第三章 MATLAB程序设计

3.1 M文件建立

3.2 函数文件

3.3 程序控制结构

3.4 其它控制语句

3.5 程序调试

3.1 M文件建立

1. M 文件

① Matlab 语言编写的程序；

② Matlab 命令组合，完成某些操作和算法；

③ 文本文件，扩展名为.m ；

④ 文本编辑器：

- M文件编辑器
- windows的记事本
- word文件

2.M 文件建立

(1) M 文件编辑器

- 新建M 文件

  - 菜单操作:

    File → New →

    1.blank m-file 表格或空白文件;

    2.function m-file 函数文件;

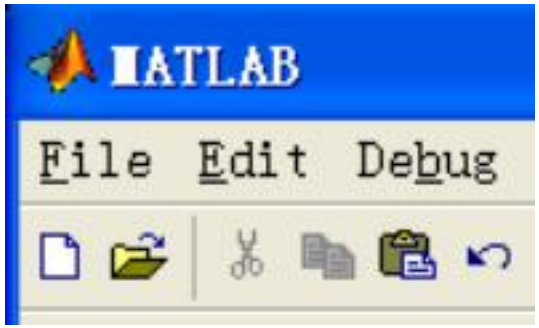
    3.class m-file 类文件。

  - 命令操作：edit

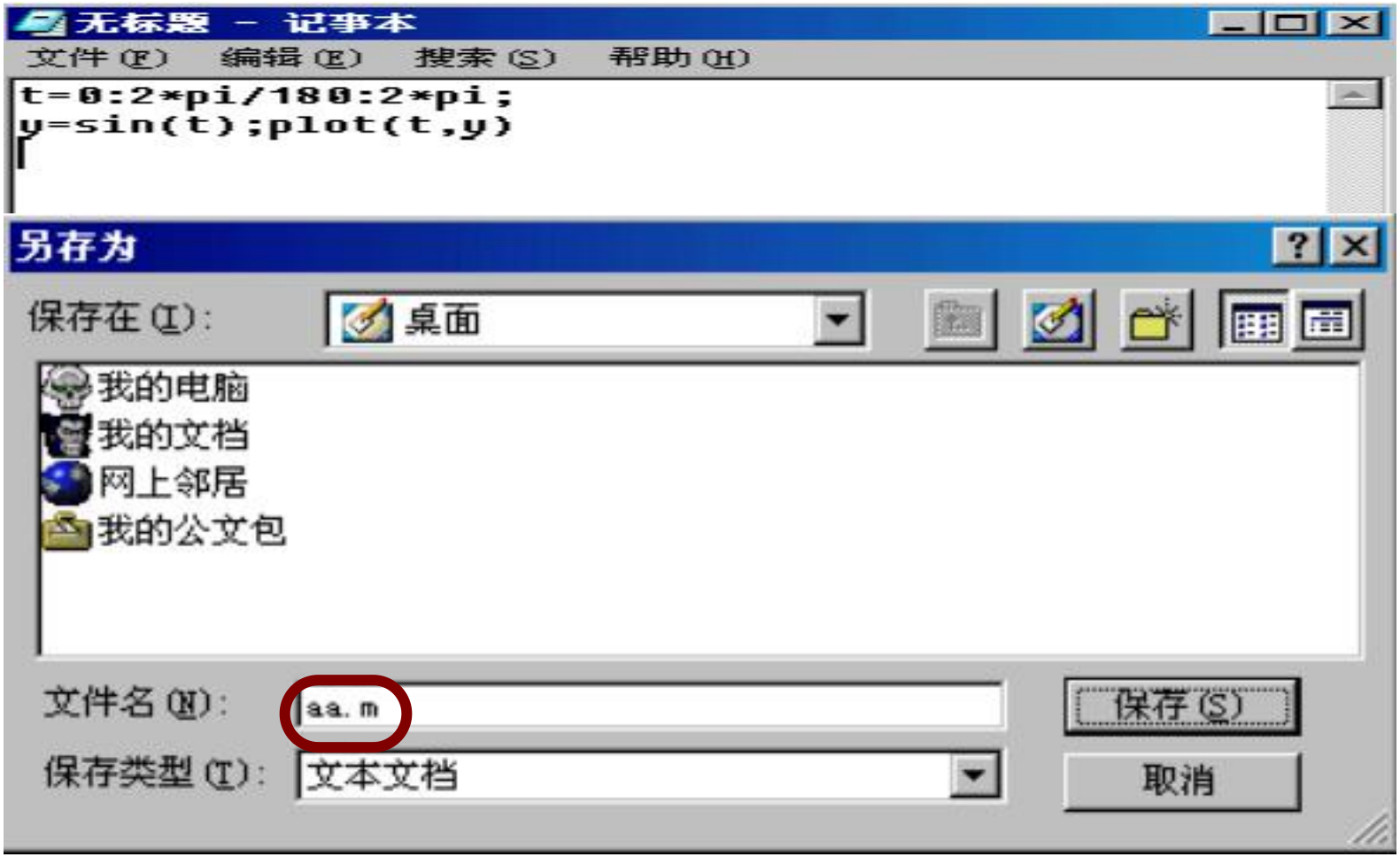
