第2章 MATLAB程序设计

本章内容

2.1 脚本M文件

2.2 程序流程语句

2.3 函数文件

2.4 全局变量

2.5 例题讲解

学时：5

2.1 脚本M文件

- 2.1.1 Matlab的两种工作方式
- 2.1.2 M文件的建立
- 2.1.3 M文件的打开
- 2.1.4 M文件的运行
- 2.1.5 M文件的调试

2.1.1 Matlab的两种工作方式

1）交互式的指令行操作方式

用户在命令窗口中按照Matlab的语法规则输入命令行并按下回车键后，系统将执行该命令并及时给出运算结果。该方式简便易行，非常适合于简单问题的数学演算、结果分析及测试。

缺点：

- 命令行操作时，Matlab窗口只允许一次执行一行上的一个或几个语句。
- 命令行方式程序可读性差，而且不能存储。

2.1.1 Matlab的两种工作方式

2）M文件的编程工作方式

用户通过命令窗口中调用M文件，从而实现一次执行多条Matlab语句的方式。

M文件是由命令或函数构成的文本文件，可以用任何文本编辑程序来建立和编辑。

一般常用且最为方便的是使用MATLAB提供的脚本编辑器。

2.1.1 Matlab的两种工作方式

- M文件的作用是：当用户在命令窗口中输入已编辑并保存的M文件的文件名并按下回车键后，系统将搜索该文件，若该文件存在，系统将按M文件中的语句所规定的计算任务**以解释的方式逐一执行语句**，从而实现用户要求的特定功能。
- M文件又分两类：

