第 5 章

函数

在程序中引入函数是软件技术发展历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，把一个大的问题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，互相协作完成整个程序要完成的功能，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一个出口，使用模块的约束条件是入口参数和出口参数。选择不同的模块或模块的不同组合就可以完成不同的系统架构和功能。这些功能模块不仅可以实现共享，并且由于功能单一容易保证设计的逻辑正确性。此外，问题划分以后更适合项目的集体开发，各个模块分别由不同的程序员编写，只要明确模块之间的接口关系，模块内部细节的具体实现就可由程序员自己随意设计，而模块之间不受影响。

5.1 函数的引出

在之前的程序设计中，已经多次使用系统提供的函数，如使用 printf 函数实现数据的输出，用 sqrt 函数进行开方运算等。这些函数功能单一，使用方便，有效地减少了程序设计的工作量。实际上，我们也可以编写实现具体功能的函数（自定义函数），使程序结构清晰，便于共享。

【问题描述 5.1】 信息学院二年级有 5 个班，输入学生的英语四级成绩，计算各班的平均分，找出 5 个班中的最高平均分；找出该年级中英语四级成绩的最高分。

分析： 先计算每个班的平均成绩，放入包含 5 个元素的数组；再找出每个班的最高分，放入另一个包含 5 个元素的数组。然后分别进行比较，找出平均分最高的班级和分数最高的学生。

算法描述：

```
step1:定义变量 Class[50], ClaAverage[5],ClaMax[5], AverMax, AllMax
step2:for(i=1; i<=5; i++)              //分别对 5 个班级进行以下操作
      {   ① 输入第 i 个班中每个同学的成绩, 将其存放到 Class[50]
          ② 计算第 i 个班的平均分 Average,将其存放到 ClaAverage[i]中
          ③ 找出第 i 个班中的最高分 max,将其存放到 ClaMax[i]中
      }
step3:在数组 ClaAverage[5]中找出最高分,将其存放到 AverMax 中
```

```
step4:在数组 ClaMax[5]中找出最高分,将其存放到 AllMax 中
step5:输出 AverMax、AllMax
```

该算法描述的还不够详细,需要将计算平均成绩及几个查找最高分的算法进行细化。这就是结构化程序设计的基本思想:由上到下,逐步求精。

注意,算法中多次用到查找最高分。不管有多少数据,可以采用相同的查找算法。因此,可以编写一个实现查找最高分功能的函数,像 C 语言的内部函数一样,在使用时调用该函数,不仅可以有效地减少程序设计的工作量,还可以使程序结构更清晰,增强可读性。如果再将数据输入模块、计算平均分模块也编写为独立函数,整个程序结构就很清晰了。

下面从一个比较简单的问题开始,说明如何编写自定义函数。

【例 5.1】 编程求解 $C_n^m=\dfrac{n!}{m!\ (n-m)!}$。

分析: 求解上面公式最重要的就是计算阶乘,求一个数的阶乘可以用一层循环来实现,分别计算出 n!、m!、(n-m)!,再按公式进行除法运算即可得到结果。

算法描述:

```
step1:输入 m 和 n
step2:计算 n!
step3:计算 m!
step4:计算(n-m)!
step5:计算 n!/(m!*(n-m)!)
step6:输出计算结果
```

在没学习函数之前,可能会编写出下面这样的程序。

```
1.  #include<stdio.h>
2.  int main()
3.  {
4.      int m, n, i;
5.      double c, c1, c2, c3;                    //阶乘的值较大,将变量定义为 double 型
6.      printf("input m, n:");
7.      scanf("%d%d", &m, &n);
8.      for(i=1, c1=1; i<=n; i++)   c1=c1*i;          //计算 n 的阶乘
9.      for(i=1, c2=1; i<=m; i++)   c2=c2*i;          //计算 m 的阶乘
10.     for(i=1, c3=1; i<=(n-m); i++)   c3=c3*i;      //计算 n-m 的阶乘
11.     c=c1/(c2*c3);
12.     printf("c=n!/(m!*(n-m)!)=%.2lf\n", c);
13.     return 0;
14. }
```

