第3章　函　　数

要点导读

本章的主要目标是学会将一段功能相对独立的程序写成一个函数，为下一章学习类和对象打好必要的基础。掌握函数定义和调用的语法形式并不难，但是要有效地应用函数，必须对函数调用的执行过程和参数的传递有深刻的认识，这也正是初学时的难点。

要很好地理解函数的调用和参数传递，尤其是嵌套调用和递归调用的执行过程，比较有效的方法是利用编译器的调试功能，跟踪函数调用的执行过程、观察参数和变量的值，实验3会引导你进行跟踪和观察。

利用引用传递参数，是函数间数据共享的一个重要方法，但是一部分读者对引用类型的理解会有困难，其实只要简单地将引用理解为一个别名就可以了。

在介绍函数的同时，本章也介绍了一些有用的算法。例3-6介绍了产生随机数序列的方法，例3-8、3-9、3-10介绍了递归算法。本章的例题程序与第2章相比显然复杂了一些，需要仔细阅读并上机调试才能完全理解。对于较复杂的程序，书中都以注释的形式给出了详细说明，请读者在阅读程序的时候务必认真阅读注释文字。递归算法是一种非常简洁高效的算法，用途很广泛，但理解起来有一定的难度，自己编写递归程序更不是件容易的事。作为初学者，对此不必着急。学习是一个循序渐进的过程，本章介绍递归算法主要是为了说明C++语言允许函数的递归调用，如果要完全理解和熟练编写递归程序，还需要学习“数据结构”课程，一般的“数据结构”书中都会详细介绍递归算法及其应用。当然，喜欢钻研的读者不妨准备一张大纸，在利用调试功能跟踪递归程序的执行过程时，记录下递归过程中各个变量的值，会有助于对递归算法的理解。

实验3　函数的应用(2学时)

一、实验目的

(1) 掌握函数的定义和调用方法。

(2) 练习重载函数的使用。

(3) 练习使用系统函数。

(4) 学习使用Visual Studio 2008以及Eclipse的Debug调试功能，使用step into追踪到函数内部。

二、实验任务

(1) 编写一个函数把华氏温度转换为摄氏温度，转换公式为：C=(F-32)*5/9。

(2) 编写重载函数Max1可分别求取两个整数，三个整数，两个双精度数，三个双精度

数的最大值。

(3) 使用系统函数 pow(x,y)计算 x^y 的值,注意包含头文件 math.h。

(4) 用递归的方法编写函数求 Fibonacci 级数,观察递归调用的过程。

三、实验步骤

(1) 编写函数 float Convert(float TempFer),参数和返回值都为 float 类型,实现算法 C=(F-32)*5/9,在 main( )函数中实现输入、输出。程序名:lab3_1.cpp。

(2) 分别编写 4 个同名函数 max1,实现函数重载,在 main( )函数中测试函数功能。程序名:lab3_2.cpp。

(3) 在 main( )函数中提示输入两个整数 x、y,使用 cin 语句得到 x、y 的值,调用 pow(x,y)函数计算 x 的 y 次幂的结果,再显示出来。程序名:lab3_4.cpp。

(4) 编写递归函数 int fib (int n),在主程序中输入 n 的值,调用 fib 函数计算 Fibonacci 级数。公式为 fib(n)=fib(n-1)+fib(n-2),n>2; fib(1)=fib(2)=1;使用 if 语句判断函数的出口,在程序中用 cout 语句输出提示信息。程序名:lab3_5.cpp。

(5) 使用 Debug 中的 Step Into 追踪到函数内部,观察函数的调用过程,参考程序如下:

```
//lab3_5
#include <iostream>
using namespace std;

int fib(int n);
int main()
{
    int n, answer;
    cout<<"Enter number: ";
    cin>>n;
    cout<<"\n\n";
    answer=fib(n);
    cout<<answer<<" is the "<<n<<"th Fibonacci number\n";
    return 0;
}

int fib (int n)
{
    cout<<"Processing fib("<<n<<")...";
    if (n<3 )
    {
        cout<<"Return 1!\n";
        return (1);
    }
    else
    {
        cout<<"Call fib("<<n-2<<") and fib("<<n-1<<").\n";
```

```
        return( fib(n-2) +fib(n-1));
    }
}
```

