CS105 Problem Solving

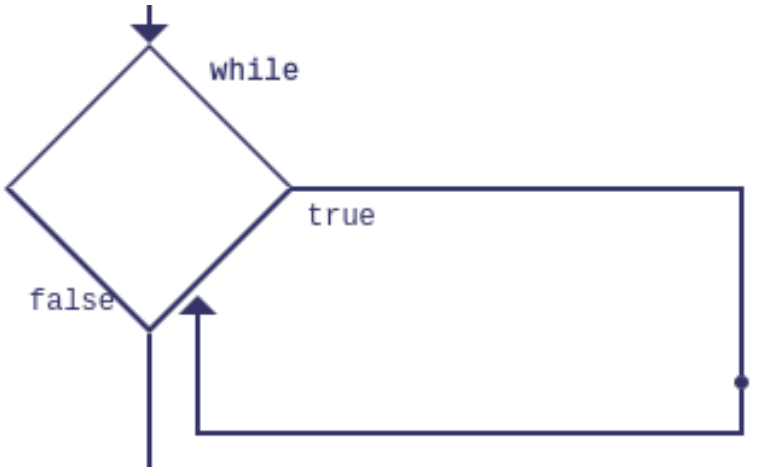
Nested Loops

Wholeness

- Nested loops are loops inside loops. These are useful when you have a collection that contains collections. Like a page of text is a collection of lines, and each line is a collection of characters.
- The whole is greater than the sum of the parts, it is not only important that a part exists, it is also important where a part exists.

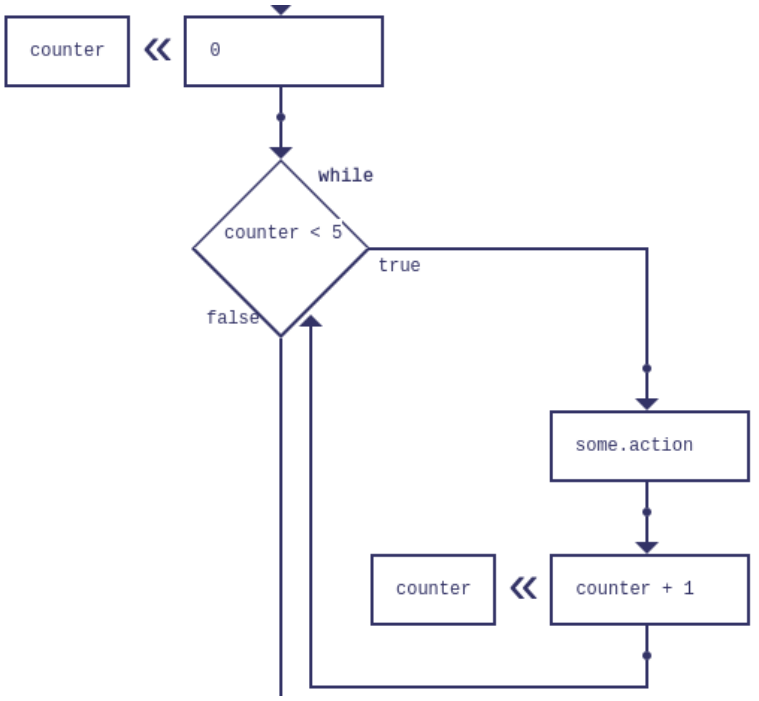
Loop Review

- A loop lets you do something multiple times



Loop with Counter Review

- Most loops do an action a specific amount of times
- The following does **`some.action`** 5 times
  - The counter starts at zero
  - 5 is in the condition
  - The action is inside the loop



In Code

```
var counter; // number
counter = 0;
while (counter < 5) {
    some.action
    counter = counter + 1;
}
```

- When we look at the code
  - Actions inside the loop are indented
  - The action we want to do 5 times is inside the loop
  - The loop does not care what the action is
    -

Solving a problems with loops

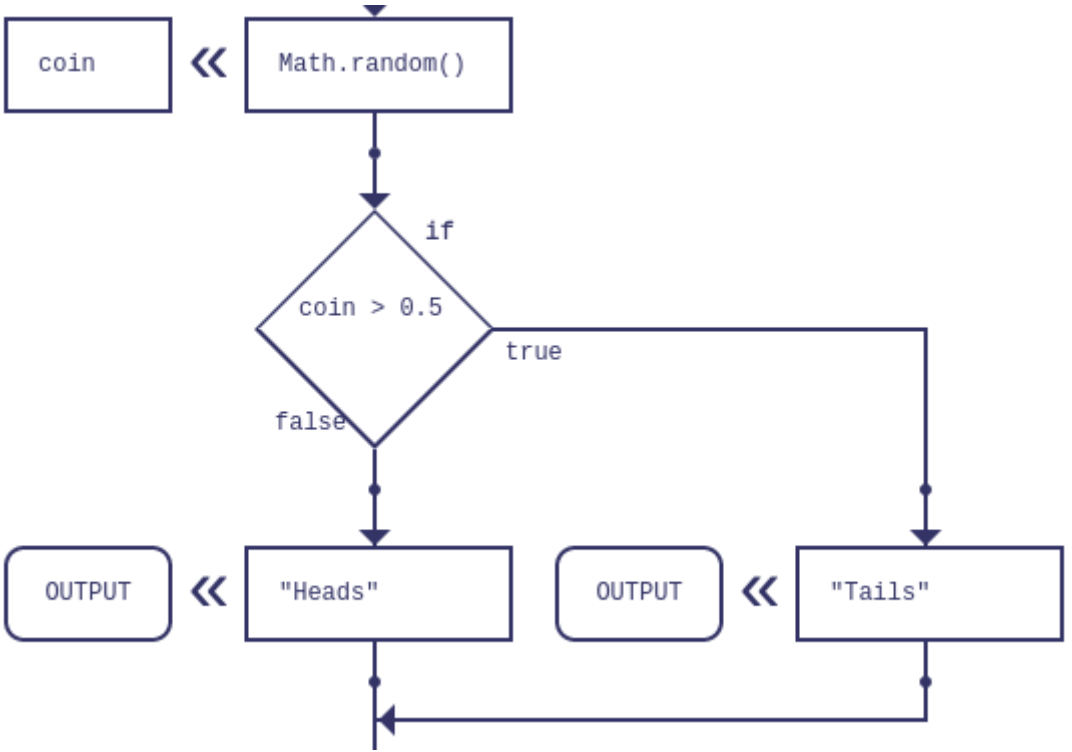
- First implement some.action without the loop

