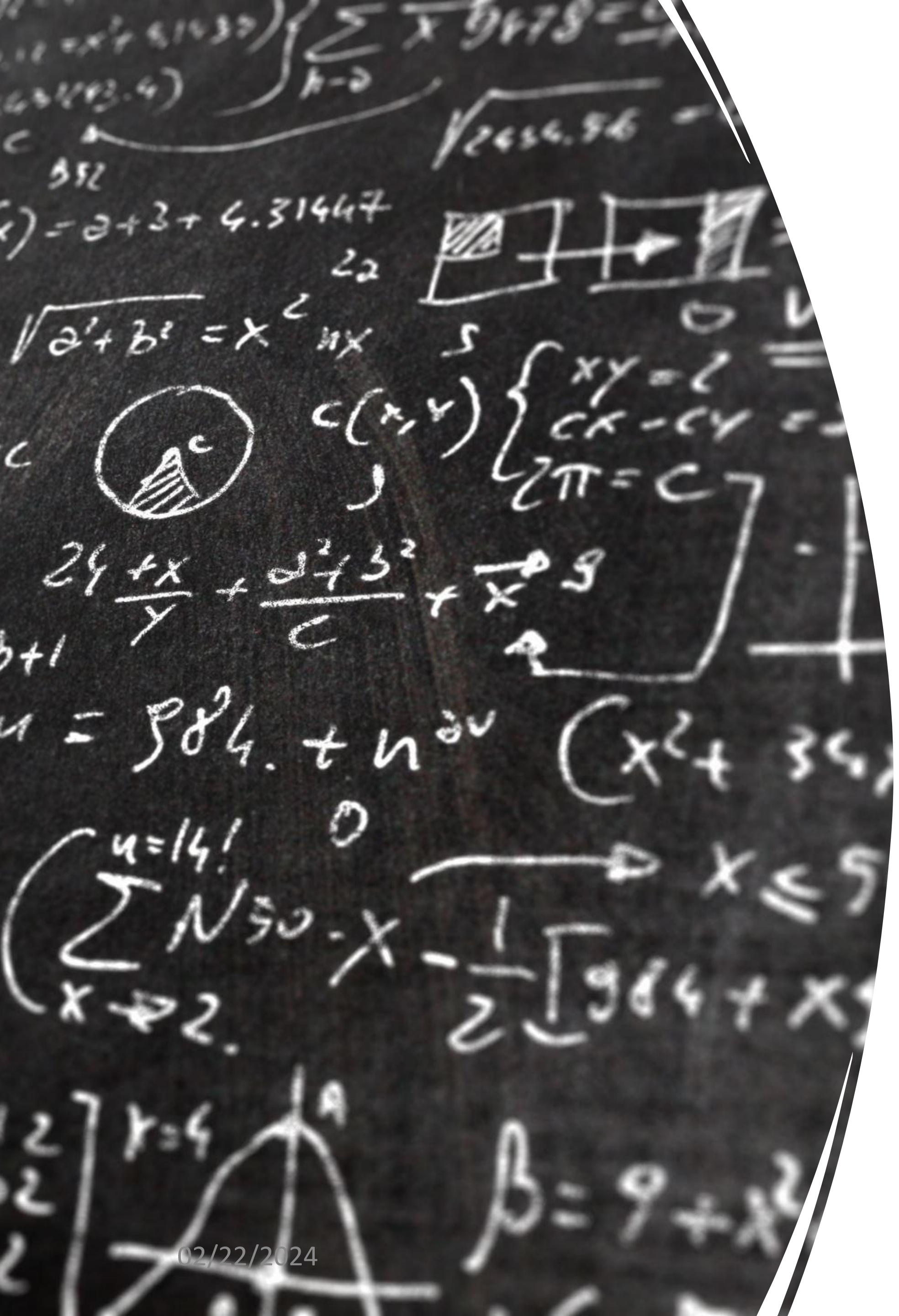


CS 113 – Computer Science I

Lecture 10 – Recursion, Arrays and Loops

Thursday 02/22/2024

Announcements

- HW03 due tonight
- Isopsephy
    - https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html
    - index from 1

Agenda

Recursion - review

Arrays – reviews

Loops

Recursion Example – printVowels

Write a recursive function that prints just the vowels in a String

Arrays

Arrays

Three ways to initialize an array

1. With an initial value

```
int[] numbers = {1, 2, 5};
```

2. With allocated space, but uninitialized

```
int[] numbers = new int[3];
```

3. With an empty array reference

```
int[] numbers = null;
```

Array Indexing

Access individual elements of an array with indexing

    array[index]

We use *zero*-based indexing

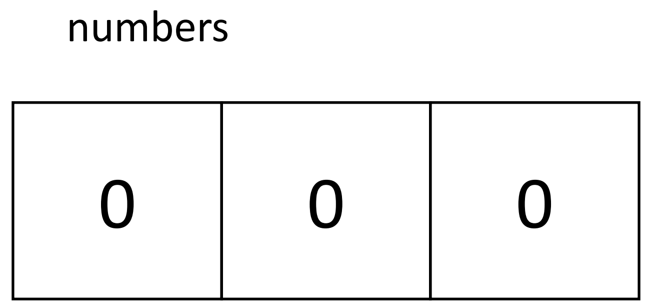
    first element is **0**

    last element is **length-1**

