第5章 函 数

有了前面的基础，我们已经能够编写基本的 C 语言程序了，将功能代码写在程序的主函数之中，然而随着程序功能的复杂程度增加，主函数也会越写越长，这会给程序的阅读和维护带来很多不必要的麻烦。

函数对我们来说并不陌生，在前面的学习中，学习了格式化输入函数、格式化输出函数、字符输入输出函数等，还有在每个 C 语言程序中所用到的主函数，它们之间有什么不同，又有什么关系？通过本章的学习读者会更深入地了解函数。

5.1 函 数 概 述

5.1.1 为什么要使用函数

早在 11 世纪的北宋时期，中国人民发明的活字印刷作为中国古代“四大发明”之一，通过使用可以移动的金属或胶泥字块，来取代传统的抄写或无法重复使用的印刷版。按照稿件把单字挑选出来，印完后再将字模拆出，留待下次排印时再次使用，这是产品领域最早的模块化设计。

模块化设计的思想是先将产品划分并设计出一系列通用部件（即模块），再通过模块的灵活选择和组合构成不同的产品。其核心思想是划分模块、组合模块，而这一思想在程序设计中也一样通用，其中，划分模块是用函数的定义来实现的，组合模块则是用函数的调用来实现的。

函数一词是从 function 翻译过来的，其另一个含义是功能，可以理解为每个函数实现一定的功能。例如，用 sin()函数实现一个数的正弦值计算，用 printf()函数来实现格式化的输出，这些常用的函数都是由系统的库函数提供的，不再需要自己动手编写，我们根据需要调用这些函数即可，这样大大提高了编程的效率，简化了调试的难度。

在实现一个较复杂的程序时，往往会按照功能进行模块划分，每个模块再通过一个或多个函数来实现。通过主函数调用其他函数，或者其他函数之间的互相调用来组合出复杂的程序。一个 C 语言程序只能有一个主函数，也是程序的执行入口，主函数不能被其他函数调用，而其他函数只有直接或间接被主函数调用才能被执行到。

总之，要善于利用函数，以减少重复编写功能相同代码的工作量，以便实现模块化的程序设计。

下面通过一个简单的函数例子来看一下使用函数的好处是什么。

【例 5-1】 输出下面的结果，使用函数调用来实现。

```
######################
      Hello C!
######################
```

程序如下。

```
#include <stdio.h>
int main( )
{
    void print_line();      //print_line()函数声明
    void print_msg();       //print_msg()函数声明
    print_line();           //调用 print_line()函数
    print_msg();            //调用 print_msg()函数
    print_line();           //调用 print_line()函数
    return 0;
}
void print_line()
{
    printf("######################\n");
}
void print_msg()
{
    printf("      Hello C!     \n");
}
```

程序的运行结果为：

```
######################
      Hello C!
######################
```

说明：print_line()和 print_msg()是两个用户自定义函数名，分别实现输出一行“#”号和文字内容。

在这个例子中，输出的两行符号在代码中只出现了一次，当改变题目中的输出符号或者输出内容时，只需要修改其中的 print_line()和 print_msg()函数就可以做到，实现一改全改。总结使用函数的好处如下。

第一，方便代码重用。所谓“重用”，是指有一些代码的功能是相同的，操作是一样的，通过将这种功能的代码提取成一个函数，以后用到这个功能时只需要调用这个函数就可以了，不需要再重复地编写同样的代码。这样可以避免重复性代码。

第二，方便代码维护。可以对出问题的函数或者需要修改的函数进行局部修改，减少修改范围。

5.1.2 函数的分类

通过 5.1.1 节的介绍，可以看出函数分为库函数和用户自定义函数，这是从用户使用的

角度对函数进行划分的。

（1）库函数。也叫标准函数，是由系统提供的，不需要用户定义，可以直接调用它们，在使用库函数时需要整理出如表 5-1 所示的函数使用说明，以库函数 abs()为例。

表 5-1　库函数 abs()的使用说明

函数原型	int abs(int x)
函数的功能	求整数 x 的绝对值
函数参数的数目、类型、顺序、意义	1 个 int 型的参数，表示对此参数求绝对值
函数返回值的类型、意义	int 型的返回值，表示求出的绝对值结果
使用时所需要包含的头文件	math.h

（2）用户自定义函数。它是用户根据代码功能的模块划分自己编写的函数。例 5-1 中的 print_line()和 print_msg()函数就是用户自定义函数。

从函数的定义形式来看，函数可分为以下两类。

（1）无参函数。这类函数在函数调用过程中不需要传入任何数据，一般用来执行一定的功能，例 5-1 中的 print_line()和 print_msg()函数都是无参函数，其作用仅仅是输出固定的内容。

