# 第5章　函　　数

在程序中引入函数是软件技术发展史上重要的里程碑之一，它标志着软件模块化和软件重用的真正开始。在进行程序设计时将一个项目按照功能划分为若干小的功能模块，每个模块完成一个确定的功能，在这些模块之间建立必要的联系，互相协作完成整个项目要完成的功能，这种方法称为模块化程序设计。通常规定模块只有一个入口和出口，使用模块的约束条件是入口参数和出口参数。模块的划分有利于团队开发，各个模块由不同的程序员开发，只要明确模块之间的接口关系，模块内部细节的具体实现可以由程序员自己设计，而模块之间不受影响。在 C 程序设计中，用函数实现这些功能模块。读者在学习函数的过程中，注意锻炼自己的工程项目分析能力和项目管理能力，同时加强自己的团队精神及合作能力的训练。

## 5.1　函 数 引 例

**【例 5-1】** 编写一个自定义函数并用它求 $x^y$ 的值。

编程提示：计算 $x^y$，就是 $y$ 个 $x$ 相乘所得的乘积，可以利用循环实现。

```
#include <stdio.h>
double mypow(double x,double y)
{
   double z=1.0;
   int i=0;
   for (i=1; i<=y; i++)  z=z*x;
   return  z;
}
int main()
{
   double a=0,b=0,c=0;
   printf("Input x y:");
   scanf("%lf%lf",&a,&b);
   c=mypow(a,b);
   printf("%.3lf,%.3lf,%.3lf\n",a,b,c);
   return 0;
}
```

程序运行结果如下：

```
Input x y:1.01 365↙
1.010,365.000,37.783
Input x y:1 365↙
1.000,365.000,1.000
```

```
Input x y:0.99 365↙
0.990,365.000,0.026
```

程序运行结果分析如下：

（1）1.01 的 365 次方等于 37.783>1，1.01=1+0.01，也就是每天进步一点点，一年后，自己的进步很大，远远大于“1”；

（2）1 的 365 次方等于 1，1 是原地踏步，一年后还是原地踏步，还是那个“1”；

（3）0.99 的 365 次方等于 0.026，0.99=1−0.01，也就是每天退步一点点，一年后将远远小于“1”，远远被人抛在后面，将“1”事无成。

程序说明：

mypow()是用户编写函数，称为自定义函数。在前面章节的例题中用到的 printf()、scanf()、sqrt()、pow()等函数由 C 系统提供，称为库函数，用户无须定义。

**【同步练习】** 编写一个求前 *n* 个自然数和的函数，然后在 main()函数中调用它，求 1+2+3+…+100 的值。

## 5.2 函数的定义与调用

### 5.2.1 函数的定义

函数在使用之前必须进行正确的定义，定义时必须根据函数需要完成的工作，确定函数名、类型名、参数个数及类型等。

**【例 5-2】** 编写一个求两个整数和的函数，并在 main()函数中调用。

```
#include <stdio.h>
int GetTwoSum(int a,int b)
{
   int z;
   z=a+b;
   return z;
}
int main()
{
   int x=3,y=4,z;
   z=GetTwoSum(x,y);
   printf("z=%d\n",z);
   return 0;
}
```

程序运行结果如下：

```
z=7;
```

定义一个求两个整数和的函数 GetTwoSum()，在 main()函数中调用该函数时，将 x、y 的值分别传递给 GetTwoSum()函数的 a、b，经过计算，将计算结果 7 返回 main()函数。

函数定义的一般形式如下：

```
类型名  函数名([类型名 形式参数1,类型名 形式参数2,…])
{
    说明语句
    执行语句
}
```

其中，“类型名　函数名([类型名 形式参数 1，类型名 形式参数 2，…])”为函数头。

（1）类型名是指函数返回值类型，含义是调用该函数后返回值的类型，指明了本函数的类型，例 5-2 中的 int GetTwoSum(int a, int b)中的 int 表示调用该函数后返回的两个整数的和为 int 类型；如果函数没有返回值用 void 类型，则表示该函数没有返回值。

（2）函数名是由用户定义的标识符。注意，为了提高程序的可读性，函数名应尽量反映函数的功能，如函数名 GetTwoSum 表示求两个数的和。

（3）用“[ ]”括起来的部分是形式参数列表，“[ ]”表示该项是可选项，可以有，也可以没有，如果没有该项表示该函数是无形式参数函数，称为无参函数，注意，没有形式参数时，函数名后面的“( )”不能少；有形式参数的函数，称为有参函数，注意，每个形式参数必须单独定义，且各个参数之间用“,”分隔开，如例 5-2 中的 GetTwoSum 函数。