说明: 上面的程序中,需要 3 次计算阶乘,除了阶乘的次数不同外,代码是一样的。因此可以将计算阶乘这样的通用代码编写为函数,下面给出使用函数编写的程序。

例 5.1 的参考程序如下。

```
1.  #include<stdio.h>
2.  double factorial (int x)              //定义求阶乘的函数
3.  {
4.      int i;   double f=1.0;
```

```
    for(i=1; i<=x; i++)                    //计算 x 的阶乘
    {   f=f*i;   }
    return f;                              //返回计算结果
}
int main()
{
    int m, n;   double c,c1,c2,c3;
    printf("input m, n:");
    scanf("%d%d", &m, &n);
    c1=factorial (n);                      //第 1 次调用 factorial 函数,计算 n 的阶乘
    c2=factorial (m);                      //第 2 次调用 factorial 函数,计算 m 的阶乘
    c3=factorial (n-m);                    //第 3 次调用 factorial 函数,计算 n-m 的阶乘
    c=c1/(c2*c3);
    printf("c=n!/(m!*(n-m)!)=%.2lf\n", c);
    return 0;
}
```

说明：① 在C语言中，除main函数外，所有函数都是平行的。函数间相互独立，一个函数并不从属于另一个函数，即函数不能嵌套定义。但是，函数之间可以相互调用。

② 一个主调函数可调用多个被调函数，同一个函数也可被一个或多个函数调用任意次。

③ 从用户使用角度看，函数分为标准函数和用户自定义函数两类。标准函数即库函数，它是由系统提供的，用户通过#include命令可以直接使用它们。用户自定义函数是用户根据需要自己定义的函数。

5-1 函数定义

5.2 函数定义与调用

5.2.1 函数的定义与调用

1. 函数定义

函数定义的一般形式：

```
数据类型 函数名(数据类型 形式参数 1,数据类型 形式参数 2,…,数据类型 形式参数 n)
{
    <说明语句>
    <执行语句>
}
```

说明：① 格式行首的数据类型是指函数返回值的数据类型；函数返回值就是函数运行产生的结果，用于返回给主调函数。

② 函数名，为函数起的名字，应符合标识符的命名规则。

③ 形式参数表，每个形式参数都要定义数据类型，各参数间用逗号分开。

④ 花括号“{ }”部分是函数的函数体，其编写方法与主函数一样。

5-2 函数调用

2. 函数调用

函数调用的一般形式：

```
函数名(实际参数 1,实际参数 2,…,实际参数 n);
```

说明： ① 函数调用时，如果有多个实际参数，则各参数间用逗号分开。

② 实际参数是传递给被调用函数的值，可以是常量、变量、表达式或函数调用，但是其值必须是确定的。实际参数和形式参数应保持数据类型一致，顺序一一对应。

【例 5.2】 编写函数求两个数中较大的数，要求在 main 函数中输入数据和输出结果。

分析： 比较两个数，选出其中较大的数的过程可以用函数 Max 实现，而输入两个数和输出结果可在 main 函数中实现。考虑 Max 函数中用于比较的两个数需要从 main 函数中获得，因此应该定义两个参数，而 Max 函数中求得的较大数也必须返回给 main 函数输出。

例 5.2 的参考程序如下。

```
1.  #include<stdio.h>
2.  int Max(int a, int b)                    //函数定义,a 和 b 为形参,返回值为 int 型
3.  {
4.      int c;
5.      if (a>b)   c=a;
6.      else   c=b;
7.      return c;                            //返回变量 c 的值
8.  }
9.  int main()
10. {
11.     int x, y, z;
12.     z=Max(3, 5);                        //调用 Max 函数,实参为常量
13.     printf("maxmum=%d\n", z);
14.     scanf("%d%d",&x, &y);
15.     z=Max(x, y);                        //调用 Max 函数,实参为变量
16.     printf("maxmum=%d\n", z);
17.     z=Max(x+3, y*2 );                   //调用 Max 函数,实参为表达式
18.     printf("maxmum=%d\n", z);
19.     z=Max(9, Max(x, y) );               //调用 Max 函数,第 2 个实参为函数调用
20.     printf("maxmum=%d\n", z);
21.     return 0;
22. }
```

说明： 从数学的角度看 Max 函数，该函数接受两个整型自变量，返回一个整型结果。

3. 函数返回值


5-3 函数返回值

函数的返回值说明如下。

(1) 函数的值是通过 return 语句返回到主调函数，return 语句的功能是计算表达式的值，并返回给主调函数。

return 语句的一般形式：

```
return (表达式);
```

或

```
return  表达式;
```

(2) 函数中可以有多个 return 语句，但每次调用只可能有一个 return 语句被执行。因

为一旦执行到其中的一个 return 语句,就立即返回到主调函数,被调函数中其他语句不再执行。

例 5.2 中的 Max 函数还可以定义成如下形式。

```
int Max(int a,int b)                    //函数 Max 的定义
{
    if (a>b)   return a;
    else   return b;
}
```

