第六章　函数和递推递归算法

- 第一节 函数
- 第二节 递推算法
- 第三节 递归算法

第一节 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abs(),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sin(1)+ sin(2)+. . . +sin(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。

例6.1　求：1!+2!+3!+……+10!

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

现在的问题是：C++不提供js(x)这样一个标准函数，这个程序是通不过的，没关系，我们编写自己的函数。如果是C++的标准函数，可以直接调用，如abs(x)，sqrt(x)．．．．．．而C++调用的标准函数需要在程序中通过#include指令加入相应的库即可。

一、函数的定义----【函数】

- 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。

2．函数定义的例子

定义一个函数，返回两个数中的较大数。

```
int max(int x,int y)
{
   return x>y?x:y;
}
```

该函数返回值是整型，有两个整型的形参，用来接受实参传递的两个数据，函数体内的语句是求两个数中的较大者并将其返回主调函数。

3. 函数从返回类型和有无参数角度可分为

一、从返回类型来看，可分为无返回值和有返回值两类

(1)无返回值函数：不返回函数值的函数，可以明确定义为“空类型”，类型说明符为“void”。如：

```
void s(int n)
{ ……
}
```

(2)有返回值的函数：此类函数被调用执行完后将向调用者返回一个执行结果， 称为函数返回值。由用户定义的这种要返回函数值的函数，必须在函数定义和函数说明中明确返回值的类型。返回值类型符是函数返回的类型说明。函数的返回值由return 语句返回。return语句的格式为：

```
return 表达式;//或者
return (表达式);
```

无参与有参数

二、从函数的形式看,函数分两类:

（1）无参函数。调用函数时不必给出参数。

定义无参函数的一般形式为

类型标识符 函数名([void])

{声明部分

语句

}

(2) 有参函数。在调用函数时,要给出参数。在主调函数和被调用函数之间有数据传递。

定义有参函数的一般形式为:

类型标识符 函数名(形式参数表列)

{声明部分

语句

}

编写一个阶乘的函数，我们给此函数取一个名字js。

```
int js(int n)                                    //函数名js，形参int n
{
   int s=1;                                  //{ }中是函数的函数体
   for (int i=1; i<=n; ++i)
   s*=i;
   return s;
}
```

在本例中，函数名叫js，只有一个int型的自变量n，函数js属int型。在本函数中，要用到两个变量i，s。在函数体中，是一个求阶乘的语句，n的阶乘的值在s中，最后由return语句将计算结果s值带回，js()函数执行结束，在主函数中js()值就是s的值。

在这里，函数的参数n是一个接口参数，说得更明确点是入口参数。如果我们调用函数：js(3)，那么在程序里所有有n的地方，n被替代成3来计算。在这里，3就被称为实参。又如：sqrt(4)，ln(5)，这里4，5叫实参。而ln(x)，sqrt(y)中的x，y叫形参。

完整的程序如下：

```
#include<iostream>
using namespace std;
int js(int);//函数的声明
int main()
{
    int sum=0;
    for (int i=1; i<=10; ++i)
      sum+=js(i);                          //函数的调用
    cout<<"sum="<<sum<<endl;
   return 0;
}

int js(int n)                              //定义的函数体
{

     int s=1;
     for (int i=1; i<=n; ++i)
       s*=i;
     return s;                             //函数的返回值
}
```

二、函数的声明和调用----【函数】

- 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。

三、函数的传值调用

- 函数传值调用的特点是将调用函数的实参表中的实参值依次对应地传递给被调用函数的形参表中的形参。要求函数的实参与形参个数相等，并且类型相同。函数的调用过程实际上是对栈空的操作过程，因为调用函数是使用栈空间来保存信息的。函数在返回时，如果有返回值，则将它保存在临时变量中。然后恢复主调函数的运行状态，释放被调用函数的栈空间，按其返回地址返回到调用函数。

- 在C++语言中，函数调用方式分传值调用和传址调用。

三、函数的传值调用

1、传值调用

这种调用方式是将实参的数据值传递给形参，即将实参值拷贝一个副本存放在被调用函数的栈区中。在被调用函数中，形参值可以改变，但不影响主调函数的实参值。参数传递方向只是从实参到形参，简称单向值传递。举个例子：

```
#include<iostream>
using namespace std;
void swap(int a,int b)
{
   int tmp=a;a=b;b=tmp;
}
int main()
{
    int c=1,d=2;
    swap(c,d);
    cout<<c<<' '<<d<<endl;
    return 0;
}          //程序输出为: 1 2
```

此例中，虽然在swap函数中交换了a,b两数的值，但是在main中却没有交换。因为swap函数只是交换c,d两变量副本的值，实参值没有改变，并没有达到交换的目的。

2.函数参数传递方式之二：地址传递

传址调用实际上还是实参到形参的拷贝，不过这次实参是要交换的两个数字的指针（即地地址），而不是要交换的两个数本身，虽然形参在swap结束后被销毁，但是形参是根据要交换的两个数的地址完成交换的，所以对这两个数字产生影响，也就完成交换

```
Void swap2(int *px, int *py)
{
int tmp=*px;
*px=*py;
*py=tmp;
print( “*px=%d,*py=%d/n” ,*px,*py);
}
main()
{
int a=4;
int b=6;
swap2(&a,&b);
Print( “a=%d,b=%d/n” , a, b);
}
```

它的输出结果是:

```
*px=6,*py=4
a=6,b=4
```

三、函数的传值调用之三：引用调用

注意这个语法在C++里合法，但是在C里面是没有的：

```
#include<iostream>
using namespace std;
void swap(int &a,int &b)                    //定义swap()函数，形参是传址调用
{
   int tmp=a;a=b;b=tmp;
}
int main()
{
    int c=1,d=2;
    swap(c,d);                              //交换变量
    cout<<c<<' '<<d<<endl;
    return 0;
}  //程序输出为：2 1
```

在此例中，因为swap函数的参数为传址调用，&a是指实参变量的地址值传递给形参，所以，在函数swap中修改a,b的值相当于在主函数main中修改c,d的值。

四、函数的应用举例----【函数】

例6.2　计算组合数C(m,n)的值(n<=m<=10)。

【分析】组合数C(m,n)可以理解为从m个数中任意取出n个数的所有情况数。求这个数值，有一个经典的计算方法：C(m,n)=m!/((m-n)!n!)。

程序如下：

```
#include<cstdio>
using namespace std;
int fac(int x);                                    //阶乘函数的声明
int main()
{
    int m,n;
    scanf("%d%d",&m,&n);
    printf("%d",fac(m)/(fac(m-n)*fac(n)));         //阶乘函数的调用
}

int fac(int x)                                     //定义阶乘函数
{
    int s=1;
    for (int i=1; i<=x; i++)
       s*=i;
    return s;                                      //阶乘函数的值返回
}
```

