CLASS 8: FLOWCHARTS – WHILE LOOPS

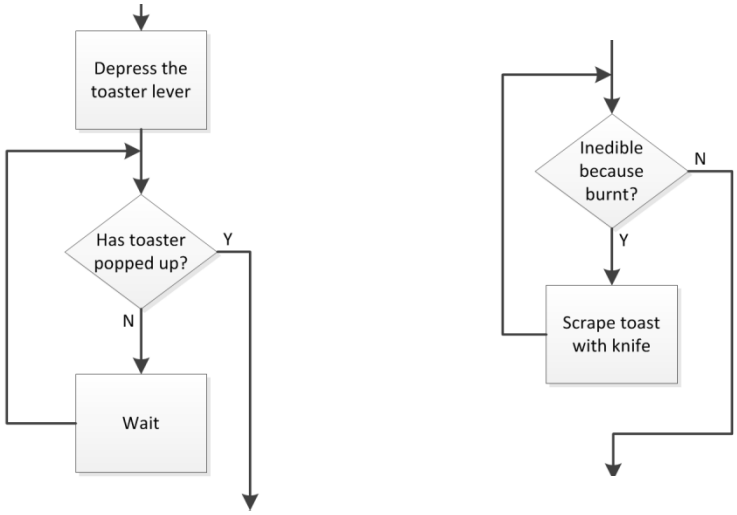
ENGR 102 – Introduction to Engineering

Loops

- *while* loops
- *for* loops

Loops

- We've already seen some examples of flow charts that contain ***loops***:



- Structures where the algorithmic flow loops back and repeats process steps
  - Repeats as long as a certain condition is met, e.g., toaster has not popped up, toast is inedible, etc.

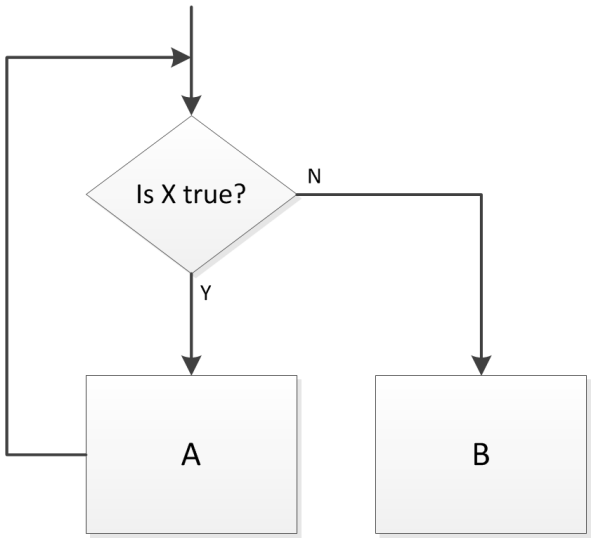
Loops

- Algorithms employ two primary types of loops:
  - ***while loops***: loops that execute as long as a specified condition is met – loop executes as many times as is necessary
  - ***for loops***: loops that execute a specified exact number of times
- Similar looking flowchart structures
  - for loop can be thought of as a special case of a while loop
  - However, the distinction between the two is very important

while Loop

while Loop

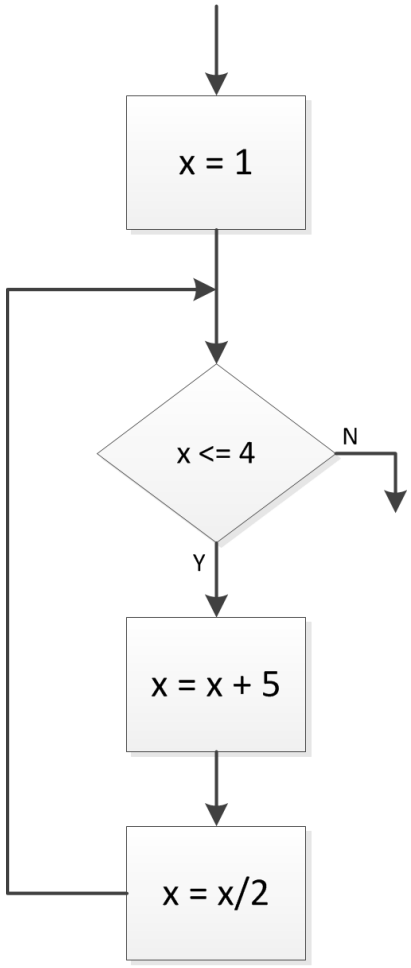
- Repeatedly execute an instruction or set of instructions as long as (*while*) a certain condition is met (is *true*)
- Repeat A *while* X is true
  - As soon as X is no longer true, *break* out of the loop and continue on to B
  - A may never execute
  - A may execute only once
  - A may execute forever – an ***infinite loop***
    - If A never causes X to be false
    - *Usually* not intentional



