第 3 章　C 语言的构成——函数

在第 2 章中介绍了算法的基本概念及算法的描述等知识，本章将介绍 C 语言的组成元素，也就是函数。一个 C 语言程序肯定包括一个或多个函数，程序的各个模块通过函数来实现其功能，函数的调用便构成了一整个程序。本章首先介绍函数的基本概念，然后介绍如何定义及调用函数，再介绍函数的参数和返回值，最后通过一个实践项目来向读者全面的总结函数的使用。本章具体内容如下：

- 函数的概述；
- 简单函数的定义和调用；
- 函数的参数；
- 函数的返回值；
- 开发实践。

3.1　函 数 概 述

C 语言是一门强大的语言，也是作为一个程序员必须掌握的一门语言，而在 C 语言的知识结构中，函数则是其灵魂和精髓。一个 C 语言程序，可以说就是由一个个函数组成的，这些函数中包含了程序的可执行代码，所以要学好 C 语言，程序员必须掌握和理解函数的使用方法和意义。

C 语言函数可以分为库函数和自定义函数。库函数是指系统建立的具有一定功能的函数的集合，用户可以直接拿来使用而不需要自己开发；而自定义函数则是指程序员自己定义的函数，需要程序员自己开发其执行逻辑。下面来具体介绍库函数和自定义函数。

3.1.1　库函数

前面介绍过，库函数指的是编译器自带的提供给程序员使用的一组函数，这些库函数极大的方便了程序员进行程序编写，使得程序员更好地实现功能而不用去管一些基本功能的实现。例如，输出函数 printf()，程序员在进行数据输出时，不用去管输出功能的实现，只管调用该函数就可以，这样就避免了程序员每次要输出数据时都得去写一次数据输出功能函数。

按照 ANSIC 的要求，C 语言编译器提供了一系列库函数，包括标准输入输出函数、文件处理函数、字符串处理函数、数学计算函数和时间处理函数等。

【例 3-1】 下面的项目 Function_Demo1 给出了标准输入输出函数的使用，用户输入

的字符串是什么，输出就是什么。

（1）在 VC IDE 中创建工程 Function_Demo1，在 Source Files 中添加文件 main.c，编写代码如下。

```
#include<stdio.h>
int main()
{
    char a[100];                    //定义字符数组，接收输入字符
    printf("input:");               //输出 input:
    scanf("%s",a);                  //输入字符串
    printf("output:%s\n",a);        //输出 a 中存放的字符串
    return 0;                       //返回
}
```

（2）编译运行程序，结果如图 3.1 所示。

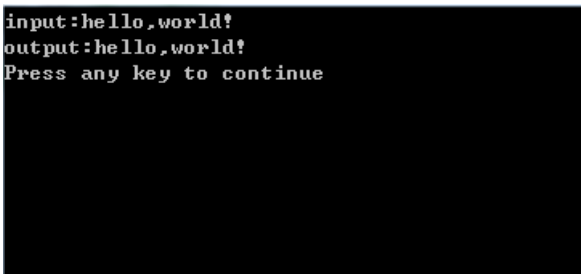

图 3.1　运行结果

可以看到，用户输入 hello，world！时，下面会显示用户的输入。上面程序中，使用了标准输入函数 printf()和标准输出函数 scanf()，读者需要注意的是，在使用这两个函数时，必须使用#include<stdio.h>，只有这样，编译才会成功。

3.1.1　自定义函数

前面介绍过，自定义函数是程序员根据一定的规则来定义而能执行一定功能的函数，程序员定义后就可以自己使用。实际上，在每一个 C 语言程序中，都有一个特别的自定义函数，即 main()函数，该函数是 C 语言程序运行的入口和出口，程序每次都从该处开始运行。前面例 3-1 中就是从 main()函数开始执行的。

在一个 C 语言程序中，往往会使用多个函数，尽管只用一个 main()函数就能编写任何程序，但是这个编写程序的方式会导致一些问题，比如程序变得很大而且复杂，因而使得调试、测试和维护工作变得困难，而多函数程序就可以避免这些问题。如果将一个程序按功能划分为多个部分，每个部分单独编码，最后将它们组合成一个整体，这样就可以更容易的对程序进行调试、测试和维护。一个多函数程序的结构图如图 3.2 所示，主程序即 main()函数可以调用多个函数，这些函数又可以继续调用函数，直到实现所需要的功能。

多函数程序设计使得程序更容易被调试、测试和维护，这正是遵循了模块化程序设

计这一软件系统设计与开发思想。这种思想将大型程序组织成小而独立的程序段，即模块，它们各有自己的名字和功能。这些模块最后集成为一个软件系统，从而满足系统的需求，如图 3.3 所示即模块化编程结构图。实际上，一个大的软件，往往由很多人一起分工合作，而它们分工的根据就来自于程序的模块化，每个人仅仅负责自己的模块，而不去管其他，这样使得分工更加明确，容易合作。

