第5章 MATLAB编程

MATLAB既是一个交互式的计算工具，也是一种效率极高的解释性程序语言，可以方便地调用所需要的MATLAB内置函数或相关工具箱，满足工作的需要。对于结构复杂、编程量比较大的程序或定义函数，不宜采用命令行输入的办法，而要通过对源程序加以修改或自己编写程序，即使用M文件，使其可以像库函数一样方便调用，甚至可以构造出新的专用工具包。本章是本书学习的重点和难点，要求重点掌握MATLAB编程的基本结构、自定义函数的编程和程序调试等。

【知识要点】

掌握和应用M文件：脚本和函数的概念；MATLAB编程的基本结构：顺序结构、循环结构和分支结构；自定义函数的几种方式和程序调试。

【学习目标】

知　识　点	学习目标			
	了解	理解	掌握	运用
脚本			★	★
函数			★	★
程序设计			★	★
自定义函数				★
程序调试			★	★

基于矩阵的MATLAB语言是世界上表示计算数学最自然的方式，它包含函数、数据结构、控制语句、输入/输出和面向对象编程。用户可以在命令行窗口中将输入语句与执行命令同步，也可以先编好一个较大的复杂的应用程序后再一起运行。图5-1所示为MATLAB程序控制输入/输出形象化示意图。

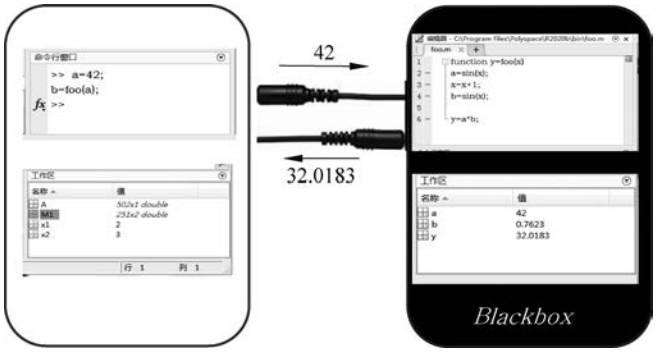

图5-1　MATLAB程序控制输入/输出形象化示意图

视频讲解

5.1 M文件

在命令行窗口中一次输入一个命令是执行MATLAB的最普遍的方式。利用MATLAB进行简单计算与绘图时,因为输入的语句不多,可以在命令行窗口中一行一行地输入并执行。由于在命令行窗口中输入无法保存且无法重复运行,因而无法实现较复杂功能。

M文件提供了另外一种执行的途径,可通过文本编辑器生成M文件。M文件有两种形式,即脚本文件(script)和函数文件(function)。

5.1.1 脚本文件

脚本文件是最简单的程序文件类型,是一系列存储于文件中的MATLAB命令。

脚本可以在命令行窗口中输入文件名来运行,或者可利用编辑器中的菜单,下拉后选取Debug及Run执行,即自动执行一系列MATLAB命令,如图5-2所示。

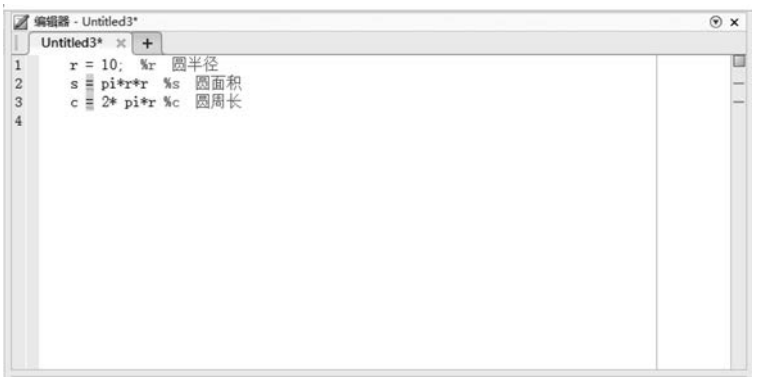

图5-2 编辑器

【例5-1】 编写函数文件,求半径为r的圆的面积和周长。

单击新建脚本,打开编辑器,键入以下命令:

```
r = 10;                    %r  圆半径
s = pi*r*r                 %s  圆面积
c = 2* pi*r                %c  圆周长
```

保存为circlesc.m文件。

在命令行窗口中调用该脚本:

```
circlesc
```

运行结果:

```
s =
  314.1593
c =
   62.8319
```

或者可以利用编辑视窗中的编辑器选项,选取 运行。

【例 5-2】 创建一个脚本以计算三角形的面积，命名为 tria.m。

单击新建脚本，打开编辑器，键入以下命令：

```
b = 5;                          %b  三角形的底
h = 3;                          %h  三角形的高
a = 0.5*(b.*h)                  %三角形面积
```

保存为 tria.m 文件.

在命令行窗口中调用该脚本：

```
tria
```

运行结果：

```
a =
    7.5000
```

要使用同一脚本计算另一个三角形的面积，可以更新 b 和 h 在脚本中的值并返回值。每次运行脚本时，都会将结果存储在名为 a 的变量(位于基础工作区中)中。脚本文件中的语句可使用工作空间中的全局变量。脚本的优缺点如表 5-1 所示。

表 5-1 脚本的优缺点

优　　点	缺　　点
适用于简单且重复性高的程序代码	不支持输入与输出变量
变量保留在基本工作空间，便于检查及除错	变量保留在基本工作空间，会因变量互相覆盖而造成程序错误

5.1.2 函数文件

函数适用于大型程序代码，可使程序代码模块化，易于改进，将脚本转换为函数可提升代码的灵活性，这样就无须每次手动更新脚本。

以 function 语句引导，表示该 M 文件是一个函数文件。

函数文件的基本结构：

```
function 输出参数 = 函数名(输入参数)
 %注释说明部分
 函数体语句
```

在编辑器中编写函数如图 5-3 所示。

【例 5-3】 编写函数文件，求半径为 r 的圆的面积和周长。

函数文件如下：

```
function [s,p] = fcircle(r)
%计算圆面积 s 和圆周长 c
%r   圆半径
%s   圆面积
%c   圆周长
s = pi*r*r;                     %求圆面积 s
c = 2* pi*r;                    %求圆周长 c
```

图 5-3 在编辑器中编写函数

在命令行窗口中输入：

```
[s,c] = fcircle(10)
```

运行结果：

```
s =
314.1593
c =
62.8319
```

【例 5-4】 编写函数文件，求三角形的面积。

程序如下：

```
function a = triarea(b,h)
 % b  三角形的底
 % h  三角形的高
a = 0.5 * (b. * h);                          % 求三角形面积
end
```

保存该文件后，可以在命令行窗口中调用函数计算不同的底和高的三角形的面积，而不用修改脚本：

```
a1 = triarea(1,5)
a2 = triarea(2,10)
a3 = triarea(3,6)
```

运行结果：

```
a1 =
   2.5000
a2 =
   10
a3 =
    9
```

函数具有它们自己的工作区，与基础工作区隔开。因此，对函数 triarea 的任何调用都不会覆盖变量 a 在基础工作区中的值，但该函数会将结果指定给变量 a1、a2 和 a3。

1. 主函数与子函数

一个 M 文件可以包含一个以上的函数。第一个函数称为主函数(primary function),其他则称为子函数(subfunction)。子函数只能被同一 M 文件中的函数调用,不可被不同文件的其他函数调用。

主函数必须出现在最上方,其后接任意数目的子函数,子函数的次序无任何限制。

【例 5-5】 编写函数文件,求二次方程的根。

主函数如下:

```
%此函数返回二次方程
function [x1,x2] = quadratic(a,b,c)                    %主函数
d = disc(a,b,c);                                       %调用子函数
x1 = (-b + d) / (2*a);
x2 = (-b - d) / (2*a);
end
```

