 1 的工程后，VC6.0 的界面

（2）再次单击“文件”（File）菜单中的“新建”（New）命令或按 Ctrl+N 组合键，第二次弹出“新建”（New）对话框，“文件”（Files）选项卡默认被选定，如图 1-15 所示。

VC6.0 默认的源文件类型为 C++，扩展名为 cpp，而 C 语言源文件的扩展名为 c，初学者不必区分。单击“确定”按钮后，VC6.0 的编辑器就自动打开了新建的 C 语言源文件，等待输入，如图 1-16 所示。

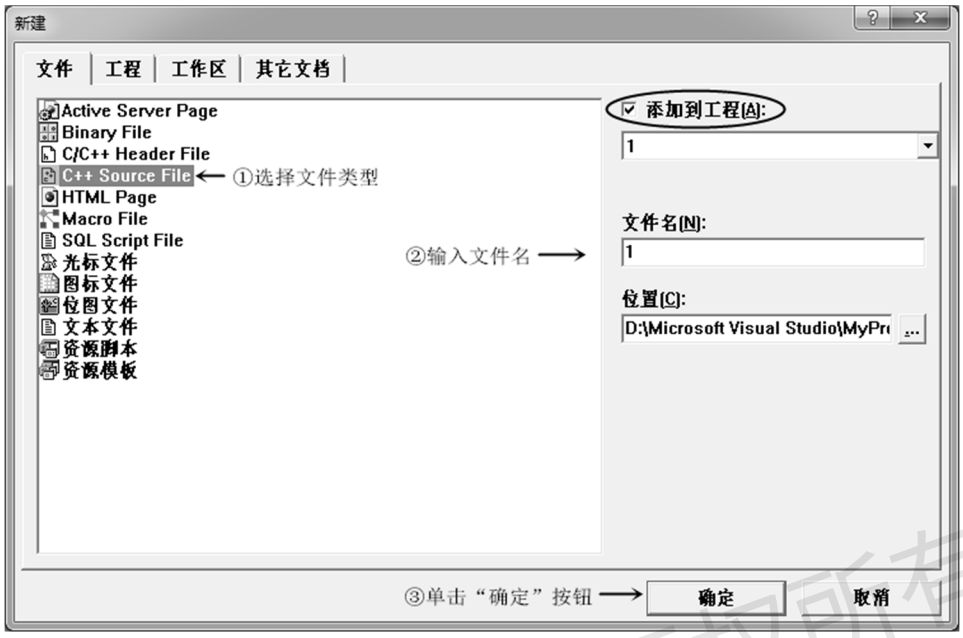

图 1-15　“新建”（New）对话框中的“文件”（Files）选项卡

图 1-16　VC6.0 中编辑器等待输入时的界面

（3）在编辑器中输入例 1-1 中的程序，为了提高输入效率，将函数体中的左花括号{放在了函数首部的末尾。

```
void main( ){
int a, b, c;
      printf("请输入两个整数:\n");
      scanf("%d%d", &a, &b);
      c = a + b;
      printf("和为%d", c);
}
```

编译、运行这个程序时，会出现图 1-17 所示的错误。

```
error C2065: 'printf' : undeclared identifier
error C2065: 'scanf' : undeclared identifier
```

图 1-17　编译、运行例 1-1 中的程序时出现的错误

错误信息提示 printf 和 scanf 是没有定义的标识符。

与变量类似，函数也必须先定义后使用。printf 函数和 scanf 函数是 C 语言中的自定义命令，需由程序员定义，但 printf 函数和 scanf 函数在 C 语言编译器中已经被定义好了，这类函数又称库函数。库函数在特定的文件中被定义，使用库函数时，只需知道包含库函数的文件名，用“#include <文件名>”的方式就可以把库函数的定义复制到源文件中。“#include <文件名>”的作用是：用指定文件的内容替换该行命令，从而把指定文件的内容合并到当前的源程序文件中。

C 语言编译器提供了许多库函数，且同类功能的库函数在一个文件中被定义。C 语言常用库函数的首部及功能说明见附录四。printf 函数和 scanf 函数在 stdio.h（标准输入输出）文件中被定义，在例 1-1 中加入#include <stdio.h>，即可得到 printf 函数和 scanf 函数的定义。#include <stdio.h>没有分号，且多在首行。求两个整数的和的完整程序如下所示。

```
#include <stdio.h>
void main( ){
      int a, b, c;
      printf("请输入两个整数：\n");
      scanf("%d%d", &a, &b);
      c = a + b;
      printf("和为%d", c);
}
```

