第5章 函　数

函数是构成C程序必不可少的基本元素，是C语言的基本构件。C程序是一系列函数的集合，每个函数都具有相对独立的功能。也就是说，一个C程序可以是一个或多个函数的集合体。前面各章中用到的printf()和scanf()等都是函数，main()是主函数。组成C程序的函数，除了系统提供的标准函数外，用户可以根据需要定义自己的函数。本章主要介绍C程序中用户函数的定义和调用方法，变量的存储属性、生存期和作用域及编译预处理命令。

5.1 概述

在结构化程序设计中，通常将复杂的功能分解成若干个相对简单的子功能，并把实现这些简单单一功能的代码封装在一个函数中，当程序需要实现函数的功能时，就可以通过调用简单的函数来实现。本节主要介绍在C语言中如何实现这种模块化的程序设计。

5.1.1 模块与函数

学习程序设计语言的目的就是使用程序设计语言编写程序来解决实际问题。对于较简单的问题，不考虑程序的设计方法和控制结构也可以编写正确的程序。一旦遇到复杂问题，采用不适当的程序设计方法和控制结构往往使编写的程序可读性较差，性能达不到预期的设计要求。

C语言是结构化的程序设计语言。结构化程序设计是一种设计程序的技术，通常采用自顶向下、逐步求精的设计方法和单入口单出口的控制结构。先进行总体设计，采用自顶向下逐步求精的方法，将一个复杂问题分解和细化成多个子问题，如果子问题仍较复杂，再将子问题细分为几个更小的子问题来处理，这样使得复杂问题转化成了由许多小问题组成、具有层次结构的系统，小问题的解决及程序的编写相对容易。通常把求解较小问题的算法及实现的程序称为“模块”，图5-1给出了分解一个复杂问题的模块结构图。

从总体设计上，采用先全局后局部、先整体后细节、先抽象后具体的逐步求精的过程，使问题的划分在结构层次上十分清楚，便于分工。从程序实现上，模块采用单入口单出口的控制结构，不使用goto语句，使程序结构清晰、容易阅读和理解。

C语言采用函数来实现这种模块型的结构关系。一个C语言程序由多个模块组成，每个模块用来实现一个特定的功能，一个或多个C语言的函数可实现一个具有特定功能的模块，函数就成了构成C语言程序的基本部件。

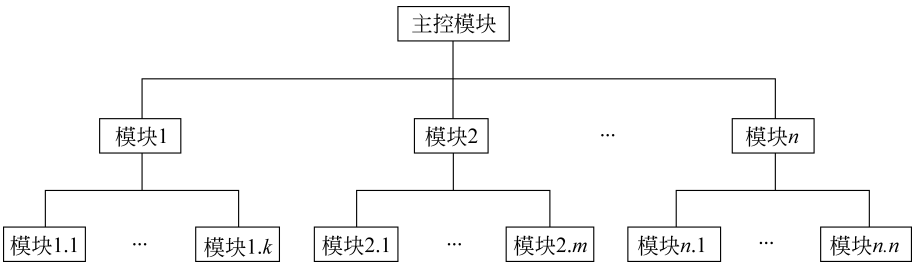

图 5-1 模块结构图

5.1.2 函数的基本概念

1. 函数的概念

所有的高级语言中都有子程序这个概念,用子程序来实现模块的功能。在 C 语言中,子程序的作用是由函数来完成的。C 语言中“函数”的概念和数学中“函数”的概念不完全相同,在英语中“函数”与“功能”是同一个单词即 function,C 语言中的“函数”实际上是“功能”的意思。当需要完成某一个功能时,就用一个函数(可以是标准库函数或自己设计的函数)去实现它。因此,一个 C 语言程序可由一个主函数和若干函数构成。由主函数调用其他函数,其他函数也可以相互调用。同一个函数可以被一个或多个函数调用任意多次。在图 5-1 中的每一个模块功能应由 C 语言的一个函数(可以是标准库函数或自己设计的函数)实现。图 5-1 模块结构图中模块之间的关系在 C 语言中表现为一个程序中函数调用函数的关系,如图 5-2 所示。