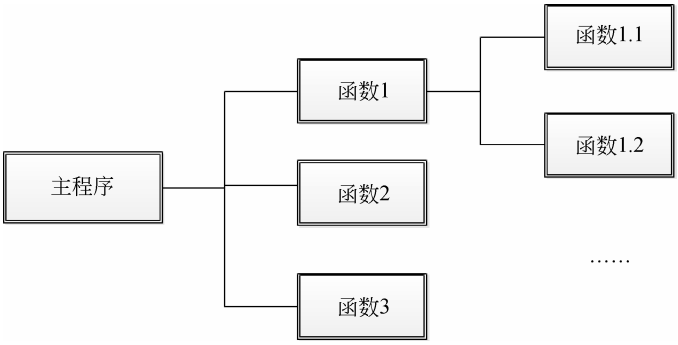

图 3.2　多函数程序编程

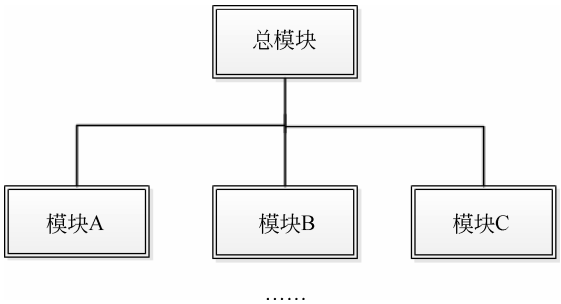

图 3.3　模块化程序设计

【例 3-2】 下面的项目 Function_Demo2 给出了多函数程序的示例。

（1）在 VC IDE 中创建工程 Function_Demo2，在 Source Files 中添加文件 main.c，编写代码如下。

```
#include<stdio.h>
/*******************************************************************************
*    函数名                :    getBrick
*    函数功能              :    搬起砖
*******************************************************************************/
void getBrick()
{
    printf("--------------------\n");
    printf("*****搬起砖功能********\n");
    printf("--------------------\n");
}
/*******************************************************************************
*    函数名                :    move
*    函数功能              :    走向目的地
*******************************************************************************/
```

```
void move()
{
    printf("--------------------\n");
    printf("***走向目的地*******\n");
    printf("--------------------\n");
}
/*******************************************************************************
*    函数名               :    droptBrick
*    函数功能             :    将砖放下
*******************************************************************************/
void dropBrick()
{
    printf("--------------------\n");
    printf("***把砖放到目的地***\n");
    printf("--------------------\n");
}
int main()
{
    getBrick();             /*执行搬起砖功能*/
    move();                 /*执行走向目的地功能*/
    dropBrick();            /*执行把砖放到目的地功能*/
    return 0;               //返回
}
```

（2）编译运行程序，得到结果如图 3.4 所示。

图 3.4　运行结果

当搬砖工要将砖从一个地方搬运到目的地，需要三个动作，分别是将砖搬起来、走向目的地和放下砖。上面示例中，搬运的三个动作分别是三个函数，在 main()函数中来调用这三个函数，就完成了搬砖功能。

3.2　函数的定义和调用

在 3.1 节中介绍了 C 语言函数的一些基本知识，主要是关于库函数和自定义函数的介绍，本节将介绍函数的定义和调用。使用 C 语言函数，首先必须得定义一个函数，而要使用一个函数就得调用定义的函数。函数的定义主要是让编译器知道这个函数的功能，包括标识符、定义的变量和运行的程序代码等；而函数的调用主要是让编译器知道调用的函数标识符，传递的参数，但函数可能会返回调用结果。下面来具体介绍函数的定义和调用。

3.2.1　简单函数的定义

在 C 语言中，函数的定义必须遵循一定的格式，虽然如此，但在 C 语言中有多种函数定义格式，无参数不带返回值的函数、有参数不带返回值的函数、无参数带返回值的函数和有参数带返回值的函数。C 语言的函数定义分为两个部分：函数头部和函数执行体，其具体语法格式如图 3.5 所示。

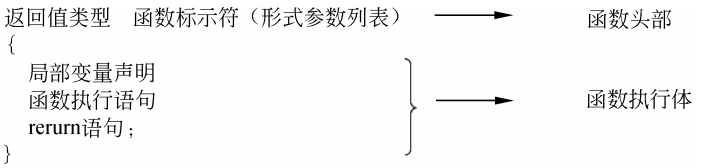

图 3.5　函数定义的语法格式

下面的代码是一个函数的代码，该函数执行加法功能，属于有参数带返回值的函数：

```
int add(int a,int b)    /*函数头部*/
{
    /*函数执行体*/
    int result;         /*局部变量声明*/
    result=a+b;         /*函数执行语句*/
    return result;      /*返回结果*/
}
```

通过上面代码和图 3.5 可以分析函数定义的过程。函数头部分为以下 3 个部分。

（1）返回值类型

返回值是一个 C 语言数据类型或后面将会介绍的自定义数据类型，在上面代码函数中返回值类型是 int。

（2）函数标识符

函数标识符主要用于编译器识别函数，所以在程序中函数标识符必须是唯一的。函数标识符遵循 C 语言命名规则：

- 标识符第一个字符必须是字母或下划线；
- 标识符只能由字符、数字或下划线组成；
- 标识符最长只有 31 个字符，如果长于 31 个字符，则只有前 31 个字符是有效的；
- 标识符不能使用 C 语言中的关键字，如 int、char、if、else、while、long、switch 和 for 等；
- 标识符不能包含空格。

在上面代码中函数标识符是 add。

（3）形式参数列表

形式参数列表是函数调用时传递的参数列表，可以简称为形参，在进行函数调用时，实际参数将会复制到这些形参变量中。函数定义时可以不带形式参数列表，即括号里什么都不写，这样就定义了无参函数。

上面代码中定义了两个参数 a 和 b。

函数执行体在函数头部下面，需要用花括号括起来，函数执行体也分为 3 个部分。

（1）局部变量声明

在函数执行体中往往需要一些变量来存储数据，这就需要声明一些局部变量。在上面代码中 result 就是一个局部变量。

（2）函数执行语句

函数执行语句就是执行函数功能，也是一个函数的核心部分。在上面的代码中，函数功能是返回 a 和 b 的加法和的结果，因此函数执行语句中先执行 a+b，再将结果存储在局部变量 result 中。

（3）return 语句

该语句一般放在函数执行体最后一行，主要返回函数执行的结果。读者需要注意的是返回结果类型必须和函数头部所要求的返回结果类型一致，例如上面代码中返回结果类型是 int，则返回的结果 result 中存储的数据类型就是 int。

【例 3-3】 下面的项目 Function_Demo3 给出了函数的定义。

（1）在 VC IDE 中创建工程 Function_Demo3，在 Source Files 中添加文件 main.c，编写代码如下。

```
#include<stdio.h>
/*********************************************************************************
*    函数名                :    add
*    函数功能              :    将两个数相加，返回和
*    函数参数              :    a 为加数 1
*                               b 为加数 2
*********************************************************************************/
int add(int a,int b)            /*函数头部*/
{
    /*函数执行体*/
    int result;                 /*局部变量声明*/
    result=a+b;                 /*函数执行*/
    return result;              /*返回结果*/
}
int main()
{
    printf("%d\n",add(10,2));   //调用函数
    return 0;                   //返回
}
```