在调用过程中,如果满足 a>b 的条件,则会执行语句 return a;,然后就返回主调函数,第二个 return 语句将不再执行。

(3) 定义函数时指定的函数值类型应该和 return 语句中的表达式的类型保持一致。如果两者不一致,则以函数值类型为准,对于数值型的数据,系统会自动进行类型转换。

对上面的例 5.2 参考程序进行修改,将函数值类型与 return 语句的表达式类型定义成不同的类型,具体代码如下。

```
1.  #include<stdio.h>
2.  int Max(float a, float b)       //定义函数值类型为 int 型
3.  {
4.      float c;                    //定义变量 c 为 float 型
5.      c=a>b?a : b;                //将例 5.2 中参考程序的 if 语句改为条件表达式
6.      return c;                   //返回 c 的值,注意 c 为 float 型,而函数值为 int 型
7.  }
8.  int main()
9.  {
10.     float x, y;
11.     int z;
12.     printf("Input x,y:");
13.     scanf("%f%f", &x, &y);
14.     z=Max(x, y);
15.     printf("maxmum=%d\n", z);
16.     return 0;
17. }
```

程序运行结果:

```
Input x,y:1.6  3.8↙
maxmum=3
```

说明: 由于定义 Max 时函数值为 int 型,而 return 语句中的 c 为 float 型,二者不一致,按上述规定,先将 c 的值 3.8 自动转换为整数 3,再将整数值 3 返回 main 函数并赋给变量 z,所以最后的输出结果是 maxmum=3。

这种函数值类型与 return 语句表达式类型不同的做法往往使程序不清晰,可读性差,容易出错,因此尽量不要使用这种方法,而应做到函数值类型与 return 语句表达式类型保持一致。

(4) 如果函数没有或者不需要返回计算结果,可以明确将函数的返回值类型定义为空类型,即 void。

【例 5.3】 编写函数计算 n 个整数的平均数，在 main 函数中输入 n 的值。

分析：如果想在 main 函数中输出计算结果，应该将函数定义成有返回值的；如果直接在函数内输出计算结果，则可以将函数定义成无返回值的，因题目要求在 main 中输入 n，所以函数应该定义一个参数。

例 5.3 的参考程序如下。

```
#include<stdio.h>
void Average(int n)              //函数功能是计算 n 个数的平均数，参数 n 表示整数个数
{
    int i, x;    double fAve, fSum=0.0;
    printf("请输入%d 个整数:\n", n);
    for(i=1; i<=n; i++)
    {   scanf("%d", &x);
        fSum=fSum+x;
    }
    fAve=fSum/n;
    printf("ave=%.2lf\n", fAve);      //在函数内输出计算结果
}
int main()
{
    int n;
    printf("计算 n 个整数的平均数，请输入 n:");
    scanf("%d", &n);                  //输入 n
    Average(n);                       //调用 Average 函数
    return 0;
}
```

说明：定义 Average 函数时，函数名前的数据类型写 void，表示该函数无返回值。另外，在 main 函数中调用 Average 函数时，不再使用赋值语句。

(5) 如果函数不需要从主调函数传入数据，在定义函数时可以没有形式参数，这样的函数称为无参函数。定义无参函数时，函数名后的圆括号中可以写 void，表示该函数没有参数。

调用无参函数的一般形式：

```
函数名();
```

注意：函数名后的圆括号不能省略。

【例 5.4】 从键盘输入一串字符，输入井号“#”时结束，将字符串的大写字母转换为小写字母输出，其他字符直接输出。

分析：编写一个函数，实现将大写字母转换为小写字母。如果输入一串字符的操作在 main 函数中实现，则该函数需要定义参数；如果直接在自定义函数中实现输入一串字符的操作，则不需要定义参数。

例 5.4 的参考程序如下。

```
#include<stdio.h>
void Change(void)                    //定义 Change 函数，该函数无参数也无返回值
{
```

```
    char ch;
    ch=getchar();                    //输入一个字符
    while(ch!='#')
    {   if(ch>='A'&& ch<='Z')        //判断 ch 是否为大写字母
            ch=ch+32;                //大写字母的 ASCII 码值加 32 就转换为小写字母
        putchar(ch);                 //输出一个字符
        ch=getchar();
    }
}
int main()
{
    Change();                        //调用 Change 函数
    return 0;
}
```

