第5章 函　数

5.1　函数定义与调用

5.1.1　知识要点

- 函数的概念及定义方法。
- 函数的声明方法。
- 函数的调用概念及方法。
- 函数间参数传递的使用方法。

【知识点讲解】 C语言中子程序的作用是由函数来完成的，该定义与数学中的函数概念不完全相同。当需要完成某一个功能时，就定义一个函数，将实现某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，就可以通过调用该函数来实现。一个C语言程序可以由一个主函数和若干子函数构成，由主函数调用其他函数，其他函数之间可以互相调用，同一个函数可以被一个或多个函数调用任意多次。函数可以分为标准函数和用户定义函数，并通过参数进行函数之间的数值传递。

一个函数定义是由函数首部和函数体构成的，函数首部用来确定函数有无返回值、返回值的类型、函数名、函数形参的定义，函数体由函数中用到变量的定义和为完成函数功能的语句组成。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。在函数调用过程中完成实参与形参之间数据的传递。本次实验涉及的知识点关系如图5-1所示。

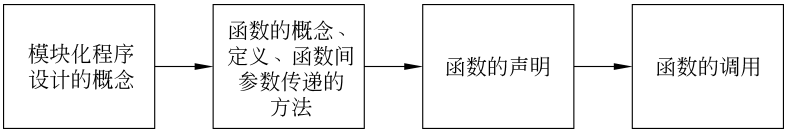

图5-1　本次实验涉及的知识点关系图

5.1.2　上机实验

1. 实验目的

- 理解函数的基本概念。
- 掌握函数定义的一般形式、函数的调用方式、函数的声明方法。

- 理解并掌握函数中的实参与形参的概念及函数间数值传递的方法。
- 了解函数类型定义与函数返回值之间的联系。

2. 实验参考

【例 5-1】 编写程序,判断输入数值是否为质数。

【源程序】

```
#include "stdio.h"
int main()
{
    int prime(int i);
    int i,j;
    printf("input a value:");
    scanf("%d",&i);
    j=prime(i);
    if(j==0)
        printf("i is not a prime number");
    else
        printf("i is a prime number");
    return 0;
}
int prime(int i)
{
    int j;
    for(j=2;j<i-1;j++)
    {
        if(i%j==0)
            return 0;
    }
    return 1;
}
```

【程序输入】

```
input a value:113<CR>
```

【程序输出】

```
i is a prime number
```

【程序分析】 如果一个整数不能够被除 1 和它本身之外的任意一个整数整除,则认为该整数为质数,也称素数。程序定义了函数 prime 实现判断任意一个整数是否为素数的功能,为了实现数据在主函数和子函数之间的传递,在函数 prime 中定义一个形参接受主函数传过来的实参数据,函数 prime 通过返回值 0 或者 1 来区分当前判断的整数是否为素数。主函数通过子函数的计算结果进行判断,最终输出结果。注意由于被调函数定义在主调函数之后,函数在使用前需要进行函数声明。

【例 5-2】 一个质数，如果其数字位置对换后仍然为质数，则称该数为绝对质数，编写程序判断一个两位数是否为绝对质数。如 13，数字位置对换后为 31，仍然为质数，所以称 13 和 31 为绝对质数。

【源程序】

```
#include "stdio.h"
int invert(int i);
int prime(int i);
int main()
{
    int i,j,k;
    printf("input a value:");
    scanf("%d",&i);
    k=invert(i);
    j=prime(i)&&prime(k);
    if(j==0)
        printf("%d is not an absolute prime number", i);
    else
        printf("%d is an absolute prime number", i);
    return 0;
}
int prime(int i)
{
    int j;
    for(j=2;j<i-1;j++)
    {
        if(i%j==0)
            return 0;
    }
    return 1;
}
int invert(int i)
{
    int j,k;
    j=i/10;
    k=i%10;
    return k*10+j;
}
```

