第5章

函　　数

C语言属于结构化程序设计语言。结构化程序设计的主要思想是以模块化设计为中心，将待开发的软件系统划分为若干相互独立、功能单一的模块，从而使每一个模块变得简单而明确。由于模块之间相互独立，因此在设计一个模块时，该模块不会受到其他模块的影响，可将原来较为复杂的问题简化为一系列简单模块的设计。在C语言中，模块功能用函数来实现。


视频讲解

5.1　C语言函数概述

C程序是由函数组成的。虽然在前面各章的程序中大多只有一个main函数，但实用程序往往由多个函数组成。函数是C程序的基本模块，调用函数模块可以实现特定的功能。C语言不仅提供了极为丰富的库函数，还允许用户自己定义函数。用户可以把求解问题的算法编成一个个相对独立的函数模块，然后用调用的方法来使用函数。可以说，C程序的全部工作都是由各种各样的函数完成的，所以也将C语言称为函数式语言。

由于采用了模块式的结构，C语言易于实现结构化程序设计，使程序的层次结构清晰，便于程序的编写、阅读和调试。

一个C程序可以由一个或多个函数组成，其中，必须有且只有一个main函数。main函数也称为主函数。在C语言中，函数的定义是平行的，不允许嵌套定义函数，即不允许在一个函数内再定义另一个函数，但函数可以嵌套调用。由主函数调用其他函数，其他函数也可以再调用函数，同一个函数可以被多个函数调用，如图5-1所示。对于两个有调用关系的函数而言，一般用主调函数和被调函数来区分，调用其他函数的函数称为主调函数。

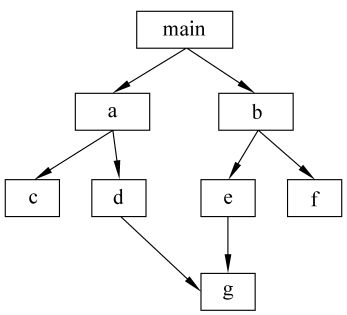

图5-1　C程序函数调用实例

在C程序中，一个函数的定义可以放在任意位置，既可放在主函数main之前，也可放在main之后。但C程序的执行总是从main函数开始，在main函数中调用其他函数，调用完成后返回到main函数，并在main函数中结束程序的运行。

从函数定义的角度而言，函数分为库函数和用户自定义函数。C 系统提供了大量的库函数，在程序中可以直接调用，而程序中大部分函数是由用户按需定义的。

无论库函数还是用户自定义函数又可分为以下类型。

(1) 有返回值函数和无返回值函数。有返回值函数被调用后将向主调函数返回一个值；而无返回值函数用于完成某项特定的功能，执行完成后不向主调函数返回函数值。

(2) 有参函数和无参函数。有参函数在进行函数调用时，主调函数向被调函数传送数据；而对于无参函数，主调函数和被调函数之间不进行数据传送。

本章主要介绍用户自定义函数的定义及使用方法。

5.2 函数的定义

一个函数的完整定义包括两部分：一是函数首部；二是函数体。函数首部即函数定义中的第一行，包括函数的数据类型、函数名和形式参数列表。函数体包括声明和语句两部分，用一对大括号“{ }”括起来，声明部分是对函数体内部所要用到的变量的类型说明，语句部分是函数的执行部分。


视频讲解

5.2.1 函数定义的一般形式

1. 无参函数的定义

无参函数的一般定义形式如下：

```
类型说明符 函数名()
{
    声明部分
    语句部分
}
```

其中，类型说明符说明函数的类型，即函数返回值的数据类型。如果函数定义时不指定函数类型，系统默认函数类型为 int 型。如果函数调用时不需要带回函数值，可将函数类型定义为 void 类型。函数名是由用户定义的合法标识符，在此函数名后的括号内为空，即没有参数，所以称为无参函数。例如：

```
void fun()
{
    printf("This is a functional program.\n");
}
```

fun 函数是一个无参函数，其功能是输出"This is a functional program."字符串。由于函数类型为 void 型，所以函数调用后不带回函数值。

