3.1 顺序结构程序

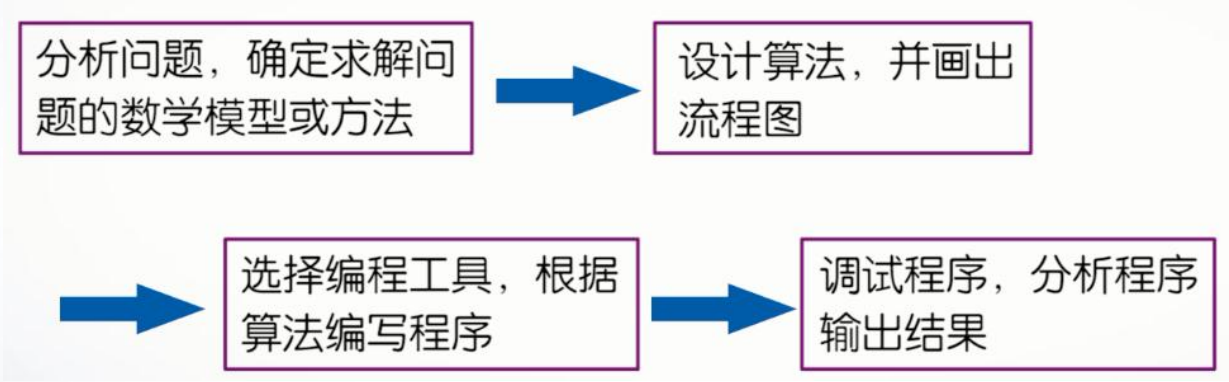
- 程序和程序设计
- 程序的三种基本结构
- 脚本文件和函数文件
- 文件的建立
- 顺序结构

1．程序和程序设计

什么叫程序？

程序是用某种计算机能够理解并且能够执行的语言来描述的解决问题的方法和步骤。

程序设计的基本步骤


分析问题，确定求解问题的数学模型或方法
设计算法，并画出流程图
选择编程工具，根据算法编写程序
调试程序，分析程序输出结果

2．程序的三种基本结构

- 顺序结构
- 选择结构
- 循环结构

3．脚本文件和函数文件

- 脚本文件是可在命令行窗口直接执行的文件，也叫命令文件。
- 函数文件是定义一个函数，不能直接执行，而必须以函数调用的方式来调用它。

4．文件的建立

- 用命令按钮创建文件。

- 用edit命令创建文件。

```
>> edit test
```

分别建立脚本文件和函数文件，求两个矩阵的乘积。

- 建立脚本文件f1.m。

```
A=[1,2,3;4,5,6];
B=[1,2;3,4;5,6];
C=A*B
```

- 在命令行窗口运行脚本文件。

```
>> f1
C =
    22    28
    49    64
```

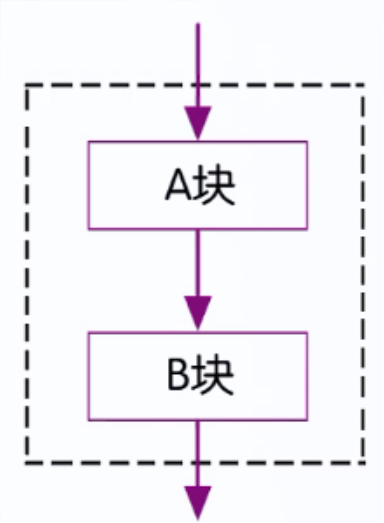
- 建立函数文件f2.m。

```
function C=f2(A,B)
C=A*B;
```

- 在命令行窗口调用函数文件。

```
>> A=[1,2,3;4,5,6];
>> B=[1,2;3,4;5,6];
>> C=f2(A,B)
C =
    22    28
    49    64
```

5．顺序结构



（1）数据的输入

A=input(提示信息,选项);

```
>> A=input('请输入变量A的值:');
请输入变量A的值:100
```

（2）数据的输出

disp(输出项);

```
>> s='Hello,World';
>> disp(s)
Hello,World
>> a=[1,2,3;4,5,6];
>> disp(a)
     1     2     3
     4     5     6
```