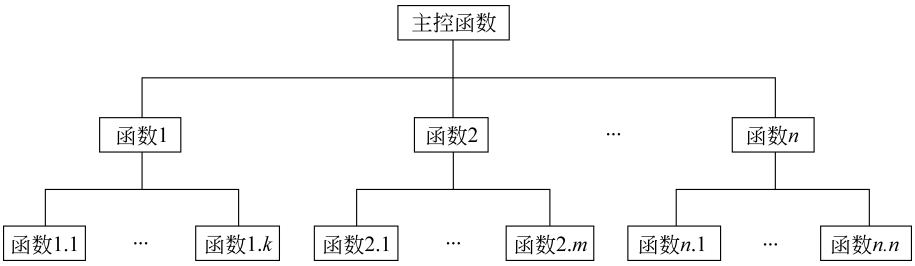

图 5-2 函数调用结构图

【例 5-1】 函数调用的简单例子。

```
1 #include<stdio.h>
2 int main()
3 {
4     void printstar();                    /* 对 printstar 函数进行声明 */
5     void print_message();                /* 对 print_message 函数进行声明 */
6     printstar();                         /* 调用 printstar 函数 */
7     print_message();                     /* 调用 print_message 函数 */
8     printstar();                         /* 调用 printstar 函数 */
9     return 0;
```

```
10 }
11 void printstar()                          /* 定义 printstar 函数 */
12 {
13     printf("*********************\n");
14 }
15 void print_message()                       /* 定义 print_message 函数 */
16 {
17     printf("How do you do\n");
18 }
```

程序运行结果：

```
*********************
   How do you do
*********************
```

程序包括三个函数，main 函数为主函数，printstar 和 print_message 都是用户定义的函数，分别用来输出一排“*”号和一行信息。

说明：

(1) 一个源程序文件由一个或多个函数组成。一个源程序文件是一个编译单位，即以源文件为单位进行编译，而不是以函数为单位进行编译。

(2) 一个 C 程序由一个或多个源程序文件组成。对较大的程序，一般不希望全放在一个文件中，而将函数和其他内容(如预编译命令)分别放在若干个源文件中，再由若干源文件组成一个 C 程序。

(3) C 程序的执行从 main 函数开始，在 main 函数中调用其他函数，程序执行流程转移到被调函数，运行完被调函数，流程返回到 main 函数，在 main 函数中结束整个程序的运行。main 函数被系统调用。

(4) 所有函数都是平行的，即在定义函数时是互相独立的，一个函数并不从属于另一个函数，即函数不能嵌套定义，但可以互相调用(注：不能调用 main 函数)。

2. 函数的分类

在 C 语言中，函数从不同的角度看，可以有以下分类方法：

(1) 从用户使用的角度看，函数有两种：

① 标准函数，即库函数。这是由系统提供的，用户不必自己定义这些函数，只需在程序前包含有该函数原型的头文件即可在程序中直接调用。

② 用户定义函数。由用户根据需要，遵循 C 语言的语法规定自己编写的一段程序，实现特定的功能。

(2) 从函数参数传送的角度看，函数分两类：

① 有参函数。定义时带有形式参数的函数。使用该函数时，必须提供必要的数据，根据提供数据的不同，可能得到不同的结果，例如函数 sin(x)。

② 无参函数。定义时没有形式参数的函数。使用该函数时，不需提供数据，函数通常

完成指定的一组操作,例如函数 getchar。

5.1.3 函数定义的一般形式

定义函数就是在程序中设定一个函数模块。一个函数是由变量定义部分与可执行语句组成的独立实体,用以完成指定的功能。具体地讲,一个函数由函数首部和函数体构成。下面从函数有无参数的角度介绍如何定义函数。

1. 无参函数定义的一般形式

无参函数定义的一般形式为:

```
类型标识符 函数名()
{
    说明部分
    语句部分
}
```

例 5-1 中的函数 printstar()和 print_message()都是无参函数。

在定义函数时要用“类型标识符”指定函数值的类型,即函数返回值的类型。例 5-1 中的函数 printstar()和 print_message()为 void 类型,表示不需要返回函数值。

2. 有参函数定义的一般形式

按有参函数的形参说明出现的位置来分,有参函数的定义有两种形式。

有参函数定义形式一:

```
类型标识符 函数名(类型名 形式参数 1,类型名 形式参数 2,…)
{
    说明部分
    语句部分
}
```

有参函数定义形式二:

```
类型标识符 函数名(形式参数 1,形式参数 2,…)
形式参数类型说明;
{
    说明部分
    语句部分
}
```

例如:

```
int max(int x,int y)                    /* 函数的首部 */
{
    int z;                              /* 函数体中的说明部分 */
    z=x>y?x:y;                          /* 语句部分 */
```

```
    return(z);                          /* 语句部分 */
}
```

或

```
int max(x,y)                            /* 函数的首部 */
int x,y;
{
    int z;                              /* 函数体中的说明部分 */
    z=x>y?x:y;                          /* 语句部分 */
    return(z);                          /* 语句部分 */
}
```

3. 空函数

空函数的一般形式为：

```
类型标识符 函数名()
{

}
```

例如：

```
int dummy()
{

}
```

调用此函数时，什么工作也不做，没有任何实际作用。在主调函数中调用该函数，仅表明调用 dummy()的位置将要调用一个函数，而现在这个函数没有起作用，等以后扩充函数功能时补充上。