2. 有参函数的定义

有参函数的一般定义形式如下：

```
类型说明符 函数名(形参列表)
{
    声明部分
    语句部分
}
```

在有参函数中,函数名后面的括号内是形式参数(简称形参)的定义。形参可以是各种类型的变量,每个形参必须有类型说明,各形参之间用逗号分隔。例如:

```
int plus (int x,int y)                    //函数首部
{
    int z;                                //声明部分
    z=x+y;                                //以下是语句部分
    return z;
}
```

plus 函数是一个有参函数,功能是求两个整型数据之和。该函数是一个整型函数,由 return 语句返回一个整数。其形参 x 和 y 均为整型变量,其值是在 plus 函数被调用时由主调函数的实参传递过来的。

注意:不能将 int plus (int x,int y) 写成 int plus (int x,y),因为每个形参必须有类型说明。


视频讲解

5.2.2 函数参数与函数返回值

1. 实际参数和形式参数

函数的参数有两类,分别是形式参数和实际参数。形式参数出现在函数定义中,在函数首部,函数名后面的括号中的参数称为形式参数,简称“形参”。实际参数出现在主调函数中,在调用函数时,函数名后面的括号中的参数(也可为表达式)称为实际参数,简称“实参”。当函数被调用时,主调函数将实参的值赋给被调函数的形参,从而实现数据的传递。

【例 5-1】 函数间的参数传递。

```
#include<stdio.h>
int plus(int x,int y)
{
    int z;
    z=x+y;
    return z;
}
int main()
{
    int i,j,k;
    scanf("%d,%d",&i,&j);
    k=plus(i,j);
```

```
    printf("i+j=%d\n",k);
    return 0;
}
```

输入：

```
3,4
```

运行结果：

```
i+j=7
```

程序分析：

本程序定义 main 函数和 plus 函数两个函数，在 main 函数中调用 plus 函数。程序从 main 函数开始执行，当遇到 plus(i,j)时，main 函数被中断，转去执行 plus 函数，将主调函数中实参的值按照参数顺序分别传递给被调函数对应位置上的形参，即将 main 函数中实参 i 的值传递给形参 x，将实参 j 的值传递给形参 y，使两个函数中的参数之间发生联系，如图 5-2 所示。plus 函数调用结束后，返回到 main 函数中，将 plus 函数返回值赋给变量 k，继续执行 main 函数中未被执行的语句，直到最后一条语句，至此整个程序执行完毕。

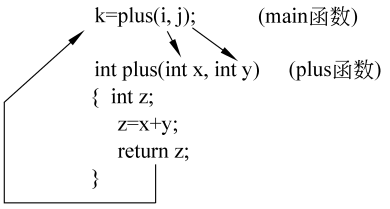

图 5-2　函数间参数传递示例

函数的实参和形参有以下特点。

(1) 实参可以是常量、变量、表达式、函数和地址等，在进行函数调用时，实参必须有确定的值，例如 plus(8,i+j)。

(2) 实参与形参的数量和顺序要严格一致，类型应相同或赋值兼容。如果实参与形参类型不同，则要按第 2 章中介绍的不同类型数据的赋值规则进行转换，即把实参的类型转换成形参的类型，然后送到形参中。

(3) 实参向形参的数据传递是单向的“值传递”，即只把实参的值传递给形参，而形参的值不会传回给实参。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，因为实参与形参被分配到不同的存储单元。

(4) 当调用函数时，形参才被分配存储单元，调用结束时，形参所占存储单元即被释放，因此形参的有效使用范围仅限于本函数内。

【例 5-2】 交换两个变量的值。

```
#include<stdio.h>
void swap(int a,int b)
{
    int temp;
    temp=a;
    a=b;
    b=temp;
    printf("a=%d,b=%d\n",a,b);
```

```
}
int main()
{
    int i=2,j=1;
    printf("i=%d,j=%d\n",i,j);
    swap(i,j);
    printf("i=%d,j=%d\n",i,j);
    return 0;
}
```

运行结果：

```
i=2,j=1
a=1,b=2
i=2,j=1
```

程序分析：

本程序定义了一个 swap 函数，其功能是交换形参 a 和 b 的值。在调用 swap 函数时，为形参 a 和 b 分配存储单元，并将实参 i 和 j 的值分别传递给形参 a 和 b，如图 5-3(a)所示。执行 swap 函数时，交换了形参 a 和 b 的值。由于实参 i 与对应的形参 a 是两个不同的变量，占用不同的存储单元(同理，实参 j 与对应的形参 b 也占用不同的存储单元)，所以，虽然形参 a 和 b 两个变量值交换了，但实参 i 和 j 的值不随形参的变化而变化，如图 5-3(b)所示。在 swap 函数调用结束时，形参 a 和 b 所占的存储单元被释放，main 函数中的 i 和 j 并未互换，如图 5-3(c)所示。