（4）用“{ }”括起来的部分称为函数体，函数体包含了实现函数功能的所有语句，是函数实现的细节。函数体中的说明语句用于定义函数中所用的变量，如例 5-2 中的 GetTwoSum 函数中的“int z;”。执行语句用于实现函数所要完成的功能，如例 5-2 中的 GetTwoSum 函数中的“z=a+b;”。

（5）如果调用函数后需要函数值，则在该函数名前给出该函数的类型，并且在函数体中用 return 语句将函数值返回如例 5-2 中的 GetTwoSum 函数名前的“int”与函数体中的“return z;”。

**【同步练习】** 编写一个求 *n* 个自然数和的函数。

### 5.2.2　函数调用

函数只有在调用时才会发挥它的作用，调用自己编写的函数与调用库函数的方法是一样的。

**【例 5-3】** 编写一个求两个整数的最大值的函数。

编程提示：注意定义函数时形参的个数、类型，函数的类型应与 return 后面表达式的类型一致。

```
#include <stdio.h>
int max(int a,int b)
{
   if (a>b)
      return a;
   else
      return b;
}
int main()
```

```
{
    int x,y,z;
    printf("Input x y:");
    scanf("%d%d",&x,&y);
    z=max(x,y);          /*调用 max 函数*/
    printf("\nmax=%d",z);
    return 0;
}
```

程序运行结果如下：

```
Input x y:3 8↙
max=8
```

程序说明：主函数中

```
z=max(x,y);
```

的作用是求两个整数中的较大值，由于库函数中没有提供该功能，因此在调用之前由用户自己定义该函数。

**1. 函数调用的一般形式**

函数调用的一般形式如下：

```
函数名([参数表])
```

对无参函数调用时则无实际参数表。实际参数表中的参数可以是常数、变量或其他构造类型数据及表达式。各实参之间用“,”分隔。

**2. 函数调用的方式**

（1）函数表达式。函数作为表达式中的一项出现在表达式中，以函数返回值参与表达式的运算。这种方式要求函数是有返回值的。例如：

```
z=max(x,y)
```

（2）函数语句。函数调用的一般形式加上分号即构成函数语句。例如：

```
printf("%d",a);
```

（3）函数实参。函数作为另一个函数调用的实际参数出现。这种情况是把该函数的返回值作为实参进行传送，因此要求该函数必须是有返回值的。例如：

```
printf("%d",max(x,y));
```

**3. 在主调函数调用某函数之前应对该被调函数进行说明（声明）**

形式如下：

```
类型说明符号 被调函数名(类型形参,类型形参,…);
```

或

```
类型说明符号 被调函数名(类型,类型,…);
```

【例 5-4】 编写一个自定义函数并用它求任意两个数的和。

编程提示：注意定义函数时形参的个数、类型，函数的类型与 return 后面表达式的类型一致。

```
#include <stdio.h>
int main()
{
   float add(float,float);  /*在主函数中对被调用函数进行说明.*/
   float a,b,c;
   scanf("%f%f",&a,&b);
   c=add(a,b);
   printf("sum is %f",c);
   return 0;
}
float add(float x,float y)
{
   float z;
   z=x+y;
   return(z);
}
```

程序运行结果如下：

```
34 78.9↙
sum is 112.900000
```

**4**. 函数调用注意事项

（1）调用库函数时必须将与该库函数相关的头文件用 include 命令包含在源文件前面，例如调用 printf()、scanf()库函数用#include <stdio.h>，调用 sqrt()、pow()库函数用#include <math.h>。

（2）调用用户自定义函数有两种方式。

① 用户自定义函数定义在函数调用的后面，必须在调用该函数的前面进行声明，如例 5-4 所示，推荐使用这种方式，优点是当程序比较长时，自定义函数声明集中写在 main()函数前面，而函数定义在调用它的后面，一般集中写在 main()函数后面，这样程序阅读者很容易知道该程序有多少个函数，而且很容易找到 main()函数。