库函数的定义实际上由函数声明文件和函数定义文件组成。函数声明文件包含库函数的首部信息，由于它常放在程序的开始部分，所以又称头文件，以.h 为扩展名。库函数定义文件中的内容是编译后的二进制形式，难以阅读和修改。使用库函数时，编译器会自动查找相关库函数的定义文件，只需把头文件复制到源文件中即可。

（4）程序输入完成后，单击“组建”（Build）菜单中的“执行”（Execute）命令或按 Ctrl+F5 组合键，编译并执行程序，如图 1-18、图 1-19 所示。

由图 1-19 可知，VC6.0 不仅把 C 语言源程序编译、链接成了可执行程序，还自动运行了编译后的可执行程序。可执行程序在 Windows 操作系统中的扩展名为.exe。

程序运行时，main 函数体中的语句自上而下依次执行。首先定义 3 个整型变量，然后用 printf 函数在运行窗口中输出一行提示信息，接着用 scanf 函数获得用户输入的两个整数。此时，程序暂停执行，等待用户输入，如图 1-19 所示。

若用户输入“23　32↙”（本书中用↙表示按下回车键），则变量 a 和 b 的值变成了 23 和 32。语句 c＝a＋b;继续执行，求出 a 与 b 的和为 55，并将 55 存储到变量 c 中。最后用 printf 函数在运行窗口中输出“和为 55”。main 函数执行完毕，程序结束运行，运行结果如图 1-20 所示。VC6.0 在程序运行窗口中增添了提示信息“Press any key to continue”，此时按下任意一个键，程序的运行窗口都会自动关闭。

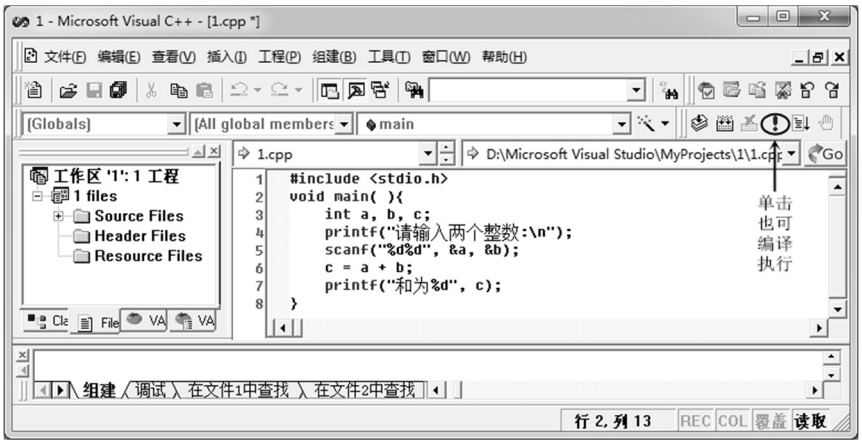

图 1-18　编译并执行程序

图 1-19　程序开始运行

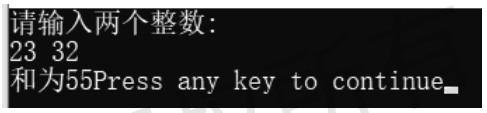

图 1-20　程序的运行结果

输入数据时，用户可以一次性输入用空格分隔的两个整数，如“23　32↙”，也可以先输入“23↙”。由于 scanf 函数需获得两个整数，程序依然暂停运行，等待用户输入另一个整数，如图 1-21 所示。

当用户再次输入“-32↙”后，程序继续运行，如图 1-22 所示。

图 1-21　scanf 需获得两个整数

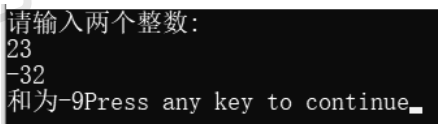

图 1-22　获得两个整数后程序继续运行

继续编写程序时，应先关闭当前的程序（工程），再重复上述过程。单击“文件”（File）菜单中的“关闭工作空间”（Close Workspace）命令，即可关闭当前程序（工程），如图 1-23 所示。

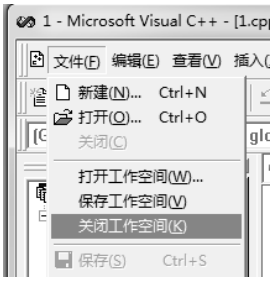

图 1-23　“文件”（File）菜单中的“关闭工作空间”（Close Workspace）命令

例 1-2 求一个整数的绝对值的程序。

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

```
void main( ){
       int m, n;
       printf("请输入一个整数：\n");
       scanf("%d", &n);
       m = abs(n);
       printf("%d 的绝对值为%d", n, m);
}
```

