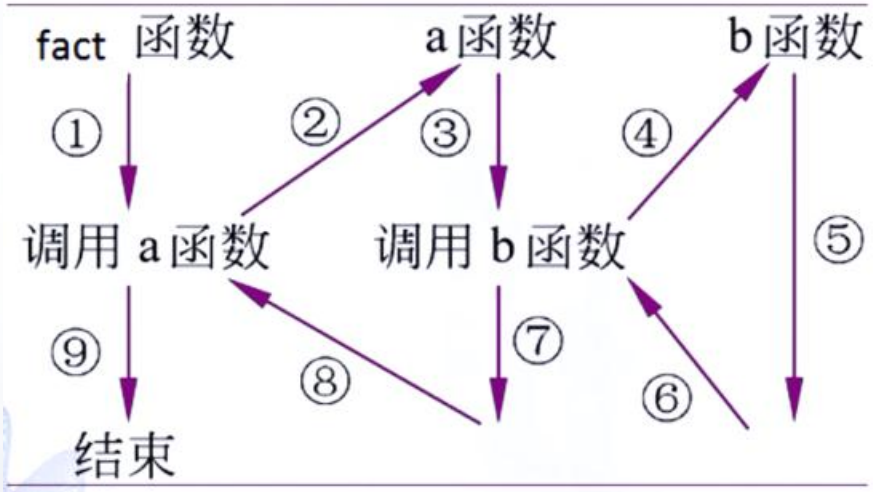
3.7　函数的递归调用

- 函数的嵌套调用
- 函数的递归调用

1．函数的嵌套调用

如果在一个函数的定义中调用了其他函数这就是函数的嵌套调用。



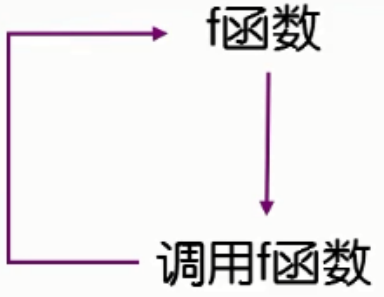
2．函数的递归调用

一个函数调用它自身称为函数的递归调用。

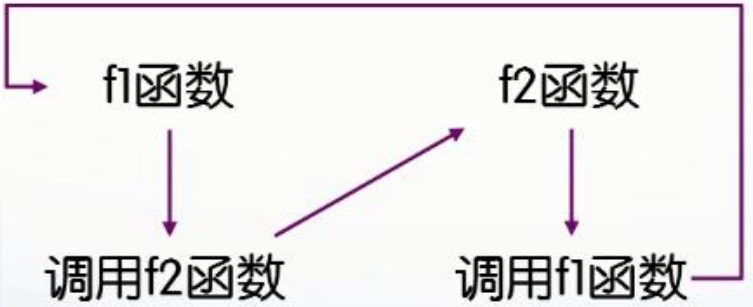
```
function f=fact(n)
…
fact(n-1)
…
```

递归　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。

（1）直接递归调用



（2）间接递归调用



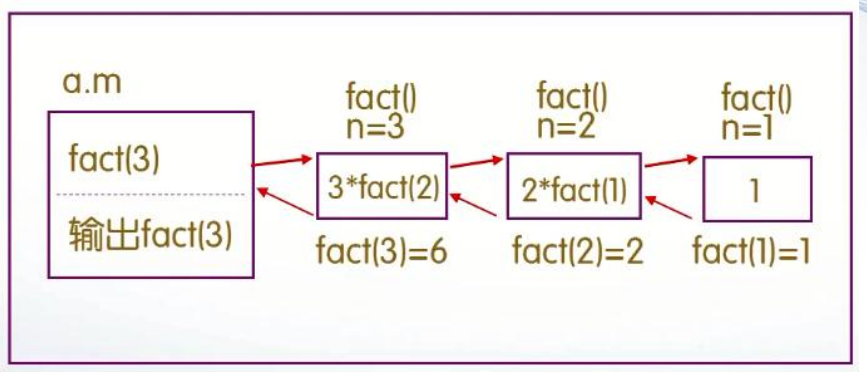
例1　利用函数的递归调用，求n！。

n!本身就是以递归的形式定义的：

$$n! = \begin{cases} 1, & n \le 1 \\ n(n-1)!, & n > 1 \end{cases}$$

函数文件fact.m如下：

```
function f=fact(n)
if n<=1
    f=1;
else
    f=fact(n-1)*n;
end
```



在脚本文件a.m中调用函数文件fact.m，求n!。

```
n=input('Please input n=');
s=fact(n);
disp(s)
```

在命令行窗口运行命令文件：

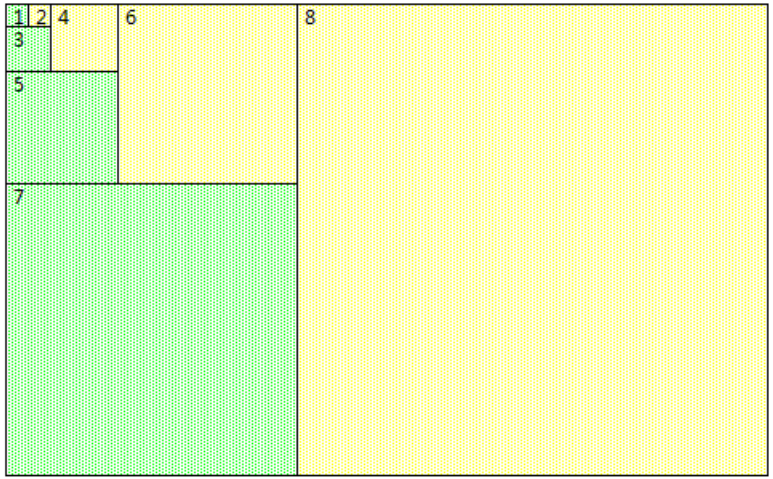
```
>> a
Please input n=3
6
```

例2　Fibonacci数列定义如下：

$$f_1=1$$
$$f_2=1$$
$$f_n=f_{n-1}+f_{n-2} \quad (n>2)$$

编写递归调用函数求Fibonacci数列的第n项，然后调用该函数验证Fibonacci数列的如下性质：

$$f_1^2+f_2^2+f_3^2+\cdots\cdots+f_n^2=f_n\times f_{n+1}$$



Fibonacci数列前八项是1，1，2，3，5，8，13，21

8个正方形的面积和=第8个正方形的边长乘以第9个正方形的边长

- 首先建立函数文件ffib.m。

```
function f=ffib(n)
if n>2
    f=ffib(n-1)+ffib(n-2);
else
    f=1;
end
```

- 建立程序文件test.m。

```
F=[];
for k=1:20
    F=[F,ffib(k)*ffib(k)];
end
sum(F)
ffib(20)*ffib(21)
```

- 运行结果为：

```
>> test
ans =
    74049690
ans =
    74049690
```