② 用户自定义函数定义在函数调用的前面，就不用在函数调用前声明了，这种方式一般在程序行数比较少时使用，如例 5-3 所示。

### 5.2.3 形式参数和实际参数

**1**. 在函数调用时要注意函数形式参数与实际参数的关系

形式参数（简称形参）必须在函数头中定义，在整个函数体内都可以使用，离开该函数后则不能使用。实际参数（简称形参）出现在函数调用时函数名后面的“( )”内。

形参和实参的功能是作数据传送。发生函数调用时，主调函数把实参的值传送给被调

函数的形参，从而实现主调函数向被调函数的数据传送。

**2. 函数的形参和实参的特点**

（1）形参变量只有在被调用时才分配内存单元，在调用结束时，即刻释放所分配的内存单元。因此，形参只有在函数内部有效。函数调用结束返回主调函数后则不能再使用该形参变量。

（2）实参可以是常量、变量、表达式、函数等，无论实参是何种类型的量，在进行函数调用时，它们都必须具有确定的值，以便把这些值传送给形参。

（3）实参和形参在数量上、类型上、顺序上应严格一致，否则会发生类型不匹配的错误。

（4）函数调用中发生的数据传送是单向的。即只能把实参的值传送给形参，而不能把形参的值反向地传送给实参。因此在函数调用过程中，形参的值发生改变，而实参中的值不会变化。

**【例 5-5】** 参数值的传递。

编程提示：体会函数调用时实参向形参传值是单向的。

```
#include <stdio.h>
void s(int n)
{
   int i;
   for (i=n-1;i>=1;i--)
   n=n+i;
   printf("n=%d\n",n);
}

int main()
{
   int n;
   printf("input number:");
   scanf("%d",&n);
   s(n);
   printf("n=%d\n",n);
   return 0;
}
```

程序运行结果如下：

```
input number:100
n=5050
n=100
```

本程序中定义了一个函数 s()，该函数的功能是求 n 个自然数的和。在 main()函数中输入 n 值，并作为实参，在调用时传送给 s 函数的形参变量 n（注意，本例的形参变量和实参变量的标识符都为 n，但这是两个不同的变量，各自的作用域不同）。在 main()函数中用 printf()输出一次 n 值，这个 n 值是实参 n 的值。在函数 s()中也用 printf()输出了一次 n 值，这个 n

值是形参最后取得的 n 值 5050。从运行情况看，输入 n 值为 100，即实参 n 的值为 100。把此值传给函数 s()时，形参 n 的初值也为 100，在执行函数过程中，形参 n 的值变为 5050。返回 main()函数之后，输出实参 n 的值仍为 100。可见实参的值不随形参的变化而变化。

### 5.2.4 函数的返回值

函数的返回值是指函数被调用之后，执行函数体中的程序段所取得的并返回给主调函数的值。对函数返回值有以下一些说明。

① 函数返回值只能通过 return 语句返回主调函数。

return 语句的一般形式如下：

```
return 表达式;
```

或

```
return(表达式);
```

该语句的功能是计算表达式的值，并返回给主调函数。在函数中允许有多个 return 语句，但每次调用只能有一个 return 语句被执行，因此只能返回一个函数值。

② 函数返回值类型和函数定义中函数的类型应保持一致。如果两者不一致，则以函数类型为准，自动进行类型转换。

③ 如函数返回值为整型，在函数定义时可以省去类型说明。

④ 不返回函数值的函数，可以明确定义为“空类型”，类型说明符为 void。如函数 s() 并不向主函数返回函数值，可定义为

```
void s(int n)
{
    …
}
```

函数一旦被定义为空类型，就不能在主调函数中使用被调函数的函数值了。例如，在定义 s 为空类型后，在主函数中写下述语句：

```
sum=s(n);
```

就是错误的，为了使程序有良好的可读性并减少出错，凡不要求返回值的函数都应定义为空类型 void。

**【同步练习】**

1. 编写一个求圆面积的函数，并在 main()函数中调用。

2. 编写判断一个整数是否是素数的函数，并在 main()函数中调用，求 100～200 的所有素数。

## 5.3 函数的嵌套调用与递归

利用函数的嵌套和递归可以解决一些相对复杂的问题。

### 5.3.1 函数的嵌套调用

**1**. 函数的嵌套调用的概念

在调用一个函数的过程中又调用了另外一个函数称为函数的嵌套调用。

**2**. 函数的嵌套调用过程

函数嵌套调用时，两个函数之间的关系如图 5-1 所示。执行 main()函数中调用 a()函数的语句时，即转去执行 a()函数，在 a()函数中调用 b()函数时，又转去执行 b()函数，b()函数执行完毕返回 a()函数的断点继续执行，a()函数执行完毕返回 main()函数的断点继续执行。