子函数如下:

```
function dis = disc(a,b,c)                             %子函数
dis = sqrt(b^2 - 4*a*c);                               %函数计算判别式
end
```

调用上述函数:

```
quadratic(2,4,-4)
```

返回以下结果:

```
ans =
   0.7321
```

MATLAB 可以在 M 文件函数中定义多个子函数。

下面的例子即为主函数调用两个子函数。

主函数如下:

```
function [max,min] = mypfun(x)                         %主函数
n = length(x);
max = mysubfun1(x,n);
min = mysubfun2(x);
```

子函数 1 如下:

```
function r = mysubfun1(x,n)                            %子函数 1
x1 = sort(x);
r = x1(n);
```

子函数 2 如下:

```
function r = mysubfun2(x)                              %子函数 2
x1 = sort(x);
r = x1(1);
```

从以上分析可以看出,脚本没有输入参数和输出参数,对数据的操作是整个工作区。函数既有输入参数,也有输出参数,对数据的操作是对局部的工作区。

函数文件中定义及使用的变量大都是局部变量(local variable),一旦退出该函数,即为无效变量。而脚本文件中定义或使用的变量都是全局变量(global variables),在退出文件后仍为有效变量。脚本与函数的不同点如表 5-2 所示。

表 5-2 脚本与函数的不同点

脚 本	函 数
没有输入参数	有输入参数
没有输出参数	有输出参数
对数据的操作是整个工作区	对数据的操作是对局部的工作区

2. 嵌套函数

所谓嵌套函数(nested function)即在函数内部再定义一个函数。嵌套函数可以访问和修改它们所在的函数工作区中的变量。

嵌套函数的基本结构:

```
function x = A(p1, p2)
...
B(p2)
   function y = B(p3)
   ...
   end
...
end
```

【例 5-6】 用嵌套函数编写求二次方程根的程序。

程序如下:

```
function [x1,x2] = quadratic2(a,b,c)
global d                          %声明为全局变量,可以由多个函数共享
  function disc                   %嵌套函数
  d = sqrt(b^2 - 4*a*c);
  end                             % 结束函数 disc
```

嵌套函数如下:

```
disc;
x1 = (-b + d) / (2*a)
x2 = (-b - d) / (2*a)
end                       %结束函数 quadratic2
```

调用函数:

```
quadratic2(2,4,-4)
```

运行结果

```
x1 =
    0.7321
x2 =
   -2.7321
```

```
ans =
    0.7321
```

通常，程序中的变量均为局部变量，这些变量独立于其他函数的局部变量和工作空间的变量。如果几个函数文件要共用一个变量，则要在这些函数文件中都定义这个变量是全局变量。

子函数与嵌套函数的区别如下。

（1）子函数和嵌套函数的区别仅在于主函数的变量对其是否可见。主函数数据对嵌套函数可见(类似全局变量)。嵌套函数可以直接操作主函数在调用嵌套函数之前声明的变量。

（2）嵌套函数之间的私有(内部)数据不互通，嵌套函数和子函数数据也不互通。

（3）嵌套函数能调用子函数，但子函数不能调用嵌套函数。

5.2 程序设计结构

在 M 文件中，既可以按照顺序执行指令，也可以使用循环结构和分支结构来控制流程，故 MATLAB 编程的基本结构包括：顺序结构、循环结构和分支结构。这种 M 文件和 C 语言中常说的程序已经非常相似了。

5.2.1 顺序结构

顺序结构是最简单的程序结构，在编写好程序后，系统会按照程序的物理位置顺次执行程序中的语句，如图 5-4 所示。由于没有控制语句，结构也比较单一，故这种程序比较容易编写。

【例 5-7】 绘制[－2π，2π]区间内正弦函数的图像。

程序如下：

```
x = -2*pi:pi/20:2*pi;
y = sin(x);
plot(x,y)
title('绘制正弦曲线 sin(x)');
```

运行后所绘制的正弦函数图像如图 5-5 所示。

图 5-4 顺序结构

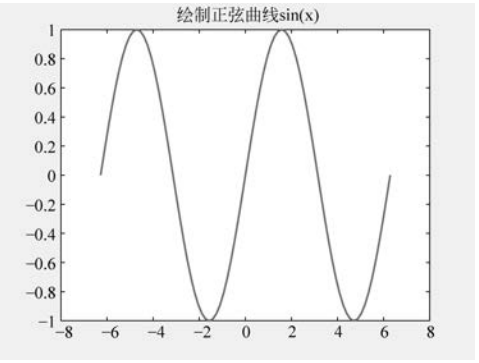

图 5-5 绘制的图像


视频讲解

5.2.2 循环控制

循环(loop)可以将一种运算不断地重复。

循环结构能够重复执行某一段相同的语句。MATLAB 提供了两种循环语句：for 循环和 while 循环。如果已知循环次数，通常用 for 循环语句；如果未知循环次数，但有循环条件，则用 while 循环语句。

1. for 循环语句

for 循环在进行指定次数的重复动作之后停止。for 循环流程图如图 5-6 所示。

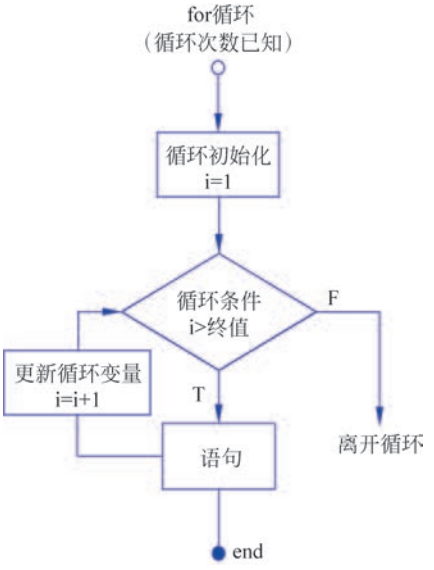

图 5-6 for 循环流程图

for 循环语句语法：

```
for 循环变量 = 初值:步长:终值
运算式
end
```

其中：初值、步长、终值可以取整数、小数、正数和负数，步长默认值为 1。当步长为正数时，表示循环变量大于终值时停止执行介于 for 和 end 之间的运算式。

若要跳出 for 循环，可用 break 指令。

【例 5-8】 用 for 循环语句编程计算 1+2+3+4+5+…+100。

程序如下：

```
sum = 0;
for i = 1:100;
    sum = sum + i;
end
fprintf('sum = %5.0f\n',sum)
```

运行结果：

```
sum =  5050
```

2. while 循环语句

while 循环语句可完成不定次数重复的循环，它与 for 语句不同，每次循环前要判别其条件，如果条件为真或非零值，则循环；否则结束循环。

while 循环语句语法：

```
while 条件式
运算式
end
```

当表达式为真时，反复执行语句体内的语句，直到表达式的条件为假时退出语句体循环。while 和 end 必须配对使用。若要跳出循环，可用 break 指令。while 循环流程图如

图 5-7 所示。

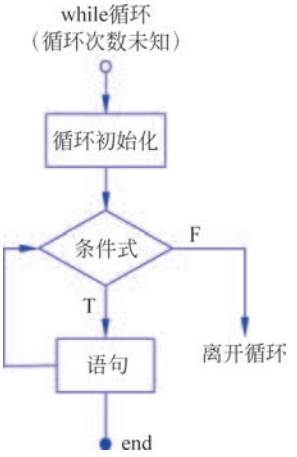

图 5-7 while 循环流程图

【例 5-9】 用 while 循环语句编程计算 1＋2＋3＋4＋5＋…＋n＞5050 最小之 n 值。

程序如下：

```
sum = 0;
n = 0;
while sum < = 5050
    n = n + 1;
    sum = sum + n;
end
fprintf('1 + 2 + ... + n > 5050 最小之 n 值 = %3.0f\n',n)
```