例6.3　输入两个数，用函数编程求出它们的最大公约数。

程序如下：

```
#include<iostream>
using namespace std;
int gcd(int x,int y);
int main()
{
    int a,b;
    cin>>a>>b;
    int g=gcd(a,b);
    cout<<g<<endl;
    return 0;
}
```

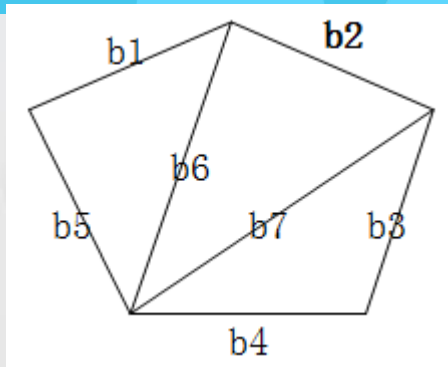
```
int gcd(int x,int y)
{
        int temp;
        while(y!=0)
        {
                temp=x%y;
                x=y;
                y=temp;
        }
        return x;
}
```

例6.4 定义一个函数check(n,d)，让它返回一个布尔值。如果数字d在正整数n的某位中出现则送回true，否则送回false。

例如：check(325719,3)==true; check(77829,1)==false;

```
#include<iostream>
using namespace std;
bool check(int,int);
int main()
{
    int a,b;
    cout<<"input n,d"<<endl;
    cin>>a>>b;
     if (check(a,b)==true) cout<<"true"<<endl;
     else cout<<"false"<<endl;
     return 0;
}
bool check(int n,int d)
{
    while (n)                    //C++中 非0为真
    {
        int e=n%10;
        n/=10;
        if (e==d) return true;
      }
    return false;
}
```

例6.5 计算如图多边形的面积。



从图中可以看出，五边形的面积是三个三角形面积之和。
程序如下：

```
#include<iostream>
#include<cstdio>      //使用printf和scanf语句，调用cstdio库
#include<cmath>
using namespace std;
double area(double,double,double);
int main()
{
    double b1,b2,b3,b4,b5,b6,b7,s;
    cout<<"please input b1,b2,b3,b4,b5,b6,b7:"<<endl;
    cin>>b1>>b2>>b3>>b4>>b5>>b6>>b7;
    s=area(b1,b5,b6)+area(b2,b6,b7)+area(b3,b4,b7);   //调用三次函数
area
    printf("s=%10.3lf\n",s);
    return 0;
}
double area(double a,double b,double c)
{
    double p=(a+b+c)/2;
    return sqrt(p*(p-a)*(p-b)*(p-c));
}
```

例6.6　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。

【分析】对于任意整数i，根据素数定义，我们从2开始，到sqrt(i)，找i的第一个约数，若找到第一个约数，则i必然不是素数。

程序如下：

```
#include<cstdio>
#include<cmath>
int prime(int x); //对于函数的声明
int main()
{
    int n;
    scanf("%d",&n);
    if (prime(n))
      printf("%s\n","yes");
    else
      printf("%s\n","no");
    return 0;
}
```

```
int prime(int x)
//判断x是否素数的函数
{
        int j;
        if (x==2) return 1;
        j=2;
        while(j<=sqrt(x) &&
x%j!=0)      j++;
        if (x%j == 0)
           return 0;
        else
           return 1;
}
```

例6.7　定义函数fa求n!。

```
#include<iostream>
#include<iomanip>
using namespace std;
void fa(int);
int t;                          //t定义为全程变量
int main()
{
    int x;
    cin>>x;
    fa(x);
    cout<<setw(5)<<x;
    cout.width(8);                  //设置域宽，表示域宽为8
    cout<<t<<endl;                 //以上两行为格式化输出，相当于printf("%8d\n",t);
    return 0;
}
void fa(int n)
{
    t=1;
    for (int i=2; i<=n; ++i) t*=i;
}
```

这里通过全程变量t，将过程中计算结果传递到t变量中。fa(x)仅仅作为程序中的一条命令被执行。

例6.8　用冒泡法对数组元素按由小到大排序（数组作为函数参数）

```
#include<iostream>
using namespace std;
void bubble(int[],int); //相当于void bubble(int a[],int n);

int main()
{                          //大数组应开为全局变量
    int array[10]={11,4,55,6,77,8,9,0,7,1};
    cout<<"排序前 ";
    for (int i=0; i<10; ++i)
      cout<<array[i]<<',';
    cout<<endl;
bubble(array,10);
     cout<<"排序后 ";
for (int i=0; i<10; ++i)
   cout<<array[i]<<',';
    cout<<endl;
    return 0;
}
```

```
void bubble(int a[],int n)
{
    for (int i=1; i<n; ++i)
{
   for (int j=0; j<n-i;++j)
        if (a[j]>a[j+1])     //判断并交换变量
        {
           int temp=a[j]; a[j]=a[j+1];
           a[j+1]=temp;
        }
    }
}
```

在前面我们已经知道数组名是该数组在内存的首地址。将数组名作为参数传给函数，实际上是把数组的地址传给函数。形参数组和实参数组的首地址重合，因此在被调用函数中对数组元素值进行改变，主调函数中实参数组的相应元素值也会改变。

例6.9　分析下列函数嵌套调用的结果。（函数的嵌套调用）

```
#include<iostream>
using namespace std;
void fun1(),fun2(),fun3();
int main()
{
   cout<<"It's in main()."<<endl;  fun2();
   cout<<"It's back in main().\n";
   // 这句语句等于 cout<<"It is back in main()."<<endl;
   return 0;
}
void fun1()
{
   cout<<"It's in fun1().\n";  fun3();
   cout<<"it's back in fun1().\n";
}
void fun2()
{
   cout<<"It's in fun2().\n";  fun1();
   cout<<"It's back in fun2().\n";
}
void fun3()
{
   cout<<"It's in fun3().\n";
}
```

程序的执行结果是：

```
    It’s in main().
    It’s in fun2().
    It’s in fun1().
    It’s in fun3().
    It’s back in fun1().
    It’s back in fun2().
    It’s back in main().
```

函数的嵌套调用指的是一个函数调用另一个函数，而被调用函数又可调用其它函数。例如，在调用A函数的过程中，可以调用B函数，在调用B函数的过程中，还可以调用C函数……当C函数调用结束后，返回到B函数，当B函数调用结束后，再返回到A函数。这就是函数的嵌套调用过程。

五、全程变量、局部变量及它们的作用域

在函数外部定义的变量称为外部变量或全局变量，在函数内部定义的变量称为内部变量或局部变量。

- 1．全局变量

全局变量的作用域是从变量定义的位置起直至本源文件结束止，即从定义位置之后的所有函数都可以访问该全局变量。

下面通过例子来说明这一点。

例6.10　全局变量的应用。

```
#include<cstdio>
#include<iostream>
using namespace std;
int x,y;          //定义全局变量x，y
int fun1(int s)
{                 //访问全局变量x，y
   x=10;
   y=x*s;
   return x+y;
}
float a,b;      //定义全局变量a，b
void fun2(int c)
{
   cout<<"x="<<x<<" y="<<y<<endl;  //访问全局变量x，y
}
int main()
{
   int m,n;
   cin>>m>>n;
   cout<<fun1(m)<<endl;
   fun2(n);
     cout<<"a="<<a<<"  b="<<b<<endl;        //访问全局变量a，b
     return 0;
}
```

