第5章　函数

为了便于管理和维护，一个大程序一般被划分为若干模块，每个模块实现特定功能，一个函数就是一个独立的功能模块，学会设计和使用函数很重要。本章详细描述函数的机制，包括函数定义、函数声明、函数调用、存储类型、参数数目可变的函数、递归函数以及多文件的C程序。

5.1　模块化程序设计

模块化程序设计就是采用“自顶向下逐步细化”的方式，把一个大的程序按照功能划分为若干相对独立的模块，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数据联系，互相协作完成总的任务。

5.1.1　函数与模块化编程

当设计解决一个复杂问题的程序时，提倡将一个复杂的任务划分为若干子任务，若子任务较复杂，还可以将子任务继续分解，直到分解成为一些容易解决的子任务为止。每个子任务设计成一个子程序，称为模块，子程序在程序编制上相互独立，而对数据处理上又相互联系；完成总任务的程序由一个主程序和若干个子程序组成，主程序起着任务调度的总控作用，而每个子程序各自完成一个单一的任务。这种自上而下逐步细化的设计方法称为模块化程序设计方法。

模块化程序设计的优点是：①子程序代码公用，当需要多次完成同样功能时，只需要一份代码，可多次调用，从而省去重复代码的编写；②程序结构模块化，可读性、可维护性及可扩充性强；③简化程序的控制流程，程序编制方便，易于修改和调试。

在C语言中，子程序称为函数，主程序称为主函数，一个解决实际问题的C程序由一个主函数和若干个其他函数组成，其他函数将最终直接或间接被主函数调用，以解决总任务。

下面通过一个简单的实例说明模块化设计思想以及函数的设计（即定义）和使用。

【例5.1】 利用函数实现输出所有的水仙花数。

分析：水仙花数是一个三位数，程序需要循环判断100～999的每个数是不是水仙花数，因此可以将“判断一个数是不是水仙花数”这个功能用函数实现，函数名为isNarcissus，参数为x，有意义的名字能够清楚地表明函数的功能。main函数每次用不同的x值调用isNarcissus判断该数是否水仙花数，主程序结构如下：

```
for(a=100;a<1000;a++)  {                    /*枚举每一个三位数*/
    if(isNarcissus(a))                      /*a是水仙花数,则输出a*/
        printf("%5d",a);
}
```

可以把函数看成一个“黑盒子”，给它一定的输入，它就会产生特定的结果并输出一个值

给使用者,函数的使用者并不需要考虑黑盒的内部行为。对于函数 isNarcissus,使用者需要给它一个整数,然后它能判断出这个整数是不是水仙花数,并把结果值 1(代表是水仙花数)或 0(代表不是水仙花数)给使用者,这个值称为函数的返回值或函数值。

程序每次循环给函数 isNarcissus 不同的 a 值,如果 a 值(如 153)是一个水仙花数,则函数返回值是 1,即函数调用表达式 isNarcissus(a)的值是 1,相当于 if(1),那么执行 printf 语句输出 a 值;如果 a 值不是一个水仙花数,则函数返回值是 0,相当于 if(0),那么不执行 printf 语句。完整的程序如下。

```
#include<stdio.h>
int isNarcissus(int x);                    /*函数原型*/
int main()
{
    int a;
    for(a=100;a<=999;a++) {
        if(isNarcissus(a))                 /*函数调用*/
            printf("%5d",a);
    }
    return 0;
}
/*********************************************
函数功能:判断某整数是不是水仙花数
函数参数:x--待判断的整数
函数返回值:x 是水仙花数,则返回 1;否则,则返回 0。
*********************************************/
int isNarcissus(int x)                     /*函数定义*/
{
    int i,j,k;
    int ans=0;                             /*默认不是水仙花数*/
    i=x/100;
    j=(x-i*100)/10;
    k=x%10;
    if(x==i*i*i+j*j*j+k*k*k)
        ans=1;                             /*是水仙花数,将 ans 设置为 1*/
    return  ans;                           /*返回结果*/
}
```

本程序包含两个函数:main 函数和 isNarcissus 函数。在 main 函数中调用了 isNarcissus 函数,因此,main 是调用函数,isNarcissus 是被调用函数。标识符 isNarcissus 在程序中出现了三次,分别出现在函数原型、函数调用和函数定义中。