  命令M文件(命令文件) —— 独立的m文件

  函数M文件(函数文件) —— 可调用的m文件

2.1.2 M文件的建立

a．打开脚本编辑器

有以下三种方法：

1）命令操作：在命令窗口输入命令edit，回车；

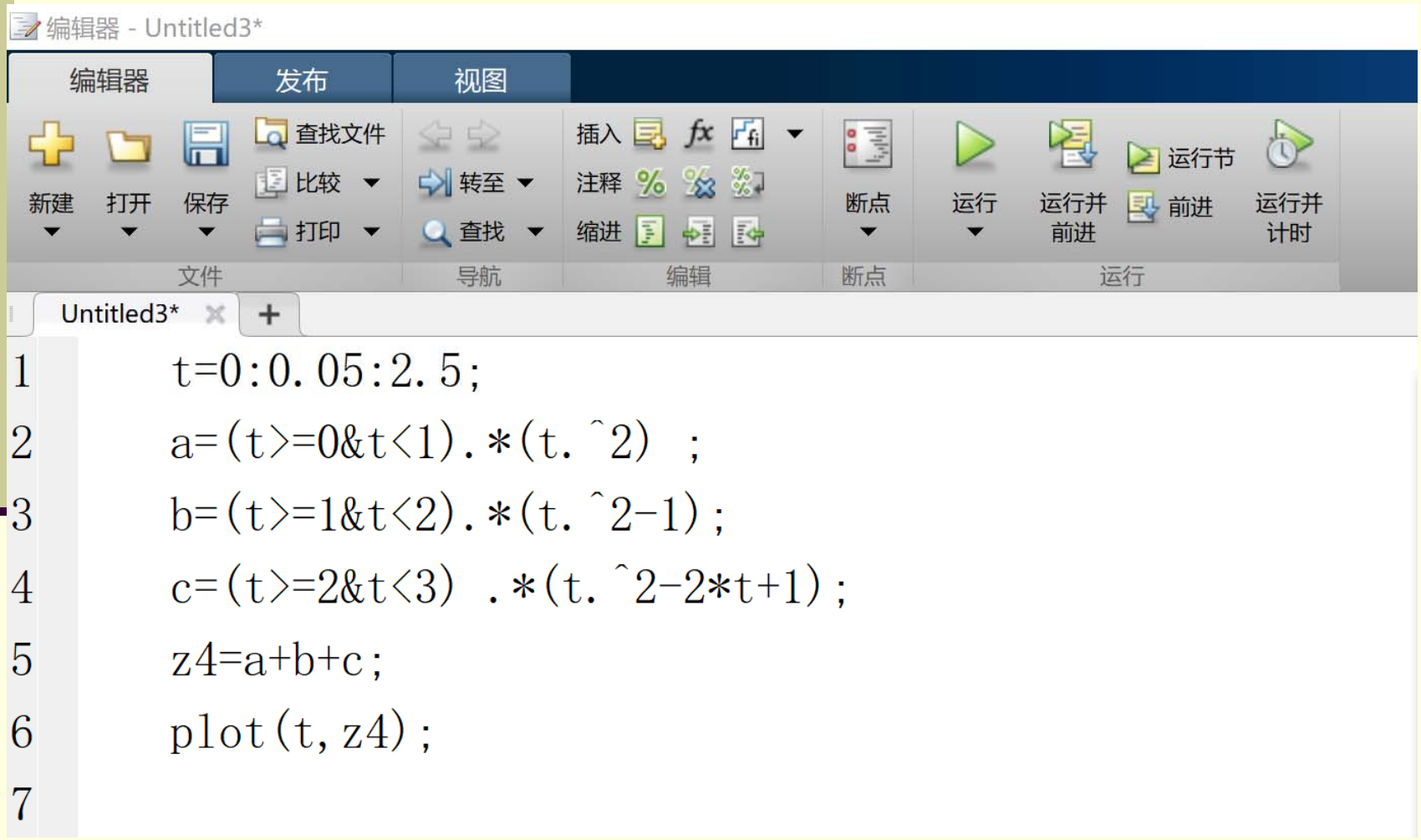
2）命令按钮操作：

Matlab操作桌面的主页窗口

3）Ctrl+N

2.1.2 M文件的建立

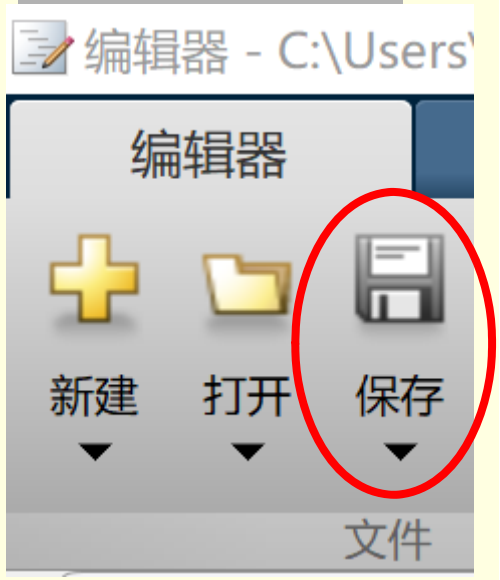
MATLAB脚本编辑器



2.1.2 M文件的建立

b. 建立新的M文件

在**编辑器的编辑器窗口**输入文件内容，输入完毕后，选择文本编辑器窗口的保存按钮，默认名字是Untitled。



注意，M文件存放位置一般是MATLAB缺省的用户工作目录，如果要选择别的目录，则应该将该目录设定为当前目录或将其加到搜索路径中，以便于文件查找。

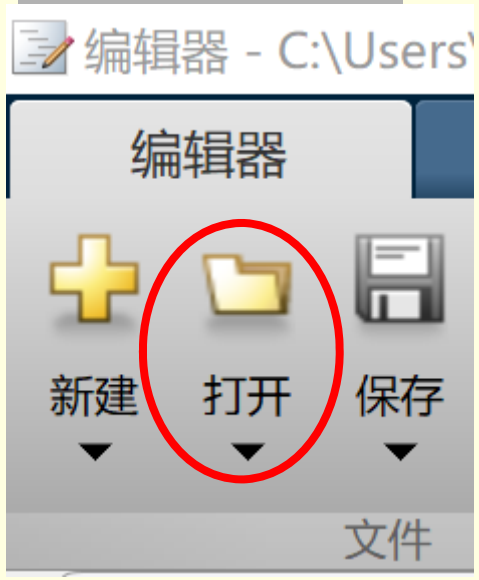