当在键盘上输入6 9后程序的执行结果是：

```
70
x=10   y=60
a=0   b=0
```

程序中主函数先调用fun1()，实参是6，将实参值传给形参s(即s=6)，函数fun1中x值为10，y值为60，return语句将x＋y的和返回，fun1()结束。主函数输出返回值之后，调用fun2()，在fun2()中输出全局变量x和y 的值，之后返回主函数。在主函数中输出全局变量a和b值。由于a、b在声明时未赋初值，系统的默认值为0。

使用全局变量的说明：

- 在一个函数内部，既可以使用本函数定义的局部变量，也可以使用在此函数前定义的全局变量。
- 全局变量的作用是使得函数间多了一种传递信息的方式。如果在一个程序中多个函数都要对同一个变量进行处理，即共享，就可以将这个变量定义成全局变量，使用非常方便，但副作用也不可低估。
- 过多地使用全局变量，会增加调试难度。因为多个函数都能改变全局变量的值，不易判断某个时刻全局变量的值。
- 过多地使用全局变量，会降低程序的通用性。如果将一个函数移植到另一个程序中，需要将全局变量一起移植过去，同时还有可能出现重名问题。
- 全局变量在程序执行的全过程中一直占用内存单元。
- 全局变量在定义时若没有赋初值，其默认值为0。

2．局部变量

⑴局部变量的作用域是在定义该变量的函数内部。换句话说，局部变量只在定义它的函数内有效。在一个子程序内定义的变量也是局部变量，其作用域是该子程序。函数的形参也是局部变量。

⑵由于局部变量的作用域仅局限于本函数内部，所以，在不同的函数中变量名可以相同，它们分别代表不同的对象，在内存中占据不同的内存单元，互不干扰。

⑶一个局部变量和一个全局变量是可以重名的，在相同的作用域内局部变量有效时全局变量无效。即局部变量可以屏蔽全局变量。

⑷这里需要强调的是，主函数main中定义的变量也是局部变量，这一点与其他程序设计语言不同。

⑸ 全局变量数组初始全部为0，局部变量值是随机的，要初始化初值，局部变量受栈空间大小限制，大数组需要注意。通俗说，局部变量的数组不能开很大，全局变量随便。

六、函数的综合应用----【函数】

例6.11　编程输入十进制整数N（N：-32767～32767），请输出它对应的二进制、八进制、十六进制数。

【分析】这是一道进行数制转换的问题，将十进制整数转换成R进制的数，算法是：除以R取余，再将余数倒过来写出即是R进制的数。本例是要求把一个十进制数同时转换成二进制、八进制、十六进制数。因此可以设计一个过程同时处理这三种的进制转换。

程序如下：

```
#include<cstdlib>
#include<iostream>
using namespace std;
void TurnData(int n,int a);
char ch[6]={'A','B','C','D','E','F'};
int main()
{
   int n;
   cin>>n;
   TurnData(n,2);      //n转成2进制数
   TurnData(n,8);      //n转成8进制数
   TurnData(n,16);     //n转成16进制数
   return 0;
}
```

这里的过程TurnData中的参数不需要把什么值返回给主程序，因此设为形参即可。

```
void TurnData(int n,int a)
{
   int x[17],i,j,k=0;
   cout<<n<<" turn into "<<a<<" :"<<endl;
   if (n<0) cout<<'-'; //负数的话，先输出负号再开始转
   j=abs(n);
   do
     {
        k++;   //用于统计转成a进制数后的总位数
        i=j%a;
        j/=a;
        x[k]=i;
     }while (j!=0);
   for (int h=k; h>=1; --h)
    if (x[h]<10) cout<<x[h];
    else  cout<<ch[x[h]-10];
   cout<<endl;
  }
```

例6.12 编写一个给一个分数约分的程序。

- 程序如下：

```
#include<iostream>
#include<iomanip>
using namespace std;
void common(int x,int y);
int main()
{   int a,b;
    cin>>a>>b;
    common(a,b);
}
void common(int x,int y)
{   int m=x,n=y,r;    //用辗转相除法求x,y的最大公约数
    do
    {    r=m%n;
         m=n;
         n=r;
    }while (r!=0);
    x/=m;              //用两者的最大公约数i对x,y进行约分
    y/=m;
    cout<<setw(5)<<x<<setw(5)<<y<<endl;
}
```

运行结果：

输入 ： 12 8

输出 ： 3 2

【课堂练习】

1. 求正整数2和100之间的完全数。
完全数：因子之和等于它本身的自然数，如6=1+2+3
2. 编程求2～n(n为大于2的正整数)中有多少个素数。
3．已知　m=，输入a,b,c，求m。把求三个数的最大数max(x,y,z)分别定义成函数和过程来做。
4. 如果一个自然数是素数,且它的数字位置经过对换后仍为素数,则称为绝对素数,例如13。试求出所有二位绝对素数。
5. 自然数a的因子是指能被a整除的所有自然数，但不含a本身。例如12的因子为：1,2,3,4,6。若自然数a的因子之和为b，而且b的因子之和又等于a，则称a,b为一对“亲和数”　。求最小的一对亲和数(a<>b)。
6. 如果一个数从左边读和从右边读都是同一个数，就称为回文数。例如6886就是一个回文数，丘出所有的既是回文数又是素数的三位数。
7．根据公式arctanx(x)=x-x3/3+x5/5-x7/7+…和=6 arctanx()，定义函数arctanx(x)，求当最后一项小于10-6时的值。
8. 哥德巴赫猜想的命题之一是:大于6 的偶数等于两个素数之和。编程将6～100所有偶数表示成两个素数之和。

【上机练习】

1. 简单算术表达式求值【1.12编程基础之函数与过程抽象01】

两位正整数的简单算术运算（只考虑整数运算），算术运算为：

+，加法运算；　　-，减法运算；

*，乘法运算；　　/，整除运算；

%，取余运算。

算术表达式的格式为（运算符前后可能有空格）：

运算数 运算符 运算数

请输出相应的结果。

输入：一行算术表达式。

输出：整型算数运算的结果（结果值不一定为2位数，可能多于2位或少于2位）。

样例输入：

32+64

样例输出：

96

【上机练习】

2. 短信计费【1.12编程基础之函数与过程抽象02】

用手机发短信，一条短信资费为0.1元，但限定一条短信的内容在70个字以内(包括70个字)。如果你一次所发送的短信超过了70个字，则会按照每70个字一条短信的限制把它分割成多条短信发送。假设已经知道你当月所发送的短信的字数，试统计一下你当月短信的总资费。