通过例5.3和例5.4可以看出,对于一个给定的问题,函数的定义灵活,是否定义参数和返回值取决于问题和函数功能设计的需要。

【练习5.1】 编写函数,计算一个整数各位数字之和。在main函数中输入这个整数,并输出计算结果。例如,输入整数123,各位数字之和为1+2+3=6。

提示:定义一个参数传递整数,函数有一个整型返回值。

5-4 函数声明

5.2.2 函数声明与函数原型

C语言中,在函数调用之前应当对所调用的函数进行声明,指出该函数的返回值的类型以及形参的个数和类型,编译器据此信息对函数调用进行语法检查,保证形参和实参的个数和类型的一致性,保证返回值的使用正确性。

函数声明不是函数定义,函数定义除了描述函数的基本特征(包括函数名、函数类型、形参表)外,核心内容是对函数功能的具体描述;而函数声明只是描述函数的基本特征,即函数原型。

函数原型的一般形式:

```
函数类型 函数名(形参类型1 形参名1,形参类型2 形参名2…);
```

说明:① 在函数声明中,由于编译系统不检查参数名,因此,形参表中可以只写形参的数据类型,而不写形参名,从而可以简化为以下形式:

```
函数类型 函数名(形参类型1,形参类型2…);
```

② 函数声明中,函数类型、函数名、参数个数、参数类型和参数顺序全部与函数定义保持一致。

③ 函数声明的位置:一种是在主调函数中对被调函数进行函数声明;另一种是在所有函数的外部进行函数声明(推荐使用)。

【例5.5】 编程求排列公式 $P_n^m=\frac{n!}{(n-m)!}$ 和组合公式 $C_n^m=\frac{P_n^m}{m!}$。

例5.5的参考程序1(在主调函数中对被调函数进行函数声明)如下。

```
#include<stdio.h>
int main()
{
    double fP, fC;   int n, m;
    double Pfun(int n, int m);          //对 Pfun 函数进行函数声明
    double Factorial(int x);            //对 Factorial 函数进行函数声明
    scanf("%d%d", &n, &m);
    fP=Pfun(n, m);
    fC=fP/Factorial(m);
    printf("P=%.2lf, C=%.2lf \n", fP, fC);
}
double Pfun(int n, int m)               //定义函数,计算排列公式 P
{
    double r;
    double Factorial (int x);           //对 Factorial 函数进行函数声明
    r=Factorial(n)/Factorial(n-m);
    return r;
}
double Factorial(int x)                 //定义函数,计算 x 的阶乘
{
    double fValue=1;
    for(int i=1; i<=x; i++)
        fValue=fValue*i;
    return fValue;
}
```

由于在 main 函数和 Pfun 函数中都要调用 Factorial 函数,所以在 main 和 Pfun 函数中都对 Factorial 函数进行了函数声明。

例 5.5 的参考程序 2(在所有函数的外部进行函数声明)如下。

```
#include<stdio.h>
double Pfun(int n, int m);              //对 Pfun 函数进行函数声明
double Factorial (int x);               //对 Factorial 函数进行函数声明
int main()
{
    double fP, fC;   int n, m;
    scanf("%d%d",&n,&m);
    fP=Pfun(n, m);
    fC=fP/Factorial(m);
    printf("P=%.2lf, C=%.2lf \n", fP, fC);
    return 0;
}
float Pfun(int n, int m)
{    …  }
float Factorial(int x)
{    …  }
```

程序在#include 命令后对 Pfun 函数和 Factorial 函数进行了函数声明,因此在 main 和 Pfun 函数中就不需要再对 Factorial 函数进行声明了。

C 语言中规定,在以下几种情况下可以省略主调函数中对被调函数的函数声明。

(1) 对标准函数的调用不需要进行函数声明,但必须在程序开头用#include命令把标准函数所在的头文件包含到本文件中。例如,在程序中使用scanf和printf等输入输出函数时,都必须在程序开头加#include<stdio.h>;若在程序中使用sin、log等数学函数时,则必须加#include<math.h>。

(2) 如果被调函数的定义出现在主调函数之前,则在主调函数中可以不对被调函数进行声明而直接调用。如例5.2中,函数Max的定义放在main函数之前,因此在main函数中可以省去对Max的函数说明。

(3) 如果在所有函数定义之前,在函数的外部已进行了函数声明,则在各个主调函数中不必对所调用的函数再进行声明。

5-5 函数参数

5.3 函数参数传递

5.3.1 简单变量作为函数参数

在C语言中,简单变量作为函数参数时,参数传递数据是单向的,只能把主调函数的实参值传给被调函数的形参,这是单向的数据传递。

(1) 实参可以是常量、变量、表达式或函数调用。在进行函数调用时,实参必须具有确定的值,以便把这些值传递给形参。

(2) 只有在函数被调用时系统才会为形参分配内存单元,在函数调用结束时,系统会立即释放形参所占用的内存单元。因此,形参只在函数内部有效,在函数调用过程中,形参的值发生改变,不会影响实参,函数调用结束返回主调函数后则不能再使用该形参变量。