2.1.3 M文件的打开

打开已有的M文件，也有多种方法：

1）命令按钮操作：

Matlab操作桌面的主页窗口

编辑器的编辑器窗口

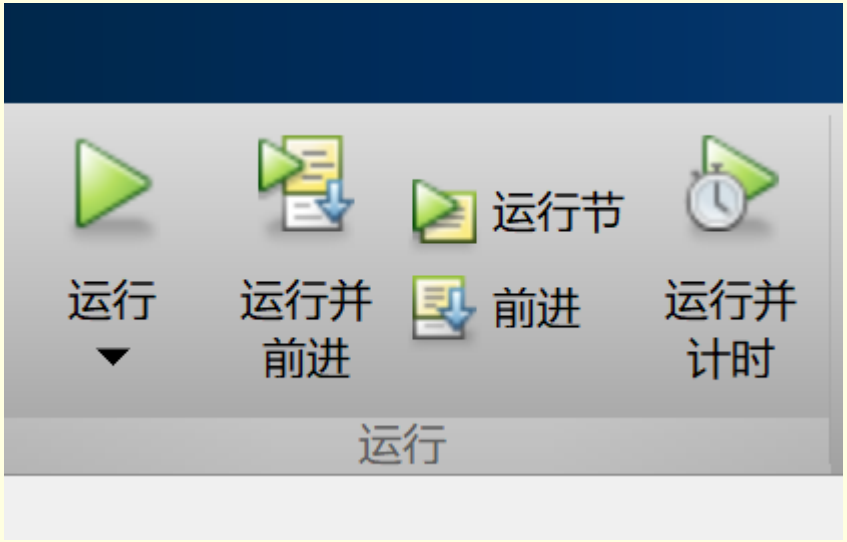


2）在**当前文件夹**窗口双击打开

3）命令操作：

在命令窗口输入命令**edit** 文件名，回车后则打开指定的M文件。如果文件不在当前路径下，还需在文件名前加上路径。

2.1.4 M文件的运行



2.1.5 M文件的调试

a. 直接调试法

1）如果在错误信息中指出了出错的行号，可先根据错误信息检查该语句是否存在语法错误或运行中变量尺寸不一致等情况。

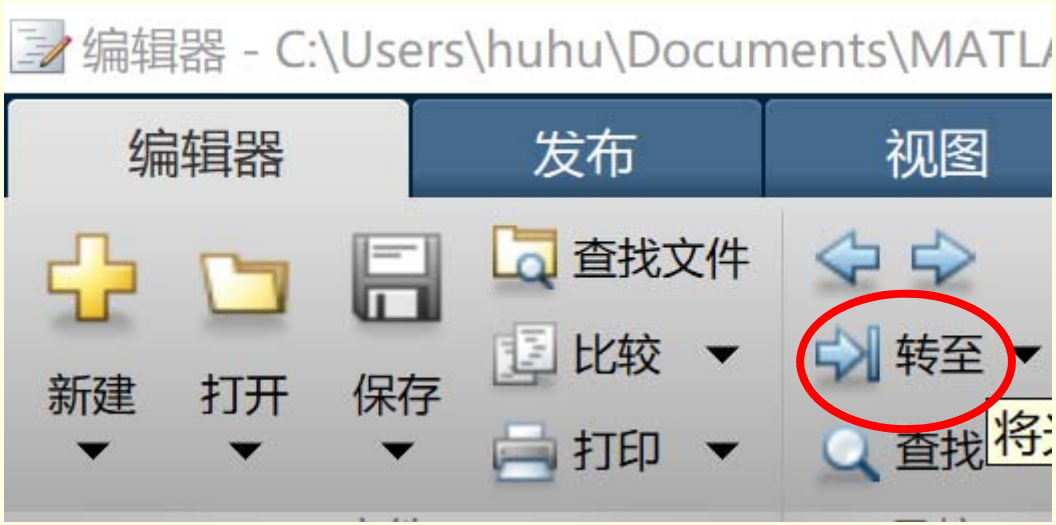
2）检查所调用函数或命令的拼写是否正确，括号（包括方括号和圆括号）是否配对，各种流程控制语句是否匹配（如for与end、while与end、switch与end等）。