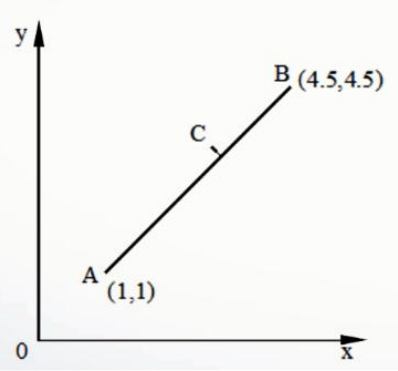
（3）程序的暂停

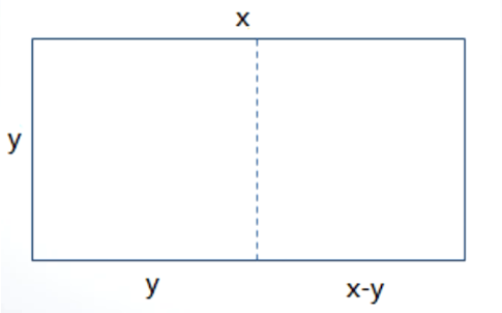
pause(延迟秒数)

若要强行中止程序的运行可使用Ctrl+C命令。

- 输入原始数据
- 对原始数据进行处理
- 输出处理结果

有一线段AB，A的坐标为(1，1)，B的坐标为(4.5，4.5)，求AB的长度，以及黄金分割点C的坐标。





A(1,1) → 1+i

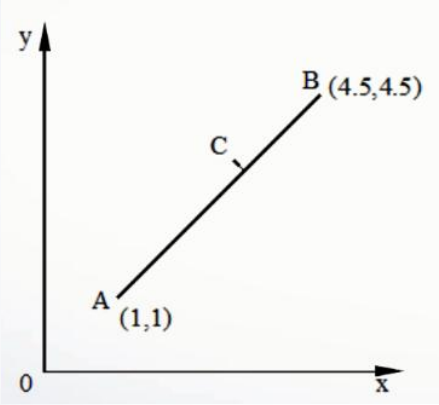
B(4.5,4.5) → 4.5+4.5i

$$\frac{x}{y}=\frac{y}{x-y} \rightarrow x^2-xy-y^2=0$$

解得 $x=\frac{y\pm y\sqrt{5}}{2} \rightarrow \frac{x}{y}=\frac{1\pm\sqrt{5}}{2}$

取 $\frac{x}{y}=\frac{1+\sqrt{5}}{2} \rightarrow y=\frac{\sqrt{5}-1}{2}x$

即 $y=0.618x$




```
a=input('a=');
b=input('b=');
c=a+0.618*(b-a);
s=abs(a-b);
disp(s)
disp(c)
```

输出结果为：

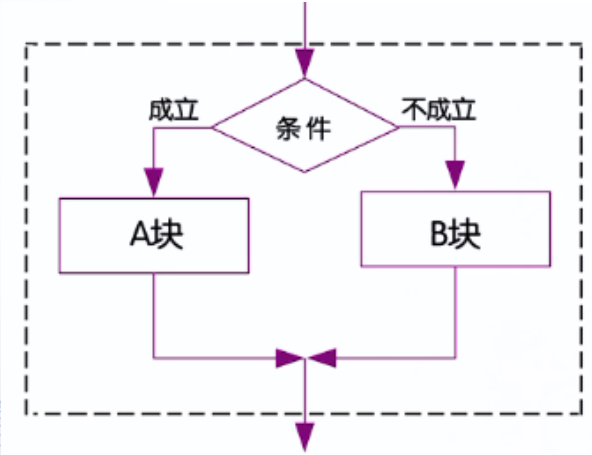
```
a=1+i
b=4.5+4.5i
    4.9497
  3.1630 + 3.1630i
```

3.2 用if语句实现选择结构

- 什么是选择结构
- 单分支if语句
- 双分支if语句
- 多分支if语句

1．什么是选择结构?