  - 命令按钮：快捷键

打开已有的 M 文件

- 菜单操作：**File → Open**
- 命令操作：**edit M 文件名**
- 命令按钮：**打开快捷键**
- 双击 M 文件



(2). windows记事本和word



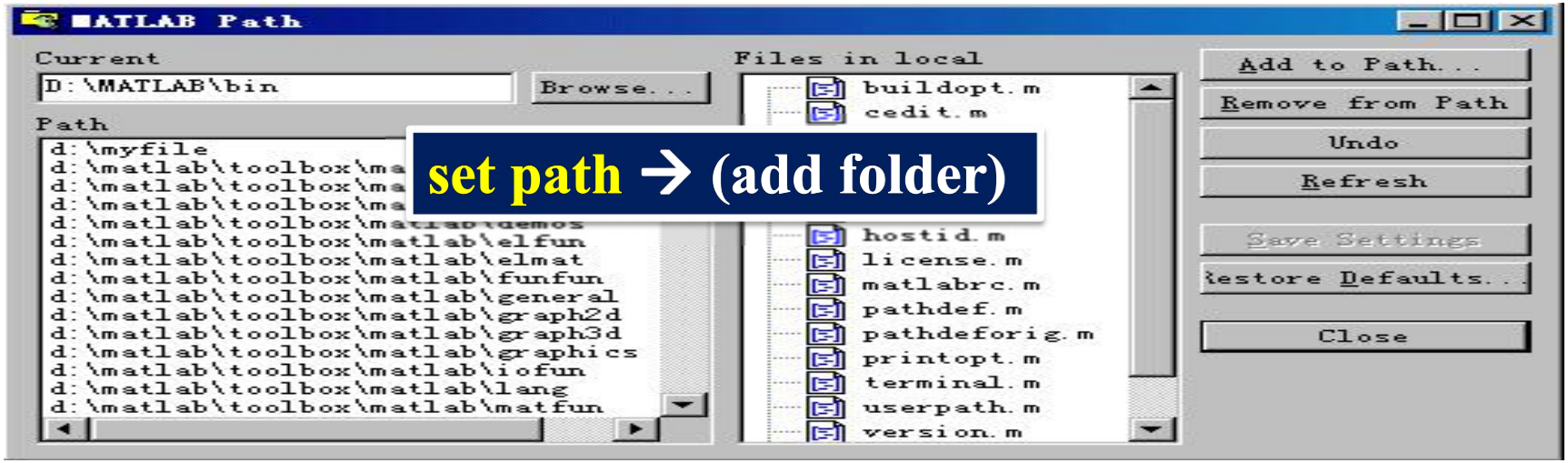
2 . matlab搜索路径的设置

显示路径： **which filename**

设定当前目录： **cd d:\myfile**

设定搜索路径： **>> addpath(' folder path')**

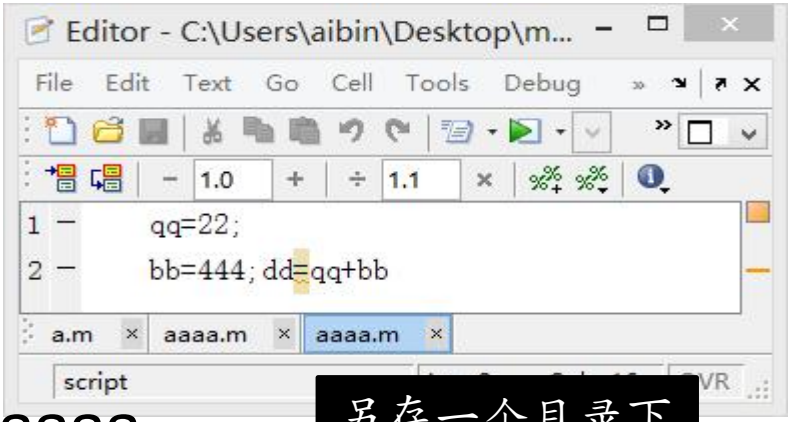
创建的m文件必须纳入matlab搜索路径或当前目录，才能在命令窗口运行



3. 显示M文件内容

调用格式： **type M文件**

>> edit



>> which aaaa

aaaa not found.

