第 5 章　函数与模块化设计

函数是程序设计语言的重要里程碑之一，它标志着程序模块化设计和软件重用的开始。简单地说，模块化设计就是把一个复杂的问题按照功能划分为若干简单的功能模块，以模块为单位进行程序设计。模块化的目的是降低程序复杂度，使程序设计、调试和维护等操作简单化。一般将复杂问题划分为不同的模块后更适合软件的团队开发，不同的模块由不同的程序员设计，只需要确定好模块之间的接口关系，就可以实现模块之间的相互调用，而模块内部的具体实现可以由程序员自己设计。在 C 语言中，模块是通过函数来实现的。

本章主要介绍函数的定义、函数中参数的传递、函数的调用、全局变量与局部变量的作用域。并通过大量的例题分析，让读者可以灵活、熟练地掌握函数（包括递归函数）的定义与调用。

5.1　概述

在前面的章节中，学习了如何编写一些简单的 C 程序。在这些程序中，经常调用诸如 scanf()、printf()之类的系统函数。这些函数都是 C 语言的标准库函数，是系统预先定义好的，可以实现输入输出等基本功能。C 语言中提供了许多标准的库函数，存放在不同的头文件（以 h 为扩展名）中，如数学函数放在 math.h 头文件中，字符串函数放在 string.h 头文件中等。用户在使用时，只需要通过 #include＜头文件名＞将其包含到程序中即可。但标准的库函数并不能满足用户所有的需求，用户还可以根据程序的需要自定义函数，通过对函数的调用来完成相应的功能。

例 5-1　计算组合数。

【问题描述】

C(n,m)表示从 n 个数中选择 m 个的组合数。计算公式为

$$C(n,m)=\frac{n!}{m!(n-m)!}$$

计算 C(n,m)。

【输入格式】

第一行是正整数 N，表示有 N 组要计算的组合数。接下来 N 行，每行两个整数 n，m（$0\leqslant m\leqslant n\leqslant 20$）。

【输出格式】

输出 N 行。每行输出一个整数表示 C(n,m)。

【样例输入】

```
3
2 1
```

```
3 2
4 0
```

【样例输出】

```
2
3
1
```

【问题分析】

根据 C(n,m)的定义，C(n,m)的计算过程实际上是 3 个阶乘的计算。根据第 3 章计算阶乘的方法，编写相关的程序。

【参考代码 1】

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int num,n,m,i;
    long long ans,fac1,fac2,fac3;
    cin>>num;
    while(num){
        cin>>n>>m;
        fac1=fac2=fac3=1;
        for(i=1;i<=n;i++)
            fac1*=i;
        for(i=1;i<=m;i++)
            fac2*=i;
        for(i=1;i<=n-m;i++)
            fac3*=i;
        ans=fac1/(fac2*fac3);
        cout<<ans<<endl;
        num--;
    }
    return 0;
}
```

【代码分析】

程序中通过 3 次循环分别求出 n!、m!和($n-m$)!。而这 3 次循环中，除了循环的次数(即终值)不同以外，其他的代码完全一样。为减少代码的重写，可以把这部分通用代码从程序中抽出来，利用函数来实现。参考如下代码。

【参考代码 2】

```
#include<bits/stdc++.h>
using namespace std;
```

```
3.  long long fac(int n){
4.      long long ans=1;
5.      int i;
6.      for(i=1;i<=n;i++)
7.          ans*=i;
8.      return ans;
9.  }
10. int main()
11. {
12.     int num,n,m;
13.     long long ans;
14.     cin>>num;
15.     while(num){
16.         cin>>n>>m;
17.         ans=fac(n)/(fac(m)*fac(n-m));
18.         cout<<ans<<endl;
19.         num--;
20.     }
21.     return 0;
22. }
```

【代码分析】

从例中可以看出，自定义函数 fac()在主函数中被调用了 3 次，这样处理有两个优点：①可以省略重复代码的编写，使程序的逻辑结构更加清楚；②利用函数可以使程序更加模块化，程序员可以把主要精力投入程序的整体设计。

对于函数的使用，有以下 3 点需要说明。

（1）从用户使用的角度看，函数包括两种：①由系统提供的、无须用户定义的标准函数，这些函数根据不同的功能存放在不同的头文件中；②用户根据需要编写具有特定功能的自定义函数。

（2）C 语言中所有的函数都是平行的，它们之间都是互相独立的。函数只能嵌套调用，不能嵌套定义。

（3）一个源程序文件中，可以由一个或多个函数及其他有关内容（如数据声明与定义）组成，但必有一个 main()函数。程序总是从 main()函数开始和结束的。

5.2 函数的定义与调用

5.2.1 函数的定义

在例 5-1 中定义了 fac()函数，具体如下：