输入:第一行是整数n，表示当月发送短信的总次数，接着n行每行一个整数，表示每次短信的字数。

输出:输出一行，当月短信总资费，单位为元，精确到小数点后1位。

样例输入:

```
10
39
49
42
61
44
147
42
72
35
46
```

样例输出:

```
1.3
```

【上机练习】

3.甲流病人初筛【1.12编程基础之函数与过程抽象03】

目前正是甲流盛行时期，为了更好地进行分流治疗，医院在挂号时要求对病人的体温和咳嗽情况进行检查，对于体温超过37.5度（含等于37.5度）并且咳嗽的病人初步判定为甲流病人（初筛）。现需要统计某天前来挂号就诊的病人中有多少人被初筛为甲流病人。

输入:

第一行是某天前来挂号就诊的病人数n。(n<200)

其后有n行，每行是病人的信息，包括三个信息：姓名（字符串，不含空格，最多8个字符）、体温（float）、是否咳嗽（整数，1表示咳嗽，0表示不咳嗽）。每行三个信息之间以一个空格分开。

输出:

按输入顺序依次输出所有被筛选为甲流的病人的姓名，每个名字占一行。之后在输出一行，表示被筛选为甲流的病人数量。

样例输入:

```
5
Zhang 38.3 0
Li 37.5 1
Wang 37.1 1
Zhao 39.0 1
Liu 38.2 1
```

样例输出:

```
Li
Zhao
Liu
3
```

【上机练习】

4.统计单词数【1.12编程基础之函数与过程抽象05】Noip2011普及组第2题

一般的文本编辑器都有查找单词的功能，该功能可以快速定位特定单词在文章中的位置，有的还能统计出特定单词在文章中出现的次数。

现在，请你编程实现这一功能，具体要求是：给定一个单词，请你输出它在给定的文章中出现的次数和第一次出现的位置。注意：匹配单词时，不区分大小写，但要求完全匹配，即给定单词必须与文章中的某一独立单词在不区分大小写的情况下完全相同（参见样例1），如果给定单词仅是文章中某一单词的一部分则不算匹配（参见样例2）。

输入:

第 1 行为一个字符串，其中只含字母，表示给定单词；

第 2 行为一个字符串，其中只可能包含字母和空格，表示给定的文章。

输出:

只有一行，如果在文章中找到给定单词则输出两个整数，两个整数之间用一个空格隔开，分别是单词在文章中出现的次数和第一次出现的位置（即在文章中第一次出现时，单词首字母在文章中的位置，位置从0开始）；如果单词在文章中没有出现，则直接输出一个整数-1。

样例输入:

样例 #1:

```
To
to be or not to be is a question
```

样例 #2:

```
to
Did the Ottoman Empire lose its power at that time
```

样例输出:

样例 #1:

```
2 0
```

样例 #2:

```
-1
```

【上机练习】

5.机器翻译【1.12编程基础之函数与过程抽象07】Noip2010提高组第1题

小晨的电脑上安装了一个机器翻译软件，他经常用这个软件来翻译英语文章。这个翻译软件的原理很简单，它只是从头到尾，依次将每个英文单词用对应的中文含义来替换。对于每个英文单词，软件会先在内存中查找这个单词的中文含义，如果内存中有，软件就会用它进行翻译；如果内存中没有，软件就会在外存中的词典内查找，查出单词的中文含义然后翻译，并将这个单词和译义放入内存，以备后续的查找和翻译。

假设内存中有M个单元，每单元能存放一个单词和译义。每当软件将一个新单词存入内存前，如果当前内存中已存入的单词数不超过M－1，软件会将新单词存入一个未使用的内存单元；若内存中已存入M 个单词，软件会清空最早进入内存的那个单词，腾出单元来，存放新单词。

假设一篇英语文章的长度为N个单词。给定这篇待译文章，翻译软件需要去外存查找多少次词典？假设在翻译开始前，内存中没有任何单词。

输入:

输入文件共2行。每行中两个数之间用一个空格隔开。

第一行为两个正整数M和N，代表内存容量和文章的长度。

第二行为N个非负整数，按照文章的顺序，每个数（大小不超过1000）代表一个英文单词。文章中两个单词是同一个单词，当且仅当它们对应的非负整数相同。

【上机练习】

输出:

共1行，包含一个整数，为软件需要查词典的次数。

样例输入:

样例 #1:

3 7

1 2 1 5 4 4 1

样例 #2:

2 10

8 824 11 78 11 78 11 78 8 264

样例输出:

样例 #1:

5

样例 #2:

6

提示:

输入输出样例 1 说明：

整个查字典过程如下：每行表示一个单词的翻译，冒号前为本次翻译后的内存状况：

空：内存初始状态为空。

1. 1：查找单词1并调入内存。
2. 1 2：查找单词2并调入内存。
3. 1 2：在内存中找到单词1。
4. 1 2 5：查找单词5并调入内存。
5. 2 5 4：查找单词4并调入内存替代单词1。
6. 2 5 4：在内存中找到单词4。
7. 5 4 1：查找单词1并调入内存替代单词2。

共计查了5 次词典。

【上机练习】

6.Vigenère密码【1.12编程基础之函数与过程抽象08】Noip2012提高组第1题

16世纪法国外交家Blaise de Vigenère设计了一种多表密码加密算法——Vigenère密码。Vigenère密码的加密解密算法简单易用，且破译难度比较高，曾在美国南北战争中为南军所广泛使用。

在密码学中，我们称需要加密的信息为明文，用M表示；称加密后的信息为密文，用C表示；而密钥是一种参数，是将明文转换为密文或将密文转换为明文的算法中输入的数据，记为k。 在Vigenère密码中，密钥k是一个字母串，$k = k_1 k_2 \dots k_n$。当明文$M = m_1 m_2 \dots m_n$时，得到的密文$C = c_1 c_2 \dots c_n$，其中$c_i = m_i ® k_i$，运算®的规则如下表所示：

Vigenère加密在操作时需要注意：

1.®运算忽略参与运算的字母的大小写，并保持字母在明文M中的大小写形式；

2.当明文M的长度大于密钥k的长度时，将密钥k重复使用。

例如，明文M=Helloworld，密钥k=abc时，密文C=Hfnlpyosnd。

明文 H e l l o w o r l d

密钥 a b c a b c a b c a

密文 H f n l p y o s n d

【上机练习】

输入:

第一行为一个字符串，表示密钥k，长度不超过100，其中仅包含大小写字母。第二行　　为一个字符串，表示经加密后的密文，长度不超过1000，其中仅包含大小写字母。

对于100%的数据，输入的密钥的长度不超过100，输入的密文的长度不超过1000，且都仅包含英文字母。

输出:

输出共1行，一个字符串，表示输入密钥和密文所对应的明文。

样例输入:

CompleteVictory

Yvqgpxaimmklongnzfwpvxmniytm

样例输出:

Wherethereisawillthereisaway

【上机练习】

7.素数对【1.12编程基础之函数与过程抽象10】

两个相差为2的素数称为素数对，如5和7，17和19等，本题目要求找出所有两个数均不大于n的素数对。

输入:

一个正整数n。1<=n<=10000。

输出:

所有小于等于n的素数对。每对素数对输出一行，中间用单个空格隔开。若没有找到任何素数对，输出empty。

样例输入:

100

样例输出:

3 5
5 7
11 13
17 19
29 31
41 43
59 61
71 73

【上机练习】

8．我家的门牌号【小学奥数7649】

我家住在一条短胡同里，这条胡同的门牌号从1开始顺序编号。

若其余各家的门牌号之和减去我家门牌号的两倍，恰好等于n，求我家的门牌号及总共有多少家。数据保证有唯一解。

输入:

一个正整数n。n<100000。

输出:

一行，包含两个正整数，分别是我家的门牌号及总共有多少家，中间用单个空格隔开。

样例输入:

100

样例输出:

10 15

【上机练习】

9．质数的和与积【小学奥数7827】

两个质数的和是S，它们的积最大是多少?

输入:

一个不大于10000的正整数S，为两个质数的和。

输出:

一个整数，为两个质数的最大乘积。数据保证有解。

样例输入:

50

样例输出:

589

【上机练习】

10.单词替换【1.7编程基础之字符串17】

输入一个字符串，以回车结束（字符串长度<=100）。该字符串由若干个单词组成，单词之间用一个空格隔开，所有单词区分大小写。现需要将其中的某个单词替换成另一个单词，并输出替换之后的字符串。

输入:

第1行是包含多个单词的字符串 s;

第2行是待替换的单词a(长度 <= 100);

第3行是a将被替换的单词b(长度 <= 100)。

s,a,b最前面和最后面都没有空格。

输出:

输出只有 1 行，将s中所有单词a替换成b之后的字符串。

样例输入:

You want someone to help you

You

I

样例输出:

I want someone to help you

【上机练习】

11.笨小猴【1.9编程基础之顺序查找06】Noip2008提高组第1题

笨小猴的词汇量很小，所以每次做英语选择题的时候都很头疼。但是他找到了一种方法，经试验证明，用这种方法去选择选项的时候选对的几率非常大！

这种方法的具体描述如下：假设maxn是单词中出现次数最多的字母的出现次数，minn是单词中出现次数最少的字母的出现次数，如果maxn-minn是一个质数，那么笨小猴就认为这是个Lucky Word，这样的单词很可能就是正确的答案。

输入:

只有一行，是一个单词，其中只可能出现小写字母，并且长度小于100。

输出:

共两行，第一行是一个字符串，假设输入的的单词是Lucky Word，那么输出“Lucky Word”，否则输出“No Answer”；

第二行是一个整数，如果输入单词是Lucky Word，输出maxn-minn的值，否则输出0。

样例输入:	样例输出:	
样例 #1:	样例 #1:	样例 #2:
error	Lucky Word	No Answer
样例 #2:	2	0
olympic		

【上机练习】

12.素数回文数的个数【1.13编程基础之综合应用05】

求11到n之间（包括n），既是素数又是回文数的整数有多少个。

输入:
一个大于11小于1000的整数n。

输出:
11到n之间的素数回文数个数。

样例输入:
23

样例输出:
1

提示:
回文数指左右对称的数，如：292，333。

【上机练习】

13.判决素数个数【1.13编程基础之综合应用10】

输入两个整数X和Y，输出两者之间的素数个数（包括X和Y）。

输入:

两个整数X和Y（1 <= X,Y <= 105）。

输出:

输出一个整数，表示X，Y之间的素数个数（包括X和Y）。

样例输入:

1 100

样例输出:

25

【上机练习】

14.最大质因子序列【1.13编程基础之综合应用21】

任意输入两个正整数m,n(1<m<n<=5000)，依次输出m到n之间每个数的最大质因子(包括m和n；如果某个数本身是质数，则输出这个数自身)。

输入:

一行，包含两个正整数m和n，其间以单个空格间隔。

输出:

一行，每个整数的最大质因子，以逗号间隔。

样例输入:

5 10

样例输出:

5,3,7,2,3,5

【上机练习】

15.区间内的真素数【1.13编程基础之综合应用23】

找出正整数M和N之间（N不小于M）的所有真素数。

真素数的定义：如果一个正整数P为素数，且其反序也为素数，那么P就为真素数。

例如，11，13均为真素数，因为11的反序还是为11，13的反序为31也为素数。

输入:

输入两个数M和N，空格间隔，1<=M<=N<=100000。

输出:

按从小到大输出M和N之间（包括M和N）的真素数，逗号间隔。如果之间没有真素数，则输出No。

样例输入:

10 35

样例输出:

11,13,17,31

【上机练习】

16. 二进制分类【1.13编程基础之综合应用36】

若将一个正整数化为二进制数，在此二进制数中，我们将数字1的个数多于数字0的个数的这类二进制数称为A类数，否则就称其为B类数。

例如：

(13)10=(1101)2，其中1的个数为3，0的个数为1，则称此数为A类数；
(10)10=(1010)2，其中1的个数为2，0的个数也为2，称此数为B类数；
(24)10=(11000)2，其中1的个数为2，0的个数为3，则称此数为B类数；
程序要求：求出1～1000之中（包括1与1000），全部A、B两类数的个数。

输入：

无。

输出：

一行，包含两个整数，分别是A类数和B类数的个数，中间用单个空格隔开。

【上机练习】

17.确定进制【1.13编程基础之综合应用34】

6*9=42对于十进制来说是错误的，但是对于13进制来说是正确的。即，6(13)* 9(13)= 42(13)， 而 42(13)=4*131+2*130=54(10)。

你的任务是写一段程序，读入三个整数p、q和 r，然后确定一个进制B(2<=B<=16) 使得 p * q = r。 如果 B 有很多选择, 输出最小的一个。

例如：p=11，q=11，r=121.则有11(3)* 11(3)= 121(3)因为 11(3)= 1 * 31+ 1 * 30= 4(10)和121(3)=1*32+2*31+1*30=16(10)。对于进制 10，同样有11(10)* 11(10)= 121(10)。这种情况下，应该输出 3。如果没有合适的进制，则输出 0。

输入:

一行，包含三个整数p、q、r。 p、q、r的所有位都是数字，并且1 <= p、q、r <= 1,000,000。

输出:

一个整数：即使得p*q=r成立的最小的B。如果没有合适的B，则输出0。

样例输入:

6 9 42

样例输出:

13

第二节 递推算法

递推和递归是编程中常用的基本算法。在前面的解题中已经用到了递推算法，下面对这两种算法的基本应用进行详细研究讨论。递推算法是一种用若干步可重复的简单运算（规律）来描述复杂问题的方法。

递推算法以初始{起点}值为基础，用相同的运算规律，逐次重复运算，直至运算结束。这种从“起点”重复相同的方法直至到达一定“边界”，犹如单向运动，用循环可以实现。递推的本质是按规律逐次推出（计算）下一步的结果。

例6.13　植树节那天，有五位参加了植树活动，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。最后问到第五位同学时，他说自己植了10棵。到底第一位同学植了多少棵树?

