ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΧΗΜΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΙΚΟ ΚΕΝΤΡΟ

Προγραμματισμός και Χρήση Ηλεκτρονικών Υπολογιστών - Βασικά Εργαλεία Λογισμικού

Μάθημα 7ο

Επανάληψη for, εμφωλευμένες(nested) επαναλήψεις
Επανάληψη υπό συνθήκη while, αέναη επανάληψη
Χρονομέτρηση
Προεκχώρηση μνήμης.

Επανάληψη **for**

```
for var = start:step:stop
   εντολές
end
```

Χρησιμοποιείται όταν ένα σύνολο εντολών πρέπει να εκτελεστεί πολλές φορές

Η μεταβλητή *var* ονομάζεται μεταβλητή επανάληψης ή μετρητής

```
var = start:step:stop
```

Η μεταβλητής *var* θα πάρει διαδοχικά τις τιμές που ορίζονται από τη σχέση *start:step:stop*

Επανάληψη **for** – Παράδειγμα 1

```
for k = 1:2:5
  disp('Hello World')
end
```

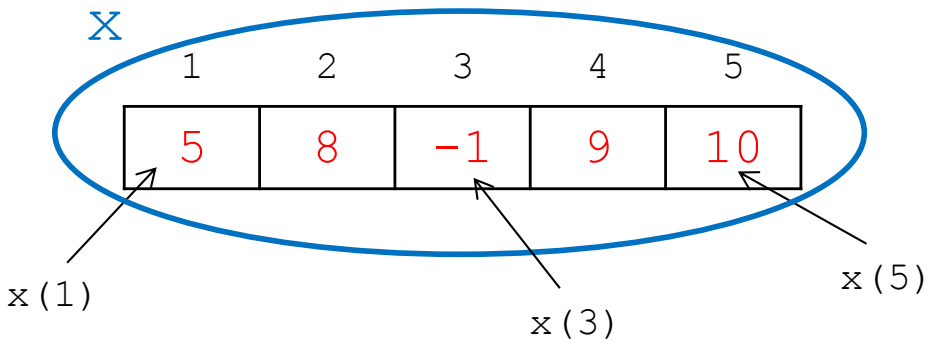
Το αποτέλεσμα της εκτέλεσης της παραπάνω επανάληψης for είναι να τυπώσει 3 φορές στη οθόνη το μήνυμα:

```
Hello World
```

Η μεταβλητή `k` θα πάρει διαδοχικά τις τιμές 1, 3, 5

```
for k = 1:2:5
  disp(k)
end
```

Επανάληψη **for** – Παράδειγμα 2




```
x=[5 8 -1 9 10];
for k = 1:2:5
  disp(x(k))
end
```

```
 5

-1

10
```

Επανάληψη `for` – Άσκηση

Δίνεται το διάνυσμα γραμμής `x=[5 8 -1 9 10]`

1. Χρησιμοποιήστε την επανάληψη for για να δημιουργήσετε ένα νέο διάνυσμα `y` με στοιχεία το 1ο 3ο και 5ο στοιχείο του διανύσματος `x`

2. Δημιουργήστε το διάνυσμα `y` χωρίς τη χρήση της επανάληψης for

Εμφωλευμένες (nested) επαναλήψεις **for**

```
for var1 = start1:step1:stop1
   εντολές
   for var2 = start2:step2:stop2
      εντολές
   end
   εντολές
end
```

Παράδειγμα

```
for k = 1:4
   for m = 1:3
      disp([k m])
   end
end
```

Επανάληψη υπό συνθήκη **while**

```
while συνθήκη
   εντολές
end
```

Χρησιμοποιείται όταν ένα σύνολο εντολών πρέπει να εκτελεστεί πολλές φορές υπό συνθήκη (εφόσον ισχύει μια συνθήκη)

```
x=0;
while x<5
   x=x+1;
   disp(x)
end
```