【程序输入】

```
input a value:31<CR>
```

【程序输出】

```
31 is an absolute prime number
```

【程序分析】 程序定义三个函数分别为主函数、判断一整数是否为素数函数prime、将任一整数翻转函数invert,在主函数中首先调用函数invert完成将输入的整数翻转,然后调用函数prime判断原数值和翻转后的数值是否均为素数。如果原数值和翻转后的数值均为质数,则该数值为绝对质数。由于本题只要求判断一个两位数,因此其翻转函数不具有普遍性,只能处理两位数的翻转,该函数可以进行扩展,以支持多位数的翻转。注意由于被调函数定义在主调函数之后,函数在调用前需要进行函数说明。

【例5-3】 编写一个程序,求所有的两位数的绝对质数。

【源程序】

```
#include "stdio.h"
int invert(int i);
int prime(int i);
void output();
int main()
{
    output();
    return 0;
}
int prime(int i)
{
    int j;
    for(j=2;j<i-1;j++)
    {
        if(i%j==0)
            return 0;
    }
    return 1;
}
int invert(int i)
{
    int j,k;
    j=i/10;
    k=i%10;
    return k*10+j;
}
void output()
{
    int i,j,k;
    for(i=10;i<100;i++)
    {
        k=invert(i);
        j=prime(i)&&prime(k);
        if(j!=0)
            printf("%d is an absolute prime number.\n",i);
```

```
    }
}
```

【程序输出】

```
11 is an absolute prime number.
13 is an absolute prime number.
17 is an absolute prime number.
31 is an absolute prime number.
37 is an absolute prime number.
71 is an absolute prime number.
73 is an absolute prime number.
79 is an absolute prime number.
97 is an absolute prime number.
```

【程序分析】 程序定义 4 个函数，分别为主函数、判断一整数是否为素数的函数 prime、将任一整数翻转的函数 invert、完成程序功能并输出结果的函数 output。在函数 output 中用 for 循环完成判断所有两位数是否为绝对质数的判断并输出结果，在循环体内首先调用函数 invert 完成一个两位整数翻转，然后调用函数 prime 判断原数值和翻转后数值是否均为素数。如果原数值和翻转后数值均为质数，则 i 为所求，输出数据 i。注意由于被调函数定义在主调函数之后，函数在调用前需要进行函数说明。

3. 实验内容

编写程序并上机调试运行。

(1) 写一个函数，求以下数列前 N 项之和。

$\frac{2}{1},\frac{3}{2},\frac{5}{3},\frac{8}{5},\frac{13}{8},\frac{21}{13},\cdots$

(2) 写一个函数，

$$S_n=a+aa+aaa+\cdots+\overbrace{aa\cdots a}^{n个a}$$

求多项式的前 n 项和，其中 a 是一个数字。例如 2＋22＋222＋2222＋22222(此时 n 为 5)，n 由键盘输入。

(3) 将 10～20 的全部偶数分解为两个素数之和。

(4) 编程利用函数求从 3～100 的双质数(给定质数 p 和 q，如果 p＝q＋2，则 p 和 q 为双质数)。

(5) 编程实现求 10 000 以内的回文数。其中，函数实现判断任一正整数是否为回文数。所谓回文数就是将一个数从左向右读与从右向左读是一样的，例如：121 和 1331 都是回文数。

(6) 写一个函数，根据程序随机生成的两位数的彩票和用户输入的两位数，计算用户赢得的奖金数。如果用户输入的数和彩票的实际顺序匹配，奖金为 10 000 元；如果用户输入的所有数字匹配彩票的所有数字，但顺序不匹配，则奖金为 3000 元；如果用户输入的数字中只有一个匹配彩票中的某个数字，则奖金为 1000 元。

(7) 假设你每月在储蓄账户上存 100 元，年利率是 5%，则每月的利率是 0.05/12＝

0.004 17。第一个月后,账户上的值变成100×(1+0.004 17)=100.417;第二个月后,账户上的值变成(100+100.417)×(1+0.004 17)=201.252;第三个月后,账户上的值变成(100+201.252)×(1+0.004 17)=302.507,以此类推。写一个函数,根据用户输入的每月的存款数、年利率和月份数,计算给定月份后账户上的钱数。