while Loop

- Algorithm loops while $x \leq 4$
  - Loops three times:

Iteration	x
0	1
1	6 3
2	8 4
3	9 4.5

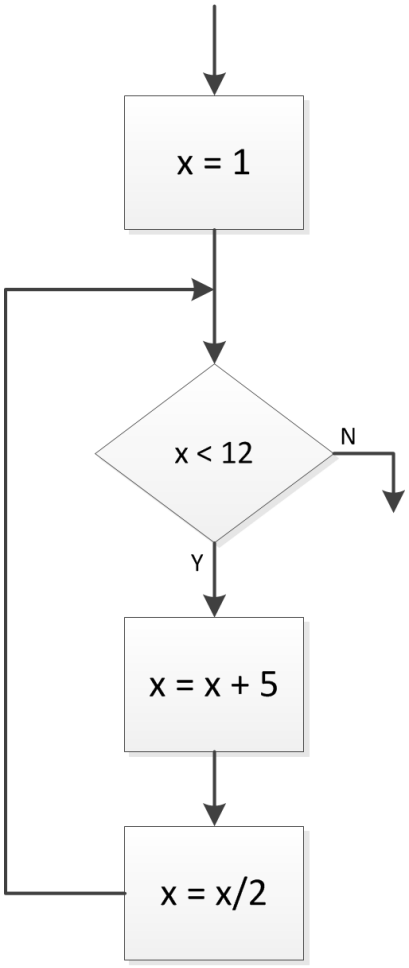


- Value of x exceeds 4 several times during execution
  - x value checked at the beginning of the loop
- Final value of x is greater than 4

while Loop – Infinite Loop

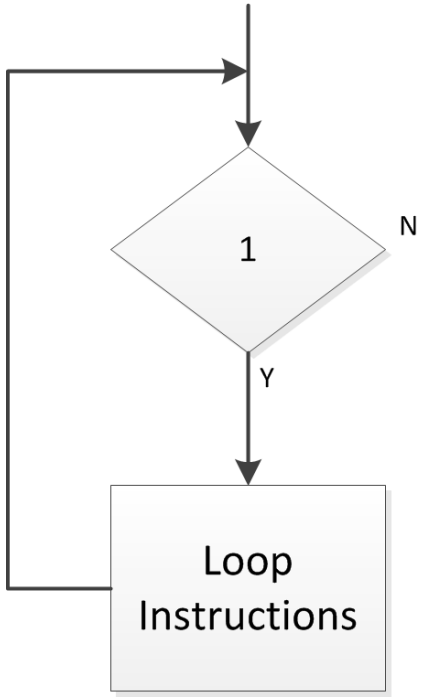
- Now looping continues as long as $x < 12$
  - x never exceeds 12
  - Loops forever – an ***infinite loop***