运行结果：

```
1 + 2 + ... + n > 5050 最小之 n 值 =  100
```

【例 5-10】 编程计算 1＋2＋…＋n＞50 最小之 n 值。

程序如下：

```
sum = 0;
n = 0;
while sum < = 50
    n = n + 1;
    sum = sum + n;
end
fprintf('1 + 2 + ... + n > 50 最小之 n 值 = %3.0f\n',n)
```

运行结果：

```
1 + 2 + ... + n > 50 最小之 n 值 =  10
```

3. 嵌套循环

嵌套循环(nested loop)中嵌套的 for 循环语句的语法如下：

```
for m = 1:j
for n = 1:k
  运算式
end
 :
 :
end
```

MATLAB 中嵌套 while 循环语句的语法如下：

```
while 表达式 1
while 表达式 2
  语句
end
 :
 :
end
```

嵌套结构的循环可以是多层的。

【例 5-11】 用嵌套的循环语句编写程序,创建 5 行 5 列的矩阵,其每一元素值按照行的平方与列的平方之和的规律得出。

程序如下:

```
for n = 1:5
    for m = 1:5
    A(n,m) = n^2 + m^2;
    end
    disp(n)
end
```

运行结果:

```
1
2
3
4
5
键入 A
A =
     2     5    10    17    26
     5     8    13    20    29
    10    13    18    25    34
    17    20    25    32    41
    26    29    34    41    50
```

4. 程序控制其他常用指令

程序控制其他常用指令有 break、continue、end、pause、return 等,如表 5-3 所示。

表 5-3 程序控制其他常用指令

命 令	描 述
break	终止执行 for 循环和 while 循环
continue	将控制传递给 for 循环或 while 循环的下一个迭代
end	终止代码块
pause	暂时停止执行
return	将控制权返回给调用函数

1) 循环控制

循环控制语句主要有 break 语句和 continue 语句。

break 语句用来终止 for 循环或 while 循环的执行。在循环中 break 语句之后出现的语句不执行;如果 break 命令用于嵌套循环的内部循环,那么只能终止内部循环,外部循环仍然继续。

以 for 循环为例,描述循环控制 break 语句的语法如下:

```
for (语句)
      {
       …
       if (条件)          break ;
       …
       }
end
```

执行过程：该循环结构的执行由循环控制条件“语句”控制，当“语句”为假时，循环结束；但在执行过程中，如果“条件”为真，则执行 break 语句，此时也会终止循环。

break 语句流程图如图 5-8 所示。

【例 5-12】 用 for 循环 100 次执行 a=a+2，如果 a 达到 100，那么就把 a 除 2，然后结束 for 循环。

程序如下：

```
a = 1;
for n = 1:100
  a = a + 2;
  if a > 100
     a = a/2;
     break
  end
end
```

for 循环 100 次执行 a=a+2，就是每执行一次加 2，使用 if 语句进行条件判断，如果 a 达到 100，那么就把 a 除 2，break 语句自动结束 for 循环，然后执行 end 下面的语句，不再执行 for 循环。

在编程中应尽量避免使用 break 命令，因为使用 break 命令的程序通常不易理解和维护，通常做法是将程序改写成不用 break 语句。

continue 语句有点像 break 语句，然而 continue 语句不是强制终止，而是迫使循环的下一次迭代发生，跳过其间的任何代码。continue 语句流程图如图 5-9 所示。

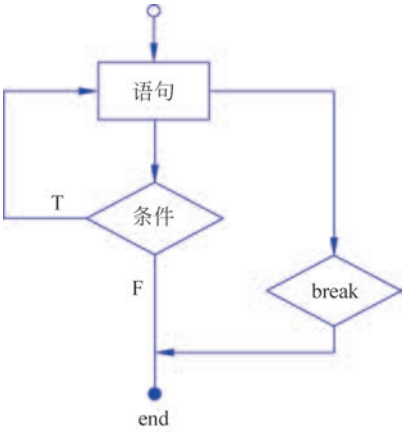

图 5-8 break 语句流程图

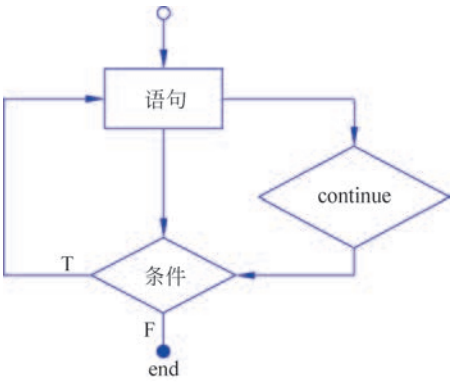

图 5-9 continue 语句流程图

continue 语句控制跳过循环体的某些语句。当在循环体内执行到该语句时，程序将跳过循环体中所剩下的语句，继续下一次循环。

【例 5-13】 用 for 循环 100 次执行 a=a+2，如果 a 小于或等于 100，那么就执行 a=a+2，如果大于 100，就执行 a=a/2 命令，然后退出。

按照之前的 for 循环例子，使用 continue 和 break 再来一次，结果一样。

程序如下：

```
a = 1;
for n = 1:100
```

```
    a = a + 2;
    if a < = 100
       continue
    end
    a = a/2;
    break
end
```

如果 a 小于或等于 100,那么就执行 continue,不执行 break 和 a=a/2,相当于如果大于 100,就执行 break 和 a=a/2 命令,如果大于 100,就不执行 continue,不恢复到检测位置,就执行 a=a/2 和 break 命令,结果和只使用 break 语句一样,那么变量 n 可以用来查看运行次数。

break 和 continue 的区别:break 直接结束循环;continue 进入下一次循环。

2) 其他常用指令

其他常用指令有 return 语句和 pause 语句。

在编写 MATLAB 程序过程中,有时会遇到当程序运行到不满足 if 条件时让程序跳出,停止运行的情况,在 MATLAB 中,使用 return 语句可实现程序跳出。

【例 5-14】 定义一个变量和一个标准量,判断变量和标准量是否相等,如果相等,在命令行窗口中打印 0;如果不相等,在命令行窗口中打印 2。

程序如下:

```
a = 1;                              % 定义一个变量 a
flag = 1;                           % 定义一个标准量
if 1
   if flag == a;                    % 判断 a 与 flag 是否相等
      disp('0');                    % 如果相等,打印 0
      return;                       % 不再向下执行
      disp('1');                    % return 后的语句不执行
   else
      disp('2');                    % 如果不相等,打印 2
   end
else
   disp('3');                       % 外层 if 对应的 else,打印 3
end
disp('4');                          % 打印 4
```

运行结果:

```
0
4
```


视频讲解

5.2.3 分支结构

分支结构(branching)需要进行判断,只有满足一定条件时才执行某些语句。在 MATLAB 中,分支结构有两类:if 语句和 switch 语句。分支结构常用的命令语句如表 5-4 所示。

表 5-4 分支结构常用的命令语句

语　句	描　述
if…end 语句	if…end 语句由一个 if 语句和一个条件表达式组成,后跟一个或多个语句
if…else…end 语句	if 语句后面可以跟一个可选的 else 语句,该语句在条件表达式为 false 时执行

续表

语　　句	描　　述
if…elseif …elseif …else…end 语句	if 语句后面可以跟一个(或多个)可选的 elseif…还有一个 else 语句,它对测试各种条件非常有用
嵌套 if 语句	可以在另一个 if 或 elseif 语句中使用一个 if 或 elseif 语句
switch 语句	switch 语句允许根据值列表测试变量是否相等
嵌套 switch 语句	可以在另一个 switch 语句中使用一个 switch 语句