(6) 调试操作步骤如下：

① Visual Studio 2008 中的调试方法。

a. 选择菜单命令 Debug|Step Into 或按下快捷键 F11，系统进入单步执行状态，程序开始运行，会出现一个 DOS 窗口，此时在源码中光标将停在 main( )函数的入口处。

b. 把光标移到语句“answer＝fib(n)”前，并在该行单击鼠标右键，在弹出的右键菜单中单击 Run to Cursor，在程序运行的 DOS 窗口中按提示输入数字 10，这时回到源码中，光标停在第 11 行，观察一下 n 的值(观察方法见实验 2)。

c. 从 Debug 菜单或 Debug 工具栏中单击 Step Into，程序进入 fib 函数，观察一下 n 的值，把光标移到语句“return(fib(n－2)＋fib(n－1))”前，并在该行单击鼠标右键，在弹出的右键菜单中单击 Run to Cursor，再单击 Step Into，程序递归调用 fib 函数，又进入 fib 函数，观察一下 n 的值。

d. 继续执行程序，参照上述方法，观察程序的执行顺序，加深对函数调用和递归调用的理解。

e. 再试试 Debug 菜单栏中别的菜单项，熟悉 Debug 的各种方法。

② Eclipse IDE for C/C++ Developers 1.2.2 中的调试方法。

a. 使用 Run|Debug As|Local C/C++ Application，或按下快捷键 F11，系统进入单步执行状态，程序开始运行，此时在源码中光标将停在 main( )函数的入口附近。

b. 把光标移到语句“answer＝fib(n)”前，并单击 Run|Run to Line，在程序运行的 Console 窗口中按提示输入数字“10”，这时回到源码中，光标停在第 11 行，观察一下 n 的值(观察方法见实验 2)。

c. 从 Run 菜单中单击 Step Into，程序进入 fib 函数，观察一下 n 的值，把光标移到语句“return(fib(n－2)＋fib(n－1))”前，从 Run 菜单中单击 Run to Line，再单击 Step Into，程序递归调用 fib 函数，又进入 fib 函数，观察一下 n 的值。

d. 继续执行程序，参照上述的方法，观察程序的执行顺序，加深对函数调用和递归调用的理解。

e. 再试试 Run 菜单栏中别的菜单项，熟悉 Debug 的各种方法。

习 题 解 答

3-1 C++ 中的函数是什么？什么叫主调函数和被调函数？二者之间有什么关系？如何调用一个函数？

解：一个较为复杂的系统往往需要划分为若干子系统，高级语言中的子程序就是用来实现这种模块划分的。C 和C++ 语言中的子程序就体现为函数。调用其他函数的函数被称为主调函数，被其他函数调用的函数称为被调函数。一个函数很可能既调用别的函数又被另外的函数调用，这样它可能在某一个调用与被调用关系中充当主调函数，而在另一个调用与被调用关系中充当被调函数。

调用函数之前先要声明函数原型。按如下形式声明：

类型标识符 被调函数名 (含类型说明的形参表)；

声明了函数原型之后，便可以按如下形式调用子函数：

函数名(实参列表)

3-2 观察下面程序的运行输出，与你设想的有何不同？仔细体会引用的用法。

源程序：

```
#include <iostream>
using namespace std;
int main()
{
    int intOne;
    int &rSomeRef=intOne;

    intOne=5;
    cout<<"intOne: \t"<<intOne<<endl;
    cout<<"rSomeRef: \t"<<rSomeRef<<endl;
    cout<<"&intOne: \t" <<&intOne<<endl;
    cout<<"&rSomeRef: \t"<<&rSomeRef<<endl;

    int intTwo=8;
    rSomeRef=intTwo; //not what you think!
    cout<<"\nintOne: \t"<<intOne<<endl;
    cout<<"intTwo: \t"<<intTwo<<endl;
    cout<<"rSomeRef: \t"<<rSomeRef<<endl;
    cout<<"&intOne: \t" <<&intOne<<endl;
    cout<<"&intTwo: \t" <<&intTwo<<endl;
    cout<<"&rSomeRef: \t"<<&rSomeRef<<endl;
    return 0;
}
```