5.1.3 实验过程中的常见问题与解决方法

【问题5-1】 函数定义导致的错误。

函数定义导致的错误比较多,常见几种错误如下:

(1) 变量重复定义错误。

```
int fun(int a, int b)
{
    int a,x;
}
```

说明:函数内部定义的局部变量不能和形式参数重名,因为形参也是函数的局部变量。

(2) 形式参数定义错误。

```
int abc(int x, y)
{
    ...;
}
```

说明:每个形式参数前都必须写数据类型,即使两个形式参数的类型相同,也不能省略其中一个不写。

【问题5-2】 函数类型不匹配导致的问题。

函数定义时需要注意两个部分:一个是函数的形式参数,另一个是函数的类型说明。形式参数部分用来向函数内部传递数据,类型说明指定函数返回的数据类型。形式参数部分不容易被忽略,而函数的返回类型部分往往被忽略,会产生此类错误。

例如:

```
void mul(int a, int b)
{
    int c;
    c=a*b;
    return c;
}
```

mul函数是计算两个数的乘积并返回结果,在函数体内明明写了return语句,却将函数返回值类型写成void。

除了在函数定义的时候出现不匹配以外,在调用函数的时候也有可能出现各式各样的不匹配现象,例如,函数的形式参数需要整型类型,但是函数在调用的时候使用了双精度的实型数据类型或者是把一个实型类型的函数返回值赋值给一个整数变量。

解决方法:牢记函数定义及传递参数时类型的一致性要求。

【问题 5-3】 函数声明和函数定义不一致。

例如，函数定义如下：

```
float fun(float x)
{
float y;
    ...
}
```

以下是错误的函数声明：

```
fun(float x);                          //没写函数返回值类型
int fun(float x);                      //函数返回值类型与函数定义中的不一致
float fun(int x);                      //参数 x 的类型与函数定义中的不一致
float fun(float x,float y);            //参数个数与函数定义中的不一致
```

解决方法：可先定义函数，然后复制函数首部，其后加上分号即为函数声明。

5.2　函数的嵌套和递归

5.2.1　知识要点

- 函数嵌套和递归的概念。
- 函数嵌套和递归调用的使用方法。

【知识点讲解】 在 C 语言中，函数的定义是同层次的，即在一个函数体中不允许再定义一个新的函数。而函数间的调用可以是任意的，允许一个函数体内部调用其他函数，也允许自己调用自己，因此可以分为嵌套调用和递归调用：嵌套调用是指 A 函数调用了 B 函数，而 B 函数又调用了 C 函数，通过函数之间的多层调用来实现程序功能；递归调用在调用一个函数的过程中又出现直接或间接地调用函数自己，称为函数的递归调用。本次实验涉及的知识点关系如图 5-2 所示。

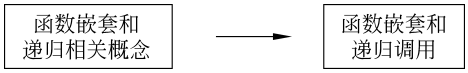

图 5-2　本次实验涉及的知识点关系图

5.2.2　上机实验

1. 实验目的

- 理解函数嵌套和递归的基本概念。
- 掌握函数嵌套和递归的调用方法。

2. 实验参考

【例 5-4】 编写程序，采用递归计算 n 的阶乘。

【源程序】

```
#include "stdio.h"
float fac(int n);
int main()
{int n;
    float m;
    printf("input n:");
    scanf("%d",&n);
    m=fac(n);
    printf("fac(%d)=%.0f\n",n,m);
    return 0;
}
float fac(int i)
{
    if(i==0 ‖ i==1)
        return 1;
    else
        return i*fac(i-1);
}
```