（2）有参函数。这类函数在函数调用过程中，通过参数传递数据，在函数内部根据参数数据来进行计算或者控制程序流程。例如，print_line()函数可以增加一个 int 型的参数，控制“#”字符的输出个数，从而使该函数更加灵活，应用于更多场合。

5.2　函 数 定 义

C 语言规定，函数和变量一样必须“先定义，后使用”。C 语言提供了极为丰富的库函数，这些函数的定义由编译系统完成，用户只需要使用预处理命令#include 将对应的头文件包含到程序中就可以使用它们。然而，库函数只能提供一些基本的功能，在实际的 C 语言程序编写过程中，所需要的大多数功能函数都需要自己定义。

C 语言中函数定义的基本格式如下。

```
函数类型 函数名(类型 1 形式参数 1，类型 2 形式参数 2，…，类型 n 形式参数 n)
{
    函数体
}
```

说明:

（1）一般将**函数类型 函数名(类型 1 形式参数 1，类型 2 形式参数 2，…，类型 n 形式参数 n)**称为函数首部或函数头。

（2）函数头中的**函数类型**表示函数返回数据（即返回值）的数据类型，如果函数不返回任何数据，可以使用 void 类型表示空，如果省略函数类型，大多数编译器默认函数类型为 int。

（3）函数头中的**函数名**要符合标识符的命名规则，且最好能够有见名知意的效果，以

便按名调用，且在程序可见范围内同一个函数名不能重复定义。

（4）函数头中在函数名后的一对小括号内是**形式参数列表**，简称形参列表，形参之间用逗号分隔，每个参数都要指定名字和类型，以便在调用函数时向它们传递数据。列表中参数个数可以是 0 个、1 个或多个，当形参个数为 0 个时，形参列表可以是空着或者使用 void 表示空，此时定义的函数为无参函数，当形参个数不为 0 个时，即形参列表至少有 1 个形参，此时定义的是有参函数。通过上述说明，无参函数的定义形式如下。

```
函数类型 函数名()
{
   函数体
}
```

或

```
函数类型 函数名(void)
{
   函数体
}
```

（5）**函数体**一般包括声明部分和语句部分。声明部分包括对函数使用的变量定义以及要调用的函数声明（具体见 5.4 节）等，在函数体中声明部分仅定义函数体中所用到的除形参以外的其他局部变量，声明部分和语句部分根据需要可省略，若两者都省略，即函数体为空时，这样的函数称为空函数。空函数的定义如下。

```
函数类型 函数名()
{ }
```

例如：

```
void dummy() { }
```

表示定义了一个函数 dummy()，函数体是空的，其他函数通过 dummy(); 来调用该空函数，调用该函数不会起到任何实际作用，什么也不会做，那么空函数有什么用呢？在程序设计初期阶段进行功能模块划分的过程中，可以对一些功能实现尚不明确或处理方式待定的模块定义相应的空函数，通过调用空函数来占一个位置，等以后功能确定的时候再来补充函数体的实现。所以通过空函数可以使程序的结构非常清楚，可读性好，而且功能扩充方便，不会影响大的程序结构。

为了便于理解函数的定义，我们将用 C 语言函数来解决一个数学函数题目。

【例 5-2】 有三个未知整数 a、b、c，其中，c = f(a,b) = a*b，用 C 语言函数实现数学函数 f 的功能。

分析：该数学函数名为 f，只要给出参数 a 和 b 的值，就可以调用函数 f，求得 c 的值，参照 C 语言的函数定义格式，将数学函数 f(a,b)对应的 C 语言函数如下。

```
int f(int a, int b)     //定义函数名为 f, 形式参数 a 和 b 为整型
{                       //函数 f 的开始
```

```
    int s;                  //声明部分，定义本函数中用到的变量 s 为整型
    s = a * b;              //语句部分，运算结果放入 s 中
    return s;               //将 s 的值返回，此语句的作用相当于调用函数 f(a,b)值为 s
}                           //函数 f 的结束
```

说明:

（1）函数 f 的功能可以实现计算任意整数 a 和任意整数 b 的乘积，并将乘积返回。这里需要注意函数头的写法，如果写成了 int f(int a, b)就是错误的，应该分别指明每个形参的数据类型。