（2）编译运行程序，结果如图 3.6 所示。

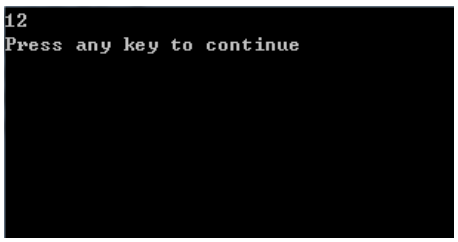

图 3.6　运行结果

从上面运行结果可以看到，add()函数成功实现了加法功能，并返回结果。

3.2.2　简单函数的调用

在实际生活中，为了完成一件事，需要用到很多工具，函数就像一个能完成某些具体功能的工具，函数的定义实际上就好比完成了工具的制作，而调用函数就是一个使用工具的过程。

函数调用的形式为函数标识符后面跟着实参列表，例如在例 3-3 中使用 add(10,2)就实现了加法函数 add()的调用。当编译器遇到函数调用时，会将控制权给被调用的函数，然后逐布执行函数执行体的语句，遇到 return 时返回一个值。

函数的调用根据其作用主要分为三种方式，主要是函数语句调用、函数表达式调用和函数参数调用，下面来一一介绍这三种方式。

1．表达式方式

函数调用表达式方式是指函数调用出现在一个表达式中，此时要求函数返回一个确定的值，这个值会参与表达式的运算。

【例 3-4】 下面的项目 Function_Demo4 给出了函数调用的表达式方式。

（1）在 VC IDE 中创建工程 Function_Demo4，在 Source Files 中添加文件 main.c，编写代码如下。

```
#include<stdio.h>
/*********************************************************************************
*    函数名                :    add
*    函数功能              :    将两个数相加，返回和
*    函数参数              :    a 为加数 1
*                               b 为加数 2
*********************************************************************************/
int add(int a,int b)        /*函数头部*/
{
    /*函数执行体*/
    int result;             /*局部变量声明*/
    result=a+b;             /*函数执行加法运算*/
    return result;          /*返回结果*/
}
int main()
{
    int result;             /*变量声明*/
    result=add(1,2)+3;      /*在表达式中调用函数*/
    printf("%d\n",result); //调用函数
    return 0;               //返回
}
```

（2）编译运行程序，结果如图 3.7 所示。

可以看到，在例 3-4 中，函数 add()被调用后返回结果继续与 3 相加，其结果赋值给 result 变量，实际上，表达式 add(1,2)+3 就相当于(1+2)+3，也就是说函数调用使用表达式方式时，函数返回结果参与表达式运算。

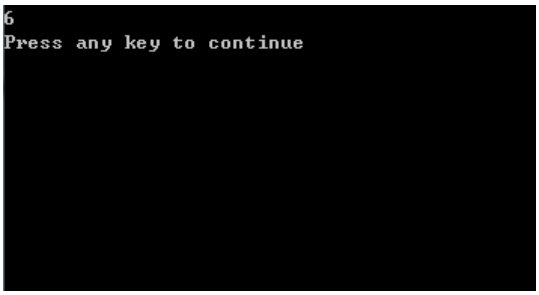

图 3.7　运行结果

2．语句方式

函数调用语句方式是指把函数调用作为一个语句，该方式常用于只要求函数完成一定的操作，不要求函数返回值。

【例 3-5】 下面的项目 Function_Demo5 给出了函数调用语句方式。

（1）在 VC IDE 中创建工程 Function_Demo5，在 Source Files 中添加文件 main.c，编写代码如下。

```
#include<stdio.h>
/*******************************************************************************
*    函数名                :    print
*    函数功能              :    打印出一串字符串
*******************************************************************************/
void print()
{
    printf("你好，哈哈哈！");
}
int main()
{
    print();                /*打印*/
    return 0;               //返回
}
```

（2）编译运行程序，结果如图 3.8 所示。

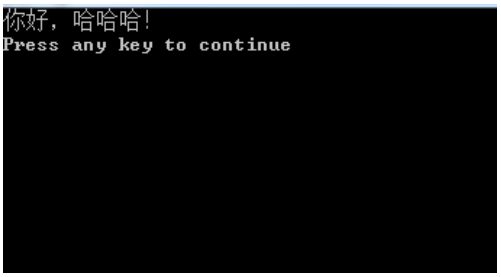

图 3.8　运行结果

可以看到，在例 3-5 中，print()函数的功能就是输出并显示一条消息，此时不需要返回值，因此调用该函数时只需要直接调用就可以。

3．参数方式

函数调用参数方式是指将函数调用作为一个函数的实参，将函数返回值作为实参传递到函数中进行使用。

【例 3-6】 下面的项目 Function_Demo6 给出了函数调用语句方式。

（1）在 VC IDE 中创建工程 Function_Demo6，在 Source Files 中添加文件 main.c，编写代码如下。

```
#include<stdio.h>
/*********************************************************************************
*    函数名                :    max
*    函数功能              :    比较两个数中的较大数
*    函数参数              :    a 为比较数 1
*                               b 为比较数 2
*********************************************************************************/
int max(int a,int b)
{
    if(a>=b)return a;           /*假如 a 大于等于 b，返回 a 的值*/
    else
    {
        return b;               /*否则返回 b 的值*/
    }
}
int main()
{
    int result;                 /*声明 result 变量*/
    result=max(1,max(2,3));     /*函数调用作为参数*/
    printf("%d\n",result);      /*输出结果*/
    return 0;                   //返回
}
```

（2）编译运行程序，结果如图 3.9 所示。

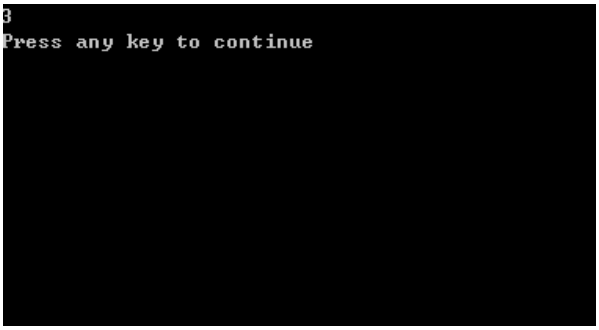

图 3.9　运行结果

可以看到，在例 3-6 中定义了一个 max()函数，该函数功能是取参数 a 和 b 中较大值并返回，而现在要想取数 1、2 和 3 中最大的数，而函数 max()只有两个参数，可以先比较两个数中比较大的数，然后较大数再和第三个数比较，这就可以采取函数作为参数的调用方式。

