6 5 4 9 8 7	
>> triu(B) ans = 1 2 3 0 5 6 0 0 9	**ملاحظـة مهمـة جـداً:** يسـتخـدم الايعـاز triu لاستخلاص الجزء المثلية العليا.
>> tril(B) ans = 1 0 0 4 5 0 7 8 9	**ملاحظـة مهمـة جـداً:** يسـتخـدم الايعـاز tril لاستخلاص الجزء المثلية السفلى.
>> det(B) ans = 6.6613e-16	**ملاحظة مهمة جداً:** يستخدم الايعاز det لحساب محدد المصفوفة (قيمة).
>> inv(B) ans = 1.0e+16 -0.4504 0.9007 -0.4504 0.9007 -1.8014 0.9007 -0.4504 0.9007 -0.4504	**ملاحظة مهمة جداً:** يستخدم الايعاز inv لحساب معكوس المصفوفة (قيمة).
>> eig(B) ans = 16.1168 -1.1168 -0.0000	**ملاحظة مهمة جداً:** يستخدم الايعاز eig لحساب القيم الذاتية المصفوفة.
>> trace (B) ans = 15	**ملاحظة مهمة جداً:** يستخدم الايعاز trace لحساب مجموع عناصر القطر الرئيسي.
ملاحظة مهمة جداً: إذا أردت أن تعرف حجم أو بعد مصـــفوفة أحادية أو ثنائية أو ثلاثية البعد غير معروفين وكنت بحاجة لحجمها لإجراء بعض العمليات الرياضـــية، فان برنامج الماتلاب يمكنك من خلال الايعاز length وsize و numel واليك الأمثلة التالية:	
>> size(B) ans = 3 3 >> [r, c] = size (B)	**ملاحظة مهمة جداً:** يسـتخدم الايعاز size لإيجاد حجم المصـفوفة إذ يعبر العنصـر الأول عن عدد الصـفوف(3) بينما يعطي العنصـر الثاني عدد الأعمدة (3).

r = 3 c = 3	
>> numel(B) ans = 9	**ملاحظة مهمة جداً:** يستخدم الايعاز numel لإيجاد العدد الكلي لعناصر المصفوفة.
>> length(B) ans = 3	**ملاحظة مهمة جداً:** يستخدم الايعاز length لإيجاد عدد العناصر الموجودة ضمن البعد الأطول للمصفوفة.

ملاحظة مهمة: هناك المصفوفات المنطقية الناتجة عن العمليات المنطقية. كما يمكن أيضاً استخدام المصفوفات المنطقية إذا كان حجمها مساويا لحجم المصفوفات المعنونة، ويتم في هذه الحالة الإبقاء على العناصر ذات القيمة (1) أي true وهي العناصر المحققة للشرط بينما يتجاهل العناصر (0) أي false وهي العناصر غير المحققة الشرط. ولنأخذ المثال التالي:

مثال (4):

```
>> x = -3: 3
x =
   -3   -2   -1    0    1    2    3
>> abs (x) > 1
ans =
     1    1    0    0    0    1    1
>> y = x (abs (x) > 1)
y =
    -3   -2    2    3
```

هنا تم إنشاء المصفوفة y من تلك العناصر من المصفوفة x التي قيمتها أكبر من الواحد.

مثال (5): ويمكن العمل مع المصفوفات الثنائية المنطقية كما عملنا مع الأحادية المنطقية، كما في المثال التالي:

```
>> B = [5   -3; 2   -4]
B =
    5   -3
    2   -4
>> x = abs (B) > 2
x =
    1    1
    0    1
```

سابعاً: العمليات الحسابية بين المصفوفة والعدد المفرد:

تجري العديد من العمليات الحسابية كعملية الإضافة والطرح والضرب والقسمة بين العدد المفرد وبين جميع عناصر المصفوفة.