>> cd C:\Users\Desktop\matlab

>> **addpath('C:\Users\Desktop\matlab')**

>> edit aaaa

>>type aaaa

```
>> type aaaa.m
qq=22;
bb=444;
dd=qq+bb;
```

3.2 函数文件

3.2.1 脚本文件

3.2.2 函数文件

3.2.3 全局变量和局部变量

3.2.4 子函数与主函数

3.2.5 函数句柄和匿名函数

3.2.1 脚本文件

1.不同调用方式两类

① **Script**：脚本文件/命令文件

② **Function**：函数文件

例题 编脚本文件求半径为r的圆的面积和周长

```
%fcircle  calculate the area and perimeter of a circle of radii r
% r       圆半径
% s       圆面积
% p       圆周长
% 2004年7月30日 编
r=22;
s=pi*r*r;
p=2*pi*r;
```

H1注释行

帮助文本区

函数体

保存rsp

调用脚本文件

>>filename

>> rsp

```
>> whos
  Name      Size            Bytes  Class     Attributes
  p         1x1                 8  double
  r         1x1                 8  double
  s         1x1                 8  double
```

- 运行后所有变量都驻留在基本工作空间（base workspace）
- 文件变量名不要和计算机命令、函数、文件名相同。

2. 判断文件是否存在

Syntax

exist name

exist name kind

A = exist('name','kind')

name返回值

0 : 不存在

1: 可以是变量名

2: 函数名、m 文件名 **3:** mex 文件、dll 文件

4: 内嵌的函数

5: p码文件

6: 目录

7: 路径

8: Java class

kind :

builtin 内嵌函数
class Java class
dir 目录
file 文件或者目录
var 变量

例

```
>> exist ones
ans =     5
>> exist pascal
ans =     2
>> aa=2
aa =     2
>> exist aa
ans =     1
>> exist aab
ans =     0
```

3. 脚本文件—

① 一串命令行的简单叠加的集合；

② 自动按顺序执行文件的命令；

③ 无输入和输出

④ 所有变量都驻留在基本工作空间中；

⑤ 所有变量均为**全局变量**。

4. 数据输入

输入数值

```
A=input(‘提示信息’,选项)
```

输入字符串变量

```
A=input(‘提示信息’, 's')
```

- 从键盘输入数据给变量A，**提示信息为字符串。**

```
A=input('Please input A: ')
```

Please input A: 33

```
name=input('What''s your name? ', 's')
```

name =lili

例题

```
clear;
r=input('Please input radii :');
s=pi*r*r;
p=2*pi*r;
```

```
>> ddd
Please input radii :33
s =  3.4212e+003
p =  207.3451
```

3.2.2 函数文件

例题 编函数文件求半径为r的圆的面积和周长

```
function [s,p]=ff(r)
% FCIRCLE  calculate the area and perimeter of a circle of radii r
% r        圆半径
% s        圆面积
% p        圆周长
% 2004年7月30日 编
s=pi*r*r;
p=2*pi*r;
```

函数定义行

H1注释行

帮助文本区

函数体

1. 函数文件的格式：

function m-file：

```
function [ output_args] = Untitled( input_args)
%UNTITLED5 Summary of this function goes here
% Detailed explanation goes here

end
```

function 输出变量=函数名（输入变量）

H1行 – 用一行文字来综述函数的功能

帮助区

函数体语句

注释说明语句段

① 定义行function 输出变量=函数名(输入变量)

- 第一行必须以**function**作为引导词;
- 文件名与函数名应一致，即函数**名.m；**
- 该行列出函数与外界交换数据的全部**输入/输出量；**
- 输入/输出量数目不限，可有可无；
- 输出多个变量，应用方括号 **[v1,v2]**。

② H1注释行

- 文件第二行，也是帮助文本的第一行；
- %开头的注释行，字符不被matlab执行；
- 包含**大写体的文件名**和运用关键词简要描述的函数功能；
- 在线帮助使用；**lookfor**只在H1行查询关键词。

帮助文本区：

- H1行后%的注释行组成；
- 详细说明函数功能，如输入/输出量总数和调用格式说明
- Help function_name显示所有的%的注释行。

显示注释：help 文件名

```
help fcircle
FCIRCLE  calculate the area and perimeter of a circle of radii r
 r        圆半径
 s        圆面积
 p        圆周长
 2004年7月30日编
```

lookfor fcircle

```
>> lookfor fcircle
fcircle      - calculate the area and perimeter of a circle of radii r
```

③ 函数体

- 实现函数文件功能的指令组成；
- 它接受输入量、程序流控制，创建输出量。

④ 注释

- %开头的注释行；
- 出现函数文件的任意位置 ，用绿色表示；
- 对语句注释说明。

2 函数调用

① 调用格式：

[输出变量]=函数名(输入变量)

```
>> rr=234
rr =   234
>> [x,y]=ff(rr)
x =  1.7202e+005
y =  1.4703e+003
```

```
function [s,p]=ff(r)
```

```
>> clear
>> ff(333)
ans =3.4837e+005
```

```
>> whos
  Name      Size              Bytes  Class
  ans       1x1                   8  double
```

```
>> whos
  Name      Size              Bytes  Class
  rr        1x1                   8  double
  x         1x1                   8  double
  y         1x1                   8  double
```