3）检查所调用的函数或载入的数据文件是否在当前目录或搜索路径上。

4）将重点怀疑的命令行后的分号删除，使得计算结果能够实时地显示在屏幕上，作为查错的依据，根据显示的结果判断问题的所在。

2.1.5 M文件的调试

（1）转至行　　光标自动移到指定的行上。

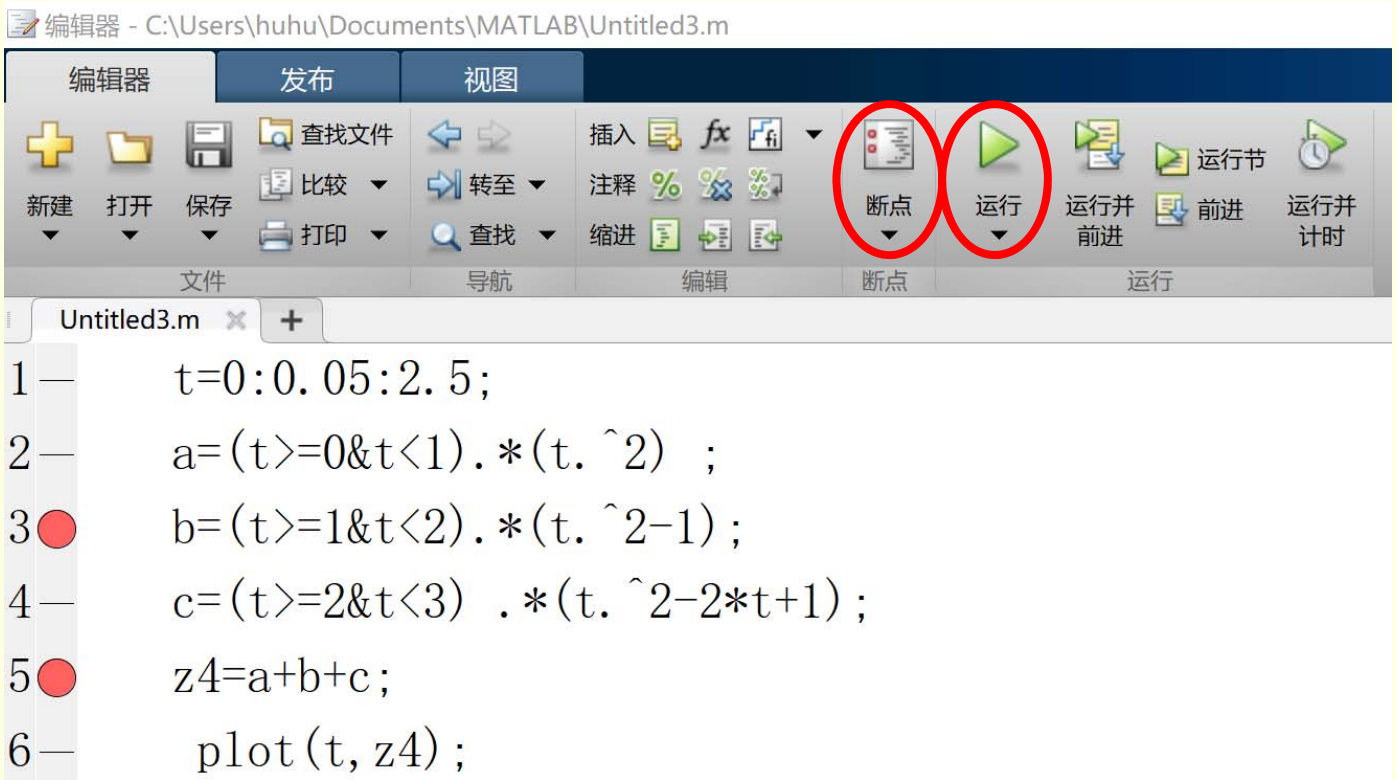


（2）“设置或清除书签”。

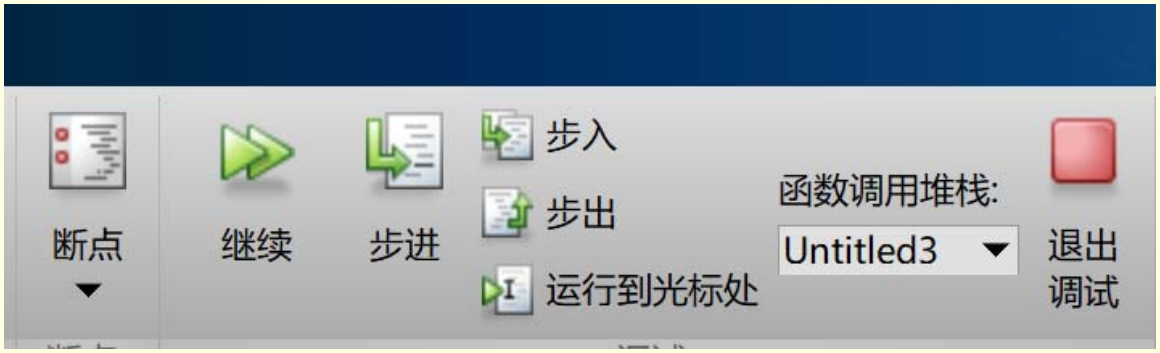
2.1.5 M文件的调试

（1）设置断点

断点是在程序特定位置设置的中断点，当程序运行至断点处时会暂停运行，此时可通过检查相关变量的内容等方法确定程序的运行是否正确。



2.1.5 M文件的调试



1）步进：单步执行。每单击一次，程序运行一次，但不进入函数。

2）步入：单步运行。遇到函数时进入函数内，仍单步运行。

3）步出：停止单步运行。如果是在函数中，跳出函数；如果不在函数中，直接运行到下一个断点处。

4）继续：运行到下一个断点时暂停运行。

5）运行到光标处：从当前位置运行到光标所在的位置。

2.2 程序流程语句

- 2.2.1 顺序结构
- 2.2.2 分支结构
- 2.2.3 循环结构
- 2.2.4 例题讲解

2.2 程序流程语句

Matlab的控制语句同c语言有相似之处，但没有c 语言复杂、灵活和多变。因而语法比较简单，容易掌握。

Matlab语言的程序结构与其它高级语言是一致的，分为顺序结构，循环结构，分支结构。

2.2.1 顺序结构

1）顺序结构 —— 依次顺序执行程序的条语句

■ 数据的输入

使用input函数从键盘输入数据，调用格式为：

A=input(提示信息，选项)；

其中提示信息为一个字符串，用于提示用户输入什么样的数据。如果在input函数调用时采用‘s’选项，则允许用户输入一个字符串。

```
>> A=input('输入矩阵A：');
输入矩阵A：[1,2,3,4,5,6]  %按照规定的格式输入矩阵的值，按回车结束
>> B=input('What's your name?');  %格式1
What's your name?'Mike'  %输入的字符串要用''界定
>> C=input('What''s your name?','s');  %格式2
What's your name?Lily  %输入的字符串不需要用''界定
```