说明：

(1) 类型标识符指明了函数返回值的类型，其数据类型可以为 int、float、double、char、void 和指针类型等。若选用 void 类型，表示该函数没有返回值。若函数值类型为 int 时，可省略其类型说明，建议不使用缺省形式的类型说明，要确保函数说明的完整性。

(2) 函数名是由用户命名的标识符，在同一个编译单位中函数名不能重名。若函数是无参函数，函数名后的括号也不可省略。

(3) 函数名的括号后无分号。

(4) 用{}括起来的部分是函数的主体，称为函数体。函数体是一段程序，确定该函数应完成的规定操作，集中体现了函数的功能。其由两部分组成：变量说明部分和语句部分。说明部分是局部说明，这是对函数体内用到变量的类型说明，其有效范围局限于该函数内部，同形参一样，不能由其他任何函数存取。

(5) 如果该函数有返回值，则函数体中必须有一条返回语句“return(表达式);”；如果该函数没有返回值，则函数体中的返回语句应为 return 或不含 return 语句。

5.2 函数的调用

一个函数一旦被定义,就可在程序的其他函数中使用它,这个过程称为函数调用。一个函数可以被其他函数多次调用,每次调用可以处理不同的数据。

5.2.1 函数调用的一般形式

函数调用的一般形式为:

函数名(实参表列);

其中:

(1) 实参表列是用逗号分隔的常量、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,它们都必须有确定的值,以便把这些值传递给形参。

(2) 函数的实参和形参是函数间传递数据的通道,因而二者应在个数、类型和顺序上一一对应,否则会发生“类型不匹配”的错误。

(3) 对于无参数的函数,调用时实参表为空,但括号不能省略。

5.2.2 函数调用的方式

按函数在程序中出现的位置来分,可以有以下三种函数调用方式。

1. 函数语句

把函数调用作为一个语句。例如:

```
printf("How are you?");
```

这时不要求函数返回值,只要求函数完成一定的操作。

2. 函数表达式

函数出现在一个表达式中,这种表达式称为函数表达式,这时要求函数带回一个确定的值以参加表达式的运算。例如:

```
c=2*max(a,b);
```

函数 max 是表达式的一部分,它的值乘以 2 再赋给 c。

3. 函数参数

把函数调用的值作为一个函数的实参,例如:

```
m=max(a,max(b,c))
```

其中 max(b,c)是一次函数调用,它的值作为 max 另一次调用的实参。这时也要求函数带回一个确定的值。

5.2.3　函数的参数

在定义函数时函数名后面括号中的变量名称为“形式参数”(简称“形参”),在主调函数中调用一个函数时,函数名后面括号中的参数(可以是一个表达式)称为“实际参数”(简称“实参”)。

【例 5-2】 调用函数时的数据传递。

```
#include <stdio.h>
int main(void)
{
    int max(int x,int y);              /* 对 max 函数的声明 */
    int a,b,c;
    scanf("%d,%d",&a,&b);
    c=max(a,b);
    printf("Max is %d",c);
    return 0;
}
int max(int x,int y)                   /* 定义有参函数 max 求两数中的较大者 */
{
    int z;
    z=x>y?x:y;
    return(z);
}
```

程序运行结果:

```
7,8<CR>
Max is 8
```

程序包含两个函数,main 函数为主函数,max 函数用来求出 x 和 y 中值较大者。

关于形参与实参的说明:

(1) 在定义函数中指定的形参,在未出现函数调用时,它们并不占内存中的存储单元,只有在发生函数调用时,形参才被分配内存单元,在调用结束后,形参所占的内存单元被释放。

(2) 实参可以是常量、变量或表达式,例如:

```
max(3,a+b);
```

但要求它们有确定的值。在调用时将实参的值传给形参。

(3) 定义函数时,形参的排列没有次序要求,但对形参表中的每个参数类型要进行说明。调用函数时,实参类型、个数及排列次序应与形参一一对应。

(4) C 语言规定,实参对形参的数据传递是“值传递”,即单向传值,实参的值传给形参,而形参的值不能传回给实参。实参与形参处在不同的函数中,作用的区域不同,即使实参与形参同名,它们也是不同的变量。

【例 5-3】 调用函数时,形参、实参值的变化。

```
 1 #include <stdio.h>
 2 int main(void)
 3 {
 4     void s(int n);                          /* 对 s 函数的声明 */
 5     int n;
 6     scanf("%d",&n);
 7     s(n);
 8     printf("\nMain: n=%d\n",n);
 9     return 0;
10 }
11 void s(int n)
12 {
13     n=n+100;
14     printf("Sub: n=%d\n",n);
15     return;
16 }
```