- Examples:
  - Output “Hello world”
  - Make a random number
- Or more complex
  - Flip a coin
  - Check if a character is upper case

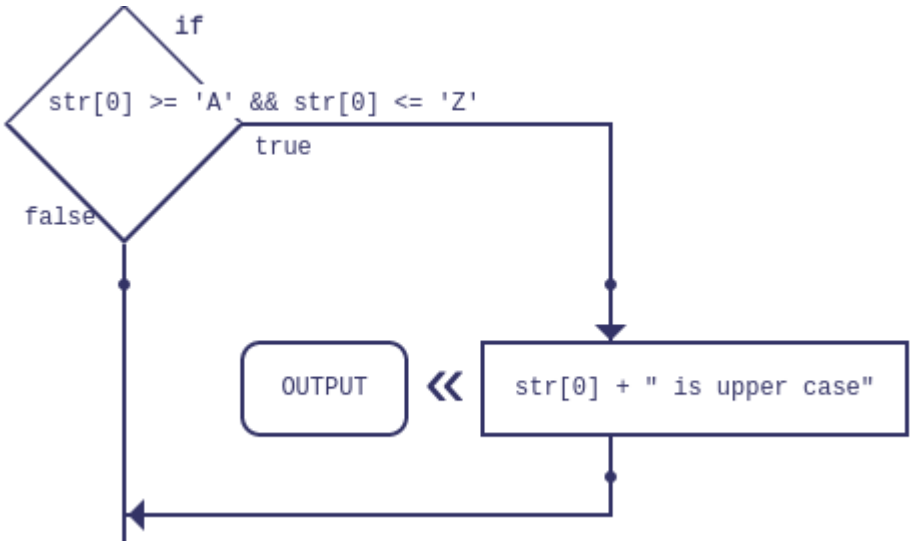
Output Hello World

Make a Random Number

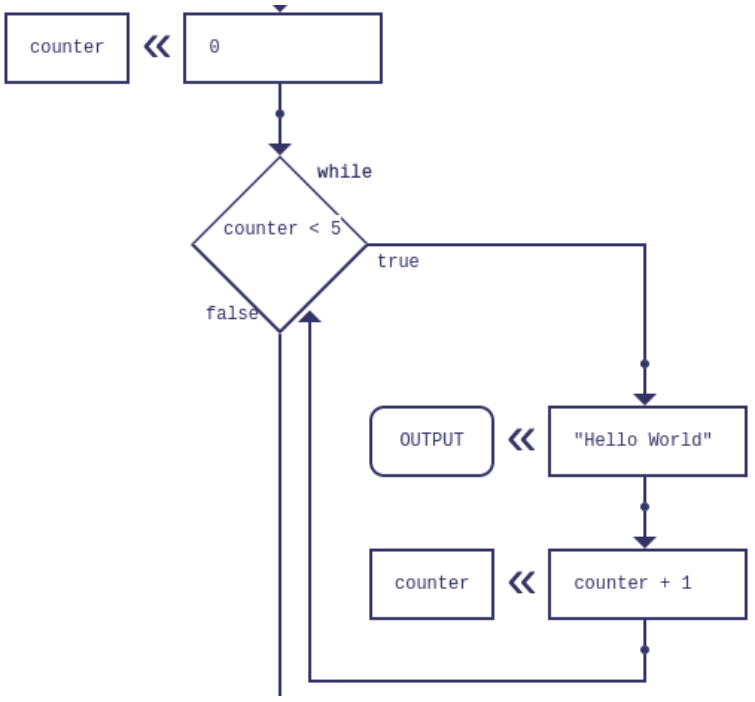
Flip a coin


coin
Math.random()
if
coin > 0.5
true
false
OUTPUT
"Heads"
OUTPUT
"Tails"

