

## 第三、四讲 程序设计

- ➔ 变量和运算符
- ➔ 流程控制
- ➔ 脚本文件和函数文件
- ➔ 主函数与子函数
- ➔ 递归思想

# 用赋值语句创建变量

变量名 = 数据

变量名 = 表达式

表达式由变量、运算符、函数、数字组成。

## 变量命名规则：

- (1) 对大小写敏感；
- (2) 第一个字符必须为英文字母，其长度不能超过31个字符；
- (3) 可以包含下划线、数字，但不能包含空格，汉字等。

## 1. 算术运算符

+	加
-	减
.	乘
.*	点乘
/	右除
./	点右除
\	左除
.\	点左除
^	次幂
.^	点次幂


## 2. 关系运算符

<	小于
<=	小于等于
>	大于
>=	大于等于
==	等于
~=	不等于

## 3. 逻辑运算符

&	与
	或
~	非

## 运算优先级

- 
- ①函数运算 `exp()`、`log()`、`sin()`、`abs()`、`fix()`、...
  - ②算术运算 `.`<sup>^</sup>、<sup>^</sup>、`.`<sup>\*</sup>、`.`<sup>/</sup>、<sup>\*</sup>、<sup>/</sup>、`+`、`-`、`;`
  - ③关系运算 `<`、`<=`、`>`、`>=`、`==`、`~=`
  - ④逻辑运算 `&`、`|`

表达式 `( 1+fix(pi) ) * mod(2, 4) + 2*3^2`结果:

`ans = 26`

## 程序控制:

条件控制; 循环控制; 错误控制; 终止运行控制

### 1、条件控制 if / else

例 计算函数值  $f(x) = \begin{cases} x^2, & x > 5; \\ x + 5, & 1 \leq x \leq 5; \\ -x, & x < 1. \end{cases}$

```
x = 2
if x >= 5
    y = x^2
elseif x >= 1 & x <= 5
    y = x+5
else
    y = -x
end
```

## 2. for 循环

```
for 变量 = 初值 : 步长 : 终值  
    可执行语句  
end
```

### 例 斐波拉奇数列

```
n = input(' pls input number: ');  
f = [1 1];  
for k = 3:n  
    f(k) = f(k-1) + f(k-2);  
end  
f(n)
```

### 3. while循环

```
while  条件表达式  
      可执行语句  
end
```

条件表达式一般由变量、数字、逻辑运算、关系运算符和一般运算符组成，以判断循环的进行和停止；

只要表达式的值(逻辑值)结果为正确(非0)，循环继续；直到表达式值为0，循环停止。

## 例 用户登陆

```
n = input('please input your PIN: ','s');  
while ~strcmp(n,'1234')  
    clc  
    n = input('Error! input PIN again: ','s');  
end  
disp('welcome!')
```



## 4. **continue** 命令

通常用于 **for** 或 **while** 循环语句中,与 **if** 语句一起使用, 跳过本次循环, 去执行下一轮循环。

## 5. **break**命令

通常用于**for**或**while**循环语句中, 与**if**语句一起使用, 中止本次循环, 跳出最内层循环。

## 6. **error('message')**

显示文本 **message**, 并中断程序执行。

**M文件**分为**命令文件(脚本文件)**和**函数文件**两种。

**命令文件**是MATLAB命令的有序集合。本质上是对文件中命令进行批处理，即从第一条命令开始按顺序执行，直到最后一条命令。如果中间某条命令出错，则中断并输出错误信息。

## 函数文件

**function** 返回变量=**函数名**(输入变量)  
**函数体**

从功能上分为**主函数**，**子函数**。

函数文件的第一行必须按特定格式书写**function**，  
允许无输入或无输出变量。

函数内所有变量是局部变量，既不影响其他M文件中同名变量，也不被其他M文件中同名变量所影响。  
函数文件中的输出变量要等于某个确定的表达式。

输入/输出变量检测命令：**nargin**、**nargout**

**nargin**      检测输入变量的个数。

**nargout**    检测输出变量的个数。

## 例 杨辉三角形

```
function Y = young(n)
if nargin == 0
    n = 3;
end
Y = eye(n);
Y(:, 1) = ones(n, 1);
for k = 3:n
    Y(k, 2:k-1) = Y(k-1, 1:k-2) + Y(k-1, 2:k-1);
end
```