1. 单分支 if…end 语句

if…end 语句的语法：

```
if  <条件>
 % 如果条件表达式为真,则执行语句
<语句>
end
```

if…end 语句流程图如图 5-10 所示。

如果表达式的计算结果为 true,则将执行 if 语句中的代码块。如果表达式的计算结果为 false,则将执行 end 语句之后的第一组代码。

2. 双分支 if…else…end 语句

if…else…end 语句的语法：

```
if <条件>
 % 如果条件表达式为真,则执行语句 1
<语句 1 >
else
<语句 2 >
 % 如果条件表达式为假,则将执行语句 2
end
```

当条件成立时,执行语句 1,否则执行语句 2,语句 1 或语句 2 执行后,再执行 if 语句的后继语句。若不需使用语句 2,则可直接省略 else 和语句 2。

if…else…end 语句流程图如图 5-11 所示。

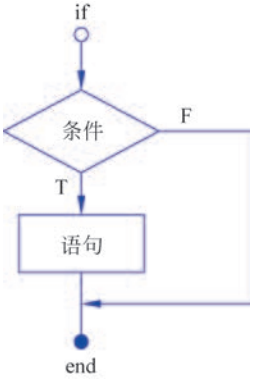

图 5-10　if…end 语句流程图

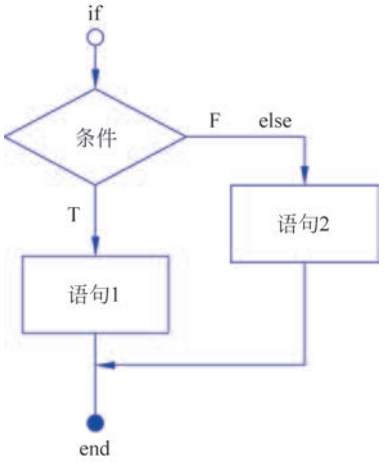

图 5-11　if…else…end 语句流程图

【例 5-15】 根据向量 y 的元素值为奇数还是偶数,来显示不同的信息。

程序代码如下:

```
y = [0 3 4 1 6];
for i = 1:length(y)
if rem(y(i),2) == 0
fprintf('y( % d) = % d is even.\n',i,y(i));
else
fprintf('y( % d) = % d is odd.\n',i,y(i));
end
end
```

运行结果:

```
y(1) = 0 is even.
y(2) = 3 is odd.
y(3) = 4 is even.
y(4) = 1 is odd.
y(5) = 6 is even.
```

上述的 if…else…end 为双分支条件,即只执行语句 1 或语句 2,不会有第三种可能。

3. 多分支 if…elseif…else…end 语句

if…elseif…else…end 语句用于实现多分支选择结构。

if…elseif…else…end 语句的语法:

```
if <条件 1 >
 % 条件 1 为真,则执行语句 1
<语句 1 >
 % 条件 1 为假,条件 2 为真,将执行语句 2
elseif <条件 2 >
<语句 2 >
else
<语句 3 >
 % 条件 2 为假,则将执行语句 3
end
```

if…elseif…else…end 语句流程图如图 5-12 所示。

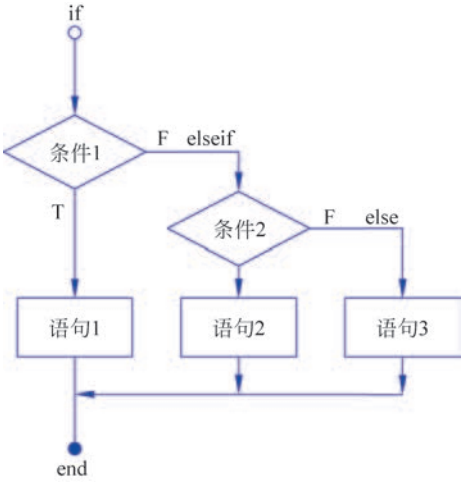

图 5-12 if…elseif…else…end 语句流程图

多分支语句可执行多个条件，若要执行更多的条件，只需不断重复 elseif 即可。

【例 5-16】 已知 $y(x)=\begin{cases} x+1 & x\leqslant 0 \\ 2x+1 & 0<x\leqslant 1 \\ x^2+2x & 1<x\leqslant 2 \end{cases}$，编写程序绘制函数的图像。

程序如下：

```
x = linspace( - 1,2,100);
for i = 1:length(x)
   if x(i)< = 0
      y(i) = x(i) + 1;
   elseif  x(i)< = 1
      y(i) = 2 * x(i) + 1;
   else
      y(i) = x(i)^ + 2 * x(i);
   end
end
plot(x,y)
```

运行程序，绘制的图像如图 5-13 所示。

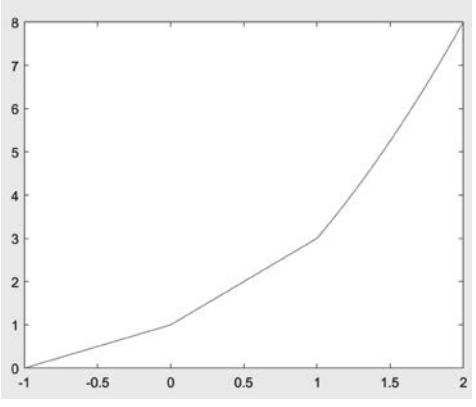

图 5-13 绘制的图像

【例 5-17】 编写程序计算 $f(x)=\begin{cases} x^3+5 & x\geqslant 0 \\ -x^3+5 & x<0 \end{cases}$ 的值，其中 x 的值为 $-10\sim 10$ 范围内，以 0.5 为步长。

程序如下：

```
x1 = 0;
for x = - 10:0.5:10
x1 = x1 + 1;
if x < 0
f(x1) = - x^3 + 5;
else
f(x1) = x^3 + 5;
end
end
```

运行结果如图 5-14 所示。

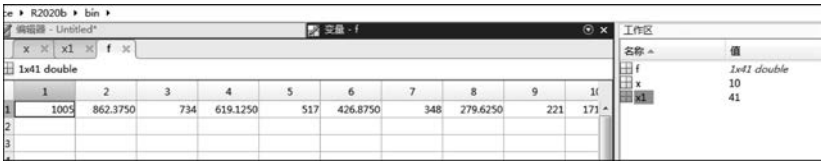

图 5-14　运行结果

4. 嵌套的 if 语句

嵌套 if 语句(nested if)的语法：

```
if <条件 1>
 % 条件 1 为真,则执行语句 1
<语句 1>
 % 条件 1 为假,条件 2 为真,则将执行语句 2
if <条件 2>
<语句 2>
end
<语句 3>
  % 条件 2 为假,则将执行语句 3
end
```

嵌套 if 语句流程图如图 5-15 所示。

嵌套 if 语句也可以嵌套 elseif…else,就像嵌套 if 语句一样。

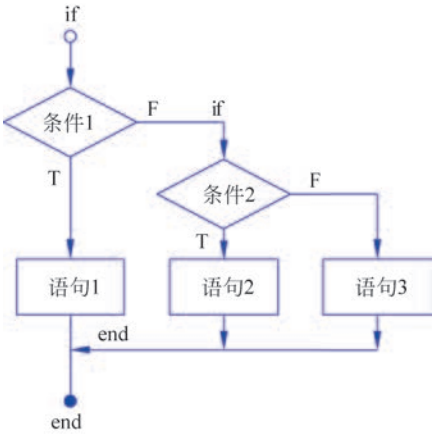

图 5-15　嵌套 if 语句流程图

【例 5-18】 判断是否为酒后驾车。如果规定车辆驾驶员的血液酒精含量小于 20mg/100ml 不构成酒驾；酒精含量大于或等于 20mg/100ml 为酒驾；酒精含量大于或等于 80mg/100ml 为醉驾。编写 MATLAB 程序判断是否为酒后驾车。

