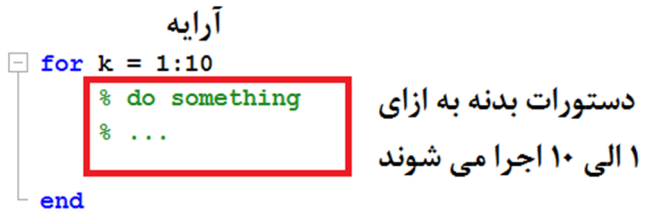
به نام خدا

جزوه جلسه سـوم درس برنامه نویسـی با متلب

1- حلقه‌های تکـــرار for

در نرم‌افزار متلب برای انجام تکراری یک عمل از دستور for استفاده می‌شود در مقابل دستور for یک آرایه به یک متغیر منتسب می‌شود. سپس دستورات داخل حلقه for به ازای تعداد عناصر آرایه اجرا می‌شود و در هر بار اجرا متغیر مقدار عناصر آرایه را به ترتیب اخذ می‌کند. در کد زیر دستورات داخل حلقه for دقیقا 10 بار اجرا می‌شوند در هر اجرا متغیر k یکی از مقادیر 1 الی 10 را به ترتیب اخذ می‌کند.



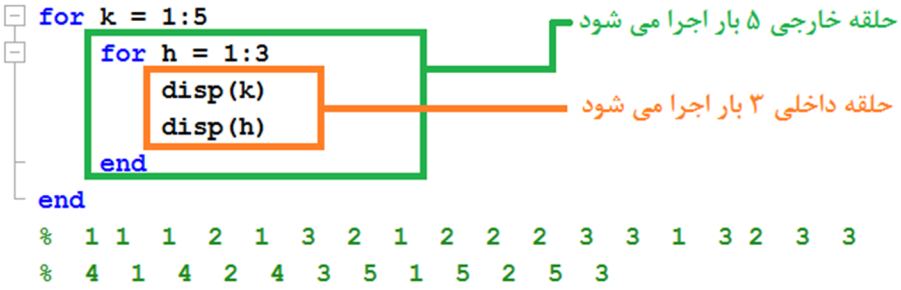
برای ایجاد آرایه از دو نقطه استفاده می‌کنیم.

```
>> k = 1:5

k =

     1     2     3     4     5

>> h = 10:15

h =

    10    11    12    13    14    15
```

```
>> m = -1:5

m =

    -1     0     1     2     3     4     5

>> g = 1:2:10

g =

     1     3     5     7     9
```

```
>> 2:-2:-20

ans =

     2     0    -2    -4    -6    -8   -10   -12   -14   -16   -18   -20

>> 10:7:100

ans =

    10    17    24    31    38    45    52    59    66    73    80    87    94
```

حلقه‌های for می‌تواند تو در تو باشد در این حالت حلقه داخلی مانند یک دستور چند بار اجرا می‌شود در هر بار اجرا دستورات داخلی خود را چند بار اجرا می کند.



مثال شماره 1: کــد زیر یک رشته از اعداد صحیح را تولید می‌کند اعداد این دنباله را بنویسید.

```
for N = 7:-2:2
    for M = 7:4:12
        2*N + M
    end
end
```

جواب : 17 21 13 21 17 25 21

مثال شماره 2 : پس از اجرای کد زیر مقدار s چند خواهد بود ؟ (پاسخ 54)

```
s = 0;
for k=1:2:4
    s = s + 2*k;
    for h=10:-1:9
        s = s + k + h;
    end
end
s
```

مثال شماره 3 : برنامه‌ای بنویسید که یک ماتریس و یک عدد را دریافت کــرده سپس عدد را به تمامی عناصر ماتریس اضافه نماید. (با دو روش)

```
clc
clearvars
A = input('Please enter Matrix A[] ');
N = input('Please enter N ');
%A = A + N;   %the first quick method
[X,Y] = size(A);
for h = 1:X
    for k = 1:Y
        A(h,k) = A(h,k) + N;
    end
end
disp(A);
```