选择结构又称为分支结构，是根据给定的条件是否成立来决定程序的执行流程。



- 用if语句实现选择结构
- 用switch语句实现选择结构。

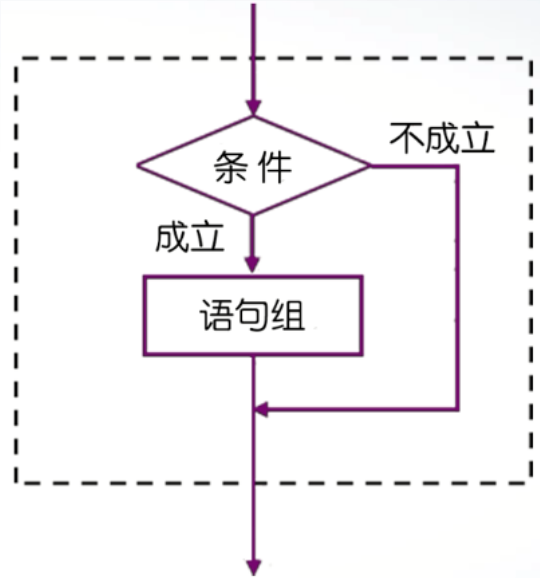
2．单分支if语句

语句格式：

```
if   条件
     语句组
end
```

关系运算或逻辑运算

可以是一条语句，也可以是多条语句



- 当条件结果为标量时，非零表示条件成立，零表示条件不成立。

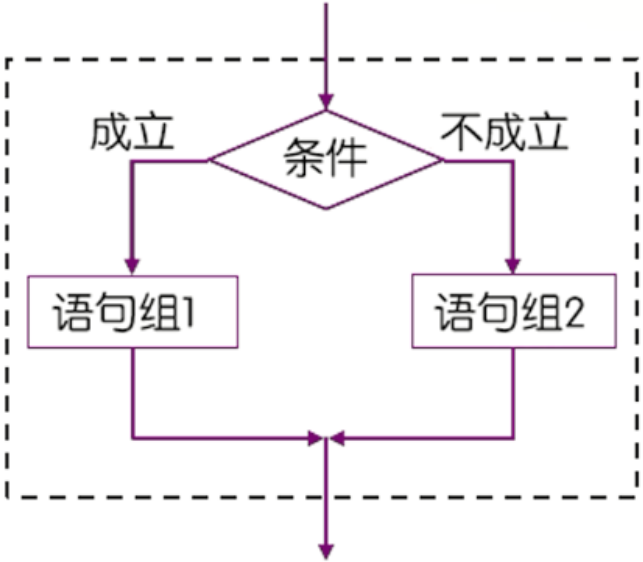
- 当条件结果为矩阵时，如果矩阵为非空，且不包含零元素，则条件成立，否则不成立。

例如，[1,2;0,4]表示条件时，条件不成立；[1,2;3,4]表示条件时，条件成立。

3．双分支if语句

语句格式：

```
if   条件
     语句组1
else
     语句组2
end
```



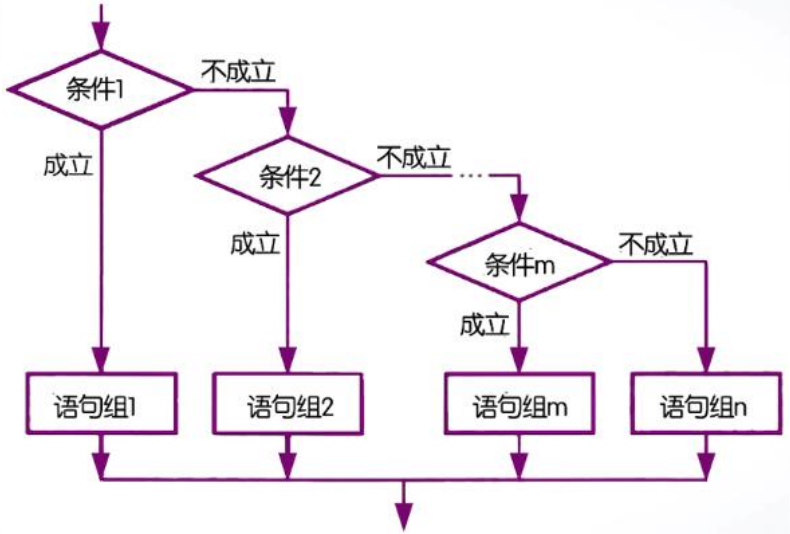
例1　输入一个整数，若为奇数则输出其平方根，否则输出其立方根。

```
x=input('请输入x的值:');
if rem(x,2)==1
    y=sqrt(x);
else
    y=x^(1/3);
end
y
```