根据题意,判断是否为酒后驾车程序流程图如图 5-16 所示。

根据流程图,编写程序代码如下：

```
proof = input("输入驾驶员每 100ml 血液酒精的含量:")
if (proof < 20)
  fprinft("驾驶员不构成酒驾")
else
  if (proof < 80)
     fprintf("驾驶员已构成酒驾")
  else
     fprintf("驾驶员已构成醉驾")
  end
  end
```

5. switch 语句

switch 语句比 if…else 语句更方便,switch 语句可从多个选择中有条件地执行一组语句,每个选择都包含在 case 语句中。

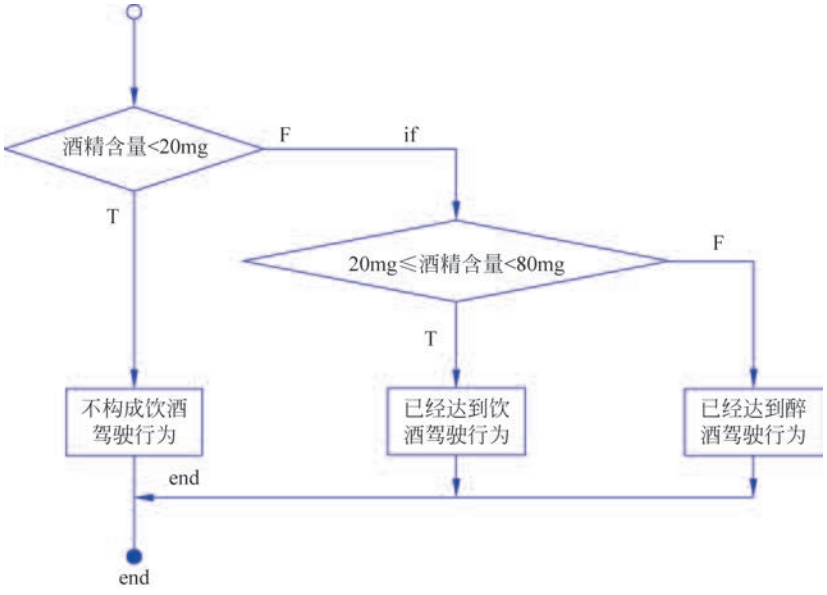

图 5-16 判断是否为酒后驾车程序流程图

switch 语句的一般形式为：

```
switch 分支条件(数值或字符串)
   case 数值(或字符串)条件 1
        语句 1
   case 数值(或字符串)条件 2
        语句 2
   case 数值(或字符串)条件 3
        语句 3
   case …
        …
   otherwise
        语句
   end
```

分支条件可以是一个函数、变量或者表达式。如果条件 1 与分支条件匹配就执行语句 1，并跳出 switch 语句；否则，检验条件 2，如果条件 2 与分支条件匹配执行语句 2，并跳出 switch 语句；否则，检验条件 3，…，当所有条件都不与分支条件匹配时就执行最后的语句。注意 otherwise 是可以省略的。

switch 语句程序流程图如图 5-17 所示。

【例 5-19】 编制根据月份判断季节的程序。

程序如下：

```
for month = 1:12
  switch month
   case {3,4,5}
        season = 'Spring';
   case {6,7,8}
        season = 'Summer';
   case {9,10,11}
```

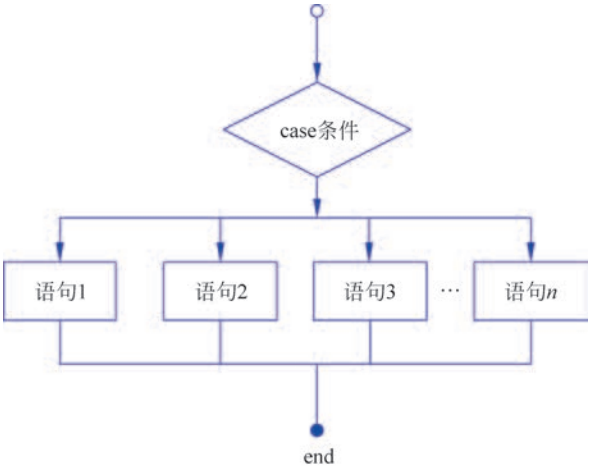

图 5-17　switch 语句程序流程图

```
        season = 'Autumn';
    case {12,1,2}
        season = 'Winter';
  end
   fprintf('Month %g ===> %s.\n',month,season);
end
```

运行结果：

```
Month 1 ===> Winter.
Month 2 ===> Winter.
Month 3 ===> Spring.
Month 4 ===> Spring.
Month 5 ===> Spring.
Month 6 ===> Summer.
Month 7 ===> Summer.
Month 8 ===> Summer.
Month 9 ===> Autumn.
Month 10 ===> Autumn.
Month 11 ===> Autumn.
Month 12 ===> Winter.
```

6. 嵌套 switch 语句

嵌套 switch 语句(nested switch)的语法：

```
switch(ch1)
  case 'A'
    fprintf('This A is part of outer switch');
    switch(ch2)
      case 'A'
      fprintf('This A is part of inner switch');

      case 'B'
      fprintf('This B is part of inner switch');
    end
```

```
  case 'B'
    fprintf('This B is part of outer switch');
end
```

【例 5-20】 使用 switch 语句实现成绩等级判断。

程序如下：

```
grade = 'A';
switch(grade)
case 'A'
   score = 'Excellent!';
   fprintf('Score ===> %s.\n',score);
case 'B'
  score = 'Well done';
  fprintf('Score ===> %s.\n',score);
case 'C'
  score = 'Well done';
  fprintf('Score ===> %s.\n',score);
case 'D'
  score = 'You passed';
  fprintf('Score ===> %s.\n',score);
case 'F'
  score = 'Better try again';
  fprintf('Score ===> %s.\n',score);
   otherwise
  score = 'Invalid grade';
  fprintf('Score ===> %s.\n',score);
end
```

运行结果：

```
Score ===> Excellent!.
```

5.3 自定义函数

读者若想自己建立函数，只需模仿 MATLAB 内建函数进行构建即可。

MATLAB 自定义函数(user-defined functions)的构建方式有七种，如表 5-5 所示。

表 5-5 自定义函数构建方式

方 式	特 点
命令文件/函数文件＋函数文件	多个 M 文件
函数文件＋子函数	一个 M 文件
inline＋命令文件/函数文件	无须 M 文件
符号表达式 syms＋subs 方式	无须 M 文件
字符串＋subs 方式	无须 M 文件
匿名函数	无须 M 文件
直接通过“@”符号定义	无须 M 文件

(1) 方式一：命令文件/函数文件＋函数文件。

命令文件/函数文件中调用函数时要注意实参与形参的匹配。被调用函数的函数名与

文件名必须一致。

【例 5-21】 编程计算 x 的取值从 1 到 10 时 $x^{1/3}$ 的值。

程序如下：

```
%命令文件/函数文件:myfile.m
clear
for t=1:10
   y=mylfg(t)
   fprintf('%4d^(1/3) = %6.4f\n',t,y);
end
```

其中函数文件“mylfg.m”为：

```
%函数文件:mylfg.m
function y=mylfg(x)
y=x^(1/3);
```

运行主函数 myfile 后结果为：

```
>> myfile
y =
     1
   1^(1/3) = 1.0000
y =
    1.2599
   2^(1/3) = 1.2599
y =
    1.4422
   3^(1/3) = 1.4422
y =
    1.5874
   4^(1/3) = 1.5874
y =
    1.7100
   5^(1/3) = 1.7100
y =
    1.8171
   6^(1/3) = 1.8171
y =
    1.9129
   7^(1/3) = 1.9129
y =
     2
   8^(1/3) = 2.0000
y =
    2.0801
   9^(1/3) = 2.0801
y =
    2.1544
  10^(1/3) = 2.1544
```