【分析】设第一位同学植树的棵数为a1，欲求a1，需从第五位同学植树的棵数a5入手，根据“多两棵”这个规律，按照一定顺序逐步进行推算：

①a5=10；

②a4=a5+2=12；

③a3=a4+2=14；

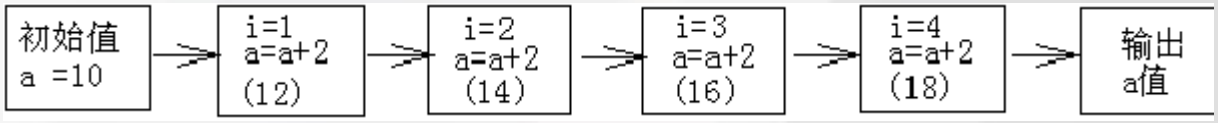
④a2=a3+2=16；

⑤a1=a2+2=18；

程序如下：

```
#include<iostream>
using namespace std;
int main()
{
   int a=10; //以第五位同学的棵数为递推的起始值
   for (int i=4; i>=1; --i) //还有4人，递推计算4次
   a+=2;                    //递推运算规律
   cout<<" The Num is "<<a<<endl;
    return 0;
}
```

本程序的递推运算可用如下图示描述：



例6.14　斐波那契数列（Fibonacci）

即：f1=1　　(n=1)

　　f2=1　　(n=2)

　　fn=fn-1 + fn-2　　(n>=3)

程序如下：

```
#include<iostream>
#include<cstdio>
using namespace std
int main()
{
   int f0=1,f1=1,f2;
   int n;
   cin>>n;
  for (int i=3; i<=n; ++i)
  {
      f2=f0+f1;
      f0=f1;
      f1=f2;
  }
  printf("%d\n",f2);
  return 0;
 }
```

- 有关Fibonacci数列，我们先来考虑一个简单的问题：楼梯有n个台阶，上楼可以一步上一阶，也可以一步上两阶。一共有多少种上楼的方法?
- 这是一道计数问题。在没有思路时，不妨试着找规律。n=5时，一共有8种方法：

1. 5=1+1+1+1+1
2. 5=2+1+1+1
3. 5=1+2+1+1
4. 5=1+1+2+1
5. 5=1+1+1+2
6. 5=2+2+1
7. 5=2+1+2
8. 5=1+2+2

- 其中有5种方法第1步走了1阶，3种方法第1步走了2阶，没有其他可能。假设f(n)为n个台阶的走法总数，把n个台阶的走法分成两类。
- 第1类：第1步走1阶，剩下还有n-1阶要走，有f(n-1)种方法。
- 第2类：第1步走2阶，剩下还有n-2阶要走，有f(n-2)种方法。
- 这样，就得到了递推式：f(n)=f(n-1)+f(n-2),不要忘记边界情况：f(1)=1,f(2)=2。把f(n)的前几项列出：1，2，3，5，8，……。

- 再例如，把雌雄各一的一对新兔子放入养殖场中。每只雌兔在出生两个月以后，每月产雌雄各一的一对新兔子。试问第n个月后养殖场中共有多少对兔子。
- 还是先找找规律。
- 第1个月：一对新兔子r1。用小写字母表示新兔子。
- 第2个月：还是一对新兔子，不过已经长大，具备生育能力了，用大写字母R1表示。
- 第3个月：R1生了一对新兔子r2，一共2对。
- 第4个月：R1又生一对新兔子r3，一共3对。另外，r2长大了，变成R2
- 第5个月：R1和R2各生一对，记为r4和r5，共5对。此外，r3长成R3。
- 第6个月：R1、R2和R3各生一对，记为r6～r8，共8对。此外，r4和r5长大。
- ……
- 把这些数排列起来：1，1，2，3，5，8，……，事实上，可以直接推导出来递推关系f(n)=f(n-1)+f(n-2):第n个月的兔子由两部分组成，一部分是上个月就有的老兔子f(n-1)，一部分是上个月出生的新兔子f(n-2)(第n-1个月时具有生育能力的兔子数就等于第n-2个月兔子总数)。根据加法原理，f(n)=f(n-1)+f(n-2)。

上机练习6.2----【递推算法】

1、猴子吃枣问题：猴子摘了一堆枣，第一天吃了一半，还嫌不过瘾，又吃了一个；第二天，又吃了剩下的一半零一个；以后每天如此。到第十天，猴子一看只剩下一个了。问最初有多少个枣子?

2、任何一个自然数的立方都可以写成一串连续奇数之和。如：

13=1

23=3+5=8

33=7+9+11=27

43=13+15+17+19=64

…….

编程输入N，求N3是由哪些奇数之和。

3、楼梯有N级台阶，上楼可以一步上一阶，也可以一步上二阶。编一递推程序，计算共有多少种不同走法?

4、兔子在出生两个月以后，就具有生殖后代的能力。假设一对兔子，每月都能生一对兔子，生出来的每一对小兔子，在出生两个月后，也每月生一对兔子。那么，由一对刚出生的小兔子开始，连续不断地繁殖下去，在某个指定的月份有多少对兔子?

5、骨牌铺法：

有1×n的一个长方形，用一个1×1、1×2和1×3的骨牌铺满方格。例如当n=3时为1×3的方格。此时用1×1、1×2和1×3的骨牌铺满方格，共有四种铺法。如下图：

第三节 递归算法

- 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。

二、递归应用

例6.15　植树节那天，有五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，最后问到第五位同学时，他说自己植了10棵。问第一位同学到底植了多少棵树?