② 函数调用特点

① 开辟临时函数工作空间(**Function workspace**) 存放中间变量;

② 运行完毕，中间变量被清除以及临时空间关闭；

③ 运行后只**保留最后结果**，不保留**中间过程；**

④ 程序中变量均为**局部变量；**

⑤ 函数调用各实参数**不必**与函数定义行中形参数**同名**；

⑥ 实参数的顺序、个数，应与形参数一致，否则出错。

④ 脚本文件和函数文件的区别

脚本式M文件	函数式M文件
无函数定义行；	有函数定义行；
无输入和输出量，也不一定要返回结果。	可以有输入和输出变量，并有返回结果；
在workspace基本工作空间中进行数据操作，运行后变量驻留其中；	中间变量存放在临时工作空间中，它随函数的调用结束而删除；
变量是全局变量。	变量是局部变量，除特别声明。

3.2.3. 全局变量和局部变量

1. 全局变量：

① 所有变量驻留在基本工作空间中，即全程有效；

② 所有函数都可对其进行存取和修改；

③ 定义全局变量是函数之间传递信息的手段。

2. 局部变量：

① 仅存于函数空间的中间变量，影响仅限于函数本身；

② 函数文件不能直接访问workspace中的全局变量，它只能读取通过参数传入的变量；

③ 函数文件中定义的变量不能被另一个函数文件引用；

④ 如果在若干函数中，把某个变量定义为全局变量，那么这些函数可以共用这个变量。

脚本文件：

a=2;
b=3;
c=a+b;
d=sin(c)
e=log(d)
保存为：abcd.m
在工作窗口中调用：
>>abcd　　%文件名即可

全局变量

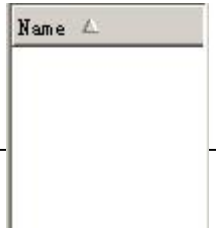
函数文件：

function abcd(a,b)
c=a+b
d=sin(c)
e=log(d)
保存为abcd.m

在工作窗口中调用：

>>a=2;b=[3,4,5,6];

>>**abcd(a,b)**



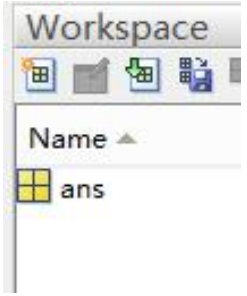
局部变量

function **e=**abcdef(a,b)
c=a+b
d=sin(c)
e=log(d)
保存为abcd.m

在工作窗口中调用：

>>a=2;b=[3,4,5,6];

>>**abcdef(a,b)**



3. 定义全局变量 global

调用格式： global 变量名

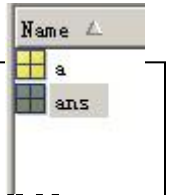
注意：变量之间以空格分隔，而不能用逗号分隔。

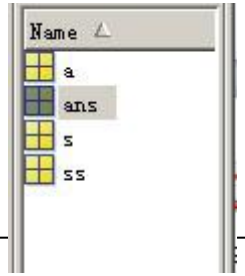
例

```
function [avgs]=test4(A)
global s  ss     %定义全局变量
[m,n]=size(A);
for i=1:m
    s(i)=sum(A(i,:))
end
ss=sum(s)
avgs=ss/(m*n)
```

调用test4：

```
>> A=[4 3 5;6 7 8;3 5 7;1 3 4];
>> test4(A)
ans =    4.6667
>> global  s  ss
>> ss
ss =    56
>> s
s = 12    21    15     8
```





3.2.4 子函数与主函数

- 一M文件可含多个函数,第一个是**主函数**,其它是**子函数**;
- 主函数必须在最前面，子函数次序可随意调整；
- 子函数仅被主函数或同一文件其它子函数所调用；
- **私有函数**是主函数的一种，只在限定函数群可见，一般放在private命名子目录中。它只对父目录中函数可见。

例题

```
function c=test(a,b)   %主函数
c=test1(a,b)*test2(a,b);
function c=test1(a,b)    %子函数1
c=a+b;
function c=test2(a,b)    %子函数2
c=a-b;
```

子函数则只能在主函数文件中编辑

3.2.5 函数句柄和匿名函数

1. 函数句柄: 是携带着函数路径、函数名以及可能存在的重载方式。

① 两种创建句柄：

```
function fv=fun(x)
    fv=x-10.^x+2
```

hfun=@+函数名

```
>> hfun=@fun
```

hfun=str2func('fun')

```
>> hfun=str2func('fun')
```

```
>>class(hd)
ans =function_handle
>>isa(hd, 'function_handle')
ans =1
```