Iteration	x
0	1
1	6 3
2	8 4
3	9 4.5
4	9.5 4.75
5	9.75 4.875
6	9.875 4.9375
⋮	⋮



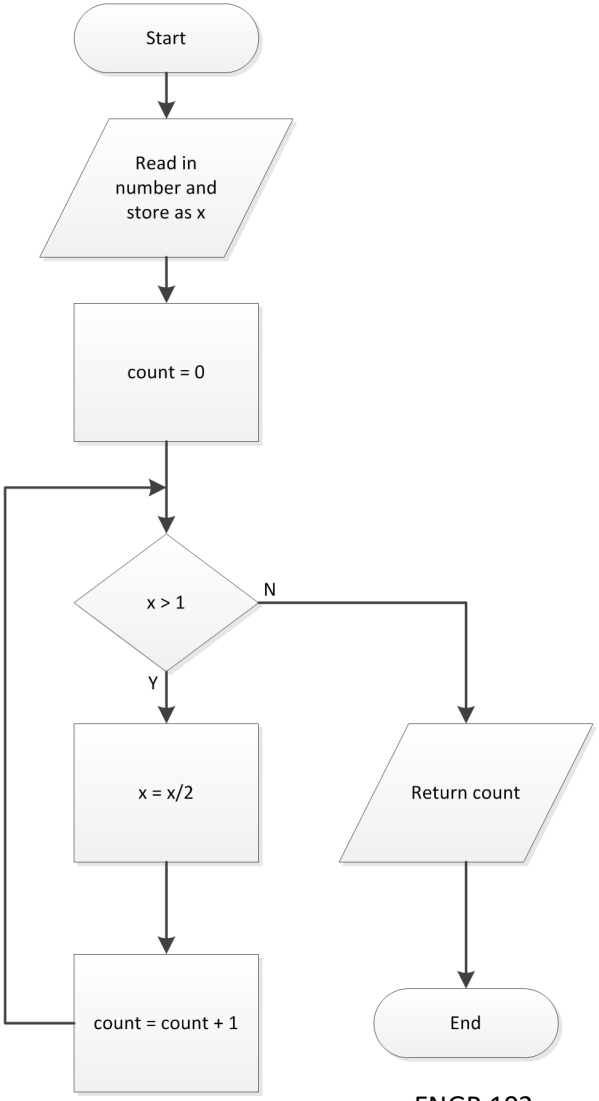
Infinite Loops

- Occasionally infinite loops are desirable
  - Consider for example microcontroller code for an environmental monitoring system
    - Continuously takes measurements and displays results while powered on
- Note the logical statement in the conditional block
  - Logical statements are either true (Y, 1) or false (N, 0)
  - 1 is the Boolean representation of true or Y



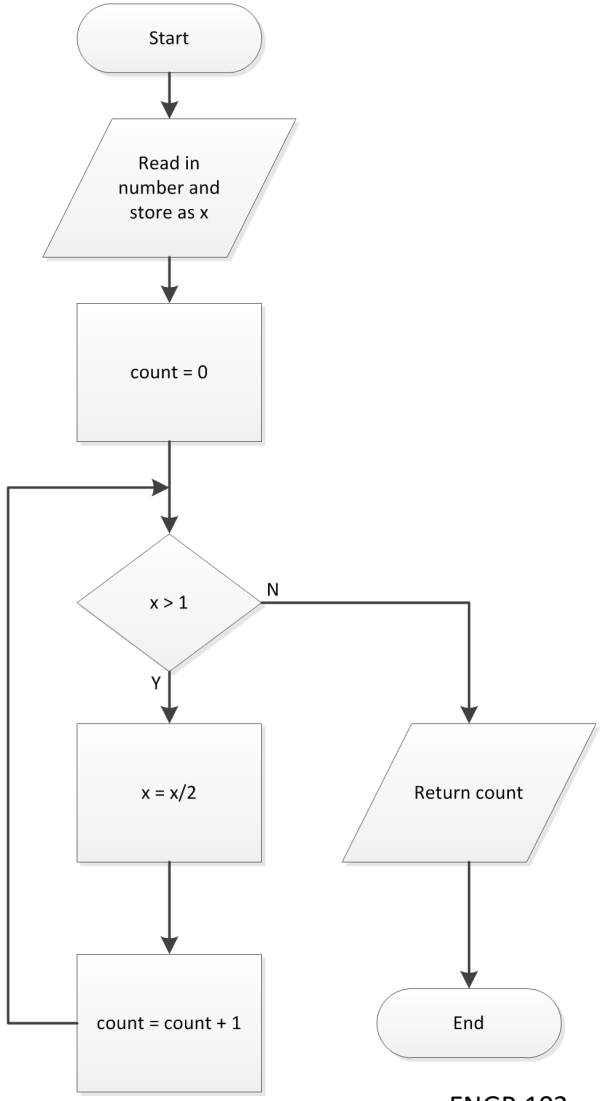
while Loop – Example 1

- Consider the following algorithm:
  - Read in a number (e.g. user input, from a file, etc.)
  - Determine the number of times that number can be successively divided by 2 before the result is ≤ 1
- Use a ***while loop***
  - Divide by 2 ***while*** number is > 1