مثال (1):

```
g =
    1    2    3    4
    5    6    7    8
    9   10   11   12
```

ملاحظة: هنا طُرح من كل عنصر من عناصر المصفوفة g العدد 2.

```
>> g-2
ans =
   -1    0    1    2
    3    4    5    6
    7    8    9   10
```

ملاحظة: هنا تم ضرب كل عنصر من عناصر المصفوفة g بالعدد 2، ثم طُرح من كل عنصر من العناصر الناتجة الرقم 1.

```
>> 2 * g – 1
ans =
     1    3    5    7
     9   11   13   15
    17   19   21   23
```

ملاحظة: أما في هذه الحالة، فقد ضُرب كل عنصر من عناصر المصفوفة g بالعدد 2، ثم قُسّم الناتج على العدد 5 وبعدها أُضيف لها الواحد.

```
>> 2 * g / 5 + 1
ans =
    1.4   1.8   2.2   2.6
      3   3.4   3.8   4.2
    4.6     5   5.4   5.8
```

ثامناً: العمليات الحسابية بين المصفوفات:

لا تعتبر العمليات الحسابية بين المصفوفات بسيطة تماماً مثل العمليات الحسابية المجراة بين المصفوفات والأعداد المفردة. وبشكل أوضح، فالعمليات الحسابية المجراة بين مصفوفات مختلفة الأبعاد والحجوم تعد عمليات صعبة التحديد، وتعد العمليات الحسابية على المصفوفات متساوية الأبعاد من جمع وطرح وضرب وقسمة من العمليات الأساسية في لغة الماتلاب واليك الأمثلة التالية:

مثال (1):

```
>> h = [1,1,1,1; 2,2,2,2;3,3,3,3]
h =

     1   1   1   1
     2   2   2   2
     3   3   3   3
```

```
>> g=[1,2,3,4;5,6,7,8;9,10,11,12]
g =

     1   2   3   4
     5   6   7   8
     9  10  11  12
```

```
>> g + h
ans =
     2    3    4    5
     7    8    9   10
    12   13   14   15
```

```
>> ans – h
ans =
     1    2    3    4
     5    6    7    8
     9   10   11   12
```

```
>> 2 * g – h
ans =
     1    3    5    7
     8   10   12   14
    15   17   19   21
```

```
>> 2 * (g – h)
ans =
     0    2    4    6
     6    8   10   12
    12   14   16   18
```

ملاحظة مهمة جداً: لاحظ أيضاً بأن العمليات الحسابية بين المصفوفات تعتمد نفس تسلسل أسبقية العمليات المعتمد عند إجراء العمليات الحسابية على الأعداد المفردة، ويمكن أيضاً استخدام الأقواس لكسر تلك الأولوية. كما ويمكن ضرب كل عنصر بالعنصر المناظر له من المصفوفة الأخرى أو قسمته شرط إن تُسبق إشارة الضرب أو القسمة بنقطة كما في المثال الاتي:

ولقد قمنا هنا بضرب المصفوفة g بالمصفوفة h عنصر بعنصر عبر استخدام إشارة الضرب المسبوقة بنقطة.

```
>> g .* h
ans =
     1    2    3    4
    10   12   14   16
    27   30   33   36
```

كما إن قسمة مصفوفتين عنصراً بعنصر ممكنة عن طريق كتابة إشارة القسمة مسبوقة بنقطة كما في المثال التالي:

ملاحظة: إذا سبقت إحدى إشارة القسمة بنقطة، عندها سيقوم برنامج الماتلاب بتقسيم المصفوفتين عنصرا بعنصر. أما إذا كانت القسمة بدون نقطة، فإننا ستحدد قسمة مصفوفات عادية.

```
>> g ./ h
ans =
    1.0000    2.0000    3.0000    4.0000
    2.5000    3.0000    3.5000    4.0000
    3.0000    3.3333    3.6667    4.0000
```

لقد وجدنا هنا مربع كل عنصـــر من عناصـــر المصفوفة g.

```
>> g .^ 2
ans =
     1    4    9    16
    25   36   49    64
    81  100  121   144
```