Is a character upper case



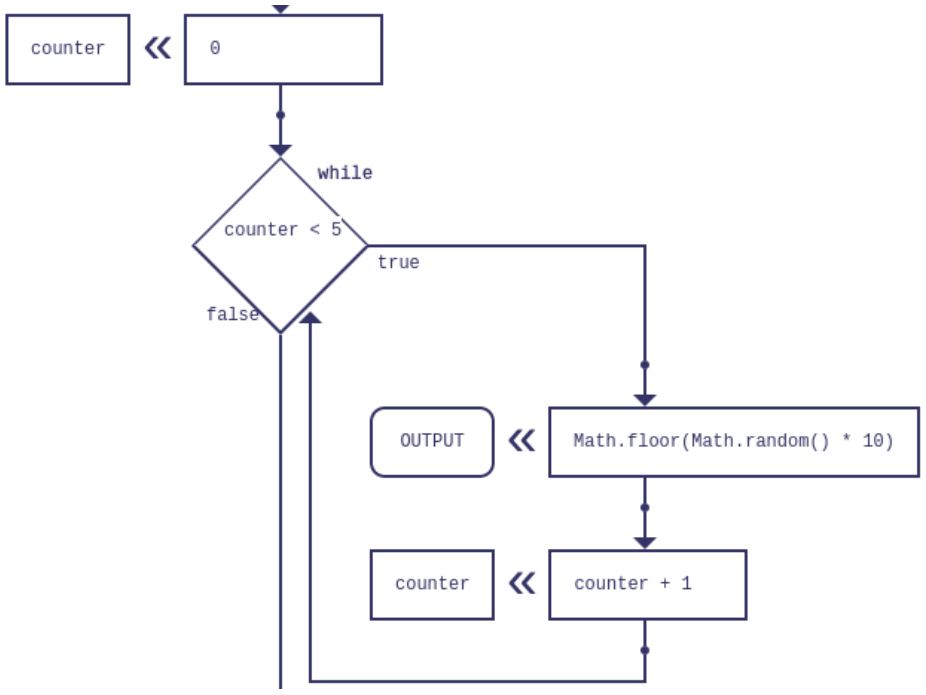
Then place it in a loop



- This outputs “Hello World” 5 times

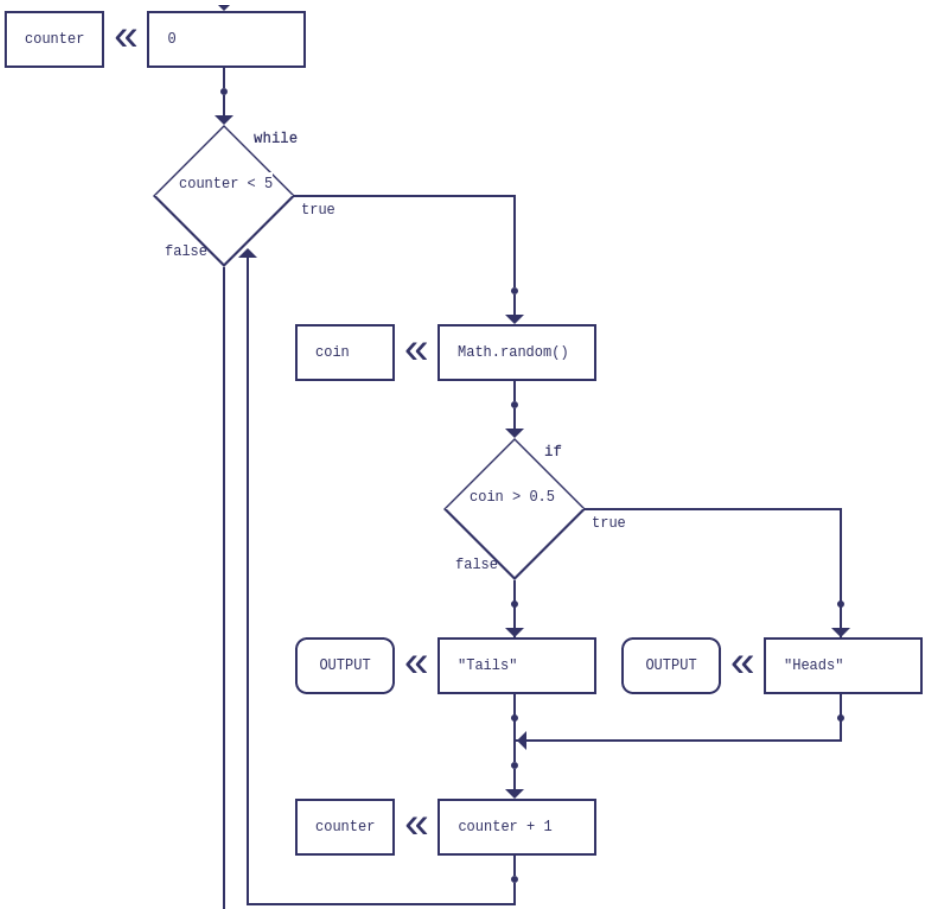
- To output it more or less times:
  - Change the loop condition