【程序输入】

```
input n:14<CR>
```

【程序输出】

```
fac(14)=1278945280
```

【程序分析】 要得到n的阶乘,必须知道n－1的阶乘,且需要初始化0和1的阶乘值。因此,当n为0或1时,fac(n)＝1,该部分是函数用于递归调用结束时使用,因此必不可少;n大于1时,fac(n)＝n＊fac(n－1)。

【例5-5】 用递归算法写一个计算组合的函数。

【源程序】

```
#include "stdio.h"
int zh(int m, int n);
int main()
{
    int m,n;
    printf("input m and n:");
    scanf("%d%d",&m,&n);
    printf("the result is %d\n",zh(m,n));
    return 0;
}
int zh(int m, int n)
{
    if(n>m-n)
```

```
        return zh(m,m-n);
    else if(n==0)
        return 1;
    else if(n==1)
        return m;
    else
        return zh(m-1,n-1)+zh(m-1,n);
}
```

【程序输入】

```
input m and n:5 2<CR>
```

【程序输出】

```
the result is 10
```

【程序分析】

组合计算的公式为 $C_m^n=\begin{cases}1 & n=0\\ m & n=1\\ C_{m-1}^{n-1}+C_{m-1}^{n} & n>1\end{cases}$，该公式可以保证函数的增量计算，进而保证函数递归的实现。

【例 5-6】 计算两个整数的最大公约数。

【源程序】

```
#include "stdio.h"
int mcf(int a, int b);
void main()
{
    int x,y,z;
    printf("input x, y:");
    scanf("%d%d",&x, &y);
    z=mcf(x,y);
    printf("the max common factor=%d\n",z);
}
int mcf(int a, int b)
{
    if(a<=0 || b<=0)
        return -1;
    if(a==b)
        return a;
    else if(a>b)
        return mcf(a-b,b);
    else
        return mcf(a,b-a);
}
```

【程序输入】

```
input x, y:16 27<CR>
```

【程序输出】

```
the max common factor=1
```

【程序分析】 最大公约数的性质包括三个：如果x>y,则x和y的最大公约数与x−y和y的最大公约数相同;如果y>x,则x和y的最大公约数与x和y−x的最大公约数相同;如果x=y,则x和y的最大公约数与x和y的值相同。

3. 实验内容

编写程序并上机调试运行。

(1) 用递归法将一个整数n转换成字符串。例如,输入483,应输出字符串“483”。n的位数不确定,可以是任意位数的整数。

(2) 用递归方法求n阶勒让德多项式的值,递归公式为

$$p_n(x)=\begin{cases}1 & (n=0)\\ x & (n=1)\\ ((2n-1)\times x-p_{n-1}(x)-(n-1)\times p_{n-2}(x))/n & (n>1)\end{cases}$$

(3) 采用递归方法计算x的n次方。

(4) 编程利用函数递归实现如下函数的计算：f(1)=1;f(2)=2;f(n)=f(n−1)*f(n−2)。

5.2.3 实验过程中的常见问题与解决方法

【问题5-4】 递归无法结束。

缺少对输入的数据进行的合法性检查过程,当输入非法数据时也执行递归,导致递归不能结束。

例如：计算n的阶乘,输入的n应该大于0,若n小于0则不应该进行计算。如果不对输入的n进行判断,当n小于0时也会进行递归,这样肯定会出错的。

解决方法：加上对n是否小于0的判断,如果小于0,则不进行递归。

5.3 函数的高级应用

5.3.1 知识要点

- 理解并掌握局部变量和全局变量的概念、定义方法及其作用域。
- 理解当局部变量和全局变量同名时,在局部变量的作用域内,全局变量被覆盖。
- 了解使用全局变量的利与弊。
- 了解static和extern说明符对变量的影响。
- 不带参数的宏定义和带参数的宏定义。

【知识点讲解】 本次实验涉及的主要内容包括变量的存储属性和作用域以及简单的编译预处理。变量的作用域和存储属性往往具有一定的关联。对于全局变量来说,其作用域