ملاحظة مهمة جداً: يجعل وجود النقطة أمام إشـــارة الضـــرب القياســـية برنامج الماتلاب يضـــرب المصفوفتين عنصرا بعنصر، بينما تخبر إشارة الضرب لوحدها البرنامج بان يقوم بضرب مصفوفات عادية.

```
>> g * h
    Error
```

لان عدد الأسطر للمصفوفة g ≠ عدد الأعمدة للمصفوفة h

تاسعاً: البحث عن مصفوفة جزئية:

مثال (1): من المفيد في بعض الأحيان إن تعرف موقع أو دليل العناصـــر التي تحقق شـــرطا معينا، والموجودة ضـــمن مصـــفوفة معينة. يقوم برنامج الماتلاب بتحقيق هذه الغاية عبر الايعاز find، والذي يعيد لك دليل أو موقع العنصر الذي تكون نتيجة تحقيقه لشرط ما true، واليك المثال التالي:

```
>> x = -3: 3
x =
    -3   -2   -1    0    1    2    3
```

```
>> k = find (abs (x) > 1)
 k =              (الموقع)
    1    2    6    7
>> y = x (k)
y =
    -3   -2    2    3
```

```
>> y = x (abs (x) > 1)
y =
    -3   -2    2    3
```

مثال (2): يستطيع الايعاز find أن يعمل في المصفوفات الثنائية البعد أيضا (عمود بعد عمود)، فمثلا:

```
>> A = [1   2   3;   4   5   6;   7   8   9]
A =
    1    2    3
    4    5    6
    7    8    9
```

```
>> [i, j] = find (A > 6)
i =
   3
```

```
   3
   3
j =
   1
   2
   3
```

ملاحظة مهمة جداً: يوفر برنامج الماتلاب الدالتين max ، min الذين يوجدان أكبر وأصغر عنصر في المصفوفة ومواقعهما. ففي حالة المصفوفة الأحادية:

مثال (2):

```
>> v = rand (1, 6)
v =
    0.3046    0.1897    0.1934    0.6822    0.3028    0.5417
```

```
>> max (v)
ans =
       0.6822
```

```
>> [mx, i] = max (v)
mx =
      0.6822
i =
    4
```

```
>> min (v)
ans =
       0.1897
```

```
>> [mn, j] = min (v)
mn =
      0.1897
j =
    2
```

مثال (3): في حالة كون المصفوفة ثنائية البعد:

```
>> A = rand (4, 6)
A =
   0.8147    0.6324    0.9575    0.9572    0.4218    0.6557
   0.9058    0.0975    0.9649    0.4854    0.9157    0.0357
   0.1270    0.2785    0.1576    0.8003    0.7922    0.8491
   0.9134    0.5469    0.9706    0.1419    0.9595    0.9340
```

```
>> [mx, r] = max (A)
```

```
mx =
   0.9134   0.6324   0.9706   0.9572   0.9595   0.9340
r =
    4    1    4    1    4    4
```

لإيجاد أكبر عنصر لكل سطر	`>> max (A')`
	`ans =`
	`0.9575  0.9649  0.8491  0.9706`

لإيجاد أكبر عنصر في مصفوفة ثنائية البعد.	`>> [mmx, i] = max (A (:))`
	`mmx =`
	`0.9706`
	`i =`
	`12`

طريقة أخرى لإيجاد أكبر عنصر في مصفوفة ثنائية البعد.	`>> z = max (max (A))`
	`z =`
	`0.9706`

```
>> [mn, r] = min (A)
mn =
   0.1270   0.0975   0.1576   0.1419   0.4218   0.0357
r =
    3    2    3    4    1    2
```

لإيجاد أصغر عنصر لكل سطر	`>> min (A')`
	`ans =`
	`0.4218  0.0357  0.1270  0.1419`

لإيجاد أصغر عنصر في مصفوفة ثنائية البعد.	`>> [mmn, i] = min (A (:))`
	`mmn =`
	`0.0357`
	`i =`
	`22`