（2）在函数体中，可以不定义变量 s，直接将表达式 a*b 的值返回，代码还可以简化如下。

```
int f(int a, int b)     //定义函数名字为 f, 形式参数 a 和 b 为整型
{                       //函数 f 的开始
   return a * b;        //调用函数 f(a,b)值为 a*b
}                       //函数 f 的结束
```

（3）return 语句后面的内容也可以用小括号括起来。例如：

```
return (s);或 return (a * b);
```

该函数实现的功能可以用图 5-1 来理解，函数需要从外部输入两个数 a 和 b，函数内部处理后可以得到一个数 s，其值为 a*b，并将值输出。

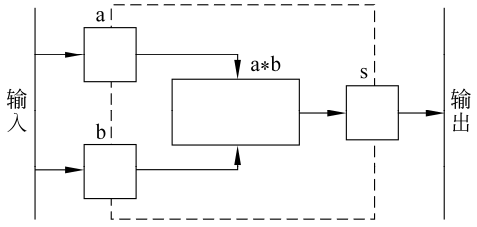

图 5-1　函数处理过程

在 C 语言中，所有函数的定义，包括主函数在内，都是“平行”的。也就是说，在一个函数的函数体内，不能再定义另一个函数，即不能嵌套定义。下面通过一个例子来对比几个简单函数定义的区别。

【例 5-3】 对比下面定义的三个函数。

```
#include<stdio.h>
int f1(void)                //定义函数名为 f1,是无参函数
{                           //函数 f1 的开始
   return 3+4;              //调用函数 f1，返回值为 7
}                           //函数 f1 的结束
int f2(int a)               //定义函数名为 f2,有 1 个形式参数 a，为整型
{                           //函数 f2()的开始
   return a+4;              //调用函数 f2(a)值为 a
}                           //函数 f2()的结束
```

```
int f3(int a, int b)        //定义函数名为 f3, 形式参数 a 和 b 为整型
{                           //函数 f3()的开始
   return a+b;              //调用函数 f3(a,b)值为 a+b
}                           //函数 f3()的结束
void main()
{
   //主函数中调用了 f1()函数、f2()函数和 f3()函数
   printf("f1=%d, f2=%d, f3=%d", f1(), f2(5), f3(7,9));
}
```

程序的运行结果为：

```
f1=7, f2=9, f3=16
```

说明：该程序中除了主函数之外，还定义了 3 个自定义函数，功能都是求和。函数 f1()没有任何参数，其功能为计算 3 与 4 的和且只能实现 3 与 4 求和；函数 f2()只需要一个参数，其功能为实现计算任意整数与 4 的和；函数 f3()则需要两个参数，其功能为实现计算任意整数与任意整数的和。

因此，在对功能进行划分阶段，需要首先设计和定义好函数首部（包括函数的返回值类型，形参的个数和类型等），使函数的功能具有很好的通用性。

5.3 函数调用

函数定义仅仅是将各个功能模块准备好，为了实现完整的功能代码，还需要将各个模块进行组装，即函数调用，一个函数只有直接或间接被主函数调用，才会在程序运行中起到作用。

5.3.1 函数调用的一般形式

函数调用在我们开始第一个 C 语言程序时就已经开始了，如前面见过的：

```
printf ("Hello World! \n");
print_line();
```

其中，printf()函数是系统库函数，print_line()函数是自己定义的函数，因此可以根据需要调用这两类函数来完成程序的功能，无论调用哪类函数，其调用的一般形式如下。

函数名(实际参数 1,实际参数 2,…,实际参数 n)

说明:

（1）在调用函数之前，函数必须是事先定义好的，一对小括号内为以逗号分隔的实际参数，切记不要在实际参数前面写类型，实际参数可以是常量、变量、表达式或者函数调用等。如果调用的函数是无参函数，则小括号内为空的。

（2）在调用函数时，要求实际参数的个数、类型都要与函数定义中的形式参数一一对应。每个实际参数要有具体的值，通过函数调用将数据传递给被调用函数对应位置的形参

变量，并执行函数的具体功能。

（3）将调用其他函数的函数称为**主调函数**，被其他函数调用的函数称为**被调函数**。如例 5-3 中的函数 f1()、f2()、f3()被 main()函数调用，main()函数是主调函数，f1()、f2()、f3()函数是被调函数。当函数调用关系变得更复杂时，一个函数可以既是主调函数又是被调函数。