② 函数句柄调用：

直接调用： **[y1, y2, ...] = hfun(x1, ..., xn)**

>> hfun(3) ans =46.0977

feval间接调用:

[y1, y2, ...] = feval(fhandle, x1, ..., xn)

>> feval(hfun,3) ans =46.0977

2. 匿名函数

f=@(xlist)expression

①以@符号开头；

②其中expression为函数体；

③xlist为输入参数列表

```
>> f=@(x)x.^3-cos(x)
>> f(33)= 3.5937e+004
```

3.3 程序控制结构

3.3.1 条件结构

3.3.2 循环结构

3.3.3 try-catch结构

3.3.4 其它指令

3.3.1 条件结构

条件结构:根据给定条件成立与否，执行不同语句。

条件结构的语句:

1. if 语句
2. switch 语句

1. if 条件语句

(1) 单分支结构

```
if expression （条件表达式：关系和逻辑）
    statements （语句组）
end
```

(2) 双分支结构

```
if expression （条件）
    statements1（语句组1）
else
    statements2（语句组2）
end
```

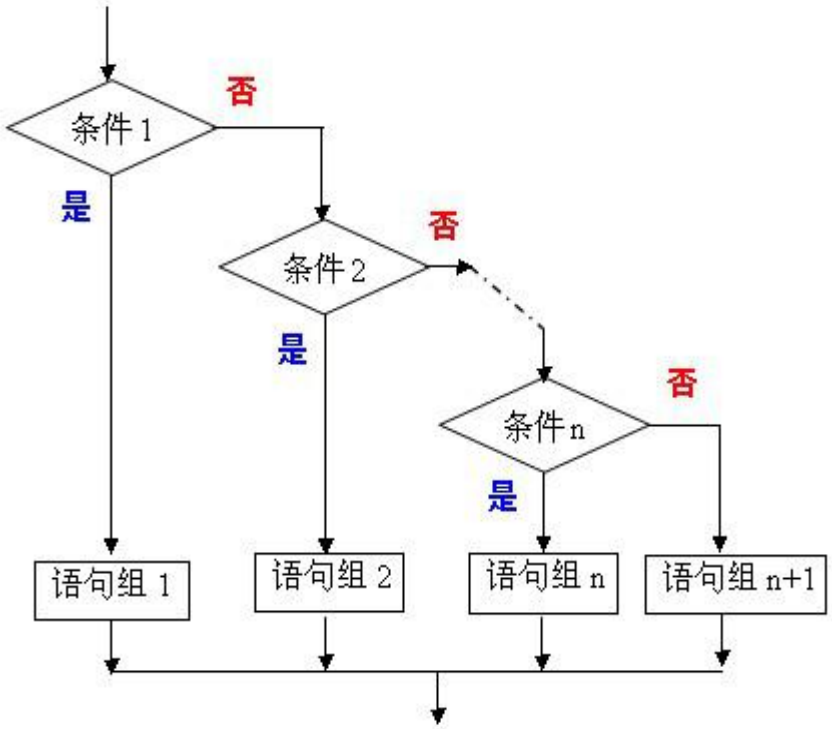
例

计算分段函数的值。

```
x=input('请输入x的值:');
if x<=0
   y=(x+sqrt(pi))/exp(2);
else
   y=log(x+sqrt(1+x*x))/2;
end
 fprintf('y=%e',y)
```

(3) 多分支结构

```
if expression1 （条件1）
   statements1（语句组1）
elseif expression2 （条件2）
   statements2（语句组2）
 ... ...
elseif expressionm （条件n）
   statementsm（语句组n）
else
   statements（语句组n+1）
end
```