طريقة أخرى لإيجاد أصغر عنصر في مصفوفة ثنائية البعد.	`>> z = min (min (A))`
	`z =`
	`0.0357`

ملاحظة: نفس الشيء لحساب المجموع sum.

```
>> z = sum (sum (A))
z =
  15.3152
```

جمل الإدخال والإخراج والجمل الشرطية

أولاً: **جمل الإدخال**: هناك عدة صيغ للإدخال بالإضافة إلى عملية التنسيب منها **الايعاز input:**

مثال (1):

```
>> x = input ('enter x: ')
enter x:
```

مثال (2): إدخال الأعداد:

```
>> x=input('First Degree is= ');
First Degree is= 55;
>> y=input('Second Degree is= ');
Second Degree is= 75;
>> Average=(x+y)/2
Average =
   65
```

مثال (3): إدخال أسماء رمزية:

```
z = input ('enter name', 's');
```

تستخدم العلامة (' ') للدلالة على ان المتغير هو رمزي وليس عددي

ثانياً: **جمل الإخراج**: هناك عدة صيغ للإخراج منها:

أ- **الايعاز disp:** يجب أن يكون محتويات disp قيمة ذات نوع بياني واحد ضمن الجملة الواحدة (كل جملة نوع بياني واحد).

مثال (1):

```
>> d = 15;
>> disp (d);
15
```

مثال (2):

```
>> a = 'ali';
>> disp (a);
ali
```

مثال (3):

ملاحظة: في حالة كون محتويات disp أكثر من قيمة ذات نوع بيانية مختلفة ضمن الجملة الواحدة (يجب ان تجمع القيم في قوسين كبيرين [ ] .

```
>> sum = 9.8;
>> disp (['sum = ', num2str (sum)]);
sum = 9.8
```

مثال (4):

```
>> disp ('computer');
Computer
```

ج- تعليمة fprintf:

مثال (1):

```
>> y = 1.2;
>> x = 100.5;
>> fprintf ('variable x is % 6.3f\n', x);
>> fprintf ('variable y is % 6.3f\n', y);
variable x is  1.200
variable y is 100.500
```

و هذا يعني بأنه تم حجز 6 مراتب منها 3 مراتب بعد الفارزة العشرية.

مثال (2):

```
>> fprintf ('% 8.3f\n', round (3.8));
    4.000
```

مثال (3):

```
>> x=15000;
>> y=12;
>> fprintf('Numer = %e',x)
Numer = 1.500000e+04
>> fprintf('Numer = %e',y)
Numer = 1.200000e+01
```

ملاحظة مهمة:

يمكن طباعة الأعداد والأســـماء والنتائج من خلال كتابة الايعازات بدون فارزة منقوطة وســـتظهر النتائج في نافذة الأمر Command Window.

الجمل الشرطية

يدعم برنامج الماتلاب العمليات المنطقية والمقارنة مثلما يدعم العمليات الرياضـــية، **وتهدف** العمليات والمعاملات المنطقية الحصول على أجوبة للأسئلة التي يجاب عنها بصح أو خطأ**(True/False) .**
تعتبر لغة الماتلاب في تعاملها مع جميع التعابير المنطقية وعمليات المقارنة إن أي عدد غير صـــفري هو **True** ويعتبر الصفر **False**، كما ويكون إخراج جميع التعابير المنطقية وعمليات المقارنة عبارة عن مصفوفات منطقية تحوي العدد **واحد** من اجل **True** والعدد **صفر** من اجل **False**.
وتعتبر المصفوفات المنطقية نوعاً خاصاً من المصفوفات العددية، كما يمكن عنونة المصفوفة المنطقية بنفس طريقة عنونة باقي المصفوفات التي استخدمها سابقاً ضمن التعابير العددية.