首先看以下函数定义的头部:

```
int isNarcissus(int x)                     /*无分号*/
```

因为该函数要接收一个整型参数,所以圆括号里包含一个名字为 x 的 int 类型变量声明,也可以使用符合标识符命名规则的其他名字,变量 x 称为形式参数。函数调用

isNarcissus(a)把 a 的值给 x,称函数调用传递一个值,这个值称为实际参数,函数实际上判断 a 是不是水仙花数,并将判断结果保存在变量 ans 中。

函数体部分可以看作一个黑盒子,里面定义的一切变量都是局部的,对调用函数 main 来说都是不可见的。变量 ans 是函数 isNarcissus 私有的,但是最后的 return 语句把 ans 的值(1 或 0)返回给了调用函数。下面的 if 语句相当于将 ans 的值作为判断条件。

```
if(isNarcissus(a))
```

函数最后如果没有 return 语句,那么变量 ans 的值送不出来,调用函数 main 也就不知道 a 到底是不是水仙花数。

在函数 main 中能用下面的语句来代替 if(isNarcissus(a)) 吗?

```
isNarcissus(a);                          /*返回值未赋值,被丢弃*/
if(ans)
```

答案是不行的,因为 ans 是局限于被调用函数的变量,调用函数 main 看不见它,不能使用。

由于函数返回值是 1 或 0,所以变量 ans 的类型声明为 int,函数定义头部的最前面也要用 int 指明返回值的类型,两者一般要保持一致。

再看 main 函数前面的一行:

```
int isNarcissus(int x);                  /*有分号*/
```

它是函数声明语句,又称函数原型,它说明了函数 isNarcissus 接收一个 int 类型的参数,返回一个 int 类型的值。编译器在调用函数 isNarcissus 前看到了这个原型,就可以根据原型中的描述对实际参数的个数和类型进行检查,以确保数量和类型上的一致,详细解释参见 5.2.2 节。

可见,函数之间可以通过参数传递和返回值实现相互通信,调用函数将实际参数传递给被调用函数的形式参数,被调用函数通过 return 语句将函数的返回值传给调用函数。

函数之间也可以不通信或者单方向通信,下面给出一个函数之间单方向通信的例子,这个例子也说明了伪随机数的使用。

5.1.2 蒙特卡洛模拟:猜数程序

模拟算法是最基本的算法。例如,编程实现抛硬币、掷骰子和玩牌等现实世界中的随机事件要用模拟算法。在程序设计中,可使用随机数函数来模拟现实中不可预测情况,这称为蒙特卡洛模拟。随机数以其不确定性和偶然性等特点在很多地方都有具体的用处,比如,在软件测试中产生具有普遍意义的测试数据,在加密系统中产生密钥,在网络中生成验证码等。

在 C 语言中,用 rand 函数生成随机数,该函数称为随机数发生器,该发生器从称为种子(一个无符号整型数)的初始值开始用确定的算法产生随机数。显然,通过种子产生第一个随机数后,后续的随机序列也就是确定的了,这种依靠计算机内部算法产生的随机数称为伪随机数。由此可见,随机数的产生依赖于种子,为了使程序在反复运行时能产生不同的随机数,必须改变这个种子的值,这称为初始化随机数发生器,由函数 srand 来实现。

【例 5.2】 编写一个猜数的游戏程序。在这个程序中，计算机想一个数(即随机产生一个 1～1000 的整数)，玩家来猜。玩家输入所猜的数，如果猜得不正确，继续猜，直到正确为止。为了帮助玩家一步一步得到正确答案，计算机会不断地发出信息“Too high”或“Too low”。

分析：程序应该允许玩家反复玩，通过询问的方式，由玩家自己选择是否继续，主程序结构如下。

```
do {
    计算机想一个数
    玩家猜数直至猜对
    printf("Play again?(Y/N) ");              /* 询问是否继续玩 */
    scanf("%1s", &choice);                    /* 玩家输入选择 */
} while (choice == 'y' ||choice == 'Y');      /* 输入 y/Y,则继续 */
```