程序中求整数绝对值的库函数 abs 在 math.h（数学库）中定义。

1.5 与虚拟 C 语言计算机深入交流

编程时需特别注意源代码的书写风格，对齐和缩进可以使代码整洁、层次清晰。输入代码时，VC6.0 会自动判断对齐和缩进的位置，在多数情况下，只需在 VC6.0 提示的位置输入代码即可。良好的代码书写风格能提高程序的可读性，可读性好的程序容易理解，不易出错。养成以说出每条语句作用的方式执行源程序的习惯，可以使读者快速提高编程能力。

1.5.1 C 语言语法规则

（1）C 程序书写格式自由，一行内可以写多条语句，一条语句也可以分写在多行上，通常一行只写一条语句。

（2）每条语句或变量定义都以分号“;”结尾，可以把分号看作 C 语言语句的结束标志。只有一个分号的语句也是一条语句，称为空语句，不表示任何命令，仅用于构造程序。include 命令不是 C 语言语句，不能用分号结尾。

（3）变量名或函数名不能使用 C 语言关键字。

（4）在 C 语言中，/*和*/之间的内容是注释。注释用于解释、说明代码，提高程序的可读性，并且会被编译器忽略。下面是一些注释的示例。

```
/*这是单行注释的示例 */
/*
       这是
       多行注释的示例
*/
VC6.0 中单行注释也可用//，如
// VC6.0 中单行注释也可如此。
```

（5）C 语言中使用英文符号（半角符号）。VC6.0 中的汉字（全角符号）只可以出现在一对双撇号（"　"）中或注释中。同时按 Ctrl 键和空格键可以在中、英文输入法之间切换。

（6）使用 scanf 函数时，需明确输入数据的类型、个数及存储对应输入数据的变量的地址，变量前面通常需加一个取地址操作符&，如 scanf("%d%d", &a, &b);。

1.5.2 printf 函数的用法

printf 函数可以把信息输出到程序运行窗口中光标指示的位置。程序运行窗口中闪烁的光标用于指示输入或输出的起始位置，可被称为输入/输出光标。程序开始运行时，该光标位于窗口中的第一行第一列，如图 1-24 所示。

输入/输出光标会自动调整位置，它始终指示下一次输入或输出的起始位置。语句 printf ("Welcome to C!");执行后，光标自动调整了位置，如图 1-25 所示。

图 1-24　输入/输出光标

图 1-25　printf("Welcome to C!");执行后，光标的位置

例 1-3　printf 函数的输出。

```
#include <stdio.h>
void main( ){
      printf("Welcome ");   /*注意空格字符*/
      printf("to C!");
}
```

例 1-3 中第二个 printf 函数的输出会紧接着第一个 printf 函数的输出，程序的运行结果也如图 1-25 所示。

printf 函数会原样输出一对双撇号内的字符，但有两类特殊的字符组合例外。反斜杠\和下一个字符的组合被称为转义序列，它有着特殊的含义。如\"表示一个双撇号"，\\表示一个反斜杠\，\n 在 VC6.0 中表示回车键。遇到字符组合\n，printf 函数会把输入/输出光标移动到下一行的第一列（相当于按下了键盘上的回车键）。语句转义序列\n 在 VC6.0 中表示回车键，如图 1-26 所示。

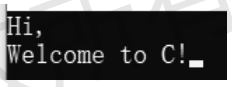

图 1-26　转义序列\n 在 VC6.0 中表示回车键

还有一类特殊的字符组合用于输出数据的占位序列，由百分号%和相邻的字符组成。遇到占位序列时，printf 函数会用对应位置上的数据代替占位序列。每种类型的数据都有相应的占位序列，整数用%d，浮点数用%f，字符用%c，占位序列又称格式字符串。

例 1-4　printf 函数中的占位序列。

```
#include <stdio.h>
void main( ){
      int a = 3;
      int b = 5;
      printf("%d + %d = %d", a, b, 3+5);//注意空格字符
}
```

程序的运行结果如图 1-27 所示。

分析：

在定义变量时，对变量赋值称为变量的初始化。语句 int a = 3;定义了一个整型变量 a，并将整数 3 存储到变量 a 中，变量 a 被初始化为 3。

printf 函数执行时，会先对实参 3 + 5 求值，第三个%d 对应整数 8。

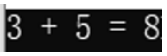

图 1-27　例 1-4 程序的运行结果

1.5.3　用 VC6.0 观察程序的运行过程

例 1-5　分析下面程序的运行过程。

```
#include <stdio.h>
int sum(int x,int y){
   int z;
```

```
    z = x + y;
    return z;
}
void main( ){
    int a, b, c;
    a = 23;
    b = -5;
    c = sum(a, b);
    printf("%d + %d = %d\n",a, b, c);
}
```