أولاً: معاملات المقارنة: Relational Operators

تتضمن معاملات المقارنة كل الإشارات المقارنة الشائعة والمدرجة في الجدول التالي:

معامل المقارنة	الوصف
<	أصغر من
<=	أصغر أو يساوي
>	أكبر من
>=	أكبر أو يساوي
==	إشارة المساواة (لكي نميزها عن =)
~=	اشارة عدم المساواة

مثال (1): يمكن اســتخدام معاملات المقارنة للمقارنة بين مصـــفوفتين لها نفس الحجم، أو للمقارنة بين مصـــفوفة وعدد مفرد وتتم هذه الحالة مقارنة كل عنصـــر من المصـــفوفة مع العدد المفرد، وتكون المصفوفة الناتجة بنفس حجم المصفوفة التي تمت مقارنتها كما يبينه المثال التالي:

```
>> a = 1; b = 5;
>> x = a > b
x =
    0
```

```
>> A = 1: 9, B = 9 - A
A =
    1   2   3   4   5   6   7   8   9
B =
    8   7   6   5   4   3   2   1   0
>> tf = A > 4
tf =
    0   0   0   0   1   1   1   1   1
```

ملاحظة:
لقد أوجدنا العناصر من A التي هي أكبر من 4، وتظهر الاصفار في المصفوفة الناتجة في مواقع العناصـــر عندما $A \leq 4$، بينما يظهر الرقم 1 عندما $A > 4$.

```
>> tf  = (A == B)
tf =  0   0   0   0   0   0   0   0   0
```

لقد تم هنا إيجاد عناصـــر A التي تســـاوي العناصر في المصفوفة B.

ملاحظة مهمة جداً:
لاحظ بان الإشـارتين (=) و (==) تعنيان شـيئاً مختلفاً، حيث يقوم (==) بمقارنة متغيرين وتعيد العدد واحد إذا كانا متسـاويين وصـفراً إذا لم يكونا متسـاويين، بينما تسـتخدم (=) لإسـناد إخراج العملية إلى متغير.

مثال (1): لتوليد مصـفوفة أحادية منطقية عناصـرها واحدات (في حالة X أكبر من Y) واصـفاراً (في حالة X أصغر من أو تساوي Y).

```
>> X= [10   17   22   0   7   3   2];
>> Y= 7;
>> y = (X > Y)
y =
    1   1   1  0   0   0   0
```

مثال (2): لتوليد مصـفوفة أحادية عناصـرها نفس العناصـر (في حالة X أكبر من Y) واصـفاراً (في حالة X أصغر من أو تساوي Y).

```
>> z = X.* (X>Y)
z =
   10   17    22   0   0   0   0
```

ثانياً: المعاملات المنطقية (العوامل المنطقية) :Logical Operators

توفر المعاملات المنطقية طريقة لدمج أو نفي تعابير المقارنة، ويظهر الجدول التالي المعاملات المنطقية الموجودة في لغة الماتلاب:

المعامل المنطقي	الوصف
&	AND (و)
|	OR (أو)
~	NOT (نفي)

مثال (1):

```
>> a = 1;
>> b = 5;
>> x = a ~= b
x =
    1
```

مثال (2):

يجب ان يتحقق الشرطين معاً

```
>> b = (1 == 1) & (2 ~= 3)
b =
    1
```

يجب ان يتحقق الشرطين معاً

```
>> b = (1==1) | (2 ~= 3)
b =     1
```

يجب ان يتحقق الشرطين معاً	>> b = (1==1) & not ((2 ~= 3)) b = 0
مثال (3):	
>> B = 9 - A B = 8 7 6 5 4 3 2 1 0	>> A=1:9 A = 1 2 3 4 5 6 7 8 9
حيث قام بإيجاد عناصـــر A التي قيمها اكبر من 4.	>> tf = A > 4 tf = 0 0 0 0 1 1 1 1 1
لقد قام البرنامج بقلب النتيجة الســابقة، وتعني استبدال مواقع الاصفار والواحدات.	>> tf = ~ (A > 4) tf = 1 1 1 1 0 0 0 0 0
حيث تعيد هذه العبارة العدد واحد عندما يكون العنصر من A أكبر من 2 واقل من 6.	>> tf = (A > 2) & (A < 6) tf = 0 0 1 1 1 0 0 0 0