将“计算机想一个数”这个功能设计成函数 getNum，其函数原型如下。

```
int getNum(void);
```

括号中的 void 表示该函数没有输入参数，即不接收任何参数，因为它不需要来自调用函数的任何信息，函数自己随机产生一个数。但是，函数需要返回所产生的数，函数名前的 int 说明返回值的类型为整型。因此，调用函数和 getNum 函数之间的通信是单向的，仅由 getNum 函数通过返回值向调用函数传递信息。该函数定义如下。

```
int getNum(void)                              /* 随机产生一个 1~MAX_NUMBER 的整数 */
{
    int  x;
    x=rand();                                 /* 调用标准库函数 rand 产生一个随机数 */
    x= x %MAX_NUMBER + 1;                     /* 将该数限制在 1~MAX_NUMBER 内 */
    return x;                                 /* 返回这个随机数给调用者 */
}
```

函数体内的 rand 是 stdlib.h 中的一个函数，它返回一个非负并且不大于常量 RAND_MAX 的随机整数，RAND_MAX 是在 stdlib.h 中定义的一个符号常量，其值取决于系统，有的为 32767，有的为 2147483647。MAX_NUMBER 是编程者用 # define 定义的符号常量，其值为 1000。当执行 return 语句时，其后表达式 x 的值被传递给调用函数。

另外，将“玩家猜数直至猜对”这个功能设计成函数 guessNum，其函数原型如下。

```
void guessNum(int x);
```

括号里的 int x 说明该函数有一个 int 类型的输入参数，因为调用函数需要传给它被猜数。玩家在猜的过程中函数只需要将猜的结果直接在屏幕显示，不需要向调用函数返回特定的数值，函数名前的 void 说明该函数无返回值，因此，函数之间的通信也是单向的，仅由调用函数通过参数向 guessNum 函数传递信息。该函数定义如下。

```
void guessNum(int x)                          /* 玩家根据提示反复猜数,直至猜对为止 */
{
    int guess;
```

```
    for (;;) {
        printf("guess it: ");
        scanf("%d", &guess);                    /*玩家输入猜的数*/
        if (guess == x)  {                      /*猜对*/
            printf("Right !\n");
            return;                             /*返回,但不带回值*/
        }
        else if (guess < x)                     /*猜小了*/
            printf("Too low. Try again.\n");
        else                                    /*猜大了*/
            printf("Too high. Try again.\n");
    }
}
```

函数体内的 for (;;) 是无限循环,一直循环到执行 return 语句时结束,关键字 return 表明函数的执行到此结束,将控制返回到调用处,但不向调用函数传递值。

这样设计后,主函数 main 的编写就比较简单了,main 调用相应的函数,解决总的任务。具体程序代码如下。

```
#include <stdio.h>
#include <stdlib.h>                             /*标准函数库的头文件*/
#include <time.h>                               /*日期和时间函数库的头文件*/
#define MAX_NUMBER 1000                         /*被猜数的最大值*/
int getNum (void);                              /*getNum 函数原型*/
void guessNum(int) ;                            /*guessNum 函数原型*/
int main(void)
{
    char choice;
    int magic;
    printf("This is a guessing game\n\n");
    srand(time(NULL));                          /*用系统时间初始化随机数生成器*/
    do {
        printf("A magic number between 1 and %d has been chosen.\n",MAX_NUMBER);
        magic = getNum( );                      /*调用 getNum 函数产生随机数供玩家猜*/
        guessNum(magic);                        /*调用 guessNum 函数让玩家猜出这个数*/
        printf("Play again?(Y/N) ");            /*询问是否继续玩*/
        scanf("%1s", &choice);                  /*玩家输入选择*/
    } while (choice == 'y' || choice== 'Y');    /*数入 y 则继续*/
    return 0;
}
```

用蒙特卡洛法模拟该猜数游戏,在开始玩游戏前(do 语句前),需要调用函数 srand 初始化随机数种子,可以采用系统时间(由 time 函数得到)作为种子,调用语句如下。

```
srand(time(NULL));
```

函数 time 返回自 1970 年 1 月 1 日以来经历的秒数,将该秒数赋值给系统设置的种子变

量，因而每次运行程序产生的随机数是不同的。函数 srand 和 rand 的原型在头文件 stdlib.h 中，函数 time 的原型在头文件 time.h 中，需要在源程序的头部用 #include 指令包含这两个头文件。