4．多分支if语句

语句格式：

```
if   条件1
     语句组1
elseif   条件2
     语句组2
     …
elseif   条件m
     语句组m
else
     语句组n
end
```



例2　输入一个字符，若为大写字母，则输出其对应的小写字母；若为小写字母，则输出其对应的大写字母；若为数字字符则输出其对应数的平方，若为其他字符则原样输出。

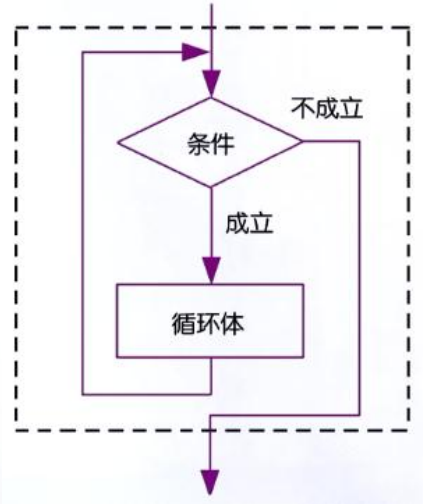
```
c=input('请输入一个字符：','s');
if c>='A' && c<='Z'
    disp(lower(c))
elseif c>='a' && c<='z'
    disp(upper(c))
elseif c>='0' && c<='9'
    disp(str2double(c)^2)
else
    disp(c)
end
```

3.4 用for语句实现循环结构

- 什么是循环结构
- for语句

1．什么是循环结构?

循环结构又称为重复结构，是利用计算机运算速度快以及能进行逻辑控制的特点来重复执行某些操作。



2．for语句

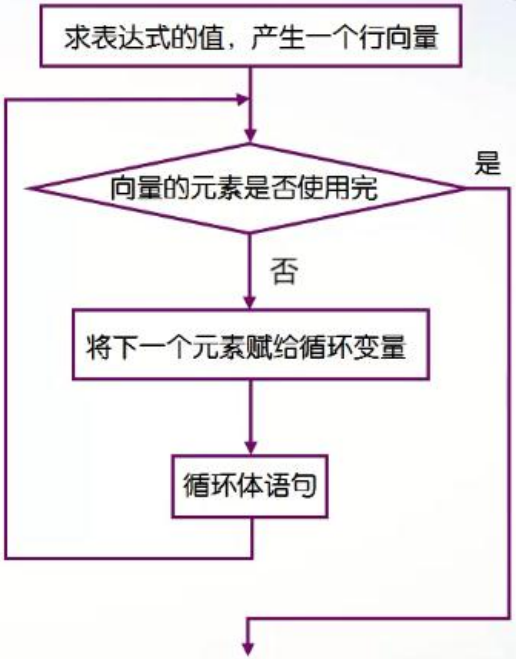
格式：

初值　　终值

```
for    循环变量=表达式1:表达式2:表达式3
       循环体语句
end
```

步长

重复执行的语句



说明：

- for语句针对向量的每一个元素执行一次循环体。

```
for k=[1,3,2,5]
    k
end
```

循环四次

- 退出循环之后，循环变量的值就是向量中最后的元素值。

```
for k=1:2:10

end

k
```

1 3 5 7 9

- 当向量为空时，循环体一次也不执行。

```
for k=1:-2:10
    k
end
```

计算圆周率π

（1）利用无穷级数展开式求π 的近似值。

$$\frac{\pi}{4}=1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots+(-1)^{n+1}\frac{1}{2n-1}$$

这是求n个数之和的累加问题，可用以下递推式来描述：

$$y_i=y_{i-1}+f_i \quad (y_0=0)$$

可用以下赋值语句来实现。

```
y=y+f
```

其中累加项f的符号可用以下赋值语句来实现（每循环一次反号一次）。

```
g=-g
```

累加项f就可用以下赋值语句来实现。

```
f=g/(2*i-1)
```

```
y=0;
g=-1;
n=input('n=?');
for i=1:n
    g=-g;
    y=y+g/(2*i-1);
end
pai=4*y
```