Regardless of the action



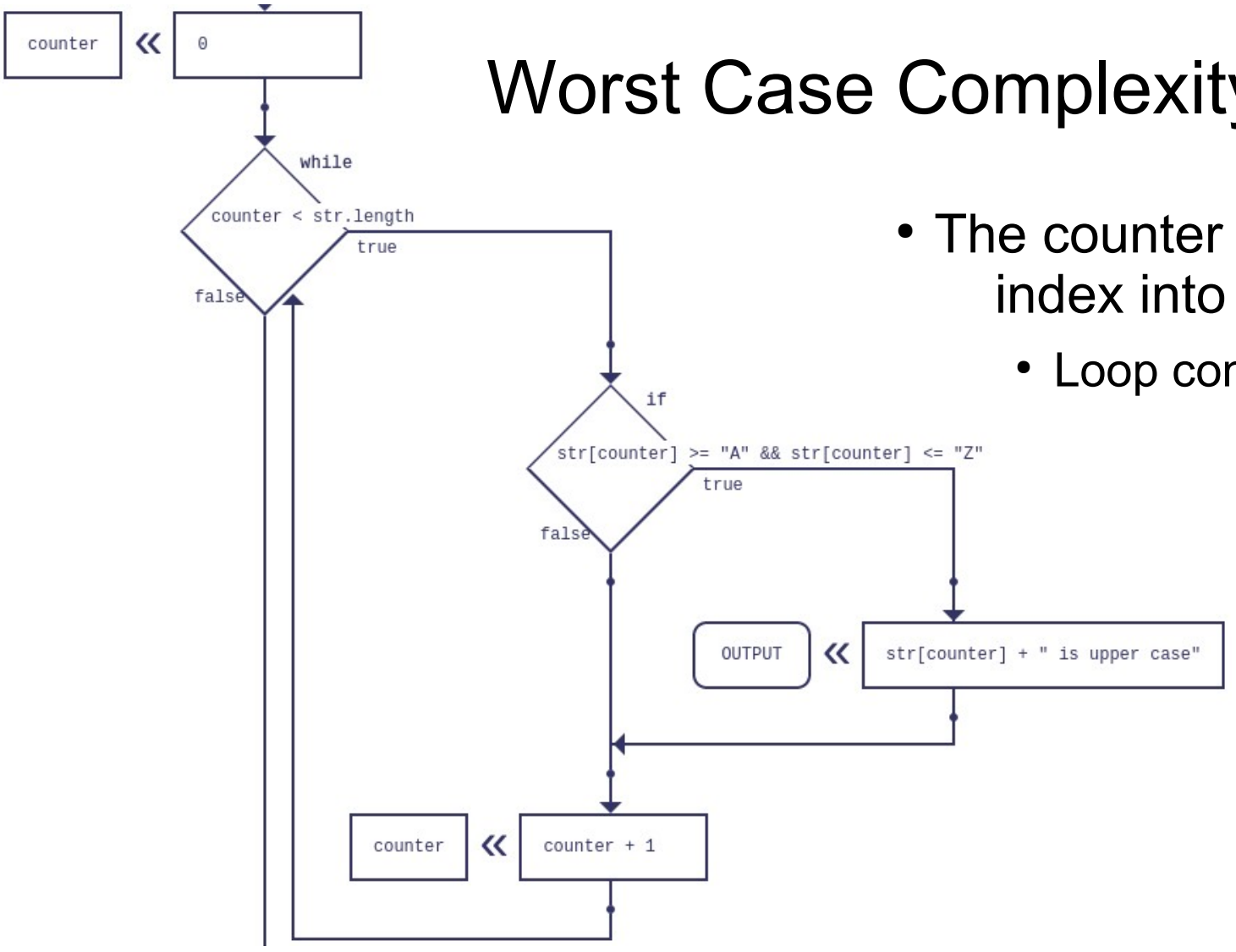
- The exact principle is true
  - Simply place it in the loop
- This outputs a random number 5 times
  - Change the loop condition to output it more or less times

Even more complex actions



- Simply place it in the loop
  - And it happens the specified amount of times (5 times)

Worst Case Complexity



- The counter is also used as index into a string or array
  - Loop condition goes to length

The Steps

1)Solve the part that needs to be done multiple times

2)Place it inside a loop

3)Adjust the condition to repeat the required amount

4)(if needed) use loop counter as an index

Exercise

- Step 1 – Write the action:
  - Write a program that asks the user for a number
    - If it is a multiple of 7 and greater than 100 output “I like this”
    - Otherwise output “Not such an interesting number”
- Step 2 – Put it in a loop:
  - Make a loop with a counter and put the previous step inside it
- Step 3 – Adjust the condition
  - The loop should repeat 100 times
- You have made:
  - Write program that asks for 100 numbers and outputs “I like this” if a number is a multiple of 7 and greater than 100, and outputs “Not such an interesting number” for numbers that are not.

Main Point

- Loops allow us to repeat a given action
- Allowing us to write less code an get more done

Nested Loops

- A nested loop is:
    - A loop inside another loop

- This happens because:
    - **some.action** happens to do something multiple times
    - The outer loop does not care about it
    - It is still just **some.action** that needs to be repeated

Solving Nested Loops

- Just like when solving a problem with a normal loop
  - Solve **some.action** first
  - Then place it inside a loop