如果将语句 srand(time(NULL))；去掉，则每次运行程序产生的随机数都是一样的（可以通过输出 magic 的值来验证），这显然达不到随机玩游戏的效果。

5.1.3 C 程序的一般结构

C 程序由一个或多个函数组成，其中有且只有一个 main 函数，程序的执行总是从 main 开始。除 main 以外的其他函数分两类：一类是由系统提供的标准函数，又称库函数，如 printf、rand 等，用户程序只需包含相应的头文件即可直接调用；另一类是需要由程序员自己编写的函数，如例 5.1 的 isNarcissus、例 5.2 的 getNum 和 guessNum，这些函数称为“自定义函数”。

组成一个 C 程序的各个函数可以放在一个源文件中，也可以编辑成多个 C 源文件，每一个 C 源文件中可以有 0 个（源文件中可以没有函数，仅有一些说明组成，如定义一些全局变量）或多个函数。各 C 源文件中要用到的一些外部变量说明、枚举类型声明、结构类型声明、函数原型和编译预处理指令等可编辑成一个.h 头文件，然后在每个源文件中包含该头文件。每个源文件可单独编译生成目标文件，组成一个 C 程序的所有源文件都被编译之后，由连接程序将各目标文件中的目标函数和系统标准函数库的函数装配成一个可执行 C 程序。图 5-1 显示了 C 语言程序的基本结构。

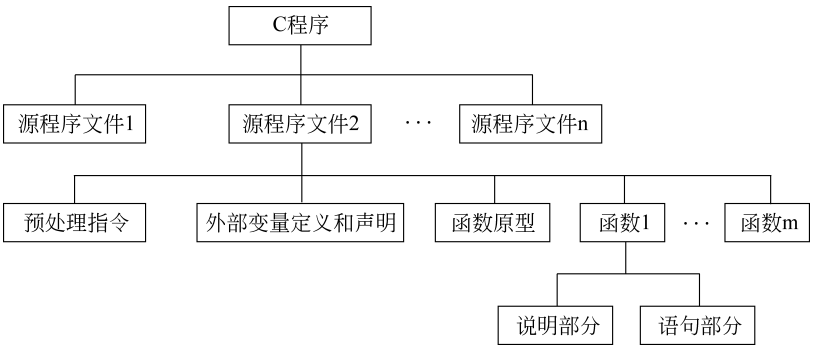

图 5-1 C 语言程序的基本结构

5.2 自定义函数

程序中若要使用自定义函数实现所需的功能，要做三件事：①按语法规则编写完成指定任务的函数，即函数定义；②有些情况下在调用函数之前要进行函数声明，即函数原型；③在需要使用函数时调用函数。

5.2.1 函数定义

1. 函数定义的形式

把描述函数做什么的 C 代码称为函数定义。函数定义的一般形式为：

```
类型名   函数名(参数列表)
{
    声明部分
    语句部分
}
```

说明：第一个花括号(左花括号)前的那部分称为函数头部，由类型名、函数名和参数列表组成；在两个花括号之间的部分称为函数体，包括声明部分和语句部分。

良好的编程风格是：在函数的顶端用"/*……*/"格式增加函数头部注释，如例5.1所示。函数头部注释包括函数名称、函数功能、函数参数、函数返回值等，如有必要还可增加作者、创建日期、修改记录(备注)等相关内容。虽然函数头部注释在语法上不是必需的，但可以提高程序的质量和可维护性，在程序设计时要遵从这一编程规范。为了节省篇幅，例5.2采用简化的注释，希望读者写程序时按例5.1的格式写头部注释。

为函数命名时，要选择有意义的名称，以增加程序的可读性。Windows风格函数名用"大小写"混排的方式，每个词的第一个字母大写，通常用"动词"或者"动词＋名词"(动宾词组)形式，如DrawBox。而UNIX通常采用"小写加下画线"的方式，如draw_box。

2. 参数列表

参数列表说明函数入口参数的名称、类型和个数，它是一个用逗号分隔的变量声明表，其中的变量名称为形式参数(简称形参)。从函数有无参数的角度，函数分为有参函数和无参函数两类。

(1) 有参函数。参数列表里有参数，每一个形参都必须声明其类型，不能像普通的变量声明那样来声明同类型的参数。例如：

```
int max(int a,b)                    /*错误的函数头,b未指定类型*/
int max(int a,int b)                /*正确的函数头*/
```