分析：

程序由求两个整数的和 sum 函数、main 函数和库函数 printf 组成。程序的运行过程就是 main 函数的执行过程。先定义 a、b 和 c 三个整型变量，再把变量 a 赋值为 23，变量 b 赋值为 −5。语句 c = sum(a, b);中，sum 函数命令先执行，即 main 函数暂停执行，sum 函数开始执行。函数执行时，首先对实参求值，实参为 23 和−5；然后用实参给形参赋值，形参 x 和 y 的值分别为 23 和−5；最后执行函数体。sum 函数先定义一个整型变量 z，再求出形参 x 与 y 的和，即 23 与−5 的和为 18，并将其赋值给变量 z。语句 return z;先将变量 z 的值存储到约定的匿名整型存储单元中，作为函数的返回值，再结束 sum 函数的执行。main 函数从暂停处恢复执行。main 函数先从约定的匿名存储单元中得到 sum 函数的返回值 18，即语句 c = sum(a, b);变成了语句 c = 18;，变量 c 赋值为 18。占位序列被对应的变量值替换后，最后一条语句就变成了 printf("23 + −5 = 18\n");。程序的运行结果如图 1-28 所示。

```
23 + -5 = 18
```

图 1-28　例 1-5 程序的运行结果

下面借助 VC6.0 的调试功能观察这个程序的运行过程。

（1）打开 VC6.0，创建一个名为 5 的工程。在编辑器窗口中输入程序，把光标定位在第 9 行，单击编译工具栏上的手形图标（或按 F9 快捷键），插入断点，如图 1-29 所示。

（2）单击“组建”（Build）菜单中的“开始调试”（Debug）子菜单中的“Go”命令（或按 F5 快捷键），进入调试执行模式，如图 1-30 所示。调试执行时，程序执行到断点处会自动停下，便于观察程序当前的状态，如变量的值。

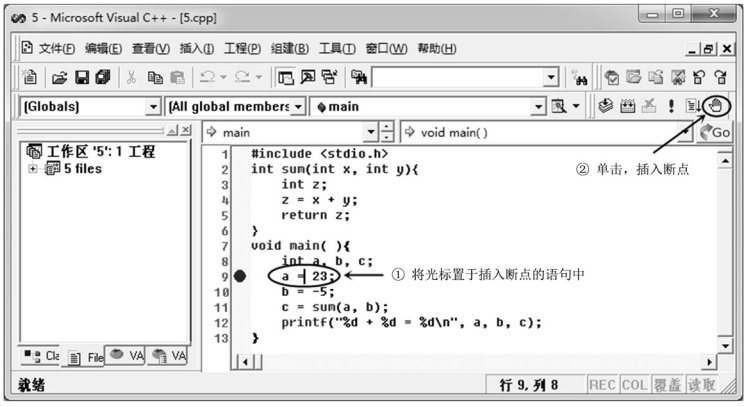

图 1-29　在程序中插入断点

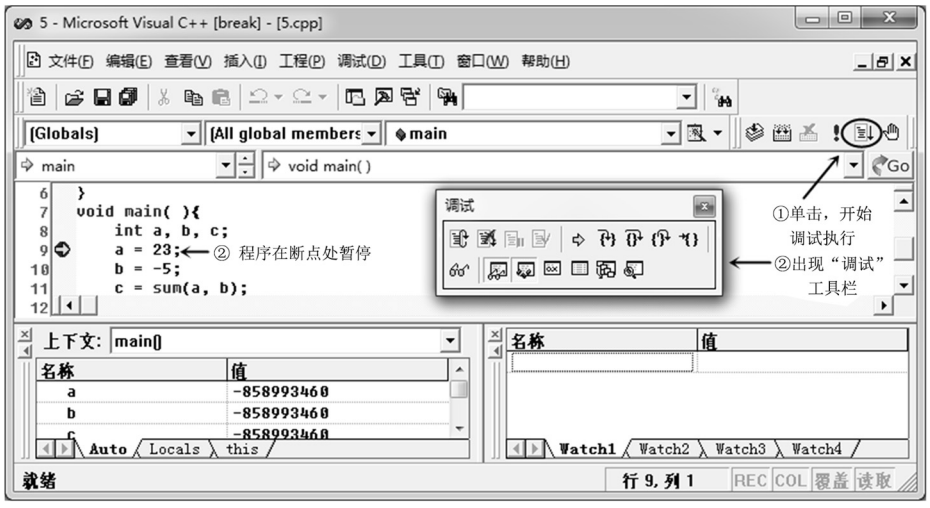

图 1-30　调试执行模式

程序暂停执行后，可以用单步执行调试命令控制程序的执行。单步执行程序时，每次只执行一条语句，执行完一条语句，程序会暂停执行。

图 1-30 中，程序暂停在有断点的第 9 行语句处。由“Auto”窗口可知，程序中整型变量 a、b、c 此时的值均为−858993460。

提示：

① 在 VC6.0 中，程序有两种执行方式：执行（按 Ctrl+F5 组合键）和调试执行（按 F5 快捷键）。遇到含有断点的语句时，如果是“调试执行”，程序会在断点处暂停执行；如果是“执行”，程序会忽略断点。