3.3　函数的参数

在 3.2 节中介绍了 C 语言简单函数的定义与调用，主要包括一个函数的整体结构，本节将具体介绍函数中很重要的一个组成部分，即函数参数。在大多数情况下，函数主要用来处理输入数据，这时就需要根据情况来输入数据，而函数参数就代表了这些数据，也就是说函数参数主要是调用函数时所需要函数处理或必须的输入数据。下面来具体介绍函数参数的相关知识。

3.3.1　带参数的函数定义

在 3.2 节中初步介绍过，带参数的函数定义主要是在函数定义时，在括号中加入参数列表，由于在参数的使用中要求不一样，因此在函数参数设计上，函数的参数类型和参数个数都可能不一样。因此，函数的参数个数可以是一个，也可以是多个；而函数的参数类型则就有可能是基本类型也可能是自定义的结构类型，这都需要根据具体函数功能来决定。

前面介绍过，在 C 语言中函数需要定义后才能通过调用来使用它，同样，带参数的函数也需要定义才能被调用，其语法格式如下。

```
函数类型 函数名(类型 参数 1, 类型 参数 2, …)
{
    函数体;
}
```

在上面带参数的函数定义中，参数 1 和参数 2 等参数都是形式参数，后面会做具体介绍。带参数的函数和不带参数的函数相比，带参数的函数仅仅多的是调用函数输入的数据，这些参数就是调用者所给的数据。例如，如果要计算一个圆的面积，这时如果有一个函数来计算圆的面积，那么必须给定圆的半径，这个半径数值就可以作为该函数的参数，其函数定义如下：

```
void area(int radius)
{
int result=3.14*radius*radius;
printf("圆的面积为: ");
    printf("%d\n", result);
}
```

可以看到，该函数实际上可以计算任意圆的面积，这时函数的作用就显示出来了。如果在一个程序中，需要计算多个不同圆的面积的时候，若每次都写一次圆面积计算公式的话，则会加大代码的可读性，也破坏了程序的模块化设计思想，这时如果将圆面积计算写成一个函数，每次调用该函数时传入半径就可以得到圆面积，这样就简单多了。

3.3.2　带参数的函数调用

带参数的函数调用和前面简单函数的调用一样，主要是根据其函数名来进行调用，

唯一不同的是这里需要注意函数调用时的参数。在进行带参数的函数调用时，其参数必须准确无误的给出，否则编译时就会报错。

带参数的函数调用语法格式如下：

```
函数名(参数 1, 参数 2, …);
```

这里需要注意的是，前面在定义函数时，其参数称为形式参数，但这里调用函数时，其参数则称为实参。

在对函数调用时，调用者需要指定实参，这里的实参实际上就是具体的数据，实参会传递给形参，这里形参和实参无论是数据类型还是数据值都是一样的，其区别就是二者是两个变量，改变一个变量对另一个变量是没有影响的，二者值相等但却是两样东西，数据装在不同的盒子里面。

下面的例子来说明实参和形参之间的值传递过程。

【例 3-7】 给定一个长方体的底、宽和高做参数，计算长方体的体积。

（1）在 VC IDE 中创建工程 Function_Demo7，在 Source Files 中添加文件 main.c，编写代码如下。

```
01  #include <stdio.h>              //包含库
02  /*****************************************************************************
03  *    函数名称             :    volumn
04  *    函数功能             :    计算长方体体积
05  *    函数参数             :    bottom：底
06  *                              width:   宽
07  *                              heigh:   高
08  *****************************************************************************/
09  void volumn(int bottom,int width, int height)
10  {
11      int result =bottom* width * height;//计算体积
12      printf("长方体的体积为：");                //输出字符串
13      printf("%d\n", result);                //输出结果
14  }
15  int main()
16  {
17      volumn(2, 3, 4);                       //调用函数，参数为 2,3,4
18      volumn(10, 19,10);                     //调用函数，参数为 10,19,10
19      return 0;                              //返回
20  }
```

（2）编译运行程序，结果如图 3.10 所示。

图 3.10　运行结果

在例 3-7 中，函数 volumn()主要起着计算长方体体积的作用，参数 bottom、width 和 height 是长方体的底、宽和高，参数类型为 int 数值型。

- 第 11 行的“*”为乘法运算，计算得出长方体的体积，然后将结果赋值给 result。
- 第 17 行代码调用函数 volumn()，将 2、3 和 4 传入 bottom、width 和 height，计算长方体体积。
- 第 18 行代码调用函数 volumn()，将 10、19 和 10 传入 bottom、width 和 height，计算长方体体积。

实际上，形式参数仅仅作用在函数里面，也就是说在调用一个函数的过程中，形式参数才起作用，其他时候形式参数不起任何作用，当形式参数接收了实际参数传递过来的值后，就可以在函数体中使用。对于不同的函数而言，二者的形式参数相互之间没有任何影响，即使两个函数的参数名称一样也是如此。

【例 3-8】 编写函数分别实现四则运算中的加、减、乘和除法。

（1）在 VC IDE 中创建工程 Function_Demo8，在 Source Files 中添加文件 main.c，编写代码如下。

```
01  #include <stdio.h>              //包含库
02  /*****************************************************************************
03  *    函数名称             :    add
04  *    函数功能             :    求和
05  *    函数参数             :    a：加数一
06  *                              b：加数二
07  *****************************************************************************/
08  void add(int a,int b)
09  {
10      int result=0;
11  result=a+b;
12  printf("%d+%d=%d\n",a,b,result);
13  }
14  /*****************************************************************************
15  *    函数名称             :    subtract
16  *    函数功能             :    求差
17  *    函数参数             :    a：被减数
18  *                              b：减数
19  *****************************************************************************/
20  void subtract(int a,int b)
21  {
22      int result=0;
23      result=a-b;
24      printf("%d-%d=%d\n",a,b,result);
25  }
26  /*****************************************************************************
27  *    函数名称             :    multiply
28  *    函数功能             :    求积
29  *    函数参数             :    a：被乘数
30  *                              b：乘数
31  *****************************************************************************/
32  void multiply(int a,int b)
33  {
34      int result=0;
35      result=a*b;
```

```
36      printf("%d*%d=%d\n",a,b,result);
37  }
38  /*****************************************************************************
39  *    函数名称              :    divide
40  *    函数功能              :    求商
41  *    函数参数              :    a：被除数
42  *                               b：除数
43  *****************************************************************************/
44  void divide(int a,int b)
45  {
46      int result=0;
47      result=a/b;
48      printf("%d/%d=%d\n",a,b,result);
49      }
50
51      int main()
52      {
53          add(1,1);                    //求 1+1
54          subtract(10,1);              //求 10-1
55          multiply(2,4);               //求 2*4
56          divide(10,2);                //求 10/2
57          return 0;                    //返回
58      }
```