（2）方式二：函数文件＋子函数。

【例 5-22】 编程计算 x 的取值从 1 到 10 时 $x^{1/3}$ 的值。

程序如下：

```
function [ ] = funtry2()
for t = 1:10
    y = lfg2(t)
    fprintf('%4d^(1/3) = %6.4f\n',t,y);
end
```

函数文件为：

```
function y = lfg2(x)
 y = x^(1/3);
```

运行主函数 funtry2：

```
>> funtry2
```

结果与例 5-21 相同。

（3）方式三：inline＋命令文件/函数文件。

对于简单的数学函数，可用 inline 命令。inline 命令可以用来定义一个内联函数。

函数格式：

```
f = inline('函数表达式','变量 1','变量 2',...)
```

调用方式：

```
y = inline(数值列表)
```

代入的数值列表顺序应与定义时的变量名顺序一致。

【例 5-23】 编程计算 x＝2，y＝3 时，f(x,y)＝x^2＋y 的值。

程序如下：

```
f = inline('x^2 + y','x','y')
z = f(2,3)
```

运行结果：

```
f =
     内联函数:
     f(x,y) = x^2 + y
z =
     7
```

这种函数定义方式是将 inline 中的函数表达式作为一个内部函数调用。其优点：基于 MATLAB 的数值运算内核，所以运算速度较快，程序效率更高。其缺点：该方法只能对数值进行代入，不支持符号代入，且对定义后的函数不能进行求导等符号运算。

(4) 方式四：syms+subs。

syms 定义一个符号表达式，用 subs 命令完成调用。

调用方法：

```
subs = (f,'x',代替 x 的数值或符号)
```

【例 5-24】 编程计算 $f(x)=\frac{1}{1+x^3}$ 在 x=2 的值。

程序如下：

```
syms x                                  %定义符号
f = 1/(1 + x^3);                        %定义一个符号表达式
subs(f,'x',2)
```

运行结果：

```
ans =
 1/9
```

这种函数定义方法的一个特点是可以用符号进行替换。

```
syms f x y                              %定义符号
f = 1/(1 + x^3);                        %定义一个符号表达式
subs(f,'x','y^2')
```

该方法的缺点也很明显，由于使用符号运算内核，运算速度会大大降低。

(5) 方式五：字符串+subs。

直接定义一个字符串，用 subs 命令完成调用。

【例 5-25】 编程计算 $f(x)=\frac{1}{1+x^3}$ 在 x=2 的值。

程序如下：

```
f = '1/(1 + x^3)';
subs(f,'x',2)
```

运行结果：

```
ans =
 1/9
```

(6) 方式六：匿名函数。

匿名函数(anonymous functions)，即使用 MATLAB 函数句柄操作符“@”，可以定义指向 MATLAB 内置函数和用户自定义函数的函数句柄，函数句柄也可以像函数一样使用。

【例 5-26】 将 cos 函数和 sin 函数组成 cell，编程用匿名函数绘制 sin(x)的图像。

程序如下：

```
x = -pi:0.1:pi;
fh = {@cos,@sin};        %fh 为两个函数组成的 cell,fh{1}表示 cos 函数,fh{2}表示 sin 函数
```

```
fh
plot(fh{2}(x))                          % fh{2}(x)即为 sin(x)
```

运行结果：

```
fh =
  1×2 cell 数组
    {@cos}    {@sin}
```

用匿名函数绘制 sin(x)的图像如图 5-18 所示。

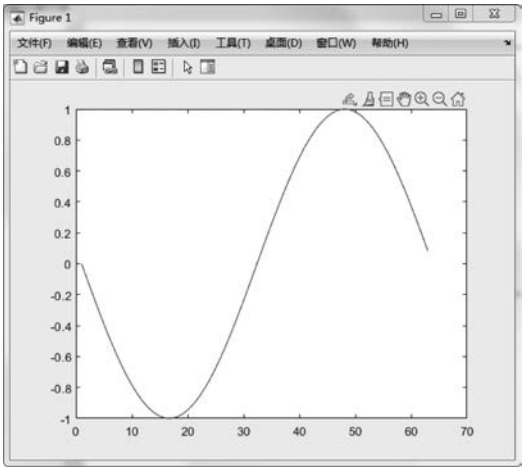

图 5-18 用匿名函数绘制 sin(x)的图像

(7) 方式七：直接通过@符号定义。

【例 5-27】 编程计算 $f(x,y)=x^2-\sin(y)$在 x=2,y=3 的值。

程序如下：

```
f = @(x,y)(x.^2 - sin(y));
f(2,3)
```

执行结果：

```
ans =
   3.8589
```

5.4 程序调试


视频讲解

程序调试(debug)在编程中很重要，意思就是找到并去除(de-)程序中的错误(bug)。程序中出现各种各样的问题是不可避免的，对于那些大型应用程序更是如此。因此，掌握调试方法是程序设计者必备的基本素质。

5.4.1 错误类型

一般说来，应用程序的错误有两类，一类是语法错误，另一类是逻辑错误。

(1) 语法错误,是指由变量名的命名不符合 MATLAB 的规则、函数名的误写、函数的调用格式发生错误、标点符号的缺漏等原因而造成的代码违背程序语言规则的错误,如图 5-19 所示。

图 5-19 语法错误

(2) 逻辑错误,是指程序运行后,得到的结果与预期设想的不一致。通常出现逻辑错误的程序都能正常运行,系统不会给出提示信息,所以很难发现。例如在循环过程中没有设置跳出循环的条件,从而导致程序陷入死循环,如图 5-20 所示。

图 5-20 逻辑错误

在 M 文件中输入一个死循环的代码,如图 5-20 所示,for 循环里面的 inf 是一个无穷大数。按下键盘上的“Ctrl+C”组合键,就可以看到 for 循环停止运行,并跳出一个 warning 的蓝色提示语句。

5.4.2 代码内调试

1. 指令调试方法

MATLAB 内置了一系列的调试函数，用于程序执行过程相关的显示、执行中断、断点设置、单点执行操作等。调试函数可以通过在 MATLAB 窗口输入以下指令获得：

```
help debug
 debug List MATLAB debugging functions

   dbstop     - Set breakpoint.                                  % 设置断点
   dbclear    - Remove breakpoint.                               % 清除断点
   dbcont     - Resume execution.                                % 重新执行
   dbdown     - Change local workspace context.                  % 变更本地工作空间上下文
   dbmex      - Enable MEX-file debugging.                       % 使 MEX 文件调试有效
   dbstack    - List who called whom.                            % 列出函数调用关系
   dbstatus   - List all breakpoints.                            % 列出所有断点
   dbstep     - Execute one or more lines.                       % 重新执行
   dbtype     - List file with line numbers.                     % 列出带行号的 M 文件
   dbup       - Change local workspace context.                  % 变更本地工作空间上下文
   dbquit     - Quit debug mode.                                 % 退出调试模式

   When a breakpoint is hit, MATLAB goes into debug mode, the debugger
   window becomes active, and the prompt changes to a K>>.   Any MATLAB
   command is allowed at the prompt.
    To resume program execution, use DBCONT or DBSTEP.
To exit from the debugger use DBQUIT.
```

【例 5-28】 名为 myprog.m 的程序代码如下：

```
clear all;
close all
clc;
x = ones(1,10);
for n = 1:10
  x(n) = x(n) + 1;
end
```

试用指令调试方法进行调试。

解：用设置断点指令 dbstop 进行调试。设置一个断点在 n >= 4 时(对应程序位置为第 6 行)，然后再运行以下程序：

```
dbstop in myprog at 6 if n>=4;
myprog;
6       x(n) = x(n) + 1;
>> K
```