程序运行输出(在不同的机器上或再次运行时地址可能有所不同)：

```
intOne: 5
rSomeRef:       5
&intOne:        0012FF7C
&rSomeRef:      0012FF7C

intOne: 8
intTwo: 8
rSomeRef:       8
&intOne:        0012FF7C
&intTwo:        0012FF74
&rSomeRef:      0012FF7C
```

3-3 比较值传递和引用传递的相同点与不同点。

解：值传递是指当发生函数调用时，给形参分配内存空间，并用实参来初始化形参（直接将实参的值传递给形参）。这一过程是参数值的单向传递过程，一旦形参获得了值便与实参脱离关系，此后无论形参发生了怎样的改变，都不会影响到实参。

引用传递将引用作为形参，在执行主调函数中的调用语句时，系统自动用实参来初始化形参。这样形参就成为实参的一个别名，对形参的任何操作也就直接作用于实参。

3-4 什么叫内联函数？它有哪些特点？

解：定义时使用关键字 inline 的函数叫做内联函数；编译器在编译时在调用处用函数体进行替换，节省了参数传递、控制转移等开销；内联函数体内不能有循环语句和 switch 语句；内联函数的定义必须出现在内联函数第一次被调用之前；对内联函数不能进行异常接口声明。

3-5 函数原型中的参数名与函数定义中的参数名以及函数调用中的参数名必须一致吗？

解：不必一致，所有的参数是根据位置和类型而不是名字来区分的。

3-6 调用被重载的函数时，通过什么来区分被调用的是哪个函数？

解：重载的函数的函数名是相同的，但它们的参数的个数和数据类型不同，编译器根据实参和形参的类型及个数的最佳匹配，自动确定调用哪一个函数。

3-7 完成函数。参数为两个 unsigned short int 型数，返回值为第一个参数除以第二个参数的结果，数据类型为 short int；如果第二个参数为 0，则返回值为－1。在主程序中实现输入/输出。

解：源程序：

```
#include <iostream>
using namespace std;

typedef unsigned short int USHORT;
short int divide(USHORT a, USHORT b)
{
    if (b==0)
        return -1;
    else
        return a/b;
}



int main()
{
    USHORT one, two;
    short int answer;
    cout<<"Enter two numbers.\n Number one: ";
    cin>>one;
    cout<<"Number two: ";
```

```
    cin>>two;
    answer=divide(one, two);
    if (answer >-1)
        cout<<"Answer: "<<answer;
    else
        cout<<"Error, can't divide by zero!";
    return 0;
}
```

程序运行输出：

```
Enter two numbers.
Number one: 8
Number two: 2
Answer: 4
```

3-8 编写函数把华氏温度转换为摄氏温度，公式为：

$$C=\frac{5}{9}(F-32)$$

在主程序中提示用户输入一个华氏温度，转化后输出相应的摄氏温度。

解：源程序见实验 3 部分。

3-9 编写函数判断一个数是否是质数，在主程序中实现输入/输出。

解：

```
#include <iostream>
#include <cmath>
using namespace std;

int prime(int i);                          //判断一个数是否是质数的函数

int main()
{
    int i;
    cout<<"请输入一个整数：";
    cin>>i;
    if (prime(i))
        cout<<i<<"是质数."<<endl;
    else
        cout<<i<<"不是质数."<<endl;
    return 0;
}

int prime(int i)
{
    int j,k,flag;
    flag=1;
```

```
    k=sqrt(i);
    for (j=2; j<=k; j++)
    {
        if(i%j==0)
        {
            flag=0;
            break;
        }
    }
        return flag;

}
```