（2）编译运行程序，结果如图 3.11 所示。

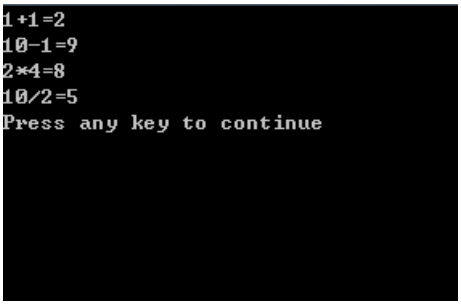

图 3.11　运行结果

在上面例 3-8 中，分别用函数 add()、subtract()、multiply()和 divide()实现了加、减、乘和除法四则运算功能，每个函数都使用相同名称的参数，在调用的时候使用的实际参数值不相等，但最后得到了正确的结果，这说明函数与函数之间的形参是没有关系的，即使名称相同也是如此。

前面提到过，在调用带参数的函数时有个规则就是必须准确无误的指定函数参数，也就是说在调用函数时，函数名、函数参数类型、函数参数顺序和函数参数个数必须和函数定义时保持一致，这样才能正确调用函数。

【例 3-9】 编写函数输出学生的相关信息。

（1）在 VC IDE 中创建工程 Function_Demo9，在 Source Files 中添加文件 main.c，编写代码如下。

```
01  #include <stdio.h>                          //包含库
02  /*****************************************************************************
```

```
03      函数名称              :   student
04  *   函数功能              :   输出学生信息
05  *   函数参数              :   name:   姓名
06  *                             score:   分数
07  *                             grade:   班级
08  ********************************************************************************/
09  void student(char* name,int score, char* grade)
10  {
11      printf("姓名: %s ", name);           //输出姓名
12      printf("分数: %d ", score);          //输出分数
13      printf("班级: %s\n", grade);         //输出班级
14  }
15  int main()
16  {
17      student("小明",80,"三年级");
18      student("小红",90,"四年级");
19      student("小刚",95,"四年级");
20      return 0;                            //返回
21  }
```

（2）编译运行程序，结果如图 3.12 所示。

图 3.12　运行结果

在例 3-9 中定义了函数 student()，该函数用来输出学生姓名、分数和年级，其参数有三个，且类型不完全一样，在调用该函数时，必须正确的输入实参，其类型、顺序和数量都必须与形参保持一致，例子中三个调用函数语句都遵循了这一规则。

3.3.3　形式参数与实际参数

在 3.3.1 和 3.3.2 节中已经初步介绍了形参和实参的关系，实际上，对于一个函数而言，函数只有通过被调用才会起作用，否则就是无用的。这里可以把函数看做成一个工具，如钳子或起子等，工具只有用于实际生活中，才能起作用，否则就是摆设。同样，在函数中的形式参数也是一样，正是因为形式参数只有通过调用然后值传递才有意义，所以被称为形式参数；而对于一个程序来说，实际参数是指正在调用函数时所用的数据，因此，实际参数具有实际意义。

通常情况下，关于形式参数和实际参数需要掌握二者之间的关系，如下例。

【例 3-10】 编写函数求 1 到 *n* 的和。

（1）在 VC IDE 中创建工程 Function_Demo10，在 Source Files 中添加文件 main.c，编写代码如下。

```
#include <stdio.h>                  //包含库
/*****************************************************************************
*    函数名称              :    sum
*    函数功能              :    求 n+（n-1）+(n-2)+(n-3)+.....+1
*    函数参数              :    n:求和的最大值
*****************************************************************************/
void sum(int n)
{
    int i;
    for(i=n-1;i>0;i--)                          //循环直到 i 为 0
        n=n+i;
    printf("求和后 n 的值为：%d\n",n);          //输出形参 n 的值
}
int main()
{
    int m=5;
    sum(m);
    printf("m 的值为：%d\n",m);                 //输出实参 m 的值
return 0;                                       //返回
}
```

（2）编译运行程序，结果如图 3.13 所示。

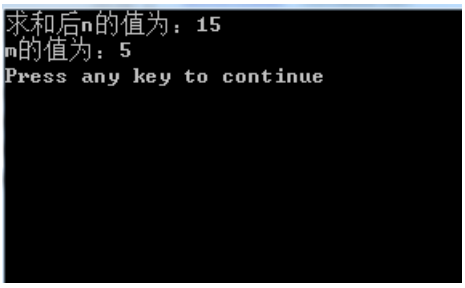

图 3.13　运行结果

在例 3-10 中，定义了求和函数 sum()，该函数中的形参变量 n 在函数调用过程中其值不断地在变化，最后值为 15，而实际参数变量 m 的值在调用 sum()函数前后的值大小是相同的。这说明，形参变量和实参变量之间是不会互相影响的。

在例 3-10 中，形参的改变并没有对实际参数造成任何影响，这里需要注意的是，在 C 语言中，实际参数和形式参数之间是值传递，所以二者之间没有本质的联系，即使两者同名，也不会有任何影响。