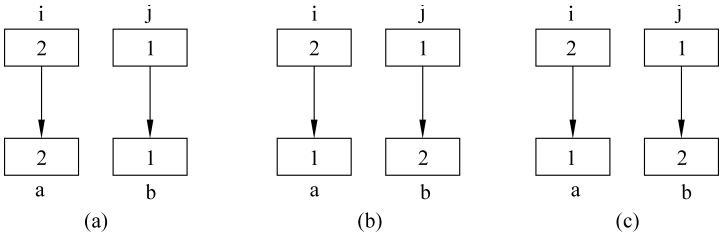

图 5-3　实参和形参变化图

由此可见，当实参和形参为普通变量时，是将实参的值传递给形参，属于单向的“值传递”方式，形参值的改变不会影响实参值。

2. 函数的返回值

函数的返回值是指函数被调用结束后所带回并返给主调函数的一个值，即函数的值。例如，调用 plus(2,4)得到值 6，即函数的返回值为 6。

说明：

(1) 函数的值是由 return 语句返回给主调函数的。

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作为函数返回值，终止函数的执行并返回到主调函数。如果函数不需要返回值，表达式可以省略，改写为以下形式：

return;

该语句的功能是终止函数的执行，并返回到主调函数。

一个函数可以有多个 return 语句，但每次调用只能有一个 return 语句被执行，因此函数调用只能返回一个值。例如：

```
int sign(int x)
{
    if(x<0)
        return -1;
    else if(x==0)
        return 0;
    else
        return 1;
}
```

(2) 如果不需要函数返回一个值，可以将函数定义为无类型或空类型，类型说明符为 void，这样，系统就能保证不使函数返回任何值。另外，在函数体中不能有以下语句：

```
return 表达式;
```

(3) 函数的类型应该和 return 语句中表达式的类型一致，如果二者不一致，则以函数类型为准，系统自动将表达式的类型转换为函数类型，即函数类型决定返回值类型。

【例 5-3】 函数类型决定返回值类型。

```
#include<stdio.h>
int min(float i,float j)
{
    float k;
    k=i<j?i:j;
    return k;
}
int main()
{
    float x,y;
    int z;
    scanf("%f,%f",&x,&y);
    z=min(x,y);
    printf("Min is %d\n",z);
    return 0;
```

```
}
```

输入：

```
3.1,6.4
```

运行结果：

```
Min is 3
```

程序分析：

min 函数的类型是 int 型。在 min 函数中，return 语句中 k 的值为 3.1，为 float 型，将 k 的值转换为 int 型，然后返回给主调函数 main。因此，主函数中 z 的值为 3，程序运行结果为 Min is 3。


视频讲解

5.3 函数的调用

函数的调用指程序从主调函数转到被调函数，执行被调函数的函数体直至执行完函数体中的语句，或者执行至遇到 return 语句为止。

5.3.1 函数调用的一般形式

函数调用的一般形式如下：

函数名(实参列表)

对于有参函数，实参的数据类型与形参的数据类型应一致，实参可以是常量、变量、函数或表达式，各实参之间用逗号分隔。对于无参函数，则无实参列表，但是括号不能省略。

在 C 语言中，函数调用有以下 3 种方式。

1. 函数语句

函数调用是以语句的形式出现的，适合于函数调用不需要返回值的情况。例如：

```
printf("x=%f,y=%f\n",x,y);
```

2. 函数表达式

函数调用作为表达式的一部分，这种表达式称为函数表达式。例如，plus(a,b) * 20 是一个函数表达式，函数调用 plus(a,b)以操作数的形式出现在算术表达式中。