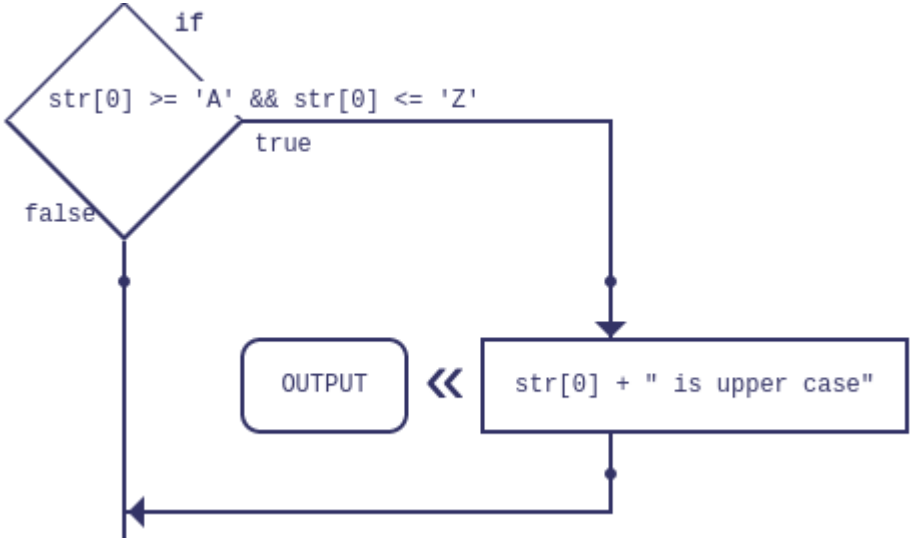
Example

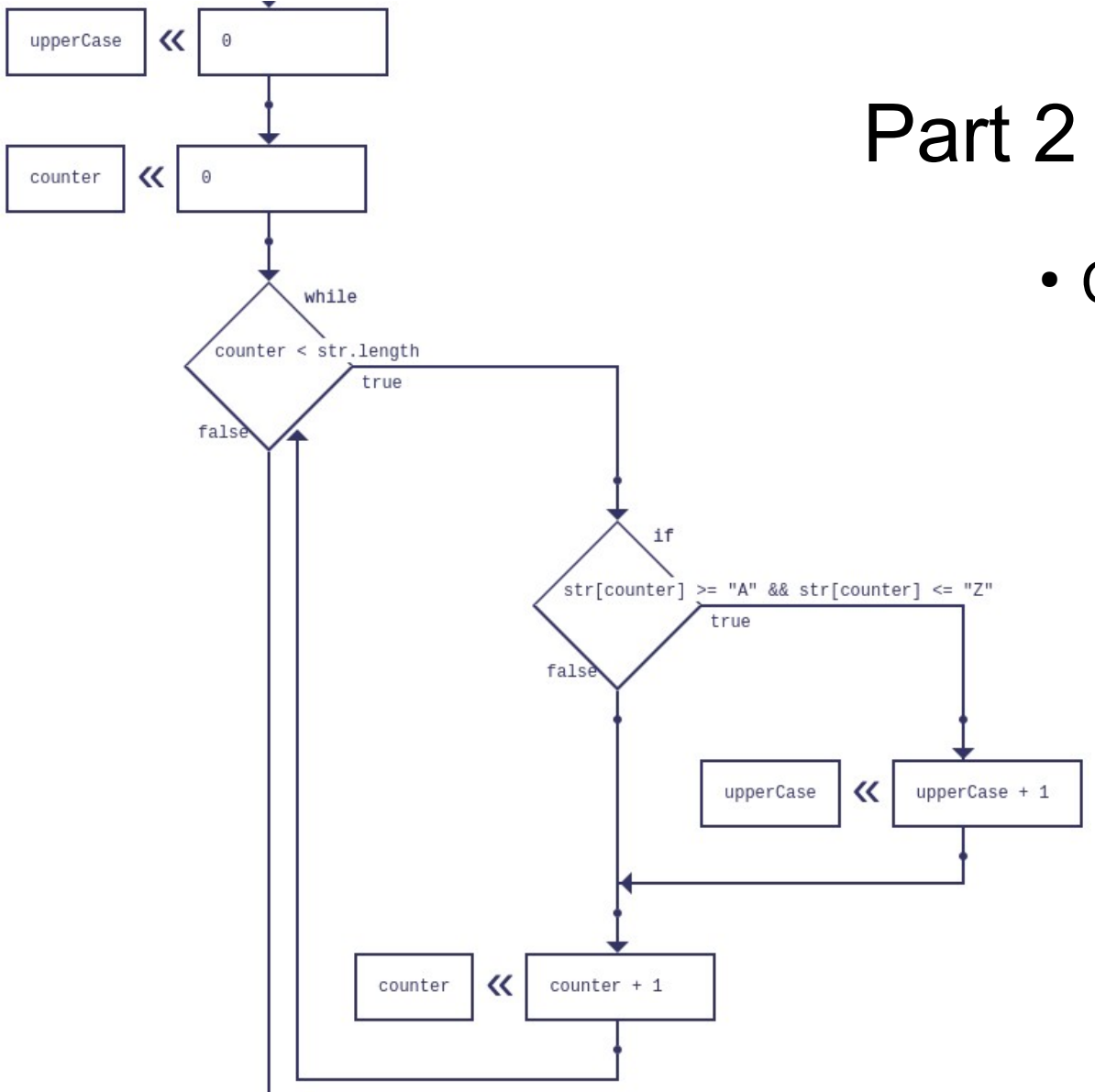
- A page has many lines of text
    - Each line of text has many characters