【例 3-11】编写函数求 1 到 *n* 的和。

（1）在 VC IDE 中创建工程 Function_Demo11，在 Source Files 中添加文件 main.c，编写代码如下。

```
#include <stdio.h>                  //包含库
/*****************************************************************************
*    函数名称              :    sum
*    函数功能              :    求 n+（n-1）+(n-2)+(n-3)+.....+1
*    函数参数              :    n:求和的最大值
*****************************************************************************/
void sum(int n)
```

```
08  {
09      int i;
10      for(i=n-1;i>0;i--)                          //循环直到 i 为 0
11          n=n+i;
12      printf("求和后 n 的值为：%d\n",n);          //输出形参 n 的值
13  }
14  int main()
15  {
16      int n=5;
17      sum(n);
18      printf("n 的值为：%d\n",n);                 //输出实参 n 的值
19  return 0;                                       //返回
20  }
```

（2）编译运行程序，结果如图 3.14 所示。

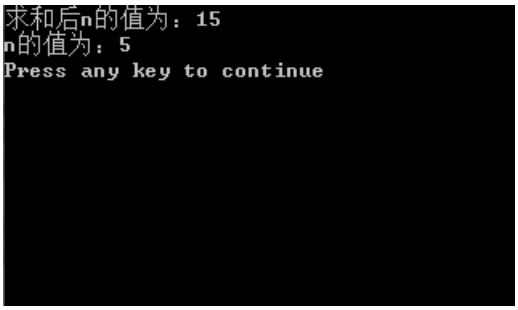

图 3.14　运行结果

例 3-11 和例 3-10 实际上是相同的，仅仅在例 3-11 中实际参数变量和形式参数变量的名称相同。在该例中同样可以看到，实际参数在函数调用前后没有发生任何变化，这说明即使形参和实参的名称相同，二者也是互不影响的。

3.4　函数的返回值

在 3.3 节中介绍了带参数的函数定义和调用，本节主要来介绍函数的返回值的相关知识。对于一个函数而言，其主要的意义是处理数据，然后返回结果，处理数据在函数体中进行，使用特定的算法来实现，而在 C 语言中返回结果一般就会通过特定的语句来返回。其具体使用方法在下面介绍。

3.4.1　带返回值的函数定义

在一个函数的调用过程中，如果数据处理完成，已经得到想得到的结果，这时就应将结果反馈给调用者，这个反馈过程是通过函数中的 return 语句实现的。当然，对于有些数据处理而言，是不需要返回结果给调用者的，这时可能就不需要 return 语句，所以函数会执行到函数体的最后一行代码。

对于一个带返回值的函数，其定义语法格式如下。

```
函数类型 函数名(类型 参数 1, 类型 参数 2, …)
{
    函数体代码;
    …
    return 返回值;
}
```

在使用 return 语句返回结果时需要注意的是，返回值的类型必须和函数类型保持一致，否则就会出现编译错误。

【例 3-12】 编写求和函数处理字符串返回第一个字符。

（1）在 VC IDE 中创建工程 Function_Demo12，在 Source Files 中添加文件 main.c，编写代码如下。

```
01  #include <stdio.h>              //包含库
02  /*****************************************************************************
04  *   函数名称             :   fun
05  *   函数功能             :   返回字符串的第一个字符
06  *   函数参数             :   str:待处理的字符串
07  *****************************************************************************/
09  char fun(char* str)
10  {
11      char result=str[0];                 //取首字符
12      return result;                      //返回结果
13  }
14  int main()
15  {
16      char* pstr="hello";
17      char firstChar=fun(pstr);
18      printf("%s 第一个字符是%c\n",pstr,firstChar);
    return 0;
}
```

（2）编译运行程序，结果如图 3.15 所示。

图 3.15　运行结果

在例 3-12 中定义了函数 fun()，该函数主要功能是将输入的字符串的首字符返回给调用者，在第 9 行就实现了取首字符的功能，这里返回值是 char 类型变量，而函数类型也是 char 类型变量。在函数定义时，必须正确的返回需要的结果，其类型必须与函数类型一致。

在实际程序开发中，一个函数内可能会有多个 return 语句，实际上，return 语句除

了有返回结果的功能，还能让程序直接返回，也就是说，当函数运行到某一行代码，这时候需要返回而不用继续往下运行的话，那么可以直接加上 return 语句，这时 return 语句后面不需要带上结果值，仅仅表示运行到此处，程序结束了。

【例 3-13】 编写函数判断给定年份是否是闰年。

```
#include <stdio.h>                          //包含库
/*****************************************************************************
*    函数名称             :    isLeap
*    函数功能             :    判断给定年份是不是闰年
*    函数参数             :    year:年份
*****************************************************************************/
char* isLeap(int year)
{
    if(year%4==0&&year%100!=0)      //能被 4 整除但不能被 100 整除就是闰年
        return "是闰年\n";
    else if(year%400==0)            //能被 400 整除就是闰年
        return "是闰年\n";
    else
        return "不是闰年\n";
}
int main()
{
    char* result;
    result=isLeap(2012);            //判断 2012 年是否是闰年
    printf("2012 年%s",result);
    result=isLeap(2013);            //判断 2013 年是否是闰年
    printf("2013 年%s",result);
return 0;
}
```

（2）编译运行程序，结果如图 3.16 所示。

图 3.16　运行结果

在例 3-13 中定义了函数 isLeap()，该函数实现了判断闰年的功能，在该函数体内有多个 return 语句，当函数运行到某个 return 语句就会直接跳过下面的语句，函数结束。可以看到，return 语句在这里不仅起到返回函数结果的作用，还起到了提前结束函数的功能。

3.4.2　带返回值的函数调用

带返回值的函数调用和其他函数调用一样，指定正确的函数名和函数参数即可。但

是对于带返回值的函数来说，需要注意的是该函数会返回一个调用结果值，因此可以将该函数当做一个数值赋值给对应的变量，或者继续作为实际参数传递进行函数调用，当然也可以对其返回值置之不理。

一般情况下，对于函数调用的返回值要根据需要来进行使用。

【例 3-14】 编写函数来计算给定参数的平均值。

```
01  #include <stdio.h>              //包含库
02  /*****************************************************************************
03  *    函数名称              :    average
04  *    函数功能              :    计算数 1 和数 2 的平均数
05  *    函数参数              :    a：数 1
06  *                               b：数 2
07  *****************************************************************************/
08  int average(int a,int b)
09  {
10     int result=(a+b)/2;
11     return result;
12  }
13  int main()
14  {
15      int result;
16      result=average(200,132);
17      printf("200 和 132 的平均数是%d\n",result);
18      result=average(1007,13);
19      printf("1007 和 13 的平均数是%d\n",result);
20
21  return 0;
22  }
```