ثالثاً: أسبقية المعامل:

يقوم برنامج الماتلاب بإيجاد قيمة تعبير مستنداً إلى مجموعة من القواعد الناظمة لأسبقية المعامل، وتحســب المعاملات ذات الأســبقية العليا قبل المعاملات ذات الأســبقية الدنيا، وتقيم المعاملات ذات الأسبقية المتساوية من اليسار إلى اليمين. ويشرح الجدول التالي قواعد أسبقية المعامل التي يعتمدها البرامج.

المعامل	مستوى الأسبقية
الأقواس ( )	الأعلى
المدور(')، القوة (^، .^)	
إشارة النفي (~)	
الضرب (*، .*)، القسمة (/، ./)	
الجمع (+)، والطرح (-)	
معامل النقطتين المتعامدتين (:)	
أصغر من (<)، وأصغر أو يساوي(=<)، أكبر من (>)، أكبر من أو يساوي (=>)، المساواة (==)، عدم المساواة (=~)	
الجمع المنطقي(&) AND	
المعامل المنطقي (|) OR	الأدنى

رابعاً: **الصيغة** IF-ELSE-END:

قد نحتاج إلى حساب مجموعة من أوامر استناداً إلى إخراج ناتج عن اختبار شرطي. وينفذ هذا الايعاز في لغة الماتلاب عبر استخدام الصيغة **if-else-end** وكما يلي:

1- IF-END:

وستنفذ الأوامر (commands) الواقعة بين العبارتين if و end إذا كانت قيمة التعبير (expression) تكون true.	if expression (commands) end

مثال (1):

```
>> x = 10;
>> if  x == 10
      disp ('ok')
   end;
```

2- IF-ELSE-END: وإذا كان لدنيا خياران، فتصبح الصيغة if-else-end كما يلي:

حيث ستنفذ المجموعة الأولى من الأوامر في حال امتلك التعبير expression القيمة true، بينما تنفذ المجموعة الثانية إذا امتلك التعبير expression القيمة false.	if expression (commands evaluated if True) else (commands evaluated if False) end

3- وإذا كانت هناك عدة حالات، فستأخذ التعبير if-else-end الشكل التالي:

```
if expression1
  (commands evaluated if expression1 is true)
elseif expression2
  (commands evaluated if expression2 is true)
elseif expression3
  (commands evaluated if expression3 is true)
elseif expression4
  (commands evaluated if expression4 is true)
.
.
else
  (commands evaluated if no other expression is true)
End
```

مثال (2):

```
>> x = 10;
>> if  x == 10
  msgbox ('ok', 'result');
```

مثال (3):

```
>> if  x ==  10
      msgbox ('ok', 'result');
   else
      msgbox ('no', 'result');
   end;
```

مثال (4):

```
>> x = 11;
>> if  x == 1
      disp ('1');
   elseif  x == 2
      disp ('2');
   else
      disp ('3');
  end;
```

الإخراج

3

مثال (5): اكتب برنامج لإيجاد ناتج الدوال التالية:

Example: Write a MatLab code for computing the following functions.

$x \geq 2 \ \ and \ \ y = 0.5$	$z = \frac{1}{\sqrt{\frac{2\pi y}{x}}}$
$x = 4$	$z = \frac{e^{-y/2x}}{\ln(y)\sqrt{x}}$
$x \geq 10 \ \ and \ \ y < 8$	$w = \sin(x)$
Else	$z = \sqrt{x} + \sqrt{y}$ $w = lnx - 3lny$

```
x=input('Please input the value of x:');
y=input('Please input the value of y:');
z=0; w=0;
if (x>=2) & (y==0.5)
disp('(x>=2) & (y==0.5)')
z=1/sqrt(2*pi*y/x)
elseif x==4
z=exp(-1*y/(2*x))/(log(y)*sqrt(x))
elseif (x>=10) & (y<8)
w=sin(x)
```

```
else
z=sqrt(x)+sqrt(y)
w=log(x)-3*log(y)
end
```