- Problem:
    - Count all the upper case characters on a page

Part 1

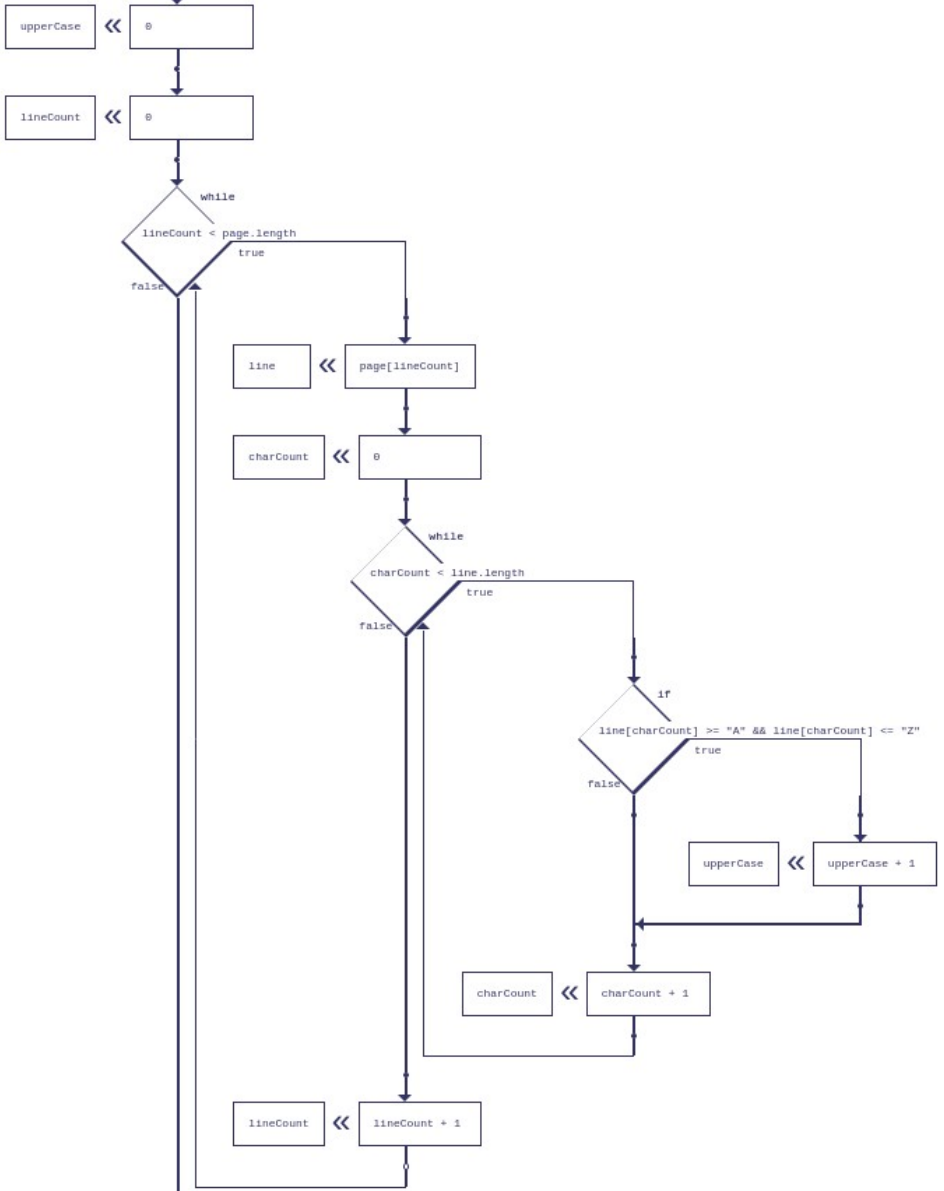
- Check if a character is upper case
  - This is our first “some.action”





Part 2

- Count upper case characters in a string
  - Take our some.action and put it in a loop
  - (add a upperCase variable that keeps track)
  - Then consider this to be "**some.action**"



Part 3

- Put “some.action” into a loop

```
upperCase = 0;
lineCount = 0;
while (lineCount < page.length) {
    line = page[lineCount];
    charCount = 0;
    while (charCount < line.length) {
        if (line[charCount] >= "A" && line[charCount] <= "Z") {
            upperCase = upperCase + 1;
        } else {
        }
        charCount = charCount + 1;
    }
    lineCount = lineCount + 1;
}
```