(2) 无参函数。参数列表是空的或说明为void，表示函数没有参数。例5.2中无参函数GetNum的函数头可以是下面任意一个。

```
int GetNum(void)
int GetNum( )
```

3. 函数的返回值

函数名前的类型名说明函数返回值的数据类型，简称函数类型或函数值的类型，可以是除数组以外的任何类型，**C99标准要求函数必须明确指定返回值的类型**。从有无返回值的角度，函数分有返回值函数和无返回值函数两类。

(1) 有返回值函数。这类函数执行完后将送出一个数值给调用者，这个值称为函数返回值，该值是通过return语句送出的。例如，例5.2的GetNum函数返回int类型数据，函数体最后的语句：

```
return x;
```

将变量x的值返回到调用处。下面这条语句的作用相当于把x的值赋给magic。

```
magic = GetNum( );
```

如果没有这条 return 语句,那么 x 值送不出去,变量 magic 就得不到函数 GetNum 产生的随机整数。所以,有返回值的函数必须使用带表达式的 return 语句,return 中表达式的值就是函数的返回值。一般来说,表达式值的类型应该与函数定义时指明的类型一致。例如,GetNum 函数的类型是 int,所以 x 的类型也声明为 int。如果两者不一致,对于基本类型,将把表达式值的类型自动转换为函数的类型;对于指针类型,必须使用强制类型符将表达式值的类型显示转换为函数的类型。

对于函数返回的值,程序可以使用它,也可以不使用它。

```
getchar();                                  /*返回值不被使用*/
c=getchar();                                /*返回值被使用*/
```

(2) 无返回值函数。当返回值类型为 void 时,函数将不返回任何值。例 5.2 的 GuessNum 函数就是无返回值函数,它仅输出结果到屏幕,没有给调用者返回数值。无返回值函数可以使用如下不带表达式的 return 语句。

```
return;
```

说明:该 return 语句的作用仅将控制返回到调用处,但不带回值。

无返回值函数也可以不包含 return 语句,如果没有 return 语句,当执行到函数结束的右花括号时,控制返回到调用处,这种情况称为离开结束。但是,在例 5.2 中,GuessNum 函数 for 循环体内的 return 语句必须有,否则函数将一直循环执行,无法离开结束。

在一个函数中可以有多个 return 语句,这种情况下的 return 语句通常被作为选择语句的子句出现,最终被执行的只是其中的一个。

【例 5.3】 写一个函数 isPrime,判断整数 n 是否为素数。如果 n 是素数,则返回 1,否则返回 0。

分析:如果 n 是 2,则 n 是素数,返回 1;如果 n 是偶数,则 n 不是素数,返回 0;如果 n 是奇数,则循环找因子,一旦找到一个因子,n 就不是素数,返回 0,如果函数执行到正常退出循环,说明 n 是素数,返回 1。一旦某个 return 语句被执行,控制立即返回到调用处,其后的代码不可能被执行。

```
int  isPrime(int n)
{
    int  k,limit;

    if (n==2) return 1;                     /*2是素数,返回1*/
    if (!(n%2)) return 0;                   /*偶数不是素数,返回0*/
    limit=n/2 ;
    for (k=3;k<=limit;k+=2)                 /*对于奇数*/
        if (!(n%k)) return 0;               /*有因子不是素数,返回0*/
    return 1;                               /*是素数,返回1*/
}
```

return 后面只能跟一个表达式。也就是说,通过 return 语句,函数只能返回一个值到调用处。怎样使函数送回多个值?通过外部变量(见 5.3.3 节)和指针参数(见 8.2.2 节)可使函

数间接送回多个值。

*** 4. 内联函数**

内联函数是指用关键字 inline 修饰的函数，也称内嵌函数，它主要的作用是提高程序的运行效率。inline 是 C99 增加的关键字，它用在函数定义的前面，告诉编译器对该函数的调用进行内联优化，即采用插入的方式处理函数调用，而不是在调用时发生控制转移。例如：

```
inline  int  add(int a,  int b)
{
    return a+b;
}
```

add 是内联函数，编译时使用函数体替换调用处的函数名，类似宏替换（见 6.1 节），这种方式不会产生调用和返回所带来的时间开销，但会增加目标程序代码量，进而增加空间开销。内联函数和宏的区别在于，宏是由预处理器对宏进行替代，而内联函数是通过编译器控制来实现的，不会产生处理宏的一些问题。

inline 仅仅是对编译器的建议，能否真正内联，还要看编译器的意思，它如果认为函数不复杂，能在调用点展开，就会真正内联。因此，inline 的使用是有限制的，内联函数应该简洁，只有几个语句，不能有循环或 switch 等复杂的结构，必须在调用之前定义，否则编译器将该函数视为普通函数。

