第 7 讲 函数与递归函数 1

第一部分 函数

前面我们曾经学习了程序设计中的三种基本控制结构（顺序、分支、循环）。用它们可以组成任何程序。但在应用中，还经常用到子程序结构。

通常，在程序设计中，我们会发现一些程序段在程序的不同地方反复出现，此时可以将这些程序段作为相对独立的整体，用一个标识符给它起一个名字，凡是程序中出现该程序段的地方，只要简单地写上其标识符即可。这样的程序段，我们称之为子程序。

子程序的使用不仅缩短了程序，节省了内存空间及减少了程序的编译时间，而且有利于结构化程序设计。因为一个复杂的问题总可将其分解成若干个子问题来解决，如果子问题依然很复杂，还可以将它继续分解，直到每个子问题都是一个具有独立任务的模块。这样编制的程序结构清晰，逻辑关系明确，无论是编写、阅读、调试还是修改，都会带来极大的好处。

在一个程序中可以只有主程序而没有子程序(本章以前都是如此)，但不能没有主程序，也就是说不能单独执行子程序。

在此之前，我们曾经介绍并使用了 C++提供的各种标准函数，如 sqrt()等等，这些系统提供的函数为我们编写程序提供了很大的方便。比如：求 sqrt (1)+ sqrt (2)+. . . +sqrt (100)的值。但这些函数只是常用的基本函数，编程时经常需要自定义一些函数。

输入正整数 n，求表达式$\sqrt{1}+\sqrt{2}+..+\sqrt{n}$ 的值。

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
    int n;
    double s=0;
    cin>>n;
    for(int i=1;i<=n;i++)
        s=s+sqrt(i);
    cout<<s<<endl;
    return 0;
}
```

一.函数的定义

1.函数定义的语法形式

```
数据类型  函数名（形式参数表）
{
    函数体                  //执行语句
}
```

关于函数的定义有如下说明：

函数的数据类型是函数的返回值类型（若数据类型为 void ,则无返回值）。

函数名是标识符，一个程序中除了主函数名必须为 main 外，其余函数的名字按照标识符的取名规则可以任意选取，最好取有助于记忆的名字。

形式参数（简称形参）表可以是空的（即无参函数）；也可以有多个形参，形参间用逗号隔开，不管有无参数，函数名后的圆括号都必须有。形参必须有类型说明，形参可以是变量名、数组名或指针名，它的作用是实现主调函数与被调函数之间的关系。

函数中最外层一对花括号“{ }”括起来的若干个说明语句和执行语句组成了一个函数的函数体。由函数体内的语句决定该函数功能。函数体实际上是一个复合语句，它可以没有任何类型说明，而只有语句，也可以两者都没有，即空函数。

函数不允许嵌套定义。在一个函数内定义另一个函数是非法的。但是允许嵌套使用。

函数在没有被调用的时候是静止的，此时的形参只是一个符号，它标志着在形参出现的位置应该有一个什么类型的数据。函数在被调用时才执行，也就是在被调用时才由主调函数将实际参数（简称实参）值赋予形参。这与数学中的函数概念相似，如数学函数：

$$f(x)= x^2+x+1$$

这样的函数只有当自变量被赋值以后，才能计算出函数的值。

3．函数的形式

函数的形式从结构上说可以分为三种：无参函数、有参函数和空函数。它们的定义形式都相同。

（1）无参函数

无参函数顾名思义即为没有参数传递的函数，无参函数一般不需要带回函数值，所以函数类型说明为 void。

（2）有参函数

有参函数即有参数传递的函数，一般需要带回函数值。例如

int max(int x,int y)函数。

（3）空函数

空函数即函数体只有一对花括号，花括号内没有任何语句的函数。

例如，

```
函数名（）
{   }
```

空函数不完成什么工作，只占据一个位置。在大型程序设计中，空函数用于扩充函数功能。

二.函数的声明和调用

1．函数的声明

调用函数之前先要声明函数原型。在主调函数中，或所有函数定义之前，按如下形式声明：

类型说明符 被调函数名（含类型说明的形参表）;

如果是在所有函数定义之前声明了函数原型，那么该函数原型在本程序文件中任何地方都有效，也就是说在本程序文件中任何地方都可以依照该原型调用相应的函数。如果是在某个主调函数内部声明了被调用函数原型，那么该原型就只能在这个函数内部有效。

下面对 js()函数原型声明是合法的。