② 整型变量被定义后，在没有赋值时，值是多少呢？可以通过输出查看，语句为 int a;printf("%d", a);。定义后没有赋值变量，值通常不确定，它与编译器有关，VC6.0 中定义后没有赋值的整型变量的值为−858993460，如图 1-30 所示。

（4）单击“调试”工具栏上的“单步执行”命令（或按 F11 快捷键），执行当前的语句，即第 9 行语句。语句执行后，程序暂停在了下一条语句处，如图 1-31 所示。

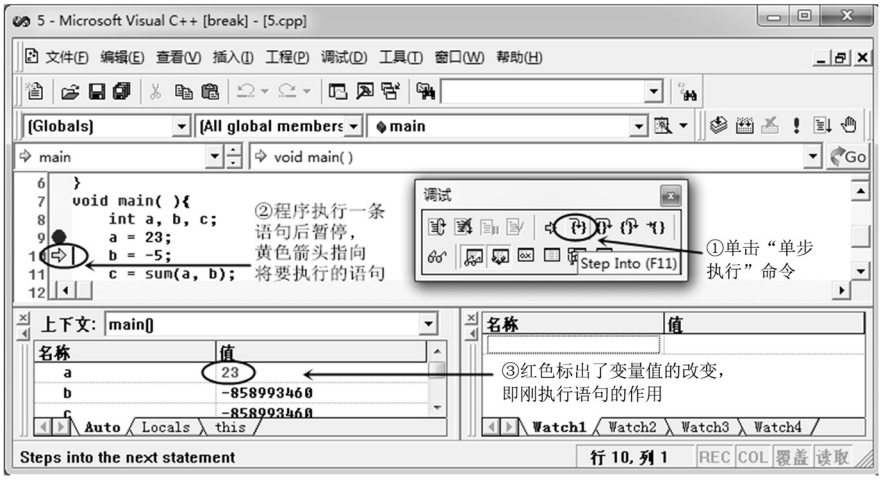

图 1-31　单击“单步执行”命令

由图 1-31 可知，语句 a = 23;执行后，整型变量 a 的值变成了 23，且被红色醒目标出。

（5）再次按 F11 快捷键，单步执行程序，观察每条语句的作用。当执行到第 11 行语句时，sum 函数被调用执行，程序会暂停在 sum 函数体的开始花括号处，如图 1-32 所示。