【例5.6】 简单变量作为函数参数,形参的变化不影响实参。

```
1.  #include <stdio.h>
2.  int fun (int x, int y)
3.  {
4.      x = x+2;    y = y * 2;
5.      printf("fun 函数:x=%d,y=%d\n", x, y);
6.      return ( x+y ) ;
7.  }
8.  int main ( )
9.  {
10.     int a, b, c;
11.     scanf("%d%d", &a, &b);
12.     c=fun (a, b);
13.     printf("mian 函数:a=%d,b=%d,c=%d\n", a, b, c);
14.     return 0;
15. }
```

程序运行结果:

```
输入:
2  5↙
输出:
fun 函数:x=4, y=10
main 函数:a=2, b=5, c=14
```

程序运行过程中实参、形参的变化情况如图 5.1 所示。

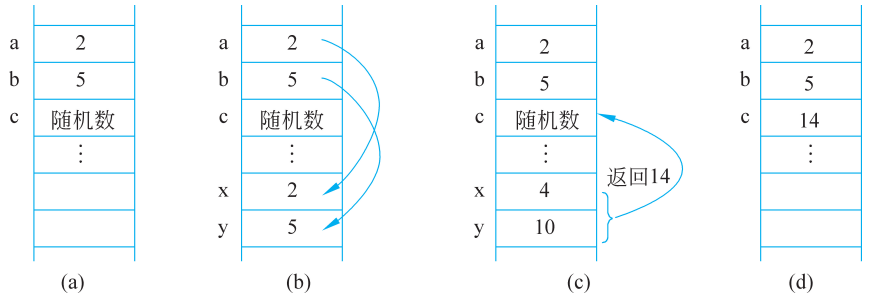

图 5.1　实参、形参变化情况

下面分别对图 5.1 中 4 个子图进行说明。

① 图 5.1(a)表示 main 函数执行 scanf 后变量存储空间的状态，a、b 已得到相应的数据，而 c 还是随机数，这时在内存中还没有为 fun 函数的形参 x、y 分配空间。

② 图 5.1(b)表示执行函数调用 fun (a, b)时存储空间的状态，系统首先为 fun 函数中的形参 x、y 分配存储空间，然后分别把实参 a、b 中的数据 2、5 传递给形参 x、y。

③ 图 5.1(c)表示 fun 函数执行到 return 语句时存储空间的状态，形参 x 的值由 2 变为 4，y 的值由 5 变为 10，并且将 x+y 的结果 14 返回到 main 函数中。

④ 图 5.1(d)表示 fun 函数调用结束后回到 main 函数执行语句 c=fun (a, b)；后变量存储空间的状态，这时系统已经释放了形参 x、y 所占用的存储空间，变量 c 得到函数返回值 14，a、b 里存放的数据仍然 2 和 5。

5.3.2　数组作为函数参数


5-6 数组作为参数

数组用作为函数参数有两种形式，一种是把数组元素作为实参使用；另一种是把数组名作为函数的实参使用。

1. 数组元素作为函数参数

数组元素就是下标变量，它与简单变量并无区别，因此它作为函数实参的使用与简单变量是完全相同的，在函数调用时，把数组元素的值传给形参，实现单向的值传递。

【例 5.7】 输入一个班的某门课程的成绩，将百分制成绩转换为 A、B、C、D、E 共 5 个等级，输出每个学生的百分制成绩和对应的等级。百分制转换为 A、B、C、D、E 这 5 个等级的规则为

A：90～100；B：80～89；C：70～79；D：60～69；E：0～59。

分析： 该题中“成绩转换”很明显是一个独立的功能，可以单独编写一个函数来实现。转换函数需要定义一个形参，其作用是从主调函数获得需要进行转换的百分制成绩，并将百分制成绩转换得到的等级返回给主调函数，所以该函数应定义成有返回值的，且返回值应为字符型。

在 main 函数中主要是实现成绩的输入和输出。另外要注意变量的定义，由于题目要求“输出每个学生的百分制成绩和对应的等级”，所以必须定义两个数组，一个数组存放学生的百分制成绩，另一个数组存放成绩对应的等级（因等级用字母表示，所以该数组应为字符数组）。