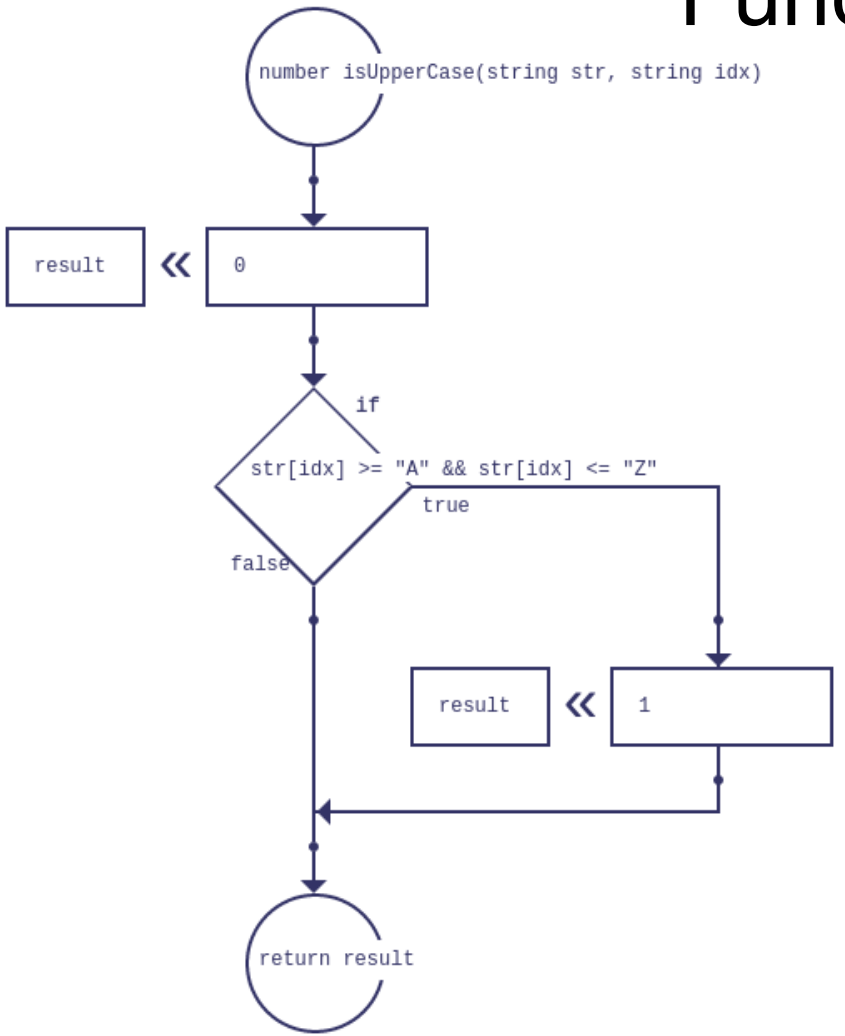
Main Point

- A nested loop is a loop inside another loop
- The outer loop does an action a certain amount of times
- That action also does something multiple times

Encapsulate with a Functions

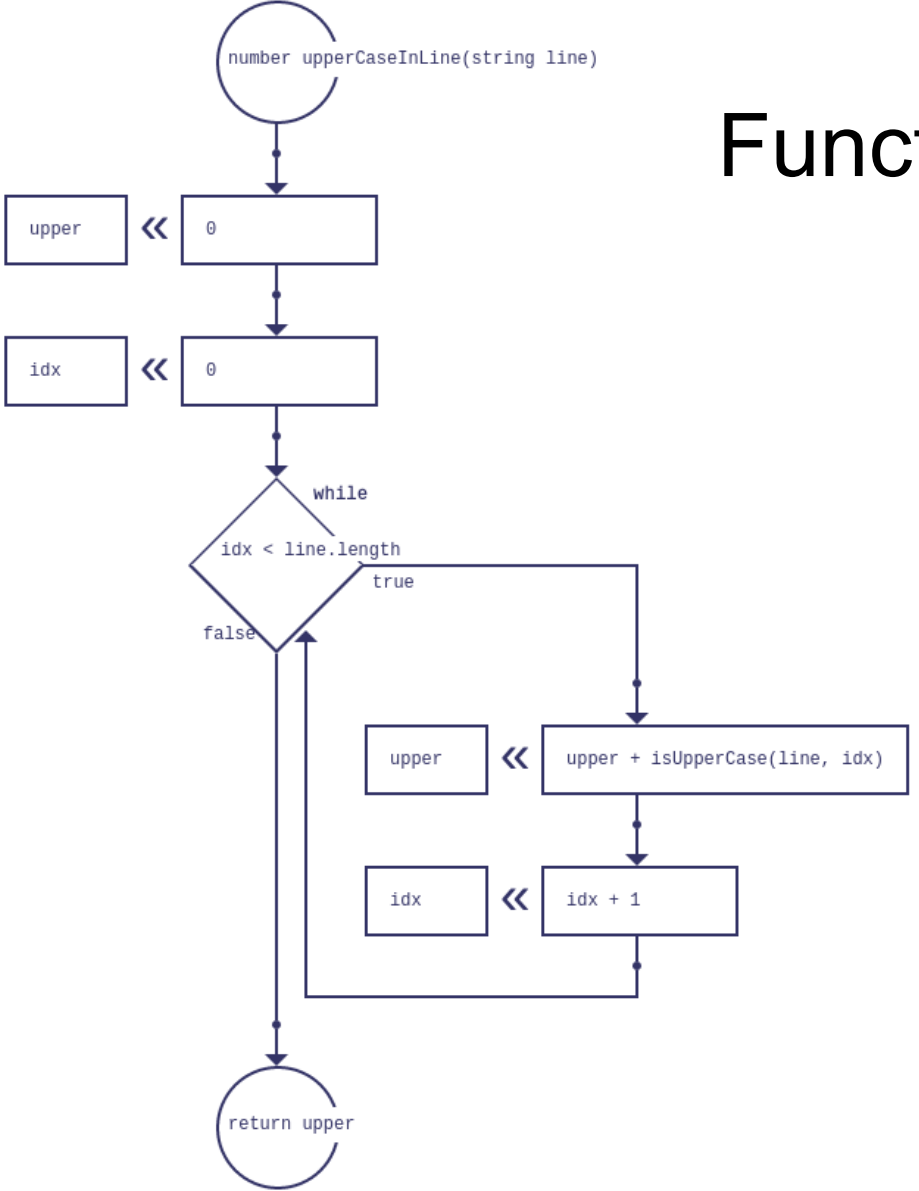
- Each part can be encapsulated into its own function
    - Doing so can help make the program clearer