```
1.  long long fac(int n){
```

```
    long long ans=1;
    int i;
    for(i=1;i<=n;i++)
        ans*=i;
    return ans;
}
```

在该函数中，fac 是函数名，它的命名规则应当满足 C 语言中标识符的命名规则，主调函数通过函数名来完成对函数的调用；第一个 long long 代表函数的返回类型，原则上它应当与函数内部的返回值 ans 类型保持一致；int n 代表函数的形参，是函数与其他函数的接口，用户通过该接口向函数传递参数；由花括号括起来的部分为函数体，完成具体的操作。

函数定义的一般形式如下：

```
数据类型 函数名(数据类型 形参 1, 数据类型 形参 2,…, 数据类型 形参 n)
{
    函数体
}
```

函数体包括函数的声明部分和执行部分。例如，可以定义如下的函数：

```
int test(int n){
    int k;                              //变量的声明
    k=n+1;
    return k;
}
```

在对函数进行定义时，有以下 4 点需要注意。

（1）在对函数的形参进行定义时，需要声明每个变量的类型，不能像通常的变量声明那样，使用变量列表，如：

```
int max(int a, int b)          //正确的参数定义
int max(int a,b)               //错误的参数定义
```

（2）函数的返回类型原则上应与函数体内 return 语句中表达式的类型保持一致，如果二者出现不一致的情况，以函数的返回类型为准。当数据的类型是数值型时，系统会进行自动类型转换。

（3）如果函数不需要返回值，函数的返回类型可定义为空类型 void。此时函数中可以没有 return 语句，或有 return 语句，但语句后没有表达式，即“return;”。

（4）在一个函数中可以有多个 return 语句，但只要执行到其中一个 return 语句，就结束该函数的调用。例如，求最大值的函数可以定义如下：

```
int max(int x, int y){
    if(x>y)
```

```
3.          return x;
4.      return y;
5.  }
```

例 5-2 闰年判断。


例 5-2

【问题描述】

给定一个年份，判断这一年是不是闰年。

【输入格式】

输入包含一个整数 y，表示当前的年份。

【输出格式】

输出一行，如果给定的年份是闰年，则输出 yes，否则输出 no。

【样例输入】

```
2013
```

【样例输出】

```
no
```

【问题分析】

设计函数时，被调函数与主调函数之间的参数传递是非常关键的，简单地将主调函数要实现的工作放在函数里实现是没有任何意义的。以本题为例，用户要输入的年份可以在主调函数里实现，被调函数的工作是判断给定的年份是不是闰年，可以通过返回 0 或 1 来表示。

【参考代码】

```
1.  #include<bits/stdc++.h>
2.  using namespace std;
3.  int judge(int year){
4.      if((year%4==0 && year%100!=0)||(year%400==0))
5.          return 1;
6.      else
7.          return 0;
8.  }
9.  int main(){
10.     int year;
11.     cin>>year;
12.     if(judge(year)==1)
13.         cout<<"yes";
14.     else
15.         cout<<"no";
16.     return 0;
17. }
```

【代码分析】

在上述程序中，函数 judge()的功能相对单一，判断给定的年份是不是闰年，年份的输入以及程序的输出由主调函数实现。此外，在 judge()函数中，通过 return 返回函数运行结果，由于函数遇到 return 会自动结束，因此上述程序中第 6 行的 else 可以省略。

例 5-3 求最大值。

【问题描述】

输入 3 个实数，求其中的最大值，使用函数编程解决。

【样例输入】

```
1.1 2.2 3.3
```

【样例输出】

```
3.3
```

【参考代码】

```
1.  #include<bits/stdc++.h>
2.  using namespace std;
3.  double imax(double a, double b){
4.      float c;
5.      if(a>b)
6.          c=a;
7.      else
8.          c=b;
9.      return c;
10. }
11. int main(){
12.     double a,b,c;
13.     cin>>a>>b>>c;
14.     cout<<imax(imax(a,b),c);
15.     return 0;
16. }
```

【代码分析】

程序运行时，如果输入 3 个 double 型的实数 1.1、2.2 和 3.3，由于函数返回值的数据类型与函数类型不一致，函数 imax()将返回值的类型自动从 float 型变为 double 型，上述程序输出的值是 3.3。

5.2.2 函数的调用

对函数进行正确的定义以后，就可以在主函数或其他函数中对函数进行调用。在例 5-1 中，通过 fac(n)、fac(m)、fac(n－m)完成对函数的调用。函数调用的一般形式如下：

函数名(实参 1, 实参 2,…, 实参 n);

例如,调用函数 test()时,只需要在主函数中加入一行“test(4);”即可。

在进行函数调用时,需要注意以下 3 点。

(1) 函数的参数有两种:定义函数时使用的参数称为形参,调用函数时使用的参数称作实参。形参必须是变量,而实参可以是常量、变量、表达式或函数,但必须有确定的值。例如:

```
c=max(9,5);                    //实参是常量
c=max(x,y);                    //实参是变量
c=max(x+2,y*3);                //实参是表达式
c=max(max(x,y),z);             //求 3 个数的最大值,实参是函数
```

(2) 实参与形参的顺序和类型应保持一致。

(3) 执行函数调用时,形参和实参是两个不同的存储单元。具体地讲,定义函数时,形参并不占用内存存储单元;进行函数调用时,函数的形参被临时分配内存单元;调用结束时,形参占用的内存存储单元被释放,但实参单元仍然保留原值,没有改变。

例 5-4 判断完全平方数。


例 5-4

【问题描述】

写一个判断平方数的函数,判断输入的 x 是否为平方数并输出结果。

【输入格式】

有多组数据,每组输入一个整数 x。

【输出格式】

每组输出一行,如果 x 是平方数输出 1,否则输出 0。

【样例输入】

```
1
25
99
```

【样例输出】

```
1
1
0
```

【问题分析】

判断一个整数是不是完全平方数主要有两种方式:一是判断它是否为某个数的平方;二是基于完全平方数是从 1 开始的若干奇数的和这个结论进行判断。

【参考代码 1】

```
1.  #include<bits/stdc++.h>
2.  using namespace std;
3.  int judge(int n){
```

```
    int i;
    i=(int)sqrt(n);
    return i*i==n;
}
int main()
{
    int n;
    while(cin>>n){
        if(judge(n)==1)
            cout<<1<<endl;
        else
            cout<<0<<endl;
    }
    return 0;
}
```

【参考代码 2】

```
#include<bits/stdc++.h>
using namespace std;
int judge(int n){
    int i=1;
    while(n>0){
        n-=i;
        i+=2;
    }
    return n==0;
}
int main()
{
    int n;
    while(cin>>n){
        if(judge(n)==1)
            cout<<1<<endl;
        else
            cout<<0<<endl;
    }
    return 0;
}
```

5.2.3 函数的声明

C 语言中，在一个函数（称为主调函数）中可以调用另一个函数（称为被调函数）。在

对函数进行调用时，首先被调函数必须是已定义的函数(库函数或用户自定义的函数)，并且对被调函数进行声明。

(1) 如果被调函数是库函数，如 printf()、scanf()等，应该将调用库函数的信息包含到本文件中。例如，在调用基本输入输出函数时，应使用语句#include<stdio.h>将头文件 stdio.h 中的库函数引入文件中；在调用 sqrt()函数时，应使用语句#include<math.h>将头文件 math.h 中与数学运算有关的库函数引入文件中。

(2) 如果被调函数是用户自定义的函数，有两种处理方式：①被调函数在主调函数之前进行定义，则不用进行函数声明，主调函数可以直接调用被调函数。②被调函数在主调函数之后进行定义，则应在主调函数调用之前对被调函数进行函数声明。

函数声明的主要作用是把用户自定义的函数信息通知编译器，包括函数名、参数的个数、参数的类型等。这样在遇到函数调用时，编译系统可以正确识别函数并且检查相关调用是否合法。下面对这种处理方式进行详细说明。

在介绍函数声明以前，首先说明一下函数声明的目的。函数声明是对函数基本特征(包括函数类型、函数名、参数表)的描述，它与函数定义是有区别的。而函数定义除了描述函数的基本特征外，核心内容是对函数功能的具体描述。函数声明只是描述函数原型(函数首部称为函数原型)，其一般形式如下：

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

编译系统在对函数进行编译时，并不检查函数中的参数名，因此有时也将函数原型中的形参名省略，其形式如下：

```
函数类型 函数名(形参类型 1,形参类型 2,…);
```

同时，函数原型中定义的形参类型、形参顺序和形参数目应与函数定义中的这些函数基本特征保持一致，否则在函数调用时容易产生错误。

例 5-5 素数回文数。


例 5-5

【问题描述】

151 既是素数又是个回文数，找出某个范围内的素数回文数。

【输入格式】

两个整数 a 和 b(2≤a<b≤10000000)。

【输出格式】

对每组数据，按从小到大输出 a，b 之间所有满足条件的素数回文数(包括 a 和 b)。

【样例输入】

```
5 500
```

【样例输出】

```
5
7
11
101
```

131
151
181
191
313
353
373
383

【问题分析】

本题对给定的数字进行两个条件的判断：一是素数；二是回文数。对回文数的判断，可以通过将数字逆序，判断逆序后的数字与给定的数字是否相等进行判断。

【参考代码】

```
#include<bits/stdc++.h>
using namespace std;
int judge1(int);                        //函数声明
int judge2(int);                        //函数声明
int main()
{
    int a,b,i;
    cin>>a>>b;
    for(i=a;i<=b;i++){
        if(judge1(i) && judge2(i))
            cout<<i<<endl;
    }
    return 0;
}
int judge1(int n){                      //判断是否为素数
    if(n==1 || n==0)
        return 0;
    for(int i=2;i*i<=n;i++){
        if(n%i==0)
            return 0;
    }
    return 1;
}
int judge2(int n){                      //判断是否为回文数
    int m=0,temp=n;
    while(temp){
        m=m*10+temp%10;
        temp/=10;
    }
    return m==n;
}
```