```
int js(int n);
```

也可以：

```
int js(int);
```

可以看到函数原型声明与函数定义时的第一行类似，只多了一个分号，成为了一个声明语句而已。

2．函数的调用

声明了函数原型之后，便可以按如下形式调用函数：

函数名（实参列表） //例题中语句 sum+=js(i);

实参列表中应给出与函数原型形参个数相同、类型相符的实参。在主调函数中的参数称为实参，实参一般应具有确定的值。实参可以是常量、表达式，也可以是已有确定值的变量，数组或指针名。函数调用可以作为一条语句，这时函数可以没有返回值。函数调用也可以出现在表达式中，这时就必须有一个明确的返回值。

3．函数的返回值

在组成函数体的各类语句中，值得注意的是返回语句 return。它的一般形式是：

return（表达式）； // 例题中语句 return s;

其功能是把程序流程从被调函数转向主调函数并把表达式的值带回主调函数，实现函数的返回。所以，在圆括号表达式的值实际上就是该函数的返回值。其返回值的类型即为它所在函数的函数类型。当一个函数没有返回值时，函数中可以没有 return 语句，直接利用函数体的右花括号“}”，作为没有返回值的函数的返回。也可以有 return 语句，但 return 后没有表达式。返回语句的另一种形式是：

return；

这时函数没有返回值，而只把流程转向主调函数。

1.求：1!+2!+3!+……+10！

格式 1：

```
#include<iostream>
using namespace std;
int js(int n){
    int s=1;
    for(int i=1;i<=n;i++)
        s=s*i;
    return s;
}
int main(){
    int sum=0;
    for(int i=1;i<=10;i++)
        sum=sum+js(i);
    cout<<sum<<endl;
    return 0;
}
```

格式 2：

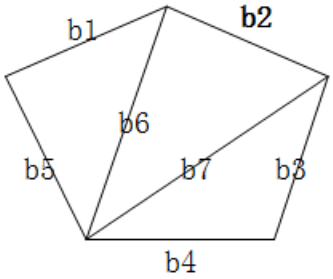
```
#include<iostream>
using namespace std;
int js(int n);
int main(){
    int sum=0;
    for(int i=1;i<=10;i++)
        sum=sum+js(i);
    cout<<sum<<endl;
    return 0;
}
```

```
int js(int n){
    int s=1;
    for(int i=1;i<=n;i++)
        s=s*i;
    return s;
}
```

2.求五边形的面积：

输入:b1,b2,b3,b4,b,b6,b7。

输出:输出五边形的面积。



3.写一个判断素数的函数，输入一个数，判断它是否是素数，是输出 yes，不是输出 no。

第二部分 递归函数

递归是一种编程技巧。

一、递归概念

当函数的定义中，其内部操作又直接或间接地出现对自身的调用，则称这样的程序嵌套定义为递归定义。

递归通常把一个大型复杂的问题层层转化为一个与原问题相似的规模较小的问题来求解，递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算，大大地减少了程序的代码量。递归的能力在于用有限的语句来定义对象的无限集合。用递归思想写出的程序往往十分简洁易懂。

例如，在数学上，所有偶数的集合可递归地定义为：

①0 是一个偶数；

②一个偶数与 2 的和是一个偶数。

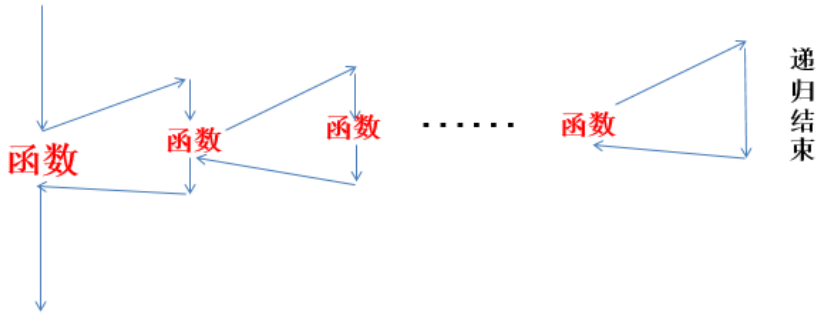
可见，仅需两句话就能定义一个由无穷多个元素组成的集合。在程序中，递归是通过函数的调用来实现的。函数直接调用其自身，称为直接递归；函数间接调用其自身，称为间接递归。

递归函数必须有终止条件，否则不停的进行下去，造成栈溢出。

递归的关键：

1.确定递归公式（关系）

2.确定边界(终止)条件



当一个函数调用另一个函数过程时：

在运行被调用函数之前，系统需要完成三件事：

（1）将所有的实在参数，返回地址等信息传给被调用的函数保存好；

（2）为被调用的函数的局部变量分配内存空间；

（3）将控制转移到被调用函数。

被调用函数结束后返回调用之前，系统也要完成三件事：

（1）保存被调用函数的计算结果；

（2）释放被调用函数的内存空间（局部变量失去作用，内存空间被回收）；

（3）依照被调用函数保存的返回地址将控制转移到调用函数。

例 1：植树

植树节那天，有 10 位同学参加了植树活动，他们完成植树的棵数都不相同。问第一位同学植了多少棵时，他指着旁边的第二位同学说比他多植了两棵；追问第二位同学，他又说比第三位同学多植了两棵；如此追问，都说比另一位同学多植两棵，最后问到第 10 位同学时，他说自己植了 5 棵。

问第一位同学到底植了多少棵树？