while Loop – Example 1

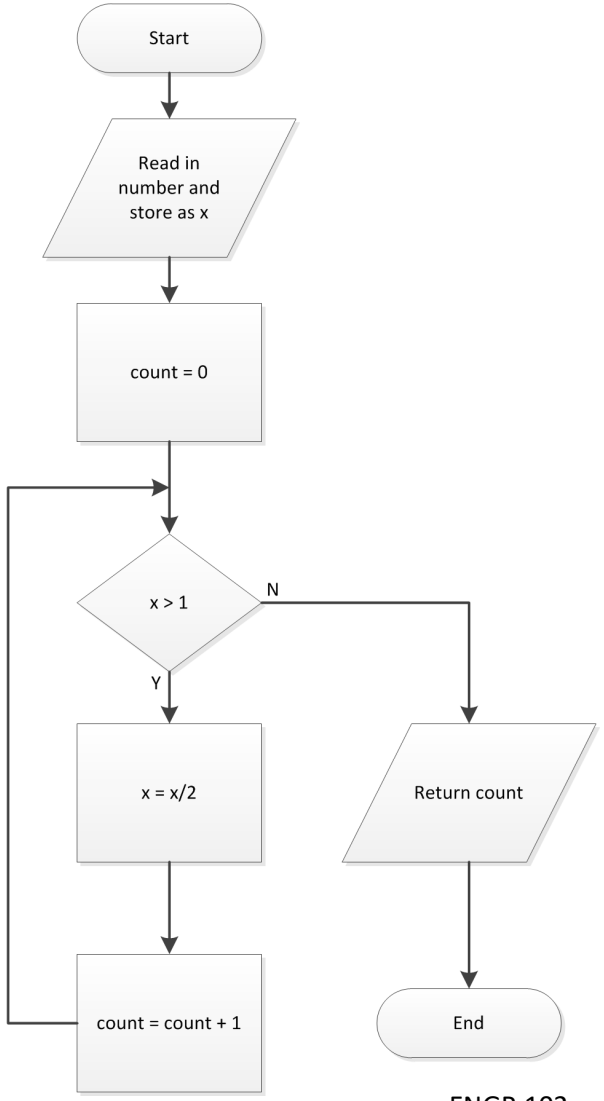
- Number of loop iterations depends on value of the input variable, x
  - Characteristic of while loops
    - # of iterations unknown a priori
  - If $x \leq 1$ loop instructions never execute
- Note the data I/O blocks
  - Typical – many algorithms have ***inputs*** and ***outputs***



while Loop – Example 1

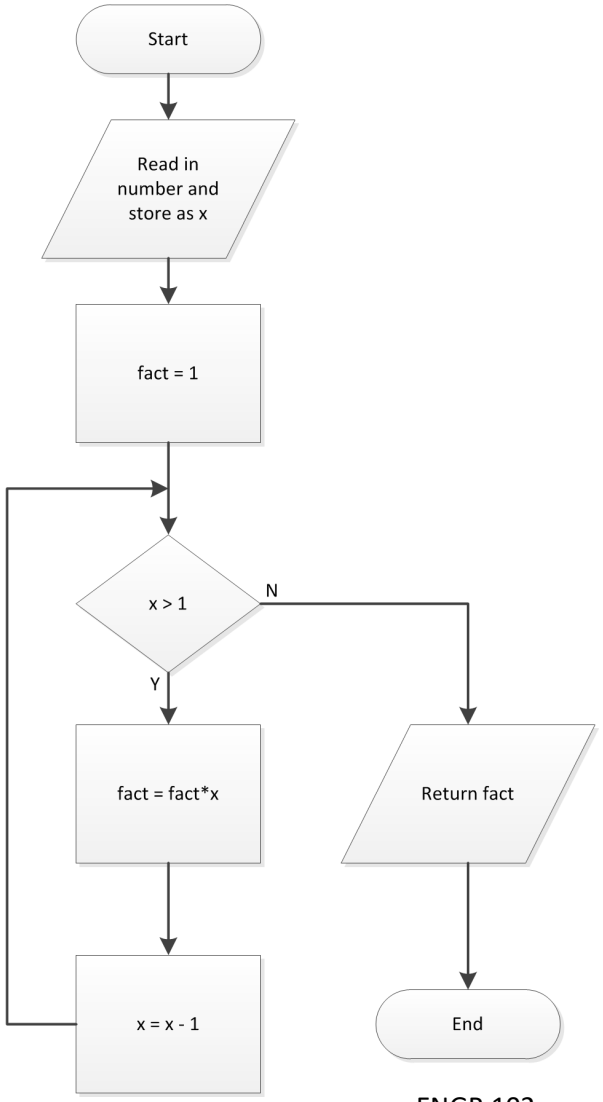
- Consider a few different input, x, values:

count	x		x		x
0	5		16		0.8
1	2.5		8		-
2	1.25		4		-
3	0.625		2		-
4	-		1		-
5	-		-		-



while Loop – Example 2

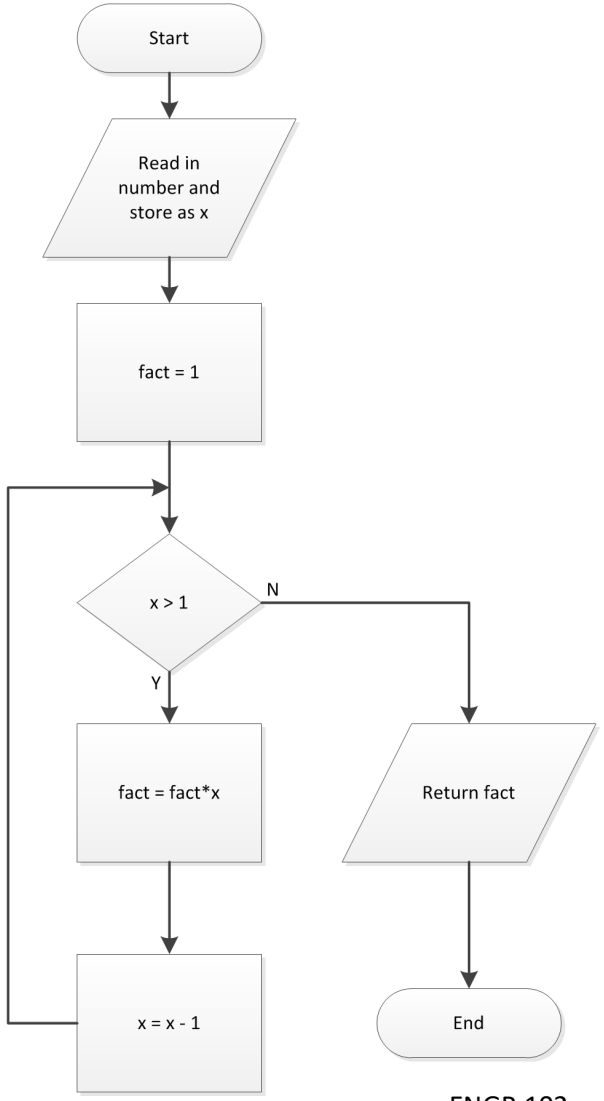
- Next, consider an algorithm to calculate x!, the ***factorial*** of x:
  - Read in a number, x
  - Compute the product of all integers between 1 and x
  - Initialize result, fact, to 1
  - Multiply fact by x
  - Decrement x by 1
- Use a ***while loop***
  - Multiply fact by x, then decrement x ***while*** $x > 1$



while Loop – Example 2

- Consider a few different input, x, values:

x	fact		x	fact		x	fact
5	1		4	1		0	1
5	5		4	4		-	-
4	20		3	12		-	-
3	60		2	24		-	-
2	120		1	24		-	-
1	120		-	-		-	-



while Loop – Example 2

- Let's say we want to define our factorial algorithm only for ***integer*** arguments
- Add ***error checking*** to the algorithm
  - After reading in a value for x, check if it is an integer
  - If not, generate an error message and exit
  - Could also imagine rounding x, generating a ***warning*** message and continuing