Function – Part 1



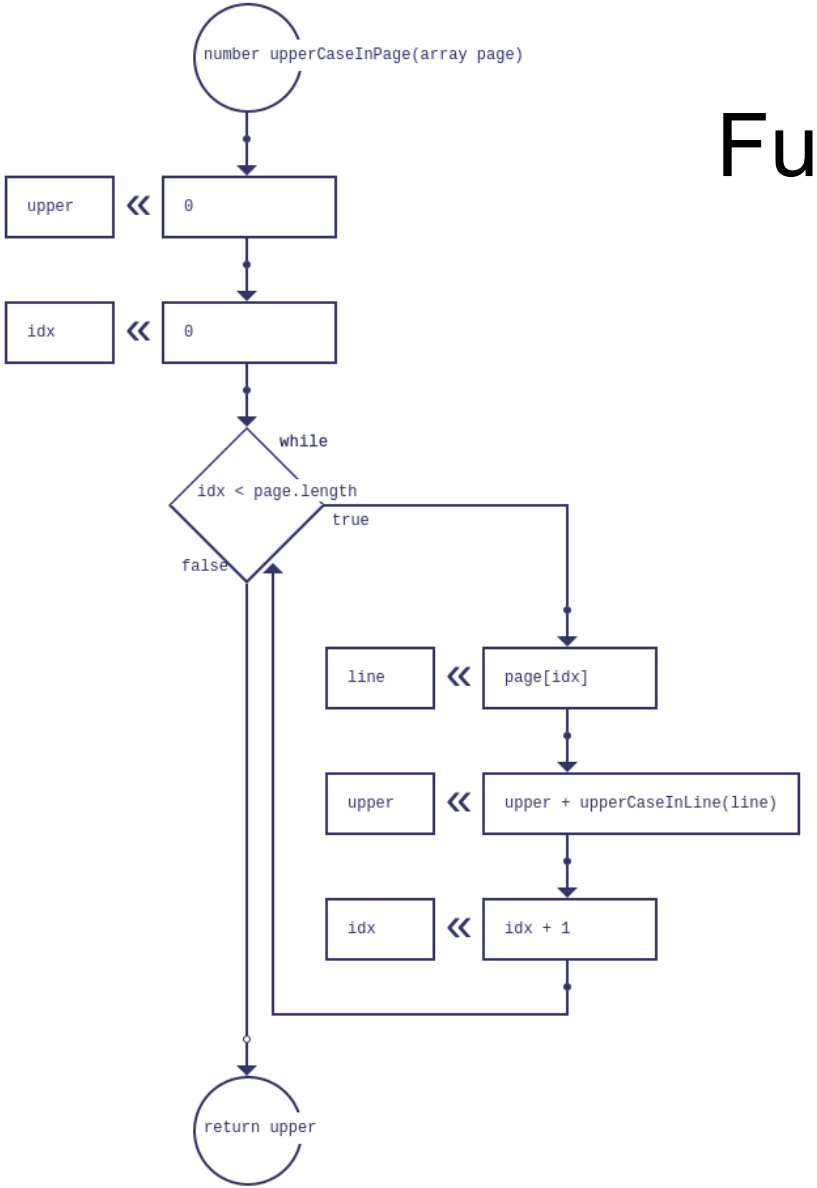

```
function isUpperCase(str, idx) {
    var result; // number
    result = 0;
    if (str[idx] >= "A" && str[idx] <= "Z") {
        result = 1;
    }
    return result;
}
```

Function – Part 2




```
function upperCaseInLine(line) {
    var upper; // number
    var idx; // number
    upper = 0;
    idx = 0;
    while (idx < line.length) {
        upper = upper + isUpperCase(line, idx);
        idx = idx + 1;
    }
    return upper;
}
```

Function – Part 3




```
function upperCaseInPage(page) {
    var idx; // number
    var line; // string
    var upper; // number
    upper = 0;
    idx = 0;
    while (idx < page.length) {
        line = page[idx];
        upper = upper + upperCaseInLine(line);
        idx = idx + 1;
    }
    return upper;
}
```

Exercise

- Make a program that counts how many characters are lower case in an array of strings (a page).

Main Point

- By encapsulating a the part with the inner loop functions can help make nested loops less complex.

For Loop

```
for(var i=0;i<5;i++) {
    // code in loop…
}
```

```
var i=0;

while(i<5) {
// code in loop…
i++;
}
```

Multidimensional Array

- [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

- [[1, 2, 3],
  [4, 5, 6],
  [7, 8, 9]]

Nested Loops

```
for(r=1;r<3;r++){     //   Outer loop
  for(c=1;c<3;c++){      //   Inner loop
    console.log(arr[r][c]);
  }
}
```

Example

- Demo printing all the numbers in a 2D array

Main Point

- When writing code a for loop is also a great way to avoid mistakes with nested loops
  - Mistakes are often made by not keeping the parts together
- A 2D array is just an array with arrays inside it

Summary

- Loops do a action (code) multiple times
- A nested loop is does an action multiple times
    - The action itself also does something multiple times
- Functions can help clean up / conceptualize nested loops

- In code for loops help clean up nested loops
- A 2D array is an array with arrays inside it