Lec 05: WHILE-Loops

Pop Quiz

Question 1

How many lines of output are produced by the following script?

```
for k = 100:200
    disp(k)
end
```

Ⓐ 99   Ⓑ 100   Ⓒ 101   Ⓓ 200

Pop Quiz

Question 2

How many lines of output are produced by the following script?

```
for k = 100:200
    if mod(k,2) == 0
        disp(k)
    end
end
```

A 50

B 51

C 100

D 101

FOR-Loop: Tips

- Basic loop header:

```
for <loop var> = 1:<ending value>
```

- To adjust starting value:

```
for <loop var> = <starting value>:<ending value>
```

- To adjust step size:

```
for <loop var> = <starting value>:<step size>:<ending value>
```

Examples

- To iterate over 1, 3, 5, . . . , 9: [*step size = 2*]

```
    for k = 1:2:9
```

  or

```
    for k = 1:2:10
```

- To iterate over 10, 9, 8, . . . , 1: [*negative step size*]

```
    for k = 10:-1:1
```

Need for Another Loop

- `For`-loops are useful when the number of repetitions is known in advance.

  "Simulate the tossing of a fair coin 100 times and print the number of Heads."

- It is not very suitable in other situations such as

  "Simulate the tossing of a fair coin until the gap between the number of Heads and that of Tails reaches 10."

  We need another loop construct that terminates as soon as $|N_{\mathrm{H}} - N_{\mathrm{T}}| = 10$.

WHILE-Loop Basics

WHILE-loop is used when a code fragment needs to be executed repeatedly *while* a certain condition is true.

```
while <continuation criterion>
    <code fragment>
end
```

- The number of repetitions is *not* known in advance.
- The continuation criterion is a boolean expression, which is evaluated at the start of the loop.
  - If it is true, the loop body is executed. Then the boolean expression is evaluated again.
  - If it is false, the flow of control is passed to the end of the loop.

Simple WHILE-Loop Examples

```
k = 1; n = 10;
while k <= n
    fprintf('k = %d\n', k)
    k = k+1;
end
```

```
k = 1;
while 2^k < 5000
    k = k+1;
end
fprintf('k = %d\n', k)
```

FOR-Loop to WHILE-Loop

A `for`-loop can be written as a `while`-loop. For example,

FOR

```
s = 0;
for k = 1:4
    s = s + k;
    fprintf('%2d %2d\n', k, s)
end
```

WHILE

```
k = 0; s = 0;
while k < 4
    k = k + 1; s = s + k;
    fprintf('%2d %2d\n', k, s)
end
```

- Note that `k` needed to be initialized before the `while`-loop.
- The variable `k` needed to be updated inside the `while`-loop body.

Up/Down Sequence

Question

Pick a random integer between 1 and 1,000,000. Call the number n and repeat the following process:

- If n is even, replace n by $n/2$.
- If n is odd, replace n by $3n+1$.

Does it ever take more than 1000 updates to reach 1?

- To generate a random integer between 1 and k, use `randi`, *e.g.*,

```
randi(k)
```

- To test whether a number n is even or odd, use `mod`, *e.g.*,

```
mod(n, 2) == 0
```

Attempt Using FOR-Loop

```
for step = 1:1000
    if mod(n,2) == 0
        n = n/2;
    else
        n = 3*n + 1;
    end
    fprintf(' %4d %7d\n', step, n)
end
```

- Note that once n becomes 1, the central process yields the following pattern:
$$1, 4, 2, 1, 4, 2, 1, \ldots$$
- This program continues to run even after n becomes 1.

Solution Using `WHILE`-Loop

```
step = 0;
while n > 1
    if mod(n,2) == 0
        n = n/2;
    else
        n = 3*n + 1;
    end
    step = step + 1;
    fprintf(' %4d %7d\n', step, n)
end
```

- This shuts down when n becomes 1!

Exercise: Gap of 10

Question

Simulate the tossing of a fair coin until the gap between the number of Heads and that of Tails reaches 10.

Summary

- `For`-loop is a programming construct to execute statements repeatedly.

```
for  <loop index values>
     <code fragment>
end
```

- `While`-loop is another construct to repeatedly execute statements. Repetition is controlled by the termination criterion.

```
while  <termination criterion is not met>
       <repeat these statements>
end
```