设置断点后运行程序的结果如图 5-21 所示。

MATLAB 调试函数对程序进行调试还有一些不足之处：①不够简便，需要输入过多的调试代码；②不够直观，对具有多重函数调用的大型程序不便使用。

2. 断点调试方法

编辑器不仅是一个文件编辑器，还是一个可视化的调试开发环境。MATLAB 程序调

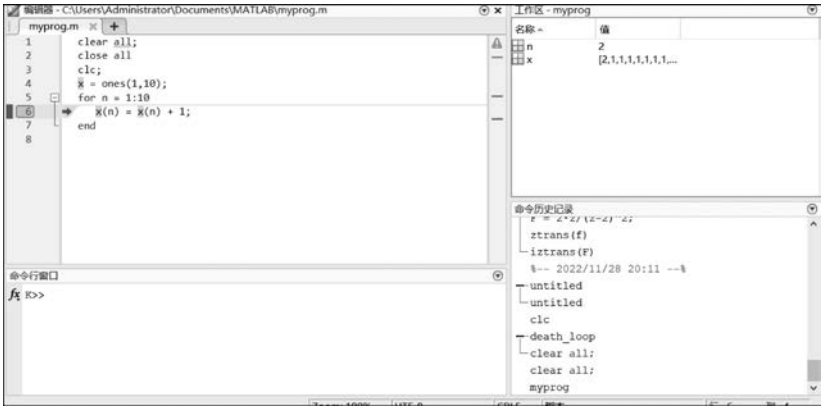

图 5-21　在 n>=4 时设置一个断点运行程序后的结果

试器集成在编辑器之中，十分有助于程序错误的调试，操作控制简单方便，功能异常强大，包括 7 个调试按钮和一个空间堆栈下拉框，图 5-22 所示为编辑器调试区域按钮。

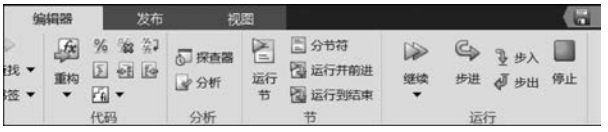

图 5-22　编辑器调试区域按钮

调试代码常用经典的设置断点调试方法，下面给出对应的快捷键。

F12：设置或清除断点。

F5：执行相邻两次断点间的所有指令，如：断点在 for 循环中，按 F5 键一次，循环执行一次。

F10：单步执行。

F11：单步执行，且碰到 function 跳入函数内执行，按 F10 键则不会跳入，这是二者的明显区别。

Shift+F11：跳入 function 之后，通过该指令退出 function。

Shift+F5：退出断点调试。

可以使用键盘快捷方式或在命令行窗口中使用函数来执行大多数调试操作。表 5-6 列出了调试操作以及相关键盘快捷方式和可用于执行这些操作的函数。

表 5-6　调试操作、键盘快捷方式及对应函数

操　作	说　明	键盘快捷方式	函　数
继续	继续运行文件，直到文件末尾或遇到另一个断点	F5	dbcont
步进	运行当前代码行	F10	dbstep
步入	运行当前代码行，如果该行包含对另一个函数的调用，则步入该函数	F11	dbstep in
步出	步入后，运行被调用函数的其余部分，离开被调用函数，然后暂停	Shift+F11	dbstep out
停止	结束调试会话	Shift+F5	dbquit
设置断点	如果不存在断点，则在当前行设置断点	F12	dbstop
清除断点	清除当前行的断点	F12	dbclear

5.4.3 断点调试实例

1. 单个“.m”文件的断点调试

在程序 j_add_array.m 文件中，在把 j 和数组 A 中的三个元素相加代码的左侧，有行数提示，若要对这里的代码进行调试，首先设置断点，直接在左侧的行数位置单击，会出现一个方框，如图 5-23 所示。

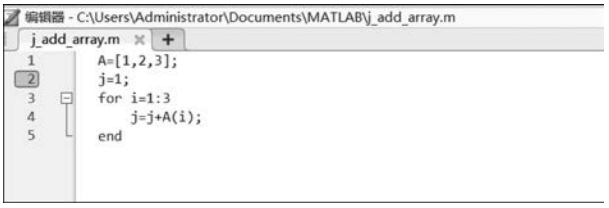

图 5-23 设置断点

设置好断点以后，单击运行就可以进行调试了，如图 5-24 所示。

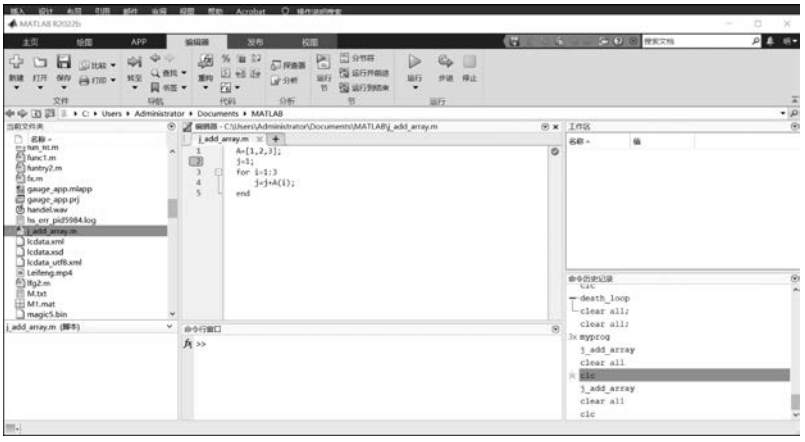

图 5-24 设置好断点以后，单击运行程序

若想一步一步观察运行效果，按 F10 键(步进)。图 5-25 所示为按 F10 键一步一步观察运行的结果。

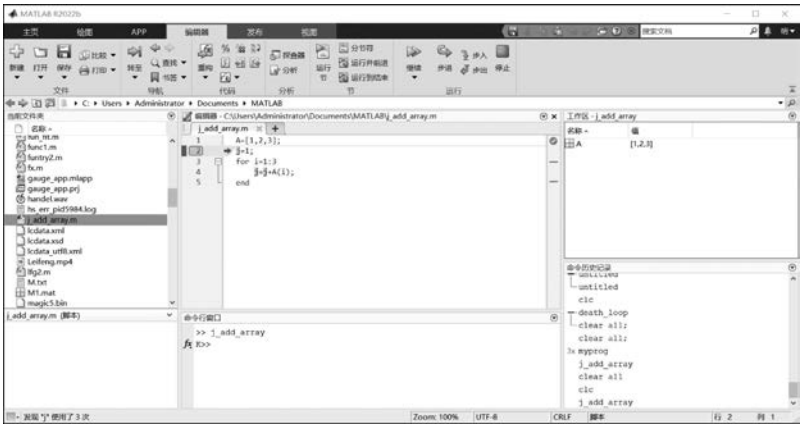

图 5-25 按 F10 键一步一步观察运行结果

如果按 F5 键,则直接结束 for 循环。图 5-26 所示为按 F5 键直接结束循环。

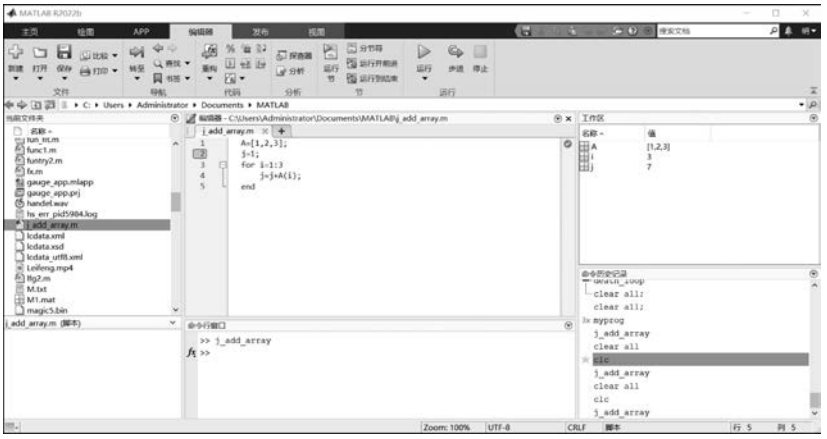

图 5-26　按 F5 键直接结束循环

2. 多个互相调用的“.m”文件的断点调试

在做一个项目或者一个大程序的时候,往往会写一个主函数和多个其他函数,即采用 main 函数+多个 function 函数的形式,在 main 函数中调用这些函数。如果按照前面介绍的在 main 函数中设置断点,按 F10 键是看不到调用其他函数的过程的,但按 F11 键可以看到程序进入子函数的过程。

