COMP 1010- Summer 2015 (A01)

Jim (James) Young

young@cs.umanitoba.ca

jimyoung.ca

passing arrays to and from functions

pass arrays to/from function

note: array variable types can be used the same as any variable:

type[] someFunction(**type[]** somevariable){…

e.g., make a function that returns an int array of size *n*, with data from *1..n*.

int[] makeArray(int n) {…

Use it in your program, draw lines from the points (array[i], 0) to the mouse.

Make a function to draw the points

The header is as normal: take the array as a parameter:

void drawLines(int[] xValues)

Arrays, functions, and memory

What happens here? Is this okay?

```
int[] numbers = new int[1];
void setup()
{
   numbers = makeArr(10);
   numbers[9]++;
}

int[] makeArr(int n)  {
   int [] data = new int[n];
   for (int i = 0; i < data.length; i++) {
     data[i] = i+1;
   }
   return data;
}
```

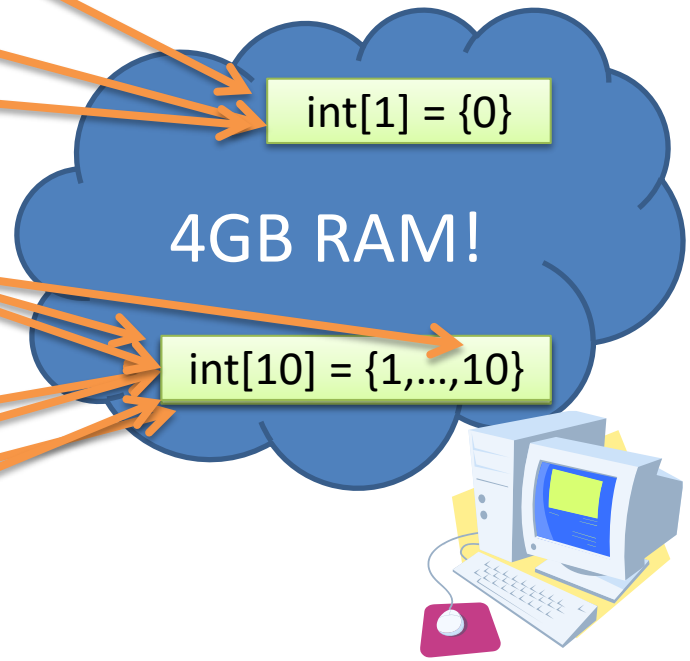
Just to peek… modify to print out the memory addresses

```
int[] numbers = new int[1];
void setup()
{
   println(""+numbers);
   numbers = makeArr(10);
   println(""+numbers);
   numbers[9]++;
}

int[] makeArr(int n)  {
   int [] data = new int[n];
   for (int i = 0; i < data.length; i++) {
     data[i] = i+1;
   }
   return data;
}
```

```
int[] numbers = new int[1];
void setup()
{
  println(""+numbers);
  numbers = makeArr(10);
  println(""+numbers);
  numbers[9]++;
}

int[] makeArr(int n)  {
  int [] data = new int[n];
  for (int i = 0; i < data.length; i++) {
    data[i] = i+1;
  }
  return data;
}
```



example: generate first n Fibonacci numbers

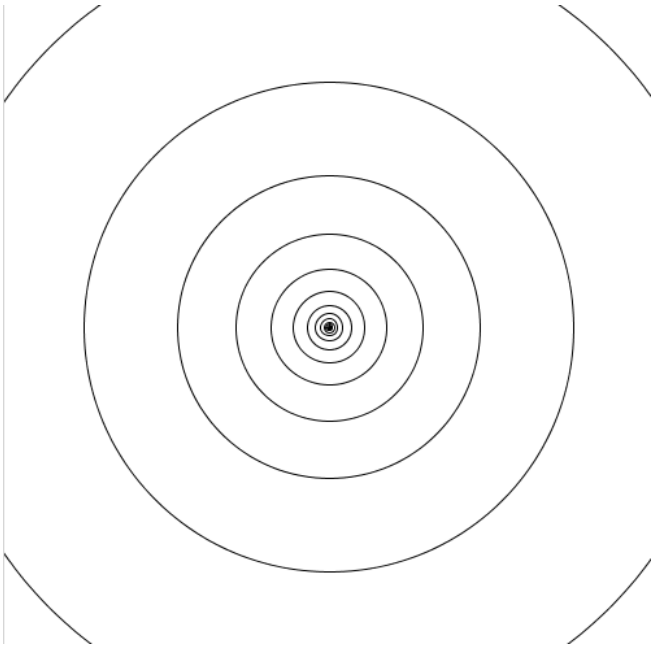
remember: Fibonacci numbers are a sequence :