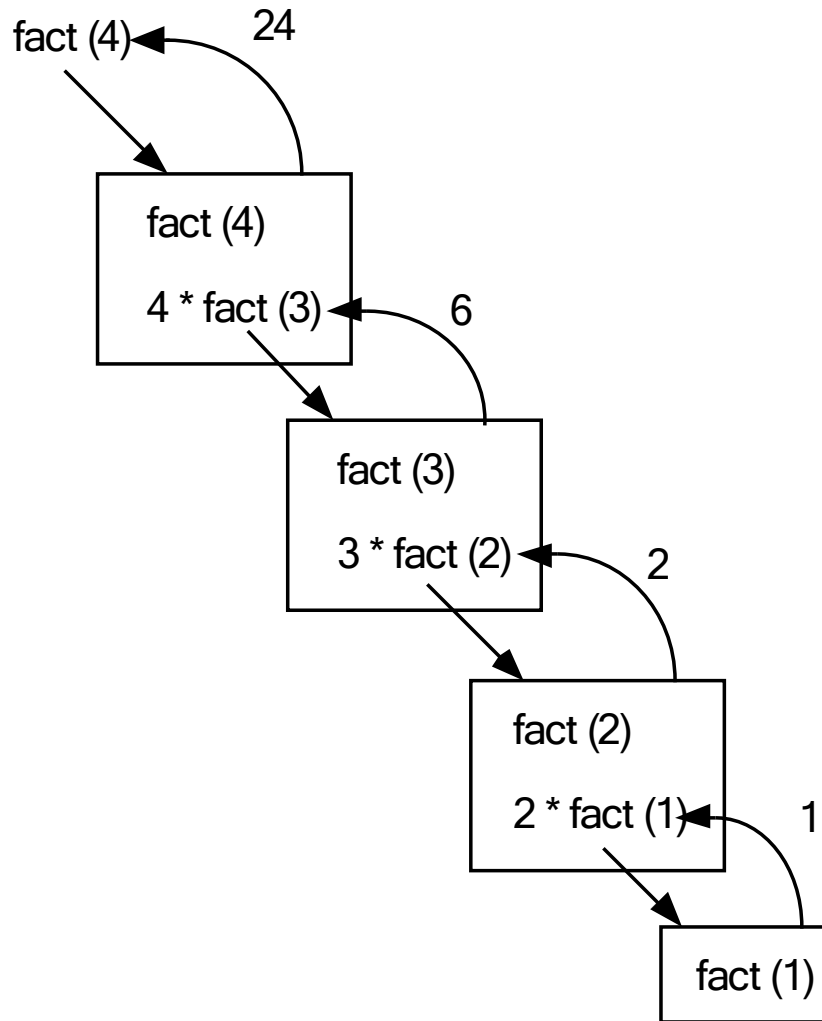
# 递归思想

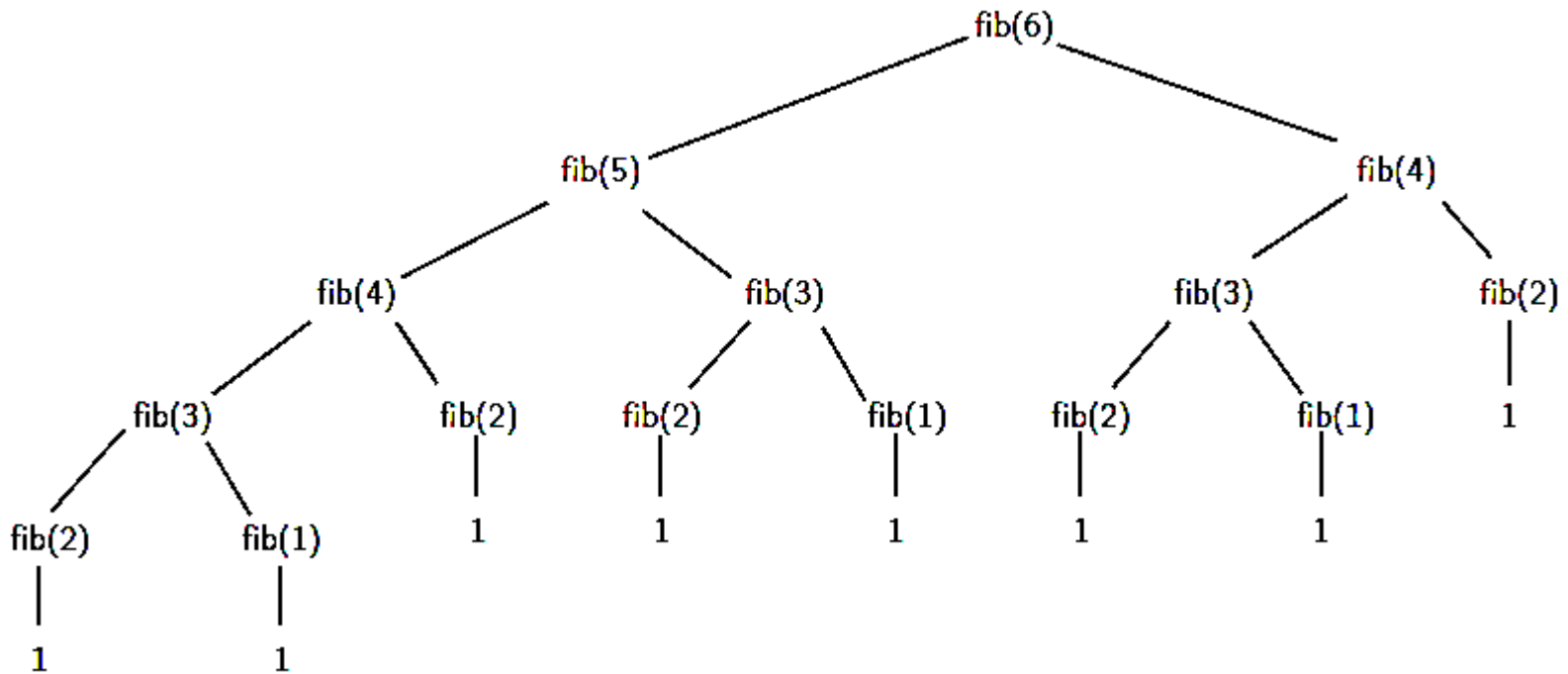


## Factorials:

$$4! = 4 \cdot 3 \cdot 2 \cdot 1$$

$$1! = 1$$





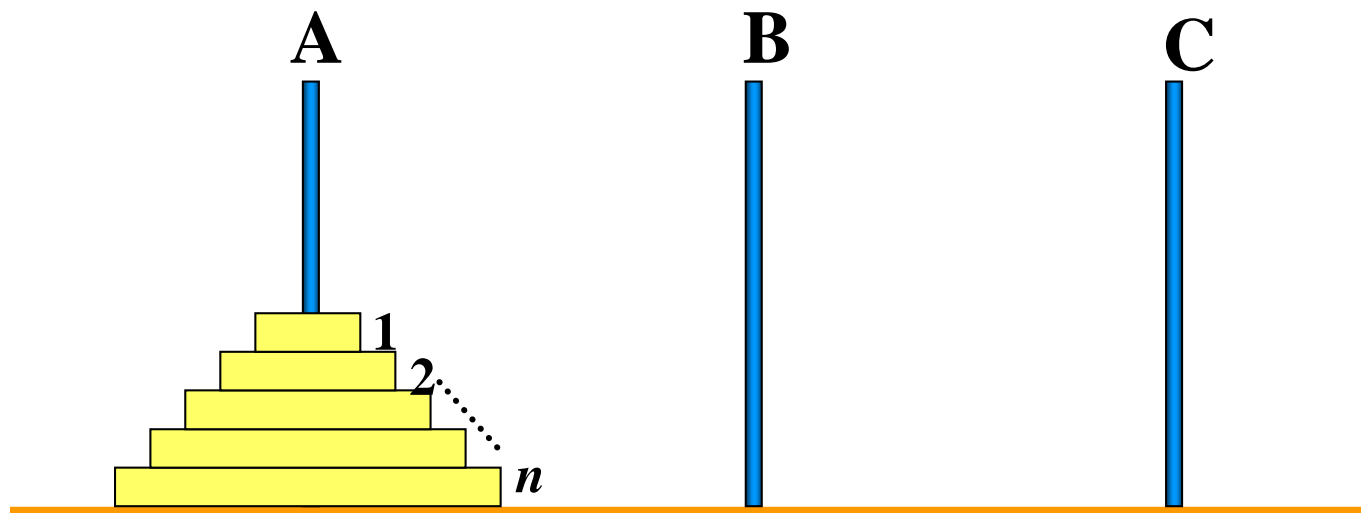
## Fibonacci sequence:

**n=6: 1 1 2 3 5 8**

**fib(1)=fib(2)=1**

## 例 Hanoi问题

有 $A$ 、 $B$ 、 $C$ 三个塔柱。柱 $A$ 上 $n$ 个有孔圆盘，由上而下由小到大叠放。要将柱 $A$ 上圆盘移到柱 $C$ 上，并仍按同样顺序叠放。移动圆盘过程中，不允许大圆盘压小圆盘，可将圆盘移至 $A$ ， $B$ ， $C$ 中任何一柱上。