```
f(x)=5;   (x=10)
f(x)=f(x+1)+2    (x<10)
```

求 f(1)的值。

参考代码：

```
#include<iostream>
using namespace std;
int f(int x){
    if(x==10) return 5;
    else return f(x+1)+2;
}
int main(){
    cout<<f(1)<<endl;
    return 0;
}
```

尝试：else 不要是否可以。

模拟执行流程图示：

例 2：利用递归求 n!=1*2*…*n。

f(1)=1

f(n)=n*f(n-1)

```
#include<iostream>
```

```
using namespace std;
int f(int x){
    if(x==1) return 1;
    return x*f(x-1);
}
int main(){
    int n;
    cin>>n;
    cout<<f(n)<<endl;
    return 0;
}
```

上机练习：

第 1 部分

1. 用递归的方法求 1+2+3+……+N 的值。
2. 用递归函数输出斐波那契数列第 n 项。0，1，1，2，3，5，8，13……
3. 用递归的方法求 Hermite 多项式的值:

对给定的 x 和正整数 n，求多项式的值。

$$h_n(x) = \begin{cases} 1, & n = 0 \\ 2x, & n = 1 \\ 2xh_{n-1}(x) - 2(n-1)h_{n-2}(x), & n > 1 \end{cases}$$

4. 阿克曼(Ackmann)函数 A(m，n)中，m，n 定义域是非负整数(m<=3,n<=10)，函数值定义为：

akm(m,n) = n+1; (m=0 时)

akm(m,n) = akm(m-1,1); (m>0,n=0 时)

akm(m,n) = akm(m-1,akm(m, n-1)); （m,n>0 时)

第 2 部分：

1. 已知：

$$f(x,n) = \sqrt{\sqrt{\sqrt{\sqrt{\sqrt{\ldots\sqrt{x}}}}}}$$

n 层开方。

计算：

x=4，n=1；

x=625，n=2；

x=1000000000，n=10；

2. 已知：

$$f(x,n) = \sqrt{n + \sqrt{(n-1) + \sqrt{(n-2) + \sqrt{\ldots + 2 + \sqrt{1+x}}}}}$$

计算 x=4.2，n=10 以及 x=2.5，n=15 时的 f 的值。

3. 已知：

$$f(x,n)=\cfrac{x}{n+\cfrac{x}{(n-1)+\cfrac{x}{(n-2)+}}}$$

$$\cdot$$

$$\cdot$$

$$\cdot$$

$$\cfrac{x}{2+\cfrac{x}{1+x}}$$

计算：

x=3.5，n=100 的值。

x=199，n=10000 的值。

x=200，n=50000 的值？

第 3 部分 用递归实现

1. 输入一个非负整数，输出这个数的倒序数。例如输入 123，输出 321。

2. 用递归算法将一个十进制数 X 转换成任意进制数 M（M<=16）。

3．在程序中定义一函数 digit(n,k)，它能分离出整数 n 从右边数第 k 个数字，如 digit(31859,3)=8，digit(2076,5)=0。

4．输入一串以‘!’结束的字符，按逆序输出。（先尝试正序输出，再逆序输出）