2.2 程序流程语句

■ 数据的输出

Matlab提供的命令窗口输出函数主要有disp函数，其调用格式为：

disp(输出项)

其中输出项既可以为字符串，也可以为矩阵。

- 例2:
- 程序如下:
- **x=input('Input x please.’); %输入x的值**
- **y=input('Input y please.’); %输入y的值**
- **z=x; %将它们的值互换**
- **x=y;**
- **y=z;**
- **disp(x); %输出**
- **disp(y);**

- 例3求一元二次方程 $ax^2+bx+c=0$的根。

- 程序如下:
- **a=input('a=?');**
- **b=input('b=?');**
- **c=input('c=?');**
- **d=b*b-4*a*c;**
- **x=[(-b+sqrt(d))/(2*a),(-b-sqrt(d))/(2*a)];**
- **disp(x);**

2.2 程序流程语句

2）选择结构

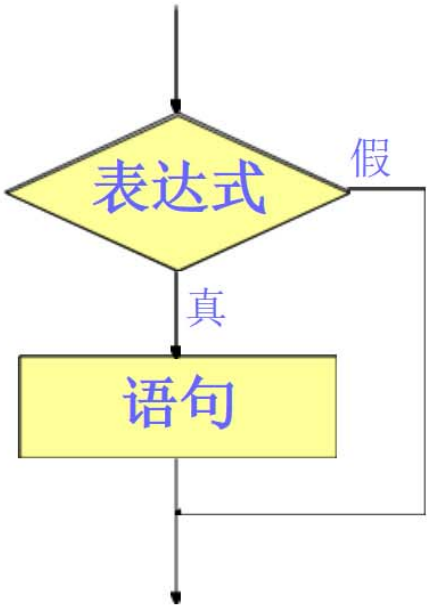
■ if语句

在Matlab中，if语句有4种格式。

(1).单分支if语句

```
if (条件表达式)
   语句;
end
```

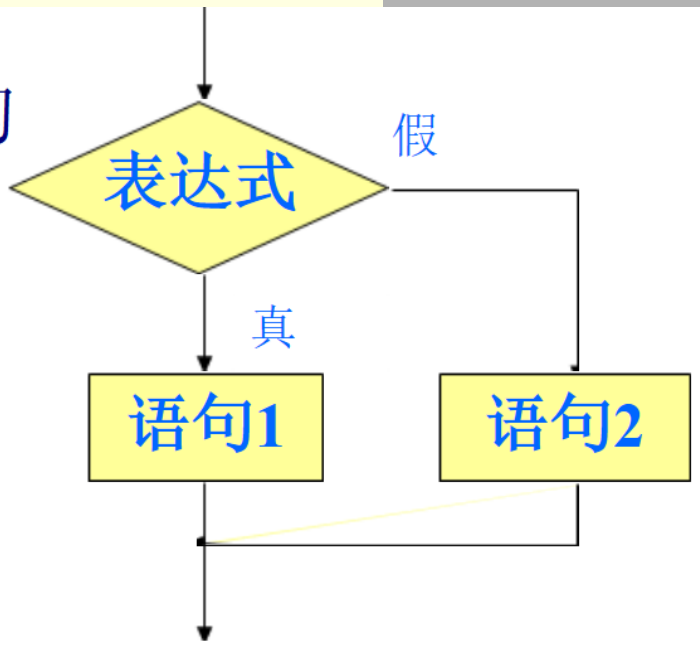
```
if (条件表达式)
   语句;
   语句;
   ……
end
```



当条件成立时，则执行语句组，执行完之后继续执行if语句的后继语句，若条件不成立，则直接执行if语句的后继语句。

(2). if语句的双分支选择结构

```
if (条件表达式)
   语句1;
else
   语句2;
end
```



当条件成立时，执行语句组1，否则执行语句组2，语句组1或语句组2执行后，再执行if语句的后继语句。

if语句的双分支形式，在语法上视为一条语句。

例题：双分支if语句

```
a=2;
if a>1
    b=1;
else
    b=2;
end
```

b=?