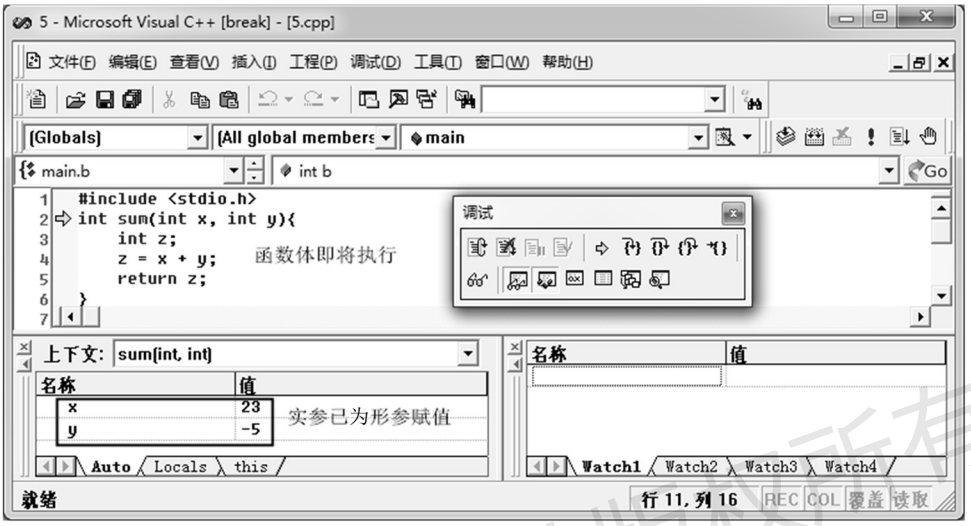

图 1-32　准备调试执行 sum 函数

由图 1-32 可知，此时实参已给形参赋值。形参 x 和形参 y 的值分别为 23 和−5。

（6）sum 函数执行完成并返回到 main 函数后，sum 函数的执行结果为返回值 18，如图 1-33 所示。

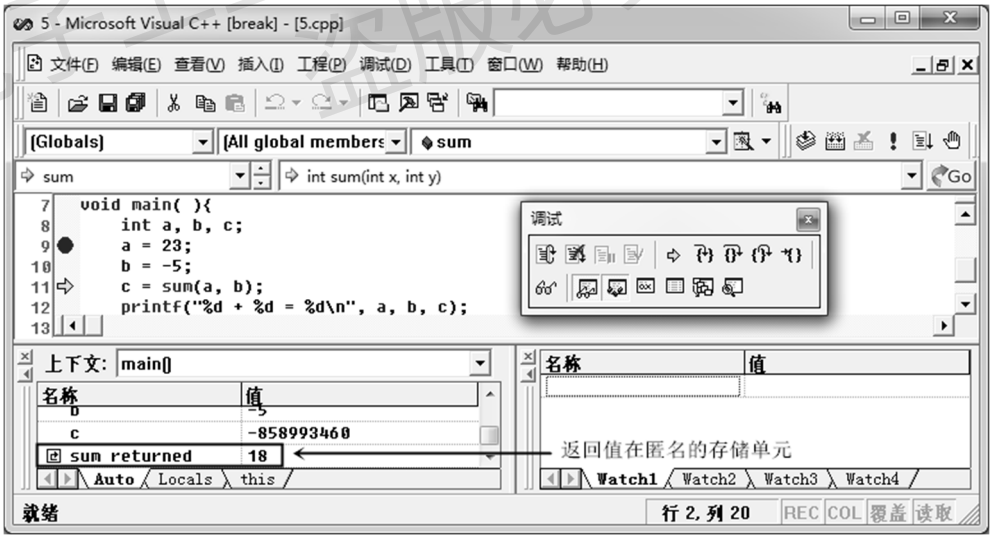

图 1-33　sum 函数调试执行的结果

（7）单步执行第 12 行语句时，若按 F11 快捷键，则程序将调试执行 printf 函数。如图 1-34 所示。printf 函数是库函数，没有必要查看其每条语句的执行过程，故使用 F10 快捷键。

按 F10 快捷键，VC6.0 不会单步执行语句中的被调函数，而是正常执行被调函数，直到被调函数执行完毕并返回才暂停。如果单步执行的语句中没有函数调用，则 F10 快捷键和 F11 快

捷键的作用相同，执行完当前语句后，程序会暂停在下一条语句处。

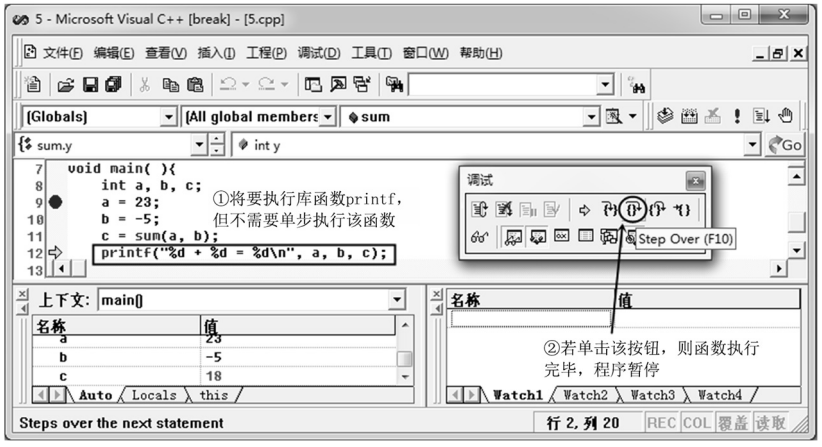

图 1-34　不调试执行库函数

提示：

① 调试执行时，如果需要用户输入数据，则可以切换到程序运行窗口输入数据。程序暂停时，也可以查看程序运行窗口的输出情况。

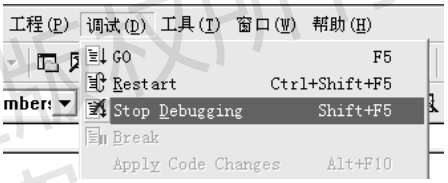

图 1-35　结束调试

② 结束调试程序可以按 Shift+F5 组合键，如图 1-35 所示。在调试执行状态，“组建”（Build）菜单自动变为“调试”（Debug）菜单。

③ 程序暂停执行时，除了以单步执行的方式继续执行程序，还可以用 F5 快捷键以调试执行的方式继续执行程序。调试执行时，只有遇到下一个断点，程序才会暂停执行。

编译程序遇到错误时，首先在信息输出窗口找到第一个错误提示，然后双击该提示，此时编译器会自动定位到可能出现错误的位置，最后结合提示信息分析出错原因。修正一个错误后，要再次尝试运行程序，而不要急着修改下一个错误。编译例 1-1 程序时，出现了图 1-36 所示的错误。