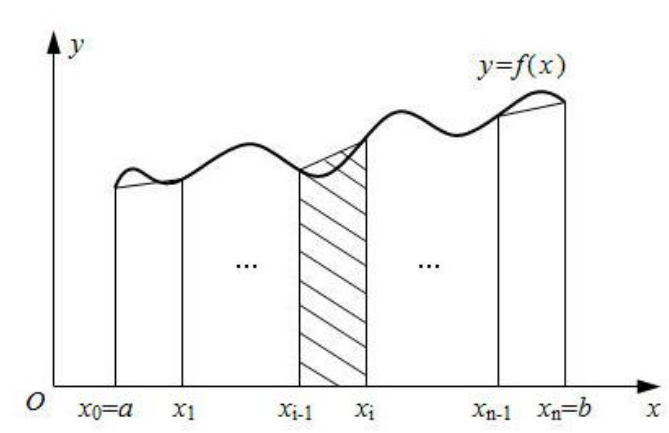
用向量求和的方法实现程序：

```
n=input('n=?');
x=1:2:(2*n-1);
y=(-1).^(2:n+1)./x;
pai=sum(y)*4
```

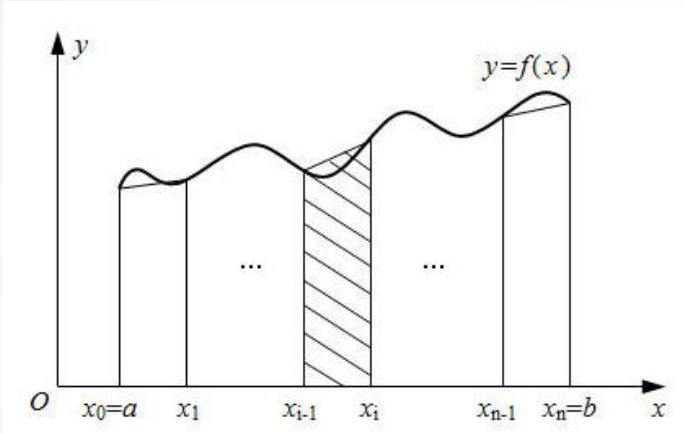
用向量计算方法写出来的程序更加简洁，也更加具有MATLAB的特点。

（2）利用定积分的近似值求π 的近似值。

设$f(x)=\sqrt{1-x^2}dx$，求$\frac{\pi}{4}=\int_0^1 f(x)\,dx$，即求四分一单位圆的面积。

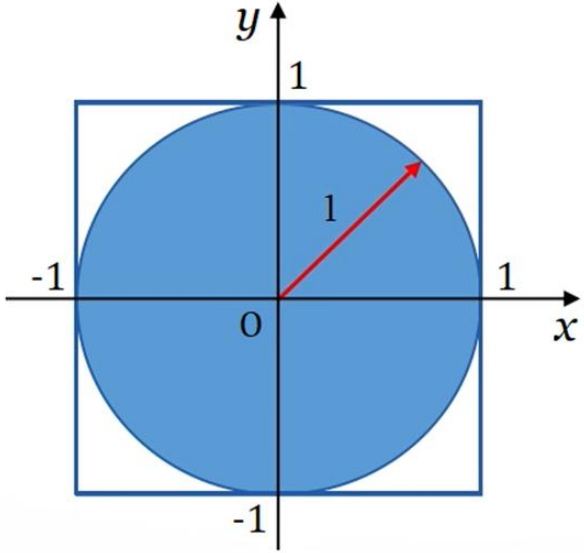


求函数f(x)在[a，b]上的定积分，就是求曲线y=f(x)与直线x=a，x=b，y=0所围成的曲边梯形的面积。近似求出每个小曲边梯形面积，然后将n个小曲边梯形的面积加起来，就得到总面积，也就是定积分的近似值。




```
a=0;
b=1;
n=input('n=?');
h=(b-a)/n;
x=a:h:b;
f=sqrt(1-x.*x);
s=[];
for k=1:n
    s1=(f(k)+f(k+1))*h/2;
    s=[s,s1];
end
pai=4*sum(s)
```