下面基于实例说明多个互相调用的“.m”文件程序的断点调试。

【例 5-29】　编程求(1*1+2*1+3)+(2*2+2*2+3)+…+(i*i+2*i+3)+(m*m+2*m+3)。

采用函数调用的方式编程,一个函数为主函数 sum1.m,另一个函数为子函数 func1.m。

主函数 sum1.m 程序代码为:

```
function result = sum1(m)              % main 函数
result = 0;
for i = 1:m
    result = result + func1(i);        % 调用 func1 函数
end
```

子函数 func1.m 程序代码为:

```
function p = func1(a)
p = a * a + 2 * a + 3;
```

在实际编程中,可以把 func1 的功能集成到 sum1 中,将程序修改为:

```
function result = sum2(m)
result = 0;
for i = 1:10
    result = result + (i * 1 + 2 * i + 3);
end
```

在 sum1.m 中 result=0 处设置断点。

在命令行窗口中输入 a=sum1(5)后按 Enter 键,会出现如图 5-27 所示窗口,说明进入到单步调试了。

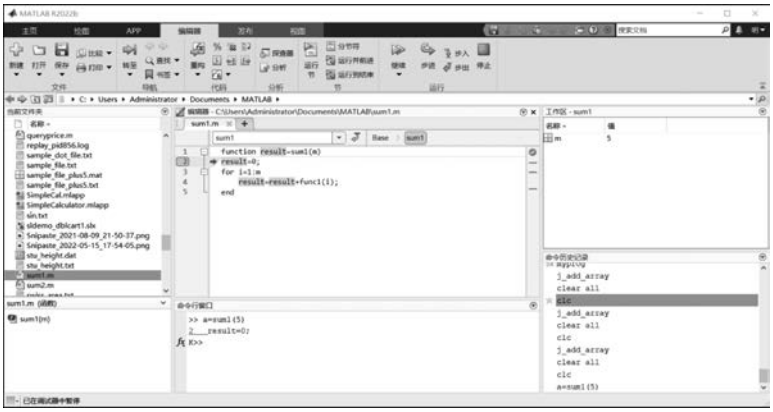

图 5-27 程序进入单步调试

这时 i 还没有被赋值，即循环语句 for i=1:m 还没有执行。

每按一次 F10 键，i、m 或 result 的值都会有变化，如图 5-28 所示。

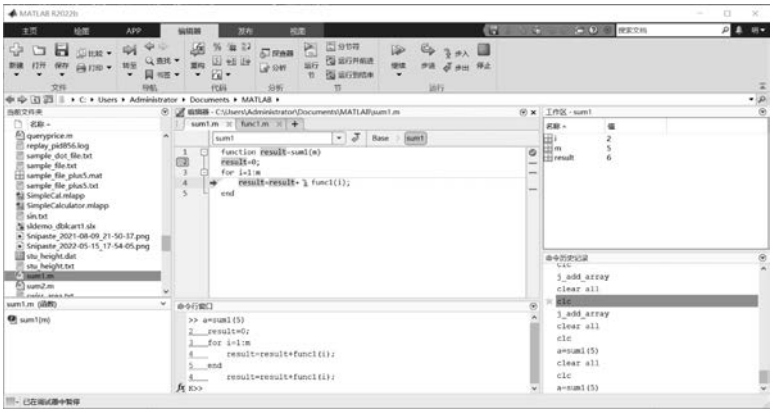

图 5-28 按 F10 键（步进）

当箭头指向“result=result+func1(i);”时，该语句还没有被执行，因 result 为 6，不等于 17，这时可通过观察 result 的值来判断该语句是否被执行。此时，不按 F10 键而按 F11 键，就能看到程序进入子函数的过程，转到如图 5-29 所示界面。如果像前面介绍的那样直接单击步进，是看不到具体过程的。

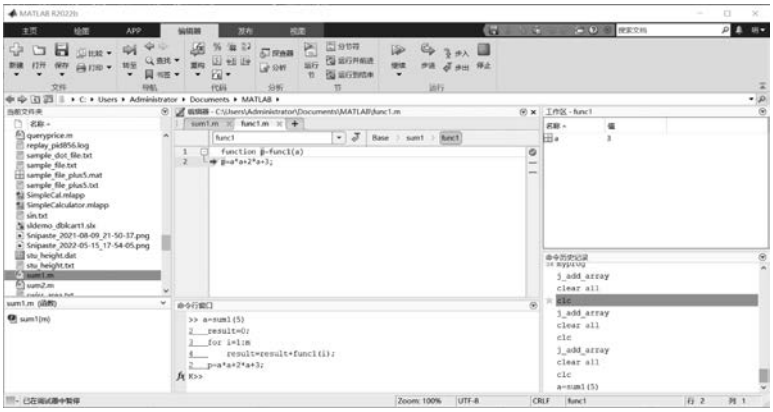

图 5-29 按 F11 键

按 F5 键程序运行结束，如图 5-30 所示。

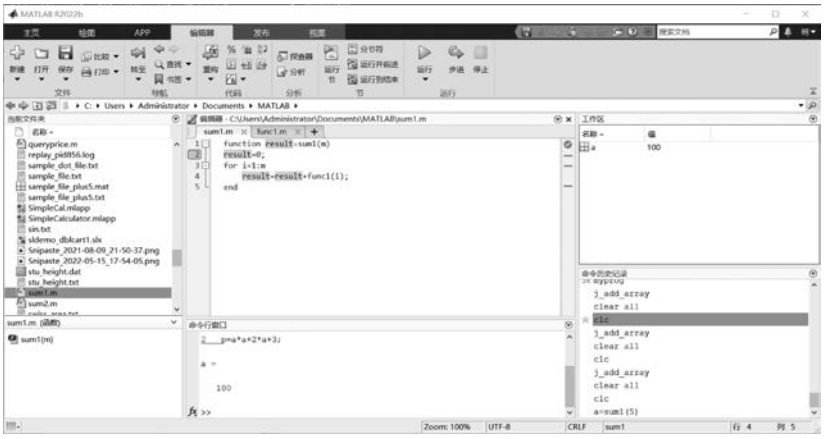

图 5-30　按 F5 键程序运行结束

本章小结

MATLAB 作为一种高级计算机语言，不仅可以用命令行方式完成操作，还具有数据结构、控制流、输入/输出等能力。本章主要分析了 MATLAB 编程中的基本概念，包括脚本、函数和控制流等。通过本章的学习，熟悉和掌握 M 文件的建立与使用方法、函数与控制程序流程的使用，具备一定的编程和程序调试能力。

【思政元素融入】

通过本章的学习，思维能力、编程能力以及独立思考能力都会得到很大的提升。循环结构作为重要的学习内容，涉及很多程序设计方法，在讲述程序实例后引出程序设计的四点感悟：识大局、拘小节、懂规矩、强能力。从程序设计的基本素养来讲述，进而引申到做人做事上，引导学生在实际生活和工作中也要识大局，注重细节，注重良好习惯的养成，做到懂规矩、守纪律，努力学习，不断提高自己的能力。