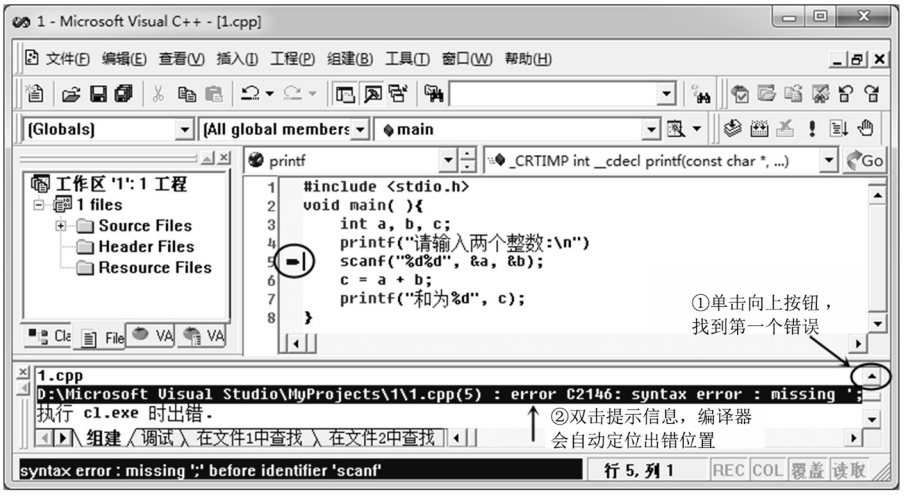

图 1-36　编译例 1-1 程序时出现的错误

由错误提示信息可知，标识符 scanf 前面少了一个;，即上一行语句的句末少了一个分号。

1.6 C 语言语句简析

C 语言语句由程序员编写并用于指挥计算机，由命令和操作数两部分组成。命令指挥计算机完成某一操作，操作数多为变量、字面量或一次操作的结果。

C 语言命令有关键字、操作符和函数 3 种。关键字是 C 语言中具有特定意义的字符串。关键字 int、float、char 与整型、浮点型、字符型有关，void 表示没有，return 用于结束函数执行，并返回函数的执行结果。操作符命令有常见的且含义相同的+、−、*、/（加、减、乘、除），有常见的但含义不同的赋值号=（C 语言中的等号是==），有专用的取地址操作符&。关键字和操作符是简单的命令，计算机可以直接执行。函数由程序员按照规定的语法定义，是完成特定功能的自定义命令。库函数由编译器在相应的头文件中定义，编程时将其引入即可使用。最常用的库函数是 stdio.h 中的 printf 函数和 scanf 函数。复杂的数学运算会用到 math.h 中的库函数，如求整数绝对值的 abs 函数、求小数绝对值的 fabs 函数等。

C 语言语句中的字符串多为标识符，用于标记 C 语言的关键字、变量和函数。关键字是规定的，变量和函数的区别在于函数带一对圆括号。应从程序员的角度理解 C 语言语句中出现的字符串，如 a + b * c 中的 a、b 和 c 就是变量，它们标识了 3 个存储单元，而非日常生活中的 abc。普通的字符串在 C 语言中需加一对双撇号，如"abc"。

printf 函数执行时，先得到一个输出字符串，然后依次输出其中的每个字符。在语句 printf("a+b=%d\n", 3 + 5);中，先执行操作符+命令，得到 3 + 5 的和为 8，再将占位序列中的%d 替换成 8，最终原语句相当于 printf("a+b=8\n");，输出结果为 a+b=8↙。

语句 int a = 3;中，int 是关键字命令，赋值号=是操作符命令，a 是变量，整数 3 是字面量。该语句定义了一个整型变量 a，且值被初始化为 3。

语句 c = sum(a, b);中，赋值号=是操作符命令，sum 是函数命令，c、a 和 b 是变量。先以 a、b 为实参调用 sum 函数，再将函数的返回值赋值给变量 c。

语句 scanf("%d", &sum);中，scanf 是函数命令，&是操作符命令；"%d"是字符串字面量，sum 是变量。取地址操作符&获得变量 sum 的地址，并将其作为实参；scanf 函数获得用户输入的一个整数，并将这个整数存储于实参地址标识的存储单元中，即变量 sum 中。

语句 printf("Hello, World!");中，printf 是函数命令，"Hello,World!"是字符串字面量，输出结果为 Hello,World!。