例题：双分支if语句

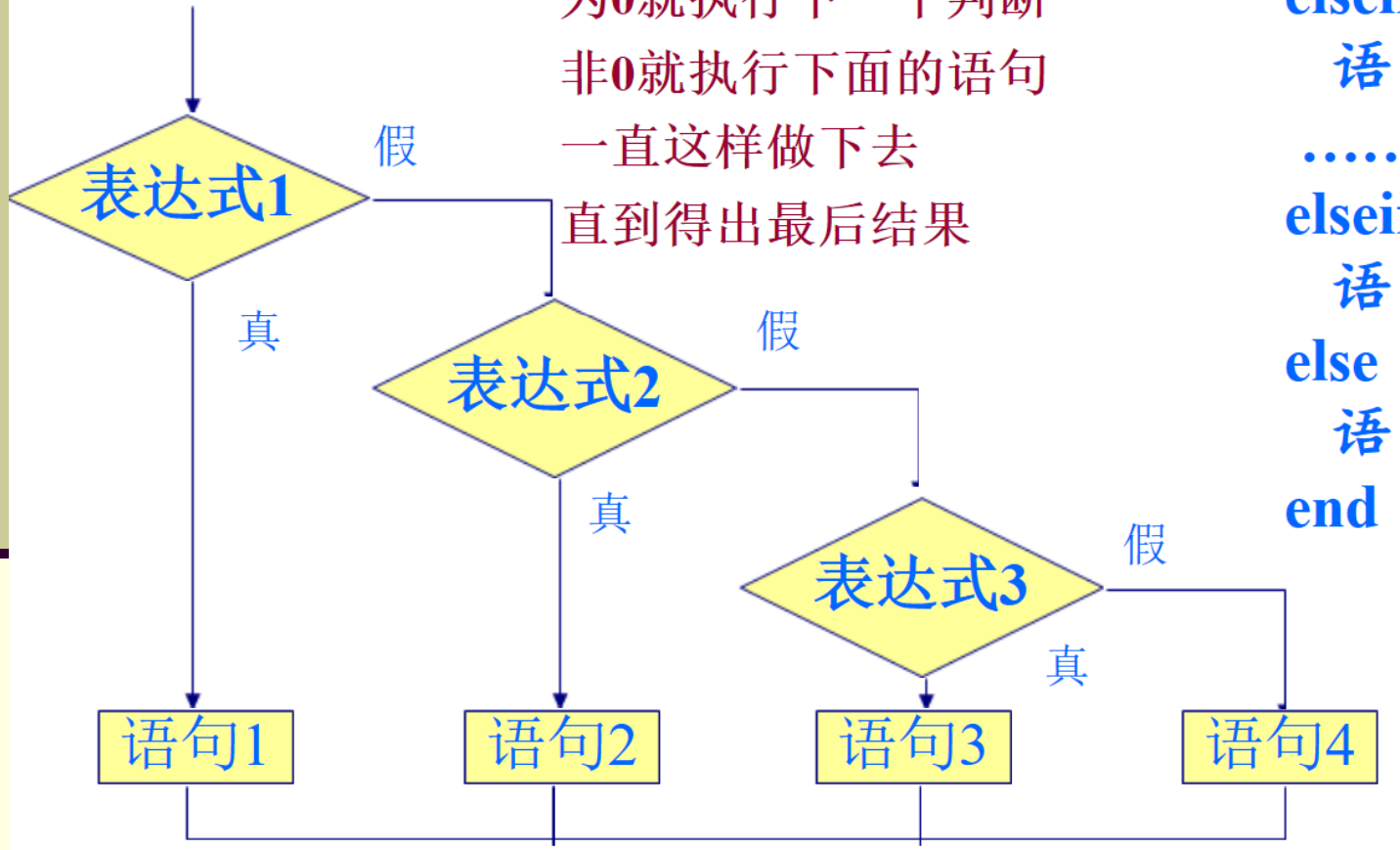
计算分段函数：

$$y=\begin{cases}\cos(x+1)+\sqrt{x^2+1}, & x=10\\ x\sqrt{x+\sqrt{x}}, & x\neq 10\end{cases}$$

```
x=input('请输入x的值:');
if x==10
  y=cos(x+1)+sqrt(x*x+1);
else
  y=x*sqrt(x+sqrt(x));
end
y
```