程序运行输出：

```
请输入一个整数：1151
1151 是质数.
```

3-10 编写函数求两个整数的最大公约数和最小公倍数。

解：源程序：

```
#include <iostream>
#include <cmath>
using namespace std;

int fn1(int i,int j);                    //求最大公约数的函数

int main()
{
    int i,j,x,y;
    cout<<"请输入一个正整数：";
    cin>>i ;
    cout<<"请输入另一个正整数：";
    cin>>j ;

    x=fn1(i,j);
    y=i*j / x;
    cout<<i<<"和"<<j<<"的最大公约数是："<<x<<endl;
    cout<<i<<"和"<<j<<"的最小公倍数是："<<y<<endl;
    return 0;
}

int fn1(int i, int j)
{
    int temp;
    if (i<j)
```

```
    {
        temp=i;
        i=j;
        j =temp;
    }
    while(j !=0)
    {
        temp=i %j;
        i=j;
        j=temp;
    }
    return i;
}
```

程序运行输出：

```
请输入一个正整数：120
请输入另一个正整数：72
120 和 72 的最大公约数是：24
120 和 72 的最小公倍数是：360
```

3-11 什么叫做嵌套调用？什么叫做递归调用？

解：函数允许嵌套调用，如果函数 1 调用了函数 2，函数 2 再调用函数 3，便形成了函数的嵌套调用。

函数可以直接或间接地调用自身，称为递归调用。

3-12 在主程序中提示输入整数 n，编写函数用递归的方法求 1＋2＋…＋n 的值。

解：

```
#include <iostream>
#include <cmath>
using namespace std;

int fn1(int i);

int main()
{
    int i;
    cout<<"请输入一个正整数：";
    cin>>i;

    cout<<"从 1 累加到" <<i<<"的和为："<< fn1(i)<<endl;
    return 0;
}

int fn1(int i)
{
```

```
    if (i==1)
        return 1;
    else
        return i +fn1(i -1);
}
```

程序运行输出：

```
请输入一个正整数：100
从 1 累加到 100 的和为：5050
```

3-13 用递归的方法编写函数求 Fibonacci 级数，公式为：

$$F_n=F_{n-1}+F_{n-2}(n>2),\quad F_1=F_2=1$$

观察递归调用的过程。

解：源程序见实验 3 部分。

3-14 用递归的方法编写函数求 n 阶勒让德多项式的值，在主程序中实现输入/输出；递归公式为：

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

解：

```
#include <iostream>
using namespace std;

float p(int n, int x);

int main()
{
    int n,x;
    cout<<"请输入正整数 n: ";
    cin>>n;
    cout<<"请输入正整数 x: ";
    cin>>x;

    cout<<"n="<<n<<endl;
    cout<<"x="<<x<<endl;
    cout<<"P"<<n<<"("<<x<<")="<<p(n,x)<<endl;
    return 0;
}

float p(int n, int x)
{
    if (n==0)
        return 1;
```

```
    else if (n==1)
        return x;
    else
        return ((2*n-1)*x*p(n-1,x) -(n-1)*p(n-2,x)) /n ;
}
```

程序运行输出：

```
请输入正整数 n: 1
请输入正整数 x: 2
n=1
x=2
P1(2)=2

请输入正整数 n: 3
请输入正整数 x: 4
n=3
x=4
P3(4)=154
```

3-15 编写递归函数 getPower 计算 x^y，在同一个程序中针对整型和实型实现两个重载的函数：

```
int getPower(int x, int y);              //整型形式,当 y<0 时,返回 0
double getPower(double x, int y);        //实型形式
```

在主程序中实现输入/输出，分别输入一个整数 a 和一个实数 b 作为底数，再输入一个整数 m 作为指数，输出 a^m 和 b^m。另外请读者思考，如果在调用 getPower 函数计算 a^m 时希望得到一个实型结果(实型结果表示范围更大，而且可以准确表示 m＜0 时的结果)该如何调用？

解：源程序：

```
#include <iostream>
using namespace std;

int getPower(int x, int y);
double getPower(double x, int y);

int main()
{
    int iNumber, power;
    double dNumber;
    int iAnswer;
    double dAnswer;
    cout<<"Enter an int base number: ";
    cin>>iNumber;
    cout<<"Enter a double base number: ";
```