（3）利用蒙特卡洛法求π 的近似值。



在正方形内随机投点，设点落在圆内的概率为P。

$P=\pi/4 \Longrightarrow \pi=4P$

P=落在圆内的点数/所投点的总数

所投的点落在圆内的充要条件是$x^2+y^2\leqslant 1$。

```
s=0;
n=input('n=?');
for i=1:n
    x=rand(1);
    y=rand(1);
    if x*x+y*y<=1
        s=s+1;
    end
end
pai=s/n*4
```

for语句更一般的格式为：

```
for 循环变量=矩阵表达式
    循环体语句
end
```

执行过程是依次将矩阵的各列元素赋给循环变量，然后执行循环体语句，直到各列元素处理完毕循环结束。

下面两个for语句引导的循环结构，其循环体执行的次数相同吗？如果不相同，分别是多少？

```
for k=[1,2,3,4]
for k=[1;2;3;4]
```

执行4次

执行1次

3.5 用while语句实现循环结构

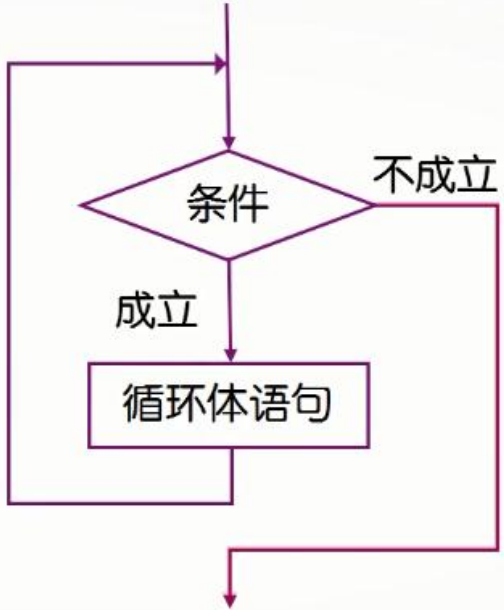
- while语句
- break语句和continue语句
- 循环的嵌套

1．while语句

循环判断条件

格式：

```
while 条件
    循环体语句
end
```



例1　从键盘输入若干个数，当输入0时结束输入，求这些数的平均值和它们之和。

```
msum=0;
n=0;
x=input('Enter a number (end in 0):');
while x~=0
    msum=msum+x;
    n=n+1;
    x=input('Enter a number (end in 0):');
end
if n>0
    msum
    mean=msum/n
end
```

- while语句多用于循环次数不确定的情况，而对于循环次数确定的情况，使用for语句更方便。
- 针对不同情况可以选择不同的循环语句，但从功能上讲两种循环语句可以相互替代。

2．break语句和continue语句

- break语句用来跳出循环体，结束整个循环。
- continue语句用来结束本次循环,接着进行下一次是否执行循环的判断。

例2　求[100，200]之间第一个能被21整除的整数。

```
for n=100:200
    if rem(n,21)~=0
        continue
    end
    n
    break
end
```

3．循环的嵌套

如果一个循环结构的循环体又包括一个循环结构，就称为循环的嵌套，或称为多重循环结构。处于内部的循环叫做内循环，处于外部的循环叫做外循环。

例3　用筛选法求某自然数范围内的全部素数。

筛选法求素数的基本思想：要找出2~m之间的全部素数，首先在2~m中划去2的倍数（不包括2），然后划去3的倍数（不包括3），由于4已被划去，再找5的倍数（不包括5），…，直到再划去不超过$\sqrt{m}$的倍数，剩下的数就都是素数了。

```
m=input('m=');
p=1:m;
p(1)=0;
for i=2:sqrt(m)
    for j=2*i:i:m
        p(j)=0;
    end
end
n=find(p~=0);
p(n)
```

3.6 函数文件的定义与调用

- 函数文件的基本结构
- 函数调用
- 匿名函数

1. 函数文件的基本结构

function 输出形参表=函数名(输入形参表)