（2）编译运行程序，结果如图 3.17 所示。

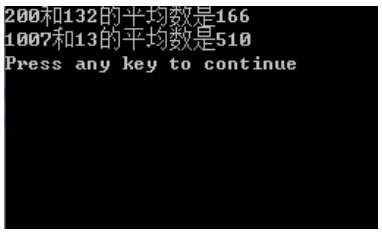

图 3.17　运行结果

在例 3-14 中 average()函数用于计算平均值，在第 16 和第 18 行的两行代码调用了 average()函数来计算平均值，其返回值为 int 类型，然后将返回值赋值给 int 类型变量 result。

3.5　开 发 实 践

【例 3-15】 求输入的整数 *n* 各位数字之积。

完整程序如下：

```
01  文件名：3-15.c
02  #include <stdio.h>
03  #include <stdlib.h>
04  long func(long num)             /*定义 func()函数*/
05  {
06      long k=1;
07      num=labs(num);              /*取 num 的绝对值*/
08      do                          /*求 num 各位的乘积*/
09      {
10          k*=num%10;
11          num/=10;
12      }while(num);
13      return(k);                  /*返回结果*/
14  }
15  main()
16  {
17      long n;
18      printf("\nPlease enter a number:");
19      scanf("%ld",&n);            /*读入 n 值*/
20      printf("\nThe product of its digits is %ld.",func(n));
                                                    /*调用 func()函数*/
21  }
```

程序运行结果为：

```
Please enter a number:12345

The product of its digits is 120.
Press any key to continue . . .
```

对程序的说明如下：

- 程序第 4 行定义了 func()函数，该函数包含了一个形式参数 num，函数的作用是计算 num 的各位数字之积。
- 程序第 7 行使用了 labs()函数，该函数的作用是求出长整型数的绝对值。
- 程序第 8 行~第 12 行使用了一个 do...while 语句，这是一个循环语句。该语句用来计算 num 各个位上的数值，并将它们依次相乘。
- 程序第 13 行使用 return 语句返回函数值。
- 程序第 15 行是 main()函数，在 main()函数中首先定义了 long 型变量 n，接着在第 19 行使用 scanf()函数读入 n 的值。
- 程序第 20 行使用 printf 语句输出计算结果，注意到，在 printf 语句中调用了 func()函数，同时将 n 作为实际参数传递给 func()函数。

【例 3-16】 编程输出下面的图形。

```
   **
  ****
 ******
********
 ******
  ****
   **
```

完整程序如下：

```
01  文件名：3-16.c
02  #include <stdio.h>
03  void a(int i)
04  {
05      int j,k;
06      for(j=0;j<=7-i;j++)
07          printf(" ");            /*输出空格*/
08      for(j=0;j<=2*i+1;j++)
09          printf("*");            /*输出“*”*/
10      printf("\n");
11  }
12  main()
13  {
14      int i;
15      for(i=0;i<3;i++)
16          a(i);                   /*调用 a()函数*/
17      for(i=3;i>=0;i--)
18          a(i);                   /*调用 a()函数*/
19  }
```

程序中定义了 a()函数，该函数用于输出一行。在 main()函数中使用两个 for()函数，这两个 for()函数都循环调用了 a()函数，每次输出一行，一共输出了 7 行。

【例 3-17】 产生 3 个小于 10 的随机整数，计算它们阶乘的和，形式为 x !+y !+z !。

完整程序如下：

```
文件名：3-17.c
#include <stdio.h>
#include <stdlib.h>
long factorial(int);                /*函数声明*/
main()
{
    int i,j;
    long sum;
    sum=0;
    for(i=0;i<=2;i++)
    {
        j=rand();                   /*产生随机数*/
        j=j%10;                     /*取随机数的各位*/
        printf("%2d!+",j);          /*函数调用*/
        sum=sum+factorial(j);
    }
    printf("\b=%ld",sum);
}
long factorial(int i)               /*函数定义，求 i 的阶乘*/
{
    long product=1;
    for(;i>=1;i--)
        product=product*i;          /*计算乘积*/
    return product;                 /*返回乘积*/
}
```

程序运行结果为：

```
1!+ 7!+ 4!=5065
Press any key to continue . . .
```

对程序的说明如下：

在 main()函数中调用了库函数 rand()，rand()函数可以产生一个小于 32767 的随机数，根据题意，对该随机数取 10 的模。factorial()函数是一个自定义函数，用于求出一个数的阶乘。

3.6　小　　结

在 C 语言中，函数的重要性不言自明，只要是 C 语言程序，都会涉及到函数的相关知识。即使一个程序再小，其程序入口只有一个 main()函数。因此，函数部分的内容应该重点掌握。本章主要介绍了函数的定义和调用的相关知识，对函数的各种用法都做了基本的介绍，这在以后的程序设计学习中是非常重要的。

3.7　习　　题

一、选择题

1．建立函数的目的之一是（　　）。

（A）提高程序的执行效率　　（B）提高程序的可读性

（C）减少程序的篇幅　　（D）减少程序文件所占内存

2．以下说法正确的是（　　）。

（A）用户若需调用标准库函数，则调用前必须重新定义

（B）用户可以重新定义标准库函数，若如此，则该函数将失去原有的意义

（C）系统不允许用户重新定义标准库函数

（D）用户若需调用标准库函数，调用前可以不使用预编译命令将该函数所在文件包含到用户源文件中，系统会自动去调用

3．下面的函数定义形式正确的是（　　）。

（A）double fun(int x,int y)　　（B）double fun(int x;int y)

（C）double fun(int x,int y);　　（D）double fun(int x,y);