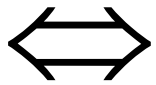
```
 1
 2
 3
 4
 5
```

η συνθήκη (x < 5) ελέγχεται στο τέλος κάθε επανάληψης

Αέναη επανάληψη **while**

Η επανάληψη υπό συνθήκη δεν χρησιμοποιείται στην πράξη με την προηγούμενη μορφή γιατί δεν έχουμε απόλυτο έλεγχο της εξόδου από την επανάληψη, πρέπει δηλαδή να ολοκληρωθούν όλες οι εντολές που βρίσκονται μέσα στην επανάληψη και μετά να τερματιστεί. Για το λόγο αυτό χρησιμοποιείται η παρακάτω μορφή:

```
while true
   εντολές
   if συνθήκη
    break
   end
   εντολές
end
```



```
while true
   εντολές
   if συνθήκη;break;end
   εντολές
end
```

Αέναη επανάληψη **while** - Παράδειγμα

```
x=0;
while true
    x=x+1;
    if x>4;break;end
    disp(x)
end
```

```
1
2
3
4
```

Επανάληψη **`while`** - Άσκηση

Φτιάξτε έναν κώδικα που θα ζητάει συνέχεια από τον χρήστη να εισάγει από το πληκτρολόγιο έναν αριθμό. Η επανάληψη θα τερματίζεται όταν ο χρήστης εισάγει το μηδέν.

1. Χρησιμοποιήστε επανάληψη υπό συνθήκη (**`while`** *συνθήκη*)
2. Χρησιμοποιήστε αέναη επανάληψη (**`while`** `true`)
3. Χρησιμοποιήστε επανάληψη **`for`**

Τροποποιήστε τους κώδικες που φτιάξατε έτσι ώστε μετά το τέλος των επαναλήψεων να τυπώνεται τα άθροισμα των αριθμών που εισήγαγε ο χρήστης.

Άσκηση

Φτιάξτε έναν κώδικα που θα υπολογίζει το άθροισμα των ν πρώτων όρων της σειράς

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\ldots$$

1. Χρησιμοποιήστε επανάληψη **`for`**
2. Χρησιμοποιήστε επανάληψη υπό συνθήκη (**`while`** *συνθήκη*)
3. Χρησιμοποιήστε αέναη επανάληψη (**`while`** `true`)

Χρονομέτρηση - tic/toc

Ενσωματωμένες εντολές που υπολογίζουν το χρόνο εκτέλεσης.

Η εντολή tic ενεργοποιεί το χρονόμετρο.
Η εντολή toc υπολογίζει το συνολικό χρόνο, από τη στιγμή που ενεργοποιήθηκε το χρονόμετρο.

Παράδειγμα

```
tic
mysum = 0;
for k=1:200000
  mysum = mysum + k;
end
toc
```

Τεχνικές Βελτίωσης Απόδοσης Κώδικα
Προεκχώρηση μνήμης (Memory pre-allocation)

```
clear; clc
n= ;  % <-- select the array size

%No memory pre-allocation
disp('No memory pre-allocation for the vector x')
tic
x(1)=1000;
for k=2:n
    x(k)=1.05*x(k-1);
end
toc

%Memory pre-allocation
disp('Memory pre-allocation for the vector y')
y=zeros(n,1);
tic
y(1)=1000;
for k=2:n
    y(k)=1.05*y(k-1);
end
toc
```

n = 1e5	OCTAVE time (s)	MATLAB time (s)
no pre-allocation	1	16
pre-allocation	1	0.002

n = 2e6	OCTAVE time (s)	MATLAB time (s)
no pre-allocation	38	>> 600
pre-allocation	25	0.04

Τεχνικές Βελτίωσης Απόδοσης Κώδικα
Διανυσματοποίηση (Vectorization)

```
clear;clc
n= ;  % <-- select the array size

disp('no vectorization')
x=zeros(n,1);
tic
  for k=1:n
    x(k)=sqrt(k);
  end
toc

disp('vectorization')
tic
  k=1:n;
  y=sqrt(k);
toc
```

n = 1e5	OCTAVE time (s)	MATLAB time (s)
no vectorization	1.4	0.3
vectorization	0.003	0.003

n = 2e6	OCTAVE time (s)	MATLAB time (s)
no vectorization	28	5
vectorization	0.07	0.07