注释说明部分

函数体语句

当输出形参多于一个时，应该用方括号括起来，构成一个输出矩阵。

- 函数文件名通常由函数名再加上扩展名.m组成，函数文件名与函数名也可以不相同。当函数文件名与函数名不相同时，MATLAB将忽略函数名，调用时使用函数文件名。
- return语句表示结束函数的执行。通常，在函数文件中也可以不使用return语句，那么被调用函数执行完成后会自动返回。

例1　编写函数文件，求半径为r的圆的面积和周长。

```
function [s,p]=fcircle(r)
s=pi*r*r;
p=2*pi*r;
```

2. 函数调用

调用格式：

[输出实参表]=函数名(输入实参表)

在MATLAB命令行窗口调用前面定义的fcircle函数。

```
>> [s,p]=fcircle(10)
s =
   314.1593
p =
    62.8319
```

3. 匿名函数

基本格式：

函数句柄变量=@(匿名函数输入参数) 匿名函数表达式

函数句柄的运算符

```
>> f=@(x,y) x^2+y^2
f =
    @(x,y)x^2+y^2
>>  f(3,4)
ans =
    25
```

函数句柄变量=@函数名

内部函数或
自定义函数

```
>> h=@sin
h =
    @sin

>> h(pi/2)
ans =
     1
```

例2　已知$y=\frac{f(40)}{f(30)+f(20)}$

①当$f(n)=n+10\ln(n^2+5)$时，y的值是多少。

②当$f(n)=1\times2+2\times3+3\times4+\cdots+n\times(n+1)$时，$y$的值是多少。

分别用匿名函数和函数文件定义函数$f(n)$。

第②问的函数文件f2.m。

```
function f=f2(n)
f=0;
for k=1:n
    f=f+k*(k+1);
end
```

脚本文件mf.m。

```
f1=@(n) n+10*log(n*n+5);
y1=f1(40)/(f1(30)+f1(20))
y2=f2(40)/(f2(30)+f2(20))
```

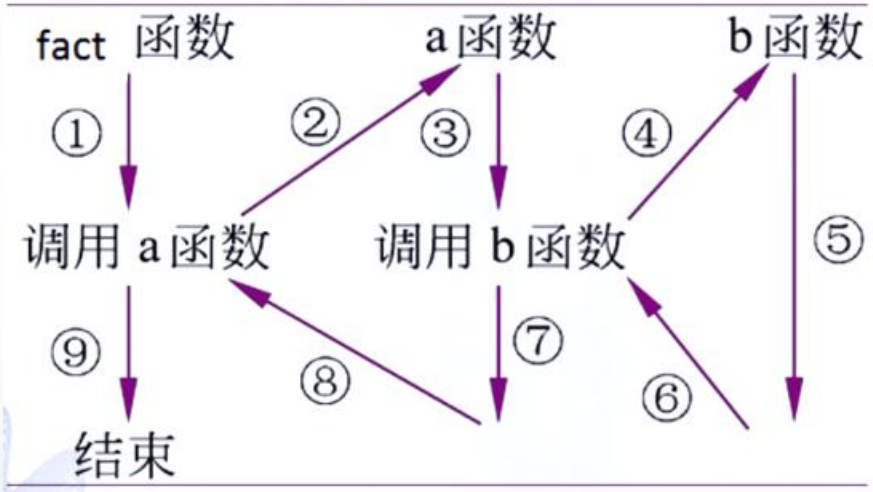
```
>> mf
y1 =
    0.6390
y2 =
    1.7662
```

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。



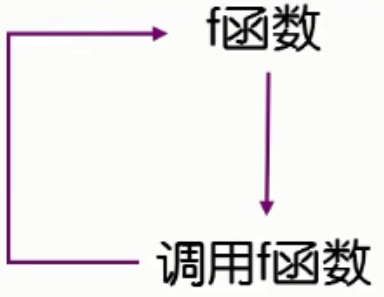
2．函数的递归调用

一个函数调用它自身称为函数的递归调用。

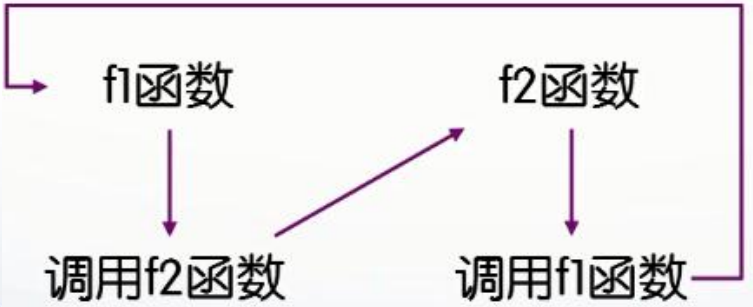
```
function f=fact(n)
…
fact(n-1)
…
```

