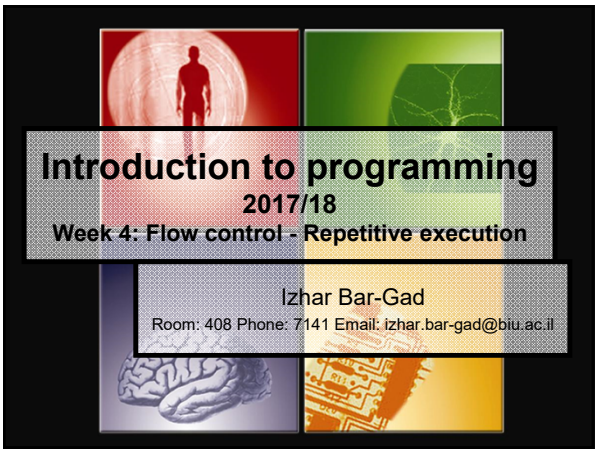
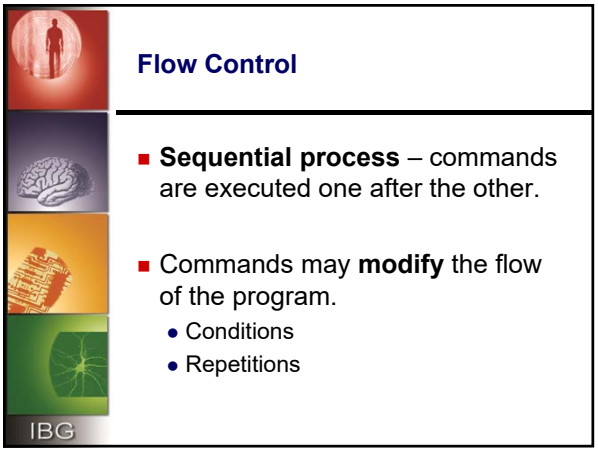
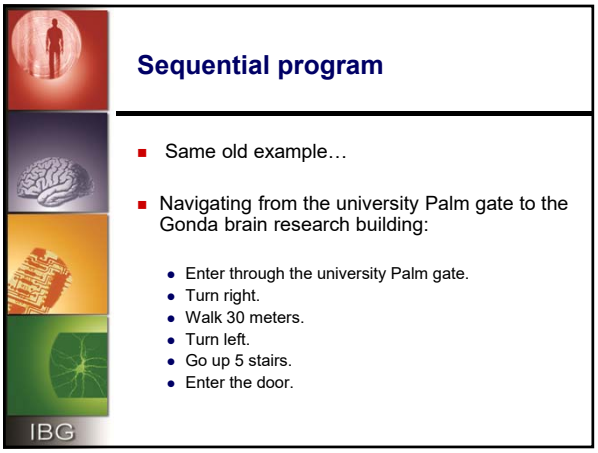

Introduction to programming
2017/18
Week 4: Flow control - Repetitive execution
Izhar Bar-Gad
Room: 408 Phone: 7141 Email: izhar.bar-gad@biu.ac.il


Flow Control
Sequential process – commands are executed one after the other.
Commands may modify the flow of the program.
Conditions
Repetitions
IBG


Sequential program
Same old example…
Navigating from the university Palm gate to the Gonda brain research building:
Enter through the university Palm gate.
Turn right.
Walk 30 meters.
Turn left.
Go up 5 stairs.
Enter the door.
IBG

Example: Repetition

- Handling an unknown number of stairs!
  - Enter through the university Palm gate.
  - Turn right.
  - Walk 30 meters.
  - Turn left.
  - Is there a stair in front of you?
    - Go up 1 stair.
    - Redo the question
  - Enter the door.

Repetitive execution

- Perform the code multiple times.
- Two main commands:
  - 'for' – iterate over a list of values.
  - 'while' – iterate until a condition is met.

The 'for' loop - simple case

- The 'for' loop repeats a group of commands a fixed, predetermined number of times.
- Simple syntax (default increment is 1):

```
for var = start : end,
      commands…
end
```

- Example:

```
a = 1;
for i = 1 : 5
    a = a * i;
end
a = a + 3;
```

What is *a* ?

The 'for' loop - still simple case

- Simple syntax (variable increment):

```
for var = start : change : end,
      commands…
end
```

- Example:

```
V = zeros(1,500);
scalar = 8;
for i = 1000 : -2 : 1
    V(501-i/2) = scalar * i;
end
```

IBG

Nested 'for' Loops I

- We can use *for* loops inside each other.
- Example:

```
for i = 1:m
      for j = 1:n
            H(i,j) = i * j;
      end
end
```

IBG

Nested 'for' Loops II

- We can use *for* loops inside each other.
- Example:

```
for i = 1:m
      for j = i:n
            H(i,j) = i * j;
      end
end
```

IBG

'for' Loops - Notes

- Increment can be negative ('going down loop').
- 'for' loops are much less efficient compared to MATLAB built-in commands and matrices operations.
- Use 'for' loop only when you can't do your calculations by built-in commands or matrices operations.
- Use correct indentation for readable code.

'for' Loops - efficiency

- a – vector with 1000 elements
- s – scalar
- Good:
  b=a+s;
- Bad:
  for i=1:1:1000
  b(i) = a(i) + s;
  end

Efficiency and timing

- **tic** & **toc** function as a stopwatch.
- The simplest way of assessing performance.
  - Not the net time
  - Affected by other processes

```
>> a=[1:1000000];
>> tic;a=a+5;toc;
Elapsed time is 0.042763 seconds.

>> tic;for i=1:1000000;a(i)=a(i)+5;end;toc;
Elapsed time is 1.728194 seconds.
```

Efficiency and timing

- Good (5msec)

```
a=rand(100000,1);
```

- Bad (200msec)

```
a=zeros(100000,1);
for i=1:100000
    a(i)=rand;
end
```

- Worse (112,500msec)

```
for i=1:100000
    a(i)=rand;
end
```

- Reallocations are devastating !

IBG

Command: *for*
The complex scenario

The actual syntax of the *for* command is much more general

Syntax:

```
for var = array
      commands…
end;
```

Example:

```
for i=[1 3 5 3 4 2]
      disp(i^2)
end
```

IBG

'while' Loop

- Repeat a group of statements an indefinite number of times under control of a logical condition.
- **Syntax:**

```
while condition,
    commands…
end
```

- **Operation:**
  1) Check condition.
  2) If condition is TRUE than execute commands otherwise exit loop.
  3) Goto 1

IBG

Example: ***while***

```
while a < b,
    a = a + 2;
    b = b – 3;
end
```

1. If initially a is 71 and b is 82, what is their final value?
2. If initially a is 82 and b is 71, what is their final value?

‘while’ Loop - Notes

- Use ‘*while*’ loop only when you can’t do your calculations by built-in commands or matrices operations.
- Use correct indentation for readable code.
- ‘while’ loops may be nested (also within ‘*if*’, ‘*switch*’ and ‘*for*’ blocks).
- The commands should change the condition value in order not get into an infinite loop (another option – ‘*break*’).