练习 1

1．根据要求写出 C 语句。

（1）向计算机申请 4 个整型存储单元，并命名为 x、y、z 和 result。

（2）在计算机的屏幕上呈现信息“请输入 3 个整数”。

（3）让计算机获得用户输入的 3 个整数，并把它们存储到变量 x、y、z 中。

（4）命令计算机求变量 x、y 与 z 的和，并把和存储到变量 result 中。

（5）在计算机的屏幕上显示变量 result 的值。

2．编写一个程序，求出用户输入的 3 个整数的和。

3．指出并改正下列语句中的错误（一条语句中可能有多个错误）。

（1）scanf("%d,value");。

（2）printf("The sum of %d and %d is %d"\n,x,y);。

（3）*/Program to determine the largest of three in integers/*。

（4）printf("The value you entered is %d"\n,&value);。

（5）int return=10;。

4．在 VC6.0 中编译本章例题中的程序和本练习第 2 题。

5．下面的标识符合法吗？

```
aBc, -245, _245, +3a, 4E2, __, 2n, n2, account_total
```

6．标识符的第一个字符为何不能是数字？

7．C 语言标识符区分大小写吗？n1 和 N1 是同一个标识符吗？利用下面的程序验证。

```
#include <stdio.h>
void main( ){
      int n1=3;
      printf("n1=%d", N1);
}
```

如果 n1 和 N1 是同一个标识符，程序会如何运行？如果 n1 和 N1 不是同一个标识符，程序又会如何运行？

8．若有 int x = 2;，写出输出结果或输出语句。

（1）printf("x=");。

（2）printf("x=%d%d", x, x);。

（3）/* printf("x+y=%d", x + y);*/。

（4）printf("\% and %%");。

（5）printf("Welcome to \\n C! and x=%z");。

（6）输出“100%”。

（7）把“This is a C program.”输出在两行，且第一行最后一个字母是 C。

9．编程输出如下内容。

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

10．分析下面程序的执行顺序，并写出程序的运行结果。

```
#include <stdio.h>
void print( ){
      printf("* * * * * * * * * * * \n");
}
void main( ){
      print( );
      print( );
      printf("    Hello,World\n");
      print( );
```

```
        print( );
}
```

11．分析下面的程序。

```
#include <stdio.h>
#include <stdlib.h>
void main( ){
        int a, b, sum;
        a = rand( );
    b = rand( );
        sum = a + b;
        printf("%d + %d = %d\n", a, b, sum);
}
```

提示：

（1）程序中的标识符 rand 是函数还是变量，为什么？

（2）查看附录中库函数 rand 的功能。什么是随机数？如何使用库函数 rand？

（3）讨论库函数的使用步骤。

12．编程求用户输入的两个小数的和。

13．重用可以简单地理解为使用别人写的代码。重用库函数，除了可以提高编码效率，还有其他方面的好处吗？

14．分析下面的 C 语言语句。

```
3+2;  z = x + y;  return 2.3 + f;  sum(x, 3) * 5; printf("sum(3, 2)\n"); 8;
```

15．查找资料了解 C 语言的历史、C 语言标准和常用的 C 语言编译器（如 GCC）。

本章讨论提示

1．例 1-1 利用加号命令求变量 a 与 b 的和，而这个程序利用 sum 函数命令求和。计算机可以直接执行加号命令，求出两个整数的和；计算机按照规定的流程执行 sum 函数命令求和。从功能上看，没必要定义 sum 函数，定义 sum 函数的目的是分析函数定义的语法和函数的执行过程，以便直观地理解函数。

2．有返回值的函数最终表现为一个具体的数，通常需要将它存储起来，以便进一步地加工、处理。如 c = sum(3, 5)中，用变量 c 存储了 sum(3, 5)的执行结果 8。若仅有语句 sum(3, 5);，则函数执行完毕后，原语句就相当于语句 8;。其中没有命令，计算机不会进行任何操作，sum(3, 5)实际上对程序没有产生任何影响，这条语句是多余的。

3．用户输入的整数需被存储到变量中，计算机中变量的存储单元可以存放“任意大”的整数吗？

4．计算机中有可以存放一个数的存储单元吗？

5．C 语言初学者没必要也没有能力把学习重点放在 C 语言的最新标准上。思考为什么是这样比了解最新的标准是什么更重要。因为初学者只会用到编译器的基本功能，所以不必纠结各种编译器的差异。初学者只要能熟练使用一种编译器，就能对其他编译器触类旁通。Windows 10 和 Windows 11 这两个版本的操作系统对中文版 VC6.0 的兼容性不好，本章中出现的中文版 VC6.0 界面仅供参考。本书使用安装了插件的英文版 VC6.0，增加了行号和代码补全功能。这两个版本的 VC6.0 在配套课件中提供，读者也可以到网上下载。