【分析】把原问题求第一位同学的植树棵数a1转化为a1=a2+2，即求a2；而求a2又转化为a2=a3+2；a3=a4+2；a4=a5+2逐层转化为求a2, a3, a4, a5且都采用与求a1相同的方法，最后的a5为已知值，用a5=10返回到上一层并代入计算出a4；又用a4的值代入上一层去求a3；…, 如此, 直到求出a1。

因此:

$$a_x = \begin{cases} 10 & (x=5) \\ a_{x+1} + 2 & (x<5) \end{cases}$$

其中求a_{x+1} 又采用求a_x的方法。所以：

①定义一个处理问题的子程序Num(x)，如果x < 5就递归调用子程序Num(x+1)；

②当递归调用到达一定条件(x=5)，就直接执行num=10，再执行后继语句，遇End返回到调用子程序的地方。

③最后返回到开头的原问题，此时所得到的运算结果就是原问题Num(1)的答案。

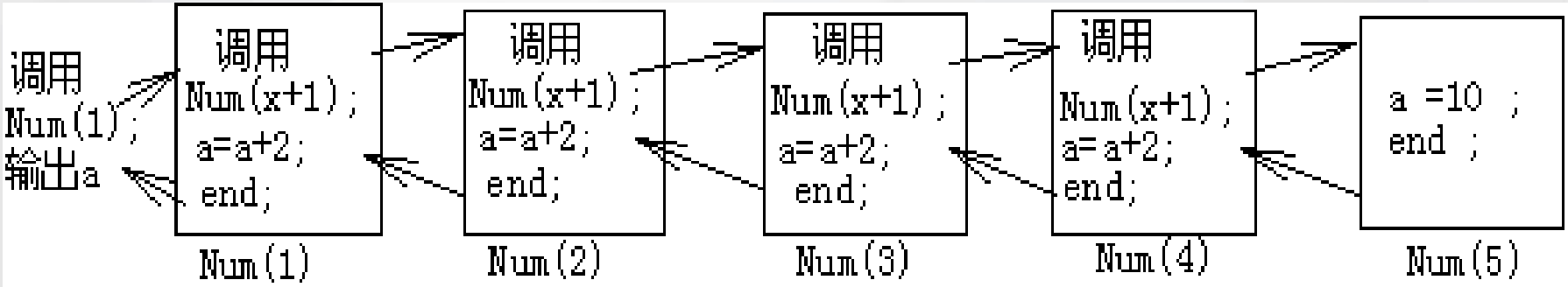
程序如下：

```
#include<iostream>
 using namespace std;
 int num(int);
 int main()
 {   cout<<" The Num is "<<num(1)<<endl;
     return 0;
 }
 int num(int x)
 {
     if (x==5) return 10;          //递归边界
     else return num(x+1)+2;    //递归式
 }
```

利用全局变量的形式，也可以传递数据，采用另一种方式编写。程序如下：

```
#include<iostream>
using namespace std;
int num(int);
int a;//定义一个全局变量a，通过全局变量传递数值
int main()
{
   num(1);  //主程序调用Num(1)求第1个人的棵数
   cout<<" The Num is "<<a<<endl;
   return 0;
}
int num(int x)
{
   if (x==5) a=10;
   else
        {
            num(x+1);      //递归调用函数Num(x+1)
            a+=2;                          //求(x+1)的棵数
         }
}
```

该程序中的递归过程图解如下：



递归方法说明如下：

①调用原问题的处理子程序（函数）时，调用程序应给出具体的子程序形参值（与形参结合的实参）；

②在处理子问题中，如果又调用原问题的处理子程序，但形参值应是不断改变的量(表达式)；

③每递归调用一次自身，系统就打开一“层”与自身相同的程序系列；

④由于调用参数不断改变，将使条件满足（达到一定边界），此时就是最后一“层”，不需再调用自身（打开新层），而是在本层往下执行后继语句，遇到end，就返回到上“层”调用此子程序的地方并继续往下执行，遇到end再向上层返回，…如此直到返回主程序；

⑤整个递归过程可视为由往返双向“运动”组成，先是逐层递进，逐层打开新的“篇章”，（有可能无具体计算值）当最终递进达到边界，执行完本“层”的语句，才由最末一“层”逐次返回到上“层”，每次返回均带回新的计算值，直至回到第一次由主程序调用的地方，完成对原问题的处理。

递归算法表现出处理问题的强大能力。然而，如同循环一样，递归也会带来无终止调用的可能性，因此，在设计递归过程（函数）时，必须考虑递归调用的终止问题，就是递归调用要受限于某一条件，而且要保证这个条件在一定情况下肯定能得到满足。

例6.16　用递归算法求x^n。

【分析】把X n 分解成：$x^0 = 1$　　（ n =0 ）

$x^1 = x * x^0$　　（ n =1 ）

$x^2 = x * x^1$　　（ n >1 ）

$x^3 = x * x^2$　　（ n >1 ）

……　　（ n >1 ）

因此将x^n 转化为：$x*x^{n-1}$，，其中求x^{n-1} 又用求x^n 的方法进行求解。

①定义子程序xn(int n)求X^n ;如果n>=1则递归调用xn(n-1) 求x^{n-1} ;

②当递归调用到达n=0时终止调用，然后执行本“层”的后继语句;

③遇到子程序运行完，就结束本次的调用，返回到上一“层”调用语句的地方，并执行其后继语句;

④继续执行步骤③，从调用中逐“层”返回，最后返回到主程序。

程序

采用函数编写程序如下：

```
#include<iostream>
using namespace std;
int xn(int);
int x;
int main()
{
   int n;
   cin>>x>>n;
    cout<<x<<'^'<<n<<"="<<xn(n
    )<<endl;
   return 0;
}
int xn(int n)
{
   if (n==0) return 1;     //递归边界
   else return x*xn(n-1);  //递归式
}
```

采用全程变量编写程序如下：

```
#include<iostream>
using namespace std;
int tt,x;     //利用全局变量tt传递结果
int xn(int);
int main()
{
   int n;
   cin>>x>>n;
   xn(n);
   cout<<x<<'^'<<n<<'='<<tt<<endl;
   return 0;
}
int xn(int n)
{
   if (n==0) tt=1;
   else
      {
         xn(n-1);//递归调用过程xn(n-1)求x n-1
         tt*=x;
      }
}
```