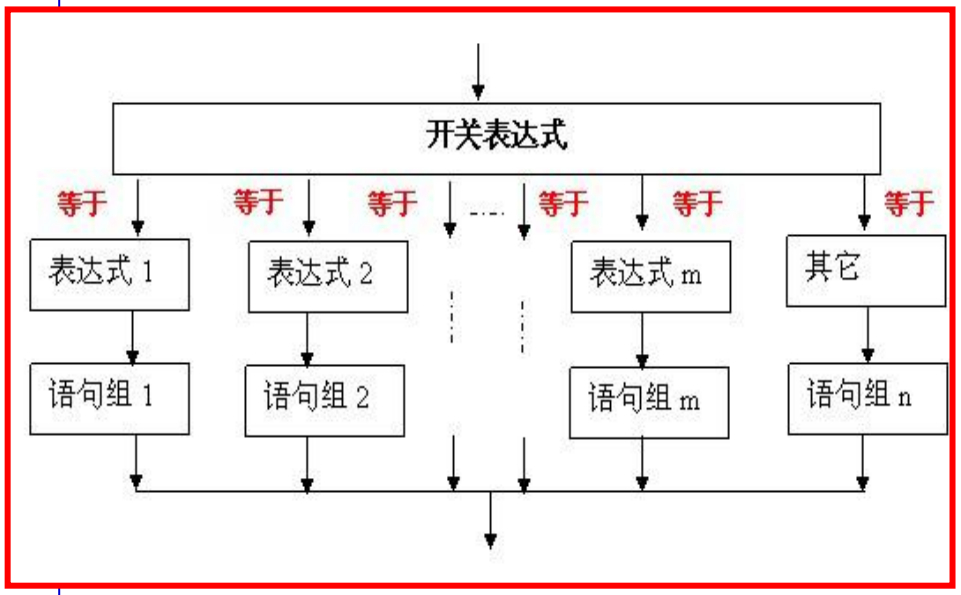


例题：

```
A=input('A=');
B=input('B=');
if A>B
    'greater'
elseif A<B
      'less'
elseif A==B
     'equal'
else
        error('A and B are
different data')
    end
```

2. switch 开关语句

```
switch expression （开关表达式）
    case value1 （表达式1）
         statement1（语句组1）
    case value2 （表达式2）
         statement2（语句组2）
       ... ...
    case valuem （表达式m）
         statementm（语句组m）
    otherwise
         statement （语句组n）
end
```



2. switch 语句特点

- 先计算 expression 的值，依次与各case 指令比较；
- 当比较结果为真时，就执行相应语句，再跳出switch 结构。
- 结果都为假，则执行otherwise 后的语句，再跳出 switch。
- otherwise 指令可以不出现。
- 开关表达式 expression 的值可以是标量或字符串。

例

```
code=input('c=')

switch code
    case -1
        disp('error');
    case 0
        disp('write in English');
    case 1
        disp('write in
Chinese');
   otherwise
        disp('write in French');
end
```

结果：　write in Chinese

3.3.2 循环结构

循环结构:按照给定的条件，重复执行指定的语句。

循环结构的语句:

for 语句

while 语句

1. for 循环

```
for variable=expression(循环变量)
    statement（循环体）
end
```

```
for 循环变量=初始值:步长:终止值
        循环体语句
end
```

=<循环次数设定>

例: $y=1+\frac{1}{3}+\frac{1}{5}+\cdots+\frac{1}{2n-1}$

当 n=100 时，求 y 的值

采用循环语句会降低其执行速度，所以程序通常由下面的程序来代替：

```
y=0; n=100;
for k=1:n
    y=y+1/(2*k-1);
end
y
```

```
n=100;
i=1:2:2*n-1
x=1./i
y=sum(x)
y
```

无循环

```
n=100;
y=sum(1./(1:2:
2*n-1));
y
```

例 一个三位整数各位数字的立方和等于该数本身则称该数为水仙花数。输出全部水仙花数。

```
for m=100:999
m1=fix(m/100);          %求m的百位数字
m2=rem(fix(m/10),10);   %求m的十位数字
m3=rem(m,10);           %求m的个位数字
if
   m==m1*m1*m1+m2*m2*m2+m3*m3*
   m3
disp(m)
end
end
```

2. 循环的嵌套

例：建立Hilbter函数

```
function H=hilbn(n)
for i = 1:n
    for j = n:-1:1
    H(i, j) = 1/(i+j-1);
    end
end
H
```

```
function H=myhilb
(n)
for i = 1:n
    for j =1:n
    H(i,j)=1/(i+j-1);
    end
end
```

3. while 循环

```
while expression （条件<逻辑变量>）
    statement（循环体）
end
```

例

求n为多少时，
2^n>100,其值多少

```
n = 0;
while 2^n < 100
    s= 2^n;
    n = n + 1 ;
end
s
n
```

4.for和while循环语句区别

- for适用已知到**循环次数**，而不知循环运算目标；
- while适用已知**循环运算目标**，而循环次数未知;
- 为了提高代码的运行效率，避免 for 循环的使用；

例.计算级数:$S=1+2+2^2+2^3+\cdots+2^{63}=\sum_{n=0}^{63}2^n$

```
s=0;
i=0;
while i<64
    s=s+2^i;
    i=i+1;
end
s
i
```

```
>> n=0:1:63;
>>s=sum(2.^n)
```

```
s=0;
i=0;
for i=1:63
    s=s+2^i;
    i=i+1;
end
s
i
```

例 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。

程序如下：

```
sum=0;
cnt=0;
val=input('Enter a number (end in 0):');
while val~=0
    sum=sum+val;
    cnt=cnt+1;
    val=input('Enter a number (end in 0):');
end
if cnt > 0
   sum
   mean=sum/cnt
end
```