0, 1, 1, 2, 3, 5…

$F_0 = 0,$

$F_1 = 1,$

$F_n = F_{n-1} + F_{n-2}$



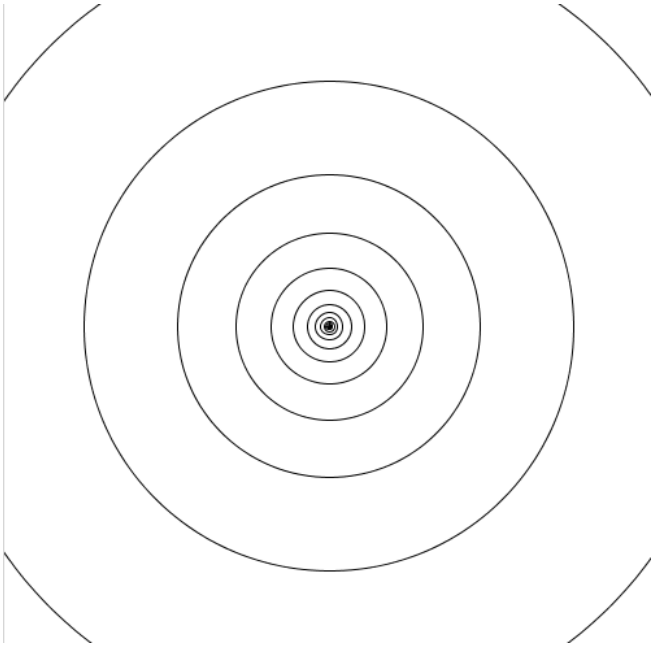
```
int[] fibonacciSequence(int n){
```

Visualize…

Draw circles with the radii as the Fibonacci numbers.

Put all the code in the setup

- not interactive



Questions..

int[] fib = fibonacci(COUNT);

Wait! I just created the variable and did not instantiate the array.. Is this okay?

changing values in a function….

consider:

```
void setup() {
        int i = 15;
        printInflation(i);
        println(i);
}

void printInflation(int number) {
        number += 1;
        println(number);
}
```

output:
16
15

The function does not alter the value of **int i** in setup because when printInflation is called, a copy of **i** is made

changing **array** values in a function….

```
void setup() {
        int i[] = {1, 2, 3};
        printInflationArry(i);
        println(i[0]);
}

void printInflationArry(int[] intarray) {
        intarray[0] += 1
        println(intarray[0]);
}
```

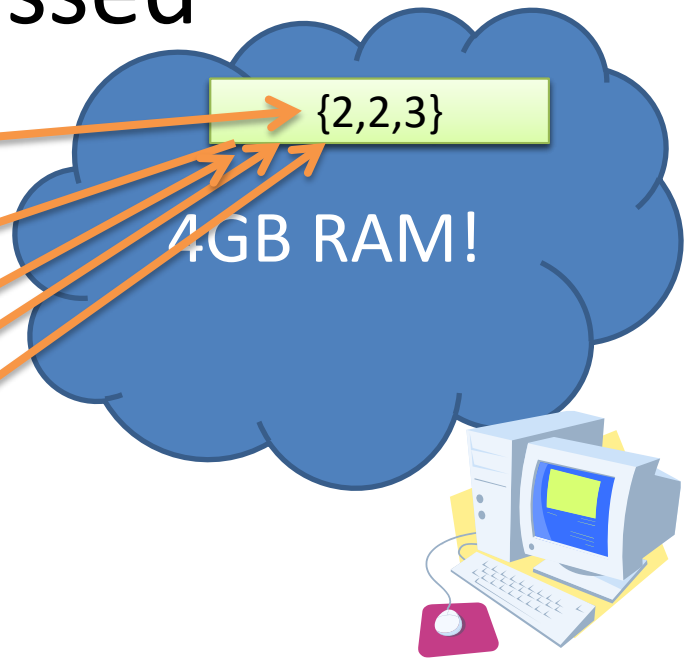
output:
2
2

Why does this happen???

Function calls only pass the **reference** to the array, the **address**. The entire array is not copied and passed

```
void setup() {
        int i[] = {1, 2, 3};
        printInflationArry(i);
        println(i[0]);
}

void printInflationArry(int[] intarray) {
        intarray[0] += 1
        println(intarray[0]);
}
```



consider…

```
void setup() {
        int i[] = {1, 2, 3};
        makeNewArray(i);
        println(i[0]);
}

void makeNewArray(int[] intarray) {
        intarray = new int[3];
        intarray[0] = 5;
        println(intarray[0]);
}
```

output:
5
1

Why does this happen??? the old array didn't change..

function passes the reference to the object. the original variable is unchanged

```
void setup() {
        int i[] = {1, 2, 3};
        makeNewArray(i);
        println(i[0]);
}

void makeNewArray(int[] intarray) {
        intarray = new int[3];
        intarray[0] = 5;
        println(intarray[0]);
}
```