例6.17　用递归函数求x！

$$x! = \begin{cases} 1 & (x=0) \\ x(x-1)! & (x>0) \end{cases}$$

【分析】

根据数学中的定义把求x！定义为求x*(x-1)！，其中求(x-1)！仍采用求x！的方法，需要定义一个求x！的函数，逐级调用此函数，即：

当x=0时，x!=1；当x>0时，x!=x*(x-1)！。

假设用函数Fac(x)表示x的阶乘，当x=3时，Fac(3)的求解方法可表示为：

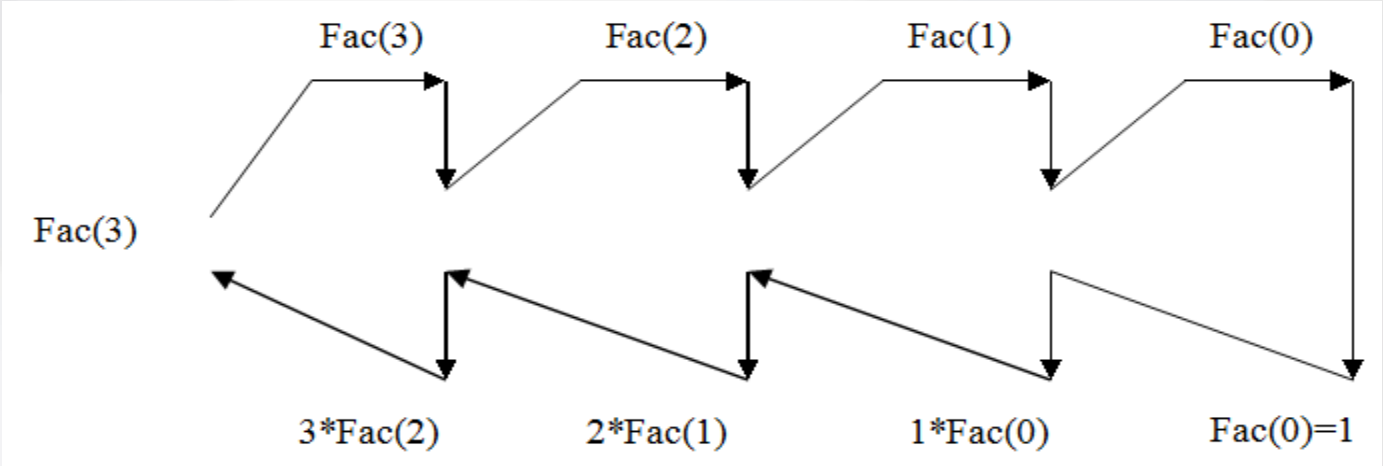
Fac(3)=3*fac(2)=3*2*Fac(1)=3*2*1*Fac(0)=3*2*1*1=6

①定义函数：int fac(int n)

如果n=0，则fac=1；如果n>0，则继续调用函数fac=n*fac(n-1)；

②返回主程序，打印fac(x)的结果。

它的执行流程如下图所示：



采用有参函数编写程序如下：

```
#include<iostream>
using namespace std;
int fac(int );
int main()
{
int x;
cin>>x;
cout<<x<<"!="<<fac(x)<<endl;    //主程序调用fac(x) 求x !
return 0;
}
int fac(int n)                          //函数fac(n) 求n !
{
      return n==0 ? 1 : n*fac(n-1); //调用函数fac(n-1)递归求(n-1) !
}
```

【说明】：

这里出现了一个小东西，三元运算符“?:”。a?b:c的含义是：如果a为真，则表达式的值是b，否则是c。所以n==0 ? 1 : n*fac(n-1)很好地表达了刚才的递归定义。

采用全程变量编写程序如下：

```
#include<iostream>
using namespace std;
int t;
int fac(int);
int main()
{
   int x;
   cin>>x;
   fac(x);
   cout<<t<<endl;
   return 0;
}
int fac(int x)
{
      if (x==1) t=1;
      else {    fac(x-1);   t*=x;    }
}
```

例6.18　用递归方法求两个数m和n的最大公约数。(m>0，n>0)

【分析】求两个数的最大公约数，可以用枚举因子的方法，从两者中较小的数枚举到能被两个数同时整除且是最大的约数的方法；也可以用辗转相除法，这里采用递归实现辗转相除算法：

①求m除以n的余数；

②如果余数不为0，则让m=n，n=余数，重复步骤①，即调用子程序；

③如果余数为0，则终止调用子程序；

④输出此时的n值。

采用有参函数编写程序如下：

```
#include<iostream>
using namespace std;
int gcd(int,int);
int main()
{
   int m,n;
   cin>>m>>n;
   cout<<"m="<<m<<" n="<<n<<"
      gcd="<<gcd(m,n)<<endl;
   return 0;
}
int gcd(int m,int n)
   {
      return m%n==0?n:gcd(n,m%n);
   }
```

采用全程变量编写程序如下：

```
#include<iostream>
using namespace std;
int d;
void gcd(int ,int );
int main()
{
      int a,b;
      cin>>a>>b;
      gcd(a,b);
      cout<<"m="<<a<<" n="<<b<<" gcd="<<d<<endl;
      return 0;
}
void gcd(int x,int y)
{   if (x%y==0) d=y;//d是用于传递结果的全局变量
          else gcd(y,x%y); //递归调用
}
```

- 例6.19　已知一个一维数组a[1..n](n<25)，又已知一整数m。 如能使数组a中任意几个元素之和等于m，则输出YES，反之则为NO。

【分析】对于一个已确定的数组a[1..n]和一个确定的数m，判断能否使数组a中任意几个元素之和等于m，等价于判断能否从数组a中取任意数使其和为m。

对于a中任意元素a[n]只有取与不取两种情况：

（1）取a[n]：

则此时问题转化为：对于一个已确定的数组a[1..n-1]和一个确定的数m-a[n]，判断能否使数组a[1..n-1]中任意几个元素之和等于m-a[n]。

（2）不取a[n]：

则此时问题转化为：对于一个已确定的数组a[1..n-1]和一个确定的数m，判断能否使数组a[1..n-1]中任意几个元素之和等于m。

若用函数sum(n,m)表示能否从数组a[1..n]中取任意数使其和为m，只要sum(n-1,m-a[n])和sum(n-1,m)当中有一个值为真，则sum(n,m)为真，否则为假。因此，可以用递归来解此题。

递归终止条件为：

```
if (a[n]==m) sum=true; else if (n==1) sum=false;
```

采用全程变量编写程序如下：

```
#include<iostream>
using namespace std;
const int max1=51;
int a[max1],n,m;
bool flag;
void sum(int ,int );
int main()
{
    cin>>n;
    for (int i=1; i<=n; ++i) cin>>a[i];
    cin>>m;
    flag=false;
    sum(n,m);
    if (flag) cout<<"YES"<<endl;
       else  cout<<"NO"<<endl;
    return 0;
}
void sum(int n,int m)
{
    if (a[n]==m) flag=true;      //利用全局变量falg传递结果
    else if (n==1) return;       //n=1作为递归边界，不再递归下去
    else                             //进行两种选择
    {
        sum(n-1,m-a[n]);
        sum(n-1,m);
    }
}
```

简单地说，递归算法的本质就是自己调用自己，用调用自己的方法去处理问题，可使解决问题变得简洁明了。

（1）递归程序在执行过程中，一般具有如下模式：

①将调用程序的返回地址、相应的调用前的变量都保存在系统堆栈中；

②执行被调用的函数；

③若满足退出递归的条件，则退出递归，并从栈顶上弹回返回地址、取回保存起来的变量值，继续沿着返回地址，向下执行程序；

④否则继续递归调用，只是递归调用的参数发生变化：增加一个量或减少一个量，重复执行直到递归调用结束。

（2）能够用递归算法解决的问题，一般满足如下要求：

①需要求解的问题可以化为子问题求解，其子问题的求解方法与原问题相同，只是规模上的增加或减少；

②递归调用的次数是有限的；必须有递归结束的条件（称为递归边界）。

课堂练习

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、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。

4、用递归算法将一个十进制数X转换成任意进制数M（M<=16）。

5、输入一串以‘！’结束的字符，按逆序输出。

上机练习

1．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)

2．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。

3．用递归的方法求Hermite多项式的值

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

- 对给定的x和正整数n，求多项式的值。

上机练习

4．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。

5．已知

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

- 计算x=4.2，n=10以及x=2.5，n=15时的f的值。
- 用递归函数求解。