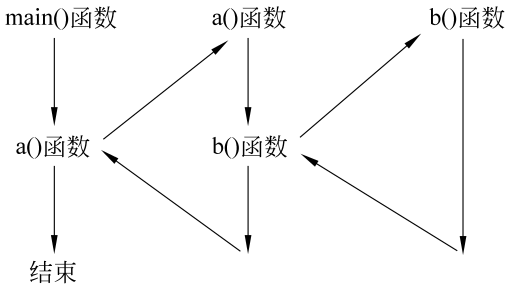


图 5-1　函数的嵌套调用

**【例 5-6】** 编写一个函数计算 $s=2^2!+3^2!$。

编程提示：编写两个函数，一个是用来计算平方值的函数 f1()，另一个是用来计算阶乘值的函数 f2。主函数先调 f1()计算出平方值，再在 f1()中以平方值为实参，调用 f2()计算其阶乘值，然后返回 f1()，再返回主函数，在循环程序中计算累加和。

程序如下：

```
#include <stdio.h>
long f1(int p)
{
   int k;
   long r;
   long f2(int);
   k=p*p;
   r=f2(k);
   return r;
}
long f2(int q)
{
   long c=1;
   int i;
   for (i=1;i<=q;i++)
   c=c*i;
   return c;
}
int main()
{
   int i;
```

```
    long s=0;
    for (i=2;i<=3;i++)
       s=s+f1(i);
    printf("\ns=%ld\n",s);
    return 0;
}
```

程序运行结果如下：

```
s=362904
```

在程序中，函数 f1()和 f2()均为长整型，都在主函数之前定义，故不必再在 main()函数中对 f1()和 f2()加以说明。在主程序中，执行循环程序依次把 $i$ 值作为实参调用函数 f1()求 $i^2$ 的值。在 f1()中又发生对函数 f2()的调用，这时是把 $i^2$ 的值作为实参去调用 f2()，在 f2()中完成求 $i^2$!的计算。f2()执行完毕把 C 值(即 $i^2$!) 返回给 f1()，再由 f1()返回 main()函数实现累加。

### 5.3.2 函数的递归调用

**1**. **函数的递归调用概念**

一个函数在它的函数体内直接或间接调用它自身称为递归调用，这种函数称为递归函数。

C 语言允许函数的递归调用。在递归调用中，主调函数又是被调函数。执行递归函数将反复调用其自身，每调用一次就进入新的一层。

例如有函数 f()如下：

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

这个函数是一个递归函数。但是运行该函数将无休止地调用其自身，这当然是不正确的。

**2**. **递归调用的条件**

为了防止递归调用无终止地进行，必须在函数内有终止递归调用的条件，满足某种条件后就不再作递归调用，然后逐层返回。

下面举例说明递归调用的执行过程。

**【例 5-7】** 求第 5 个人的年龄。第 5 个人自称比第 4 个人大 2 岁。问第 4 个人有多大，他自称比第 3 个人大 2 岁。问第 3 个人有多大，他自称比第 2 个人大 2 岁。问第 2 个人有多大，他自称比第 1 个人大 2 岁。问第 1 个人有多大，他说 10 岁。第 5 个人的年龄是多少？

编程提示：经过分析可得

$$\text{age}(n)=\begin{cases}10, & n=1\\ \text{age}(n-1)+2, & n>1\end{cases}$$

即

age(5) =age(4)+2；
age(4) =age(3)+2；
age(3) =age(2)+2；
age(2) =age(1)+2；
age(1) =10。
程序如下：

```
#include <stdio.h>
int age(int  n)
{
   int c;
   if (n==1)
      c=10;
   else
      c=age(n-1)+2;
   return  c;
}
int  main()
{
   printf("%d\n",age(5));
   return 0;
}
```

程序运行结果如下：

```
18
```

**3. 递归调用函数的通用格式**

递归调用函数一般由 if…else 条件语句组成，其格式如下：

```
if (终止条件)
    发生终止值
   else
   递推公式
```

**【例 5-8】** 编写一个函数，用递归方法求 *n*!。

编程提示：分析可得

$$n!=\begin{cases}1, & n=1\\ (n-1)*n, & n>1\end{cases}$$

程序如下：

```
#include <stdio.h>
int fac(int  n)
{
   int f;
   if ((n==0)||(n==1))
```