خامساً: **الصيغة** SWITCH-CASE-OTHERWISE :

عندما يتوجب علينا تنفيذ أوامر اعتماداً على استخدام متكرر لاختيار كمي لوسط ما، عندها من السهل استخدام الصيغة switch-case التي لها الصيغة العامة التالية:

```
switch expression
case test-expression1
    (commands1)
 case test-expression2
    (commands2)
  otherwise
     (commands3)
end
```

يجب أن يكون expression هنا أما عدداً مفرداً أو سلسلة رمزية. يقارن التعبير expression الموجود في الصيغة السابقة بالتعبير test-expression1 الموجود في عبارة case الأولى. وإذا تساوى التعبيران، سيتم تنفيذ الأوامر (commands1) وتخطي التعليمات الواقعة بعدها حتى العبارة end. أما إذا لم يتحقق الشرط الأول، فسيختبر الشرط الثاني، حيث سيقارن expression في المثال السابق مع العبارات test-exoression2 الموجودة في عبارة case الثانية. وإذا تساوى التعبيران، سيتم تنفيذ (commands2) وتهمل بقية العبارات حتى عبارة end. إذا لم تحقق أي عبارة case المساواة مع التعبير expression، عندها ستنفذ الأوامر (commands3) التي تلي العبارة otherwise.

لاحظ من الشرح الذي أوردناه عن صيغة switch-case بأنه سيتم تنفيذ أحدى مجموعات الأوامر المكونة للصيغة switch-case واليك الأمثلة التالية:

مثال (1):

```
x = 1;
switch x
   case {1, 2, 3, 4, 5}
      disp ('1..5');
   case {9, 10}
      disp ('9..10');
   otherwise
      disp ('this is impossible');
end;
```

الإخراج

```
1..5
```

مثال (2):

```
clc;
clear;
```

```
n = 3;
switch n
  case {0}
    m = n + 3;
  case {2}
    m = 'ali';
 case {3}
    m = magic (n);
  otherwise
    disp ('error');
end;
disp (m);
```

الإخراج

```
     8     1     6
     3     5     7
     4     9     2
```

مثال (3):

```
x = 2.7;
units = 'm';
switch units
  case {'inch', 'in'}
    y = x * 2.54;
  case {'meter', 'm'}
    y = x / 100;
  case {'feed', 'ft'}
    y = x * 2.54 / 12;
  case {'millimeter', 'mm'}
    y = x * 10;
  case {'centimeter', 'cm'}
    y = x;
 otherwise
   disp (['Unknown Units:'    units]);
end;
```

الإخراج

```
y = 0.027
```

جمل الدوران والتكرار

توفر لغة الماتلاب مجموعة من جمل الدوران والتكرار وهي:

1- جملة: for تقوم حلقات for بإعادة تنفيذ مجموعة من الأوامر لعدد معين من المرات وبخطوة معينة، وتعطى الصيغة العامة لحلقة (for) كما يلي:

for i = x1: x3: x2 (commands) end;	حيث يعاد تنفيذ الأوامر(commands) الواقعة بين عبارتي for وend من القيمة الابتدائية x1 إلى القيمة النهائية x2 وبزيادة مقدارها x3. كما في المثال التالي:

مثال (1):

ويمكن تفسير هذه الدورة: من اجل كل قيمة لـ n من 1 إلى 5 يجب حساب قيمة العبارة الموجودة حتى عبارة end التالية، تكون قيمة n في الدورة الأولى n = 1، وتكون في الدورة الثانية n = 2 وهكذا حتى تصل إلى n = 5.

```
>> for n=1:5
x(n)=sin(n/10)
end
x =
   0.0998
x =
   0.0998    0.1987
x =
   0.0998    0.1987    0.2955
x =
   0.0998    0.1987    0.2955    0.3894
x =
   0.0998    0.1987    0.2955    0.3894    0.4794
```

ملاحظة:

الفرق بين الادخال الاعتيادي والادخال بصيغة التكرار (for) هو ان الادخال بصيغة التكرار يستند الى دورات متلاحقة كل مرة يثبت قيمة ويبدئ بالأخرى وهكذا...