(3). if语句的多分支选择结构

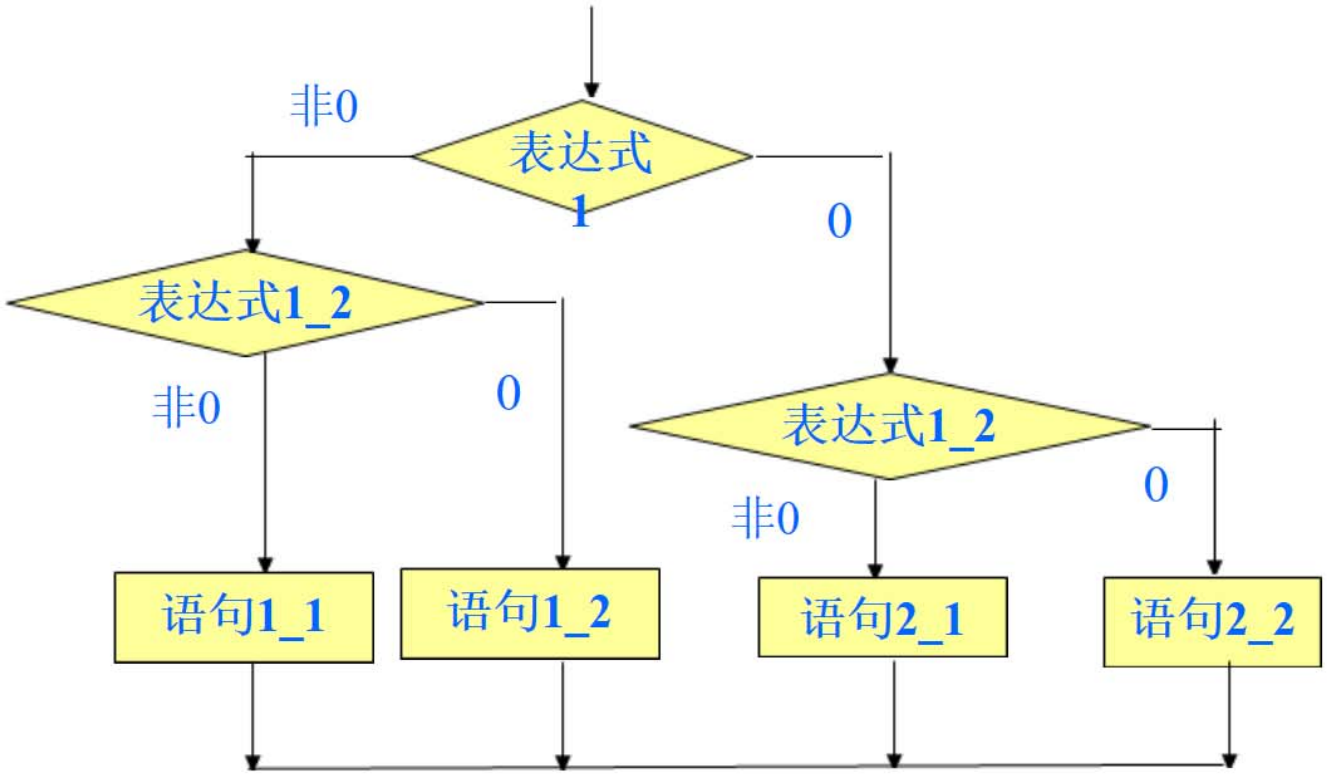
执行过程： 先判断表达式1的值
为0就执行下一个判断
非0就执行下面的语句
一直这样做下去
直到得出最后结果




```
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
```

(4). if语句的二层嵌套结构

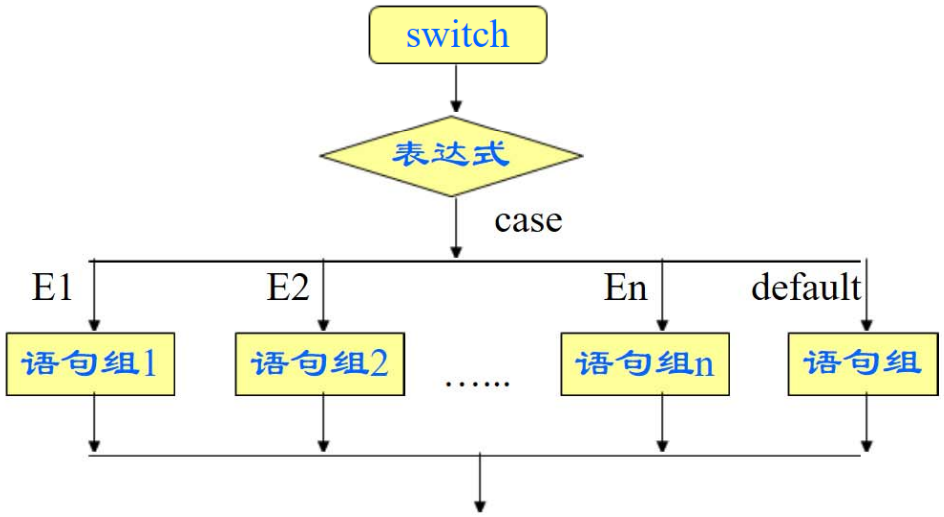
```
if(表达式1)
  if(表达式1_2)
    语句1_1;
  else 语句1_2;
end
else
  if(表达式2_1)
    语句2_1;
  else 语句2_2;
end
```