**问题分析：n张盘片，A、B、C三根塔柱**

**将A做为开始塔柱，C为目标塔柱，B为中间塔柱。**

- ① 将A上的n-1个盘转移到B上**
- ② 将A上第n号盘转移到C上**
- ③ 将B上的n-1个盘转移到C上。**

**第一步是n-1个盘问题(A开始，C中间，B目标)；**

**第二步是1个盘问题；**

**第三步是n-1个盘问题(B开始，A中间，C目标)。**

**将三步操作按次序编写函数文件，第一步操作和第三步操作需要调用函数本身，即自己调用自己。**

```
function hanoi(n,A,B,C)
    %n--圆盘数,A--开始,B--中间,C--目标
if nargin==1, A=' A';B=' B';C=' C';end
if n==1
    disp(strcat('No',int2str(n),':',A,' -> ',C))
else
    hanoi(n-1,A,C,B);
    disp( strcat('No',int2str(n),':',A,' -> ',C))
    hanoi(n-1,B,A,C);
end
```

递归技术实现的关键是设置边界条件（即一个盘的情况）。程序运行结果表明，三个盘的汉诺塔问题需要七步操作。

**hanoi(3)**

- No1: A -> C**
- No2: A -> B**
- No1: C -> B**
- No3: A -> C**
- No1: B -> A**
- No2: B -> C**
- No1: A -> C**

**hanoi(4)**

- No1: A -> B**
- No2: A -> C**
- No1: B -> C**
- No3: A -> B**
- No1: C -> A**
- No2: C -> B**

- No1: A -> B**
- No4: A -> C**
- No1: B -> C**
- No2: B -> A**
- No1: C -> A**
- No3: B -> C**
- No1: A -> B**
- No2: A -> C**
- No1: B -> C**