2- حلقه تکرار while

در این حلقه بعد از کلمه while یک شرط قرار می‌گیرد تا زمانی که شرط درست باشد دستورات حلقه اجرا می‌شوند

```
k = 10;
while (k>=1)  شرط حلقه
    disp(k);
    k = k - 1;   دستورات درون حلقه
end
```

بطور معمول درون حلقه یک یا چند متغیر به نحوی تغییر می‌کنند که شرط حلقه پس از اجرای تعداد دلخواه نادرست می‌شود و برنامه از حلقه خارج می‌شود.

مثال شماره 4 : برنامه‌ای بنویسید که یک بردار و یک عدد را دریافت کـــرده و تعیین کند آیا عدد عضو بردار است یا خیر. (از حلقه while استفاده کنید)

```
clc
clearvars
V = input('Please enter vector V[] ');
N = input('Please enter N ');
k = length(V);
flg = 0;
while (k > 0)
    if(V(k)==N)
        disp('N is in V');
        flg = 1;
    end
    k = k - 1;
end

if (flg ~= 1)
     disp('N is not in V');
end
```

هرگاه شرط حلقه همچنان درست باشد اما بخواهیم از حلقه خارج شویم می‌توانیم از دستور break استفاده کنیم در کد زیر به محض آنکه مقدار m برابر با 3 می‌شود برنامه از حلقه خارج می‌شود.

```
k = 10;
m = 0;
while (k > 0)
    disp(k);
    if(m==3)
        break
    end
    m =  m + 1;
    k = k - 1;
end
```

دستور دیگری که در حلقه while مورد استفاده قرار می‌گیرد دستور continue است هرگاه اجرای برنامه به دستور continue برسد بلافاصله بقیه دستورات داخل حلقه while را نادیده گرفته و به ابتدای حلقه while می‌رود. در مثال زیر می‌خواهیم معکوس اعداد 5- تا 5+ را محاسبه نماییم اما می‌خواهیم وقتی به صفر رسیدیم معکوس آن را نشان ندهیم لذا در شرط قرار می دهیم هرگاه مقدار k برابر با صفر شد ابتدا مقدار k را اضافه کن سپس از بقیه دستورات (شامل نمایش معکوس k) صرف نظر کن و حلقه را ادامه بده.

```
clc
clearvars
k = -5;
while (k <= 5)
    if(k==0)
        k = k + 1;
        continue
    end
    disp(1/k)
    k = k + 1;
end
```

برخلاف دستور for که متغیر حلقه خود به خود مقادیر بعدی را اخذ می‌کند در حلقه while مقداردهی به متغیر حلقه بر عهده برنامه‌نویس است.

برای شرط حلقه می‌توانیم چند شرط را ترکیب نماییم در این صورت باید از عملگرهای منطقی AND و OR استفاده نماییم.

عملگرهای منطقی در متلب عبارتند از & برای AND ، | برای OR و ~ برای NOT یا نقیض.

نکته : توصیه می‌شود به جای & از && و به جای | از || استفاده شود در این صورت هرگاه در AND یکی از شروط نادرست شود و یا در OR یکی از شروط درست شود سایر شروط ارزیابی نمی‌شوند.

مثال شماره 5 : پس از اجرای کد زیر مقدار متغیر k را بدست آورید

```
clc
clearvars
a = 12;  b = 8;
k = 1;
while (rem(k,a)~=0) || ~(rem(k,b)==0)
    k = k + 1;
end
disp(k)
```

پاسخ عدد 24 خواهد بود که کوچکترین مضرب مشترک دو عدد a و b است.

نکته با استفاده از دستورات lcm و gcd می‌توانیم کوچکترین مضرب مشترک و بزرگترین مقسوم‌علیه مشترک را محاسبه نماییم