5.2.2 函数原型

和变量一样，函数也应该先声明后使用，有两种声明函数的方法：一是给出完整的函数定义；二是提供函数原型。为确保程序正确性，在函数调用之前必须给出它的函数定义或函数原型，或者两者都给出。

1. 函数定义起函数声明的作用

C 程序书写格式很自由，函数定义的次序可以随意，如果函数定义出现在函数调用之前，那么函数定义起函数声明作用。例如，例 5.2 中的三个函数定义的次序可为：

```
#include<stdio.h>
void guessNum(int x);                          /* 函数原型 */
int getNum( )                                  /* getNum 函数定义 */
{  ...  }
int main()
{
    …
    magic = getNum( );                         /* getNum 函数调用 */
    guessNum(magic);                           /* guessNum 函数调用 */
    …
}
void guessNum(int x)                           /* guessNum 函数定义 */
{    …   }
```

main 中调用了函数 getNum 和 guessNum，getNum 的定义在 main 函数之前，函数定义起到了函数声明的作用，因此前面可以不写 getNum 函数的原型声明（写上也无妨）。而函

数 guessNum 的定义在 main 函数之后，函数定义不能起到函数声明的作用，因此前面必须写 guessNum 函数的原型声明。有了函数原型信息，编译器就可以检查函数调用是否和其原型声明相一致，比如检查参数个数是否正确、参数类型是否匹配等。如果参数个数不对，如函数调用写成 guessNum(a,b)，则编译器会给出一个错误信息，告知调用函数 guessNum 时传递的参数太多。如果参数类型不匹配，如函数调用为 guessNum(6.8)，则编译器会根据形式参数的类型来转换实际参数值，将 6.8 转换为 6 再传递给形参 x。

2. 函数原型

如果函数定义出现在函数调用之后或者被调用函数在其他文件中定义(对于多文件的 C 程序)，则必须在函数调用之前给出函数原型。函数原型告诉编译器函数返回值类型、参数个数和各参数类型。编译器用函数原型来检查函数调用的正确性，从而避免错误的函数调用所导致的致命运行错误或者微妙而难以检测的非致命逻辑错误。函数原型的一般形式为：

类型名 函数名(参数类型表)；

说明：函数原型是声明语句，必须以分号结束，参数类型表通常是用逗号隔开的形参类型列表，而形参名可以省略，除此之外，函数原型和函数定义的头部是相同的。例如，函数 max 的原型声明可以是下面中的任意一个。

```
int max(int,int);
int max(int x,int y);
int max(int a,int b);
```

该函数原型表明：函数 max 返回值类型为 int，有两个类型为 int 的参数，类型后的变量名是虚设的名字，不必与函数定义中的形参名一致，编译器忽略函数原型中像 x 和 y 这样的名字，使用形参名的目的是为了方便阅读文档。

对于无参函数，函数原型的参数类型表需要指定为 void。例 5.2 中 getNum 函数原型应为：

```
int getNum(void);
```

函数原型一般位于文件开头部分(即所有函数定义的前面)，这样该函数可被本文件中的任意函数调用。函数原型也可以位于调用函数体内，如例 5.2 中的 main 函数调用了 getNum 和 guessNum 函数，其函数原型可出现在 main 函数的声明语句部分。

```
int main( )
{
    int getNum(void);                          /* 函数原型位于调用函数体中 */
    void guessNum(int);
```

该函数原型表明 main 函数可以调用它。如果有其他函数要调用它，则在调用它的函数体内也要写函数原型代码。

为充分利用 C 语言的类型检查能力，在程序中应包含所有函数的函数原型。系统库函数的函数原型在相应的头文件中，所以，程序中用到的库函数要使用 #include 预处理指令包含相应的头文件。