2.2.2、 switch语句（多分支选择语句）

if语句只有两个分支可供选择，如果分支较多,则嵌套的if语句层数多，程序冗长而且可读性降低。 Matlab提供了switch语句可以更方便地完成。

switch语句根据表达式的取值不同，分别执行不同的语句，其语句格式为：




```
switch 表达式
  case 表达式1
     语句组1;
  case 表达式2
     语句组2;
  ……
  case 表达式m
     语句组m;
  otherwise
     语句组n;
end
```

说明：先计算表达式的值，再按顺序与case语句后面的数组值进行比较，如果相等则执行该组语句，然后执行end后的语句，不再继续比较。当表达式的值不等于任何一个case语句后面的数组值时，程序将执行otherwise语句后的语句组，再执行 end后的语句。注意：这种情况下缺省otherwise语句，程序会提示出错。

例：多分支switch语句

```
a=78;
switch fix(a/10)        %fix为向零取整函数
    case {9,10}
        b='优秀';
    case 8
        b='良好';
    case 7
        b='中等';
    case 6
        b='及格';
    otherwise
        b='不及格';
end
```

2.2.3、while语句

while语句是条件循环语句，在条件（多为关系表达式）控制下重复执行，直到条件不成立为止。while循环的一般形式是：

```
while 表达式
  语句体
End
```

说明：先计算表达式的值，如果非零，语句体就执行一次；执行完毕再次计算表达式的值，如果仍然非零，语句体就再执行一次；如此循环，直到表达式的值为零。如果表达式的值总是非零，该循环将无休止地进行（即死循环），程序设计时一定要避免。

【例】求1+2+3+…… +100的和

- s=
- 5050

【例】求1+2+3+…… +100的和

```
clear
clc
k=0;
s=0;
while k<=100
        s=s+k;
        k=k+1;
end
s
```

```
s=
     5050
```

2.2.4、for语句

for语句为计数循环语句，在许多情况下，循环条件是有规律变化的，通常把循环条件初值、终值和变化步长放在循环语句的开头，这种形式就是for语句的循环结构。for循环的一般形式是：

```
for 循环变量名=表达式1：表达式2：表达式3
  语句体
end
```

说明：

表达式1的值是循环变量的初值

表达式2的值是循环步长

表达式3的值是循环变量的终值。

初值、步长和终值可以取整数、小数、正数和负数，步长可以缺省，缺省值为1。

【例】求1+2+3+…… +100的和

```
clear
clc
s=0;
for  k=1:100
            s=s+k;
end
s
```

```
s=
     5050
```

2.2.5、 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。

任一种循环语句的循环体部分都可以包含另一个循环语句，多重循环嵌套的层数可以是任意的。

习惯上按照嵌套层数，分别叫做二重循环、三重循环等。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】有一数列：$1^1+1^2+1^3\ldots+1^{10}+2^1+2^2+2^3+\ldots+2^{10}+3^1+3^2+3^3+\ldots+3^{10}$，求这些项的和

- s=
- 90628

【例】有一数列：$1^1+1^2+1^3\ldots+1^{10}+2^1+2^2+2^3+\ldots+2^{10}+3^1+3^2+3^3+\ldots+3^{10}$，求这些项的和

```
clear
clc
s=0;
for  k=1:3                %外层循环，分别产生1，2，3
      for  m=1:10         %内层循环，分别产生1~10
           s=s+k^m;       %求和
      end
end
s                         %输出结果
```

```
s=
    90628
```

2.2.6、其他语句

1．continue语句

continue语句用于控制for循环或while循环跳过某些执行语句，当出现continue语句时，则跳过循环体中所有剩余的语句，继续下一次循环，即结束本次循环。

2．break语句

break语句用于终止for循环和while循环的执行。当遇到break语句时，则退出循环体，继续执行循环体外的下一个语句，即中止循环。在嵌套循环中，break往往存在于内层的循环中。

【例】预测如下程序的结果

```
k=0;
a=0;
b=0;
for  k=1:10
    a=a+1;
    if k==4
        continue;
    end
    b=b+1;
end
```

a=?

b=?

```
k=0;
a=0;
b=0;
for  k=1:10
    a=a+1;
    if k==4
        break;
    end
    b=b+1;
end
```

a=?

b=?