4．以下正确的函数形式为：（　　）。

（A）

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

（B）

```
fun(int x,y)
{
    int z;
```

```
    return z;
}
```

（C）

```
fun(x,y)
{
    int x,y;
    double z;
    z=x+y;
    ruturn z;
}
```

（D）

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

5．以下正确的说法是（　　）。

（A）实参和与其对应的形参各占用独立的存储单元

（B）实参和与其对应的形参共占用一个存储单元

（C）只有当实参和与其对应的形参同名时才共同占用一个存储单元

（D）形参是虚拟的，不占用存储单元

6．若调用一个函数，且此函数中没有 return 语句，则该函数（　　）。

（A）没有返回值　　（B）返回若干个系统默认值

（C）能返回一个用户所希望的函数值　　（D）返回一个不确定的值

7．以下说法正确的是（　　）。

（A）定义函数时，形参的类型说明可以放在函数体内

（B）return 后边的值不能为表达式

（C）如果函数值的类型与返回值类型不一致，以函数值类型为准

（D）如果形参与实参的类型不一致，以实参类型为准

8. C 语言规定，简单变量作实参时，它和对应的形参之间的数据传递方式是（　　）。

（A）地址传递　　（B）单向值传递

（C）由实参传给形参，再由形参传回给实参　　（D）由用户指定传递方式

9．C 语言允许函数值类型缺省定义，此时该函数值隐含的类型是（　　）。

（A）float 型　　（B）int 型　　（C）long 型　　（D）double 型

10．C 语言规定，函数返回值的类型是由（　　）。

（A）return 语句中的表达式类型所决定

（B）调用该函数时的主调用函数类型所决定

（C）调用该函数时系统临时决定

（D）在定义该函数时所指定的函数类型所决定

11．下面函数调用语句含有实参的个数为（　　）。

```
func((exp1,exp2),(exp3,exp4,exp5));
```

（A）1　　（B）2　　（C）4　　（D）5

12．以下程序是选出能被 3 整除且至少有一位是 5 的两位数，打印出所有这样的数及其个数，并选择填空。

```
sub(int k, int n)
{
    int a1,a2;
    a2=【1】;
    a1=k-【2】;
    if(k%3= =0&&a2= =5)||(k%3= =0&&a1= =5))
{
    printf("%d",k);
    n++;
    return n;
}
else
    return n;
}
main()
{
    int n=0,k,m;
    for(k=10;k<=99;k++)
    {
        m=sub(k,n);
        if(m!=-1)
            n=m;
}
printf("\nn=%d",n);
}
```

【1】

（A）k*10　　（B）k%10　　（C）k/10　　（D）k*10%10

【2】

（A）a2*10　　（B）a2　　（C）a2/10　　（D）a2%10

13．以下程序的功能是计算函数 F(x,y,z)=(x+y)/(x-y)+(z+y)/(z-y)的值，请选择填空。

```
#include <stdio.h>
#include <math.h>
float f(float, float)
main()
{
    float x,y,z,sum;
    scanf("%f%f%f",&x,&y,&z);
    sum=f(【1】)+f(【2】);
    printf("sum=%f\n",sum);
}
float f(float a, float b)
{
    float vluae;
value=a/b;
return(value);
}
```

【1】

（A）x-y,x+y　　（B）x+y,x-y　　（C）z+y,z-y　　（D）z-y,z+y

【2】

（A）x-y,x+y　　（B）x+y,x-y　　（C）z+y,z-y　　（D）z-y,z+y

二、填空题

1. 在 C 语言中，一个函数一般由两部分组成，它们分别是_____和_____。

2．若输入的值为-125，那么下面程序的运行结果是_____。

```
#include <math.h>
main()
{
    int n;
    scanf("%d",&n);
    printf("%d=",n);
    if(n<0)
        printf("-");
    n=fabs(n);
    fun(n);
}
fun(int n)
{
    int k,r;
    for(k=2;k<=sqrt(n);k++)
    {
        r=n%k;
        while(r= =0)
        {
            printf("%d",k);
            n=n/k;
if(n>1)
    printf("*");
r=n%k;
}
}
if(n!=1)
    printf("%d\n",n);
}
```

3．下面程序的运行结果是：_____。

```
#include <stdio.h>
main()
{
    int i=2,x=5,j=8;
    fun(j,5);
    printf("i=%d;j=%d;x=%d\n",i,j,x);
}
fun(int i,int j)
{
    int x=7;
    printf("i=%d;j=%d;x=%d\n",i,j,x);
}
```

4．下面程序的运行结果是：_____。

```
#include <stdio.h>
main()
{
```

```
    increment();
    increment();
    increment();
}
increment()
{
    int x=0;
    x+=2;
    printf("%d",x);
}
```

5．下面程序的运行结果是：_____。

```
#include <stdio.h>
main()
{
    int a=2,b=5,c;
    c=max(a,b);
    printf("max is %d\n",c);
}
max(int x,int y)
{
    int z;
    z=(x>y)?x:y;
    return (z);
}
```

三、编程题

1．已有如下的变量定义和函数调用语句：

int a=1,b=-5,c;

c=fun(a,b);

fun()函数的作用是计算两个数之差的绝对值，并将差值返回调用函数，请编写 fun()函数。

fun(int x,int y)

{　　}

2．已有变量定义和函数调用语句：

int x=57;

isprime(x);

函数 isprime()用来判断一个整型数 *x* 是否为素数，若是素数，则函数返回 1，否则返回 0。请编写 isprime()函数。

3．已有变量定义语句：

double a=5.0;

int n=5;

和函数调用语句 mypow(a,n);用以求 a 的 n 次方。请编写 double mypow(double x,int y)函数。

4．编写子函数：（1）用冒泡法将一个数组排成升序的函数——SUB1；（2）在升序数组中插入一个数，并且保持该数组仍为升序数组的函数——SUB2。主函数：①输入

任意 10 个正整数给数组；②调用 SUB1 对数组进行排序；③从键盘输入一个正整数，调用 SUB2 将其插入该数组。

5．编写函数：（1）用选择法将数组排成降序的函数——SUB1；（2）用折半查找法查找某数是否在给定的数组当中的函数——SUB2。主函数：输入任意 10 个正整数给数组，调用 SUB1 对数组进行排序，从键盘输入一个正整数，调用 SUB2 在数组中进行查找，找到后输出“OK”，没有找到则输出“NO FOUND！”。

6．编写一个求 X 的 Y 次幂的递归函数，X 为 double 型，Y 为 int 型，要求从主函数输入 X，Y 的值，调用函数求其幂。