مثال (2): الادخال الاعتيادي:

```
>> x=1:5;
>> y=sin(x)
y =
   0.8415    0.9093    0.1411   -0.7568   -0.9589
```

مثال (3): الادخال بصيغة التكرار (for) بدلالة (x):

```
>> for x=1:5;
y(x)=sin(x)
```

```
end
y =
   0.8415
y =
   0.8415    0.9093
y =
   0.8415    0.9093    0.1411
y =
   0.8415    0.9093    0.1411   -0.7568
y =
   0.8415    0.9093    0.1411   -0.7568   -0.9589
```

مثال (4): الادخال بصيغة التكرار (for) فقط

```
>> for x=1:5
y=sin(x)
end
y =
   0.8415
y =
   0.9093
y =
   0.1411
y =
  -0.7568
y =
  -0.9589
```

مثال (5): توليد 10 أعداد عشوائية قيمتها (1...10).

سـيأخذ متغير الحلقة n هنا قيماً عشوائية بين (1) و (10) معطاة بالمصفوفة array.

```
>> array = randperm (10)
array =
        8    2    10    7    4    3    6    9    5    1
>> for n = array
      x (n) = sin (n * pi / 10);
    end;
>> x
x =
0.3090    0.5878    0.8090    0.9511    1.0000    0.9511
0.8090    0.5878    0.3090    0.0000
```

أمثلة (6):

```
>> for i = 10: -2: 1
     disp (i);
   end;
```

الإخراج

10
8
4
2

```
>> for i = 0:2:10
     disp (i);
end;
```

الإخراج

0
2
4
6
8
10

```
>> for i = 1: 10
    disp (i);
  end;
```

الإخراج

1
2
3
.
10

مثال (7): جد Factorial 10 (10!):

```
F=1;
for k=1:10;
F=F*k;
end
disp(F)
```

الإخراج

3628800

مثال (8): يمكن إنشاء عدة حلقات for متداخلة، كما في المثال التالي:

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

الإخراج

1
2
3
4
5

```
>> 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
```

مثال (9): طباعة جدول الضرب:

```
>> for i =1: 10
       for j = 1: 10
          mult (i, j) = i * j;
       end;
      end;
        1    2    3    4    5    6    7    8    9    10
        2    4    6    8   10   12   14   16   18   20
        3    6    9   12   15   18   21   24   27   30
        4    8   12   16   20   24   28   32   36   40
        .    .    .    .    .    .    .    .    .    .
        .    .    .    .    .    .    .    .    .    .
       10   20   30   40   50   60   70   80   90  100
```

مثال (10):

```
A=   5   10   15
     20   25   30
     35   40   45
     50   55   60
>> s=0
for r=1:4
for c=1:3
s=s+5;
A(r,c)=s;
end
end
disp(A)
A =
    5   10   15
   20   25   30
   35   40   45
   50   55   60
```

```
A=   0    5   10
     15   20   25
     30   35   40
     45   50   55
>>  s=0
for r=1:4
for c=1:3
A(r,c)=s;
s=s+5;
end
end
disp(A)
A=
     0    5   10
    15   20   25
    30   35   40
    45   50   55
```

```
A=   5    5    5
     5    5    5
     5    5    5
     5    5    5
>> s=0
for r=1:4
for c=1:3
d=s+5;
A(r,c)=d;
end
end
disp(A)
A =
    5    5    5
    5    5    5
    5    5    5
    5    5    5
```

```
A =   4    8   12
     10   14   18
     16   20   24
     22   26   30
```

```
A= 4    8   12   16   20
   18   22   26   30   34
   32   36   40   44   48
   46   50   54   58   62
```

```
A= 3    5    7
   9   11   13
B= 4    6    8
  10   12   14
```