程序运行结果:

```
15<CR>
Sub: n=115
Main: n=15
```

程序定义了一个函数 s,功能是将形参 n 加 100 后输出。在主函数中定义变量 n,并从键盘输入 n 的值 15,系统将 15 存入变量 n 所对应的单元中。程序执行语句“s(n);”时,函数 s 被调用。实参 n 将其值 15 传送给 s 函数的形参 n(注意,本例的形参变量和实参变量的标识符都为 n,但这是两个不同的量,各自的作用域不同),这样形参 n 对应的存储单元中值也为 15,如图 5-3 所示。程序流程转到函数 s,执行函数体中语句“n=n+100;”,形参 n 的值变为 115;执行语句“printf("Sub: n=%d\n",n);”,即输出形参 n 的值;执行语句“return;”,程序流程从被调函数 s 返回主函数,执行主函数中的语句“printf("\nMain: n=%d\n",n);”,输出的是实参 n 的值(实参 n 的值并没有变化,仍为 15)。参数传递过程如图 5-3 所示。可见实参的值不随形参的变化而变化。

图 5-3 形参与实参间的数据传递

5.2.4 函数的返回值

函数的返回值是指函数被调用并执行完后,返回给主调函数的值。

返回语句的一般形式为:

return (表达式); 或 return 表达式; 或 return;

说明:

(1) 函数的返回值只能有一个。

(2) 当函数不需要指明返回值时,可以写成:return;或者不写任何返回语句,函数运行

到右大括号后自然结束。

(3) 当函数没有指明返回值,或没有 return 语句时,函数执行后,实际上不是没有返回值,而是返回一个不确定的值,有可能会给程序带来某种影响。因此为了确保函数不返回任何值,此时应在定义函数时用 void 类型加以说明。例如:

```
void bell()
{
    printf("********");
    return;
}
```

(4) 函数中可以出现多个 return 语句,执行到哪一个 return 语句,哪个语句起作用。例如:

```
int max(int x,int y)
{
    if(x>=y) return x;
    return y;
}
```

(5) 返回值的类型应该与函数类型相同,如果不相同,以函数类型为准,先转换为函数类型后,再返回。例如:

```
int max()
{
    float z;
    …
    return z;
}
```

在例中 z 变量是 float 型,在返回时,先转换成 int 型,再返回。

(6) 为了确保参数和返回值类型正确,一般须在调用函数中对函数的类型在程序开始处加以说明。例如:

```
int max(int,int); 或 int max(int x,int y);
```

5.2.5　对被调函数的声明

主调函数调用某函数之前应对该被调函数进行声明,这与使用变量之前要先进行变量声明一样。对被调函数的声明有以下几种情况。

1. 库函数的声明

一般在调用库函数时应在程序清单的开头处写上含有该库函数定义的头文件组成的包含命令(只有标准的 I/O 函数 scanf()和 printf()可以不加头文件)。其一般形式为:

#include "头文件名.h" 或 #include <头文件名.h>

例如:

```
#include "stdio.h"          /*输入输出函数*/
#include "math.h"           /*数学函数*/
#include "string.h"         /*字符函数和字符串函数*/
```

2. 用户定义的函数声明

如果使用用户自己定义的函数,而且该函数与调用它的函数(即主调函数)在同一个文件中,一般还应该在文件的开头或在主调函数中对被调函数的类型进行说明。其一般形式为

```
函数类型 函数名();
```

或

```
函数类型 函数名(参数类型1,参数类型2,…,参数类型n);
```

或

```
函数类型 函数名(参数类型1 参数名1,参数类型2 参数名2,…,参数类型n 参数名n);
```

在编制应用程序时,提倡使用后两种形式。后两种函数声明称为函数原型。

注意:末尾加分号。

3. 不需要对被调函数声明

有三种情况可以不加声明:

(1) 当被调函数的函数定义出现在主调函数之前时,主调函数可不对被调函数再作声明,而直接调用。

(2) 当被调函数在主调函数之后定义,其被调函数的返回值是int型或char型,可以不对被调函数作声明,而直接调用。

(3) 如果已在文件的开头(在所有函数之前)对本文件中所调用的函数进行了声明,则在函数中不必对其所调用的函数再作声明。例如:

```
char letter( char,char);            /*以下三行在所有函数之前*/
float f(float,float);
int i(float,float);
int main()                          /*在main函数中要调用letter,f和i函数*/
{                                   /*不必对它所调用的这三个函数进行声明*/
    …
}
/*下面定义被main函数调用的三个函数*/
char letter( char c1,char c2)       /*定义letter函数*/
{
    …
}
float f(float x,float y)            /*定义f函数*/
```