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



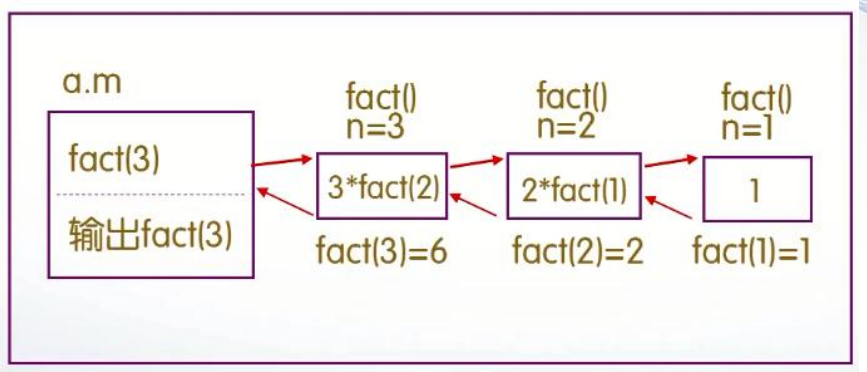
例1　利用函数的递归调用，求n！。

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：

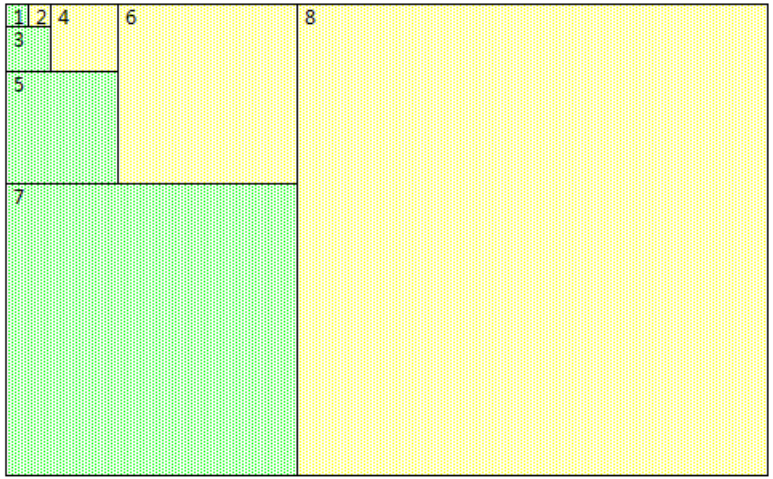
```
>> a
Please input n=3
6
```

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
```

3.8 函数参数与变量的作用域

- 函数参数的可调性
- 全局变量与局部变量

1．函数参数的可调性

nargin ⟶ 输入实参的个数

nargout ⟶ 输出实参的个数

建立函数文件test.m。

```
function fout=test(a,b,c)
if nargin==1
    fout=a;
elseif nargin==2
    fout=a+b;
elseif nargin==3
    fout=(a*b*c)/2;
end
```

```
>> fout=test(2)
fout =
     2
>> fout=test(2,3,2)
fout =
     6
```

2．全局变量与局部变量

- 局部变量：在程序中只在特定过程或函数中可以访问的变量。
- 全局变量 ：所有的函数都可以对它进行存取和修改。

全局变量定义格式：

global　变量名

建立函数文件wad.m。

```
function f=wad(x,y)
global ALPHA BETA
f=ALPHA*x+BETA*y;
```

在命令行窗口中输入命令并得到输出结果。

```
>> global ALPHA BETA
>> ALPHA=1;
>> BETA=2;
>> s=wad(1,2)
s =
     5
```