3.3.3 try-catch语句

语句格式：

try

　语句组1

catch

　语句组2

end

lasterr　%显示出错原因

Try

① 检测程序代码是否出错；

② 先试探语句组1，如出现错误，则将错误信息赋给lasterr保留；

③ 并转去执行语句组2。

例 矩阵乘法运算要求两矩阵的维数相容，否则会出错。先求两矩阵的乘积，若出错，则自动转去求两矩阵的点乘。

```
A=[1,2,3;4,5,6]; B=[7,8,9;10,11,12];
try
   C=A*B;
catch
   C=A.*B;
end
C
lasterr                %显示出错原因
```

3.4 其它程序控制语句

在程序设计中经常遇到提前终止循环、跳出子程序、显示出错信息等情况，主要有**continue、break、return、pause、echo、error**等。

1. break 和 continue

- break和continue与循环结构for 和while相关的语句，它们一般与if语句配合使用。
- break 终止循环的执行，即跳出最内层循环；
- continue结束本次循环，进行下一次循环.

例

求[100，1000]之间第一个能被21整除的整数。

```
for n=100:1000
if rem(n,21)~=0
      continue
end
break
end
n
```

2. keyboard

① 停止运行,控制权给键盘，命令窗口 “>>”变成 “K>>”;

② 当输入return后，控制权将交回文件。

③ 对程序调试和程序运行中修改都很方便。

```
function abcde(a,b)
c=a+b;
keyboard
d=sin(c);
e=log(d)
```

```
>> abcde(3,4)
K>> dd=66+c
dd =    73
K>> return
e =   -0.4201
```

```
function abcdef(a,b)
c=a+b;
d=sin(c);
return
e=log(d)
```

3. return

① 使正在运行的函数正常退出，并返回它的代码段继续运行，

② 可强制结束该函数的执行，也可以中断keyboard

4. 程序的暂停 pause

pause 或 pause(n)

- n 是延迟时间，以秒为单位；
- 缺省，将暂停程序，直到用户按任意键后继续

- pause off 屏蔽程序中所有 pause 的作用
- pause on 打开 pause 的作用

若想强行终止程序的运行，可以使用 Ctrl+c

例如

```
function abcd (a,b)
c=a+b
d=sin(c)
e=log(d)
```

```
function abcd (a,b)
c=a+b
d=sin(c)
pause
e=log(d)
```

```
>> abcd(3,4)
c =   7
d =  0.6570
任意键
e = -0.4201
```

5. echo

在命令窗口显示执行过程的M文件的命令

echo on	%显示其后所有执行文件的指令
echo off	%不显示其后所有执行文件的指令
echo	%在上述两种情况之间切换
echo filename on	%显示 filename 所指定文件的指令
echo filename off	%不显示 filename 所指定文件的指令
echo on all	%显示所有文件的指令
echo off all	%不显示所有文件的指令

例

```
for n=100:1000
if rem(n,21)~=0
      continue
end
break
end
n
```

保存为ed

```
>> echo on
>> ed
for n=100:1000
if rem(n,21)~=0
      continue
if rem(n,21)~=0
      continue
if rem(n,21)~=0
      continue
if rem(n,21)~=0
      continue
if rem(n,21)~=0
      continue
if rem(n,21)~=0
end
break
end
n

n =  105
```

```
function abcd (a,b)
c=a+b
d=sin(c)
e=log(d)
```

```
>> echo  abcd on

>> abcd(3,4)
c=a+b;
d=sin(c);
e=log(d)

e =   -0.4201
```

6. error

此命令可显示指定的出错信息并终止当前程序的运行。语法规则如下：

```
error('message')
```

类似的还有 warning 命令，二者区别在于 warning 显示指定警告信息后程序仍继续运行。

```
function abcd (a,b)
c=a+b
d=sin(c)
cc
e=log(d)
```

```
>> abcd(3,4)
c =   7
d =  0.6570
??? Undefined function
or variable 'cc'.
Error in ==> abcd at 4
cc
```