一般地，函数调用在程序中出现的形式可以有如下 3 种。

1．函数调用语句

函数调用作为一条执行语句（语句末尾加上分号;）出现在主调函数中。仅执行被调函数的功能和操作，而对于被调函数的返回值没有要求，即被调函数的函数类型可以是 void 类型，也可以是其他类型，但是在主调函数中不使用其返回的数据。如例 5-1 中的函数调用“print_line();”，以及我们经常调用的 printf()和 scanf()函数。

2．参与表达式运算

函数调用参与表达式运算。如“ch=getchar();”或者“c=f(3,4);”这两个都是赋值表达式语句，赋值运算符右侧是一个函数调用（getchar()函数是一个字符输入库函数，f()函数是一个自定义的函数），将被调函数的返回值赋值给赋值运算符左侧的变量。

这种形式的函数调用要求被调函数的函数类型不能是 void 类型，即必须有数据返回。在主调函数中使用其返回数据的值参加表达式的运算，包括我们学习过的算术运算、逻辑运算、关系运算、条件运算等。例如：

```
c=5*f(2,5);
d=f(4,3)>f(3,4);
```

若 f()函数为例 5-2 中定义的功能为计算两个数乘积，那么通过上述语句，c 的值为 50（即 5 乘以函数调用 f(2,5)的返回值 10 的积），d 的值为 0（即函数调用 f(4,3)返回值 12 大于函数 f(3,4)返回值 12 的关系表达式为假）。

3．作为函数实参

函数调用还可以作为另一个函数调用的实参。这种形式的函数调用也要求被调函数的函数类型不能是 void 类型，必须有数据返回。如例 5-3 中的 main()函数中的语句：

```
printf("f1=%d, f2=%d, f3=%d", f1(), f2(5), f3(7,9));
```

首先，从被调用函数 printf()的角度看，这是一个函数调用语句，函数的实际参数有 4 个，第一个是输出字符串的格式，第二个实际参数是函数调用 f1()，第三个实际参数是函数调用 f2(5)，第四个实际参数是函数调用 f3(7,9)，后面三个实际参数的值分别为这 3 个函数调用的返回值。又如：

```
c=f(3,f(4,5));
```

内层的函数调用 f(4,5)是作为外层 f()函数调用的第二个参数。

无论函数调用以哪种形式出现，函数调用本身并不需要分号结尾，只需要使用函数名及所需的实际参数即可，而分号在表示一个完整的 C 语言可执行语句结束的时候才需要。

5.3.2 函数调用的过程分析

有了函数之间的调用，那么程序是如何执行的呢？通过下面的例子来分析程序运行的过程。

【例 5-4】 编写一个完整的程序，调用例 5-2 中定义的 c = f(a,b) = a*b 的函数，实现 15 与 4 的积。

```
#include<stdio.h>
int f(int a, int b)                    //定义函数名字为 f, 形式参数 a 和 b 为整型
{
   int s;
   s = a * b;
   return s;
}
int main()
{
   int c;                              //定义存放结果的变量
   c=f(15,4);                          //调用函数 f, 将得到的值赋给 c
   printf("Result is %d\n",c);         //输出 c 的值
   return 0;
}
```

程序运行结果为：

```
Result is 60
```

程序的执行顺序为：

（1）任何程序的入口都是主函数，所以程序先进入主函数，然后根据声明语句分配变量 c 的存储空间，执行“c=f(15,4);”语句，这是一个赋值表达式语句，赋值运算符的右边是调用函数 f(15,4)，程序需要向前面的代码行查找 f()函数的定义或者声明，找到后跳转到 f()函数去执行。

（2）调用函数 f()时，C 编译系统会将实参 15 和 4 依次按顺序赋值给形参 a 和 b，这个过程称为**参数传递**。这样 a 和 b 都有了具体的值，继续执行 f()函数内部的语句，计算 a 与 b 的积并赋值给变量 s。通过"return s;"语句将结果 60 输出返回给主函数，程序执行到函数 f()的结束大括号位置，函数调用结束。

（3）程序继续回到主函数中调用 f()的位置，将调用函数 f()获得的返回值 60 赋给变量 c，再执行"printf(“sum is %d\n”,c);"，将结果 60 输出。

（4）程序继续执行主函数中的语句，执行"return 0;"，主函数正确返回，程序结束。

标记了执行顺序的完整程序如下。

```
    #include<stdio.h>
⑤int f(int a, int b)                    //定义函数名为 f, 形式参数 a 和 b 为整型
⑥{
⑦     int s;
```

```
⑧     s = a * b;
⑨     return s;
⑩ }
① int main()
② {
③       int c;                          //定义存放结果的变量
④⑪   c=f(15,4);                        //调用函数 f(), 将得到的值赋给 c
⑫    printf("Result is %d\n",c);        //输出 c 的值
⑬        return 0;
⑭ }
```