3. 函数参数

函数调用以实参的形式出现在另一个函数的调用中。例如：

```
m=plus(plus(a,b),c);
```

其中，plus(a,b)是一次函数调用，它的值作为 plus 函数另一次调用的实参，m 的值是 a、b、c 三个数的和。

5.3.2 被调用函数的声明

在主调函数调用另一个函数之前，应对被调用函数进行声明（说明）。函数的声明又称函数原型，作用是把函数类型，函数名称，形参的类型、数量、顺序告知编译系统，便于编译系统对函数调用形式进行语法检查，以及在调用函数时对函数返回值的类型进行处理。

被调用函数声明的一般形式如下：

```
类型说明符  被调用函数名(参数类型 1  形参 1,参数类型 2  形参 2,…);
类型说明符  被调用函数名(参数类型 1,参数类型 2,…);
```

例如：

```
float plus(float x, float y);
```

或

```
float plus(float, float);
```

上述是函数声明的两种形式。

对被调用函数的声明应该放在主调函数中的声明部分，也可以在所有函数定义之前，在函数外对各个函数进行声明。

【例 5-4】 编写函数求 n!。

```
#include<stdio.h>
int main()
{
    long fac(int n);                    //fac 函数的声明
    int n;
    long k;
    scanf("%d",& n);
    k=fac(n);
    printf("k=%ld\n",k);
    return 0;
}
long fac(int n)
{
    long k=1;
    int i;
    for(i=1;i<=n;i++)
            k=k * i;
    return k;
}
```

输入：

```
5
```

运行结果：

```
k=120
```

以下两种情况可以省略对被调函数的声明。

(1) 如果被调用函数的定义出现在主调函数之前，在主调函数中可以不必声明而直接调用函数。

(2) 对库函数的调用不必声明，但必须用 include 命令将相应的头文件放在源文件开头，例如：

```
#include<stdio.h>
#include<math.h>
```

【例 5-5】 对例 5-4 做简单的修改。

```
#include<stdio.h>
long fac(int n)
{
    long k=1;
    int i;
    for(i=1;i<=n;i++)
        k=k*i;
    return k;
}
int main()
{
    int n;
    long s,k;
    scanf("%d",&n);
    k=fac(n);
    printf("k=%ld\n",k);
    return 0;
}
```

输入：

```
10
```

运行结果：

```
k=3628800
```

程序分析：

由于被调用函数 fac 的定义出现在主调函数 main 之前，所以在 main 函数中省略了对 fac 函数的声明。

另外，若在程序开头对各函数进行声明，则在之后出现的各函数中不必对被调用函数进行声明，可直接使用，例如：

```
#include<stdio.h>
char f1(char,char);
int f2(int);
float f3(float,float,float);
int main()
{…}
char f1(char c1,char c2)
{…}
int f2(int a)
{…}
float f3(float x,float y,float z)
{…}
```

5.4 函数的嵌套调用与递归调用


视频讲解

5.4.1 函数的嵌套调用

C 语言不允许在一个函数定义中定义另一个函数，但允许调用另一个函数。函数的嵌套调用指在被调函数中又调用了其他函数，其关系如图 5-4 所示。该图表示两层嵌套关系，即 main 函数调用 n 函数，n 函数又调用 p 函数。其执行过程如下。

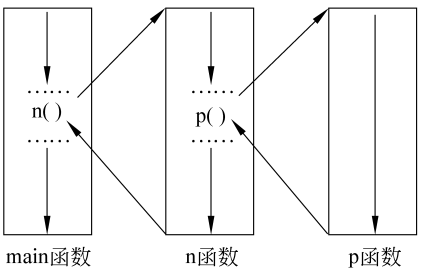

图 5-4 函数嵌套调用图

(1) 程序从 main 函数开始执行，遇到调用 n 函数时，main 函数被中断，转去执行 n 函数。

(2) 在 n 函数中调用 p 函数时，n 函数被中断，转去执行 p 函数。

(3) 当 p 函数执行完后，返回 n 函数的断点，继续执行后续语句。

(4) 当 n 函数执行完后，返回 main 函数的断点，继续执行后续语句，直到 main 函数执行完毕，至此整个程序执行完毕。

【例 5-6】 求 3 个数中最大数和最小数的差值。

```
#include<stdio.h>
int dif(int x,int y,int z);                    //dif 函数声明
int max(int x,int y,int z);                    //max 函数声明
int min(int x,int y,int z);                    //min 函数声明
int main()
{
    int a,b,c,d;
    scanf("%d%d%d",&a,&b,&c);
```