3.5 程序调试

M文件错误种类：

语法错误：函数参数输入类型，括号，矩阵运算

运算错误 ：运行过程中死机或溢出，与程序本身有关。

M文件设计应避免情况：Inf，nan或空矩阵

避免方法：可能出现异常地方提供识别语句，同时采用其他方法。

识别语句：isinf, innan, isempty

设置断点

这是最重要的部分，可以利用它来指定程序代码的断点，使得程序在断点前停止执行，并进入调试模式，从而可以检查当前各个**变量的值**。

- dbstop in mfile

  在文件名为mfile的M文件的**第一个可执行语句**前设置断点。

- dbstop in mfile at lineno

  在文件名为mfile的M文件的第lineno行设置断点。如果第lineno行为非执行语句，则在其后的第一个可执行语句前设置断点。

- dbstop in mfile at subfun

  在文件名为mfile的M文件的子程序subfun的第一个可执行语句前设置断点。

- dbstop if error

  在程序运行遇到错误时，自动设置断点。这里的错误不包括try…catch之间的错误。

- dbstop if all error

  在程序运行遇到错误时，自动设置断点。这里的错误包括try…catch之间的错误。

- dbstop if warning

  在程序运行遇到警告时，自动设置断点

- dbstop if caught error

  在程序运行try…catch间代码遇到错误时，自动设置断点。

- dbstop if naninf 或 dbstop if infnan

  当程序运行遇到**无穷值或者非数值**时，自动设置断点。

其它调用函数

- dbtype显示行号的M文件文本
- dbstatus显示断点信息
- dbcont 继续执行 到程序结束，或者下个断点
- dbstep将从断点处继续执行M文件
- dbclear in mfile清除断点
- dbquit 退出调试状态

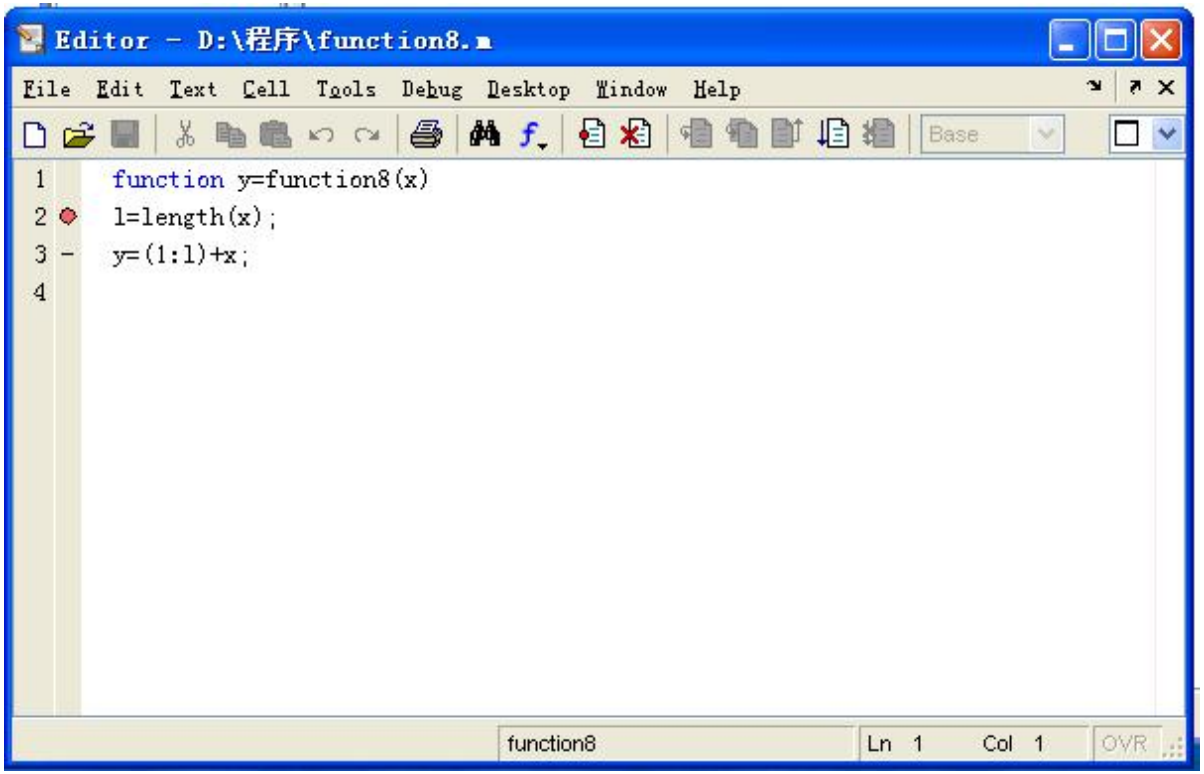
例　以函数 function8()为例说明如何使用命令调试程序，它的内容如下：

```
function y=test1(x)
l=length(x);
y=(1:l)+x;
```

```
function y=t (x)
l=length(x);
y={1:l}+x
```

由程序不难看出，函数 function8()中的输入只能为向量，如果输入的是矩阵则会产生错误。

在命令窗口输入dbstop in function，并打开文件function.m就可看到如下图所示的界面，它在第一个可执行语句前设置了断点。



单击图中红点，会发现红点被取消，此时回复到初始状态。然后在命令窗口依次输入dbstop if error和test(magic(3))，可得到如下的运行结果和如下图所示的界面。