总之，main()函数是主函数，它可以调用其他函数，但不允许被其他函数调用。C 语言程序的执行总是从主函数开始的，也是到主函数结束的，即使自定义的其他函数位于主函数的前面，程序仍然从主函数开始执行。如果执行到函数调用则程序转去执行被调用的函数，完成函数调用后再返回到函数调用前的位置继续往下执行，最后由主函数结束整个程序。一个 C 语言程序必须有且仅有一个主函数。

在函数调用过程中，系统会把实参的值传递给被调函数的形参，也就是形参变量的值是从实参得到的，该值只有在函数调用期间有效，也只能在该函数内参与运算。在参数传递过程中，当实参的数据类型与对应的形参变量类型不一致时，会根据数据类型转换原则，根据形参的数据类型获取实参的数据。

5.3.3 函数的返回值

当函数调用在表达式中或者是作为函数参数时，在主调函数中希望通过函数调用得到一个值来参与运算或传递参数。这个值就是被调函数的返回值。如例 5-4 的主函数中有语句

```
c=f(15,4);
```

从 f()函数的定义可知，函数调用 f(15,4)的值是 60，这个 60 就是函数的返回值，再通过赋值语句将 60 赋给变量 c。

对函数返回值需要做如下说明。

（1）**函数的返回值仅能通过函数内的 return 语句得到**。return 语句能够将被调函数中的运算结果返回给主调函数，return 语句也表示本次函数调用过程的结束，若被调函数内无 return 语句，则函数调用执行到函数定义的“}”时，函数调用自动返回。如果需要使用 return 语句返回数据，那么被调函数的函数类型一定不能是 void 类型。

在一个函数中可以有一个以上的 return 语句，根据程序的执行顺序，第一个被执行到的 return 语句起作用。return 语句后面的括号是可有可无的，如“return (s);”或“return s;”是等价的。return 后面的值可以是一个常量、变量、表达式或者是另一个函数调用的返回值，例如：

```
return getchar();
```

表示返回一个用户输入的字符。等价于：

```
char c=getchar();
return c;
```

（2）**函数的返回值类型**。在函数定义时指定了函数类型，这个类型就表示该函数返回值的类型。例如下面的函数头：

```
int bigger(float a, float b)          //函数返回值为 int 型
double min(int x, int y)              //函数返回值为 double 型
```

因此虽然函数的返回值是通过 return 语句返回的，但是返回值的数据类型是由函数定义的函数头中的函数类型决定的。函数类型最好与 return 语句返回的值类型保持一致，前面例子中的函数返回值与函数类型都是一致的。若二者不一致，函数返回数据以函数类型为准，由 C 编译系统自动完成类型转换，即**函数类型决定返回值的类型**。

【例 5-5】 修改例 5-4 中的函数，将形参及内部的变量 s 变为 float 类型，分析程序运行结果。

```
#include<stdio.h>
int f(float a, float b)         //定义函数名为 f, 形式参数 a 和 b 为实数类型
{
    float s;
   s = a * b;
   return s;
}
int main()
{
    int c;                      //定义存放结果的变量
    c=f(1.5,4.5);               //调用函数 f(), 将得到的值赋给 c
    printf("Result is %d\n",c);//输出 c 的值
  return 0;
}
```

程序运行结果为：

```
Result is 6
```

分析：在主函数中执行了赋值语句 c=f(1.5,4.5);，将调用函数 f()的返回值赋值给变量 c，在调用 f()函数时传递的实参是 float 型的常量，与 f()函数定义的形参类型一致，函数调用后，形参变量 a 和 b 的值分别为 1.5 和 4.5，因此计算出来的 s 的值为 6.75。然而函数 f()的函数头定义的函数类型为 int，而在函数体中 return 语句返回的数据 s 是 float 型，二者出现不一致。按照赋值规则，系统会将 s 的值转换为 int 型，得到整数 6 作为函数的返回值。因此主函数中 c 的值输出为 6。

（3）**无返回值函数，函数类型应定义为 void**。这样的函数一般只执行一定的操作，不返回任何数据，因此这类函数调用仅能作为函数调用语句，不能参与表达式运算或作为函数实参。这类函数的函数体中不能使用带值的 return 语句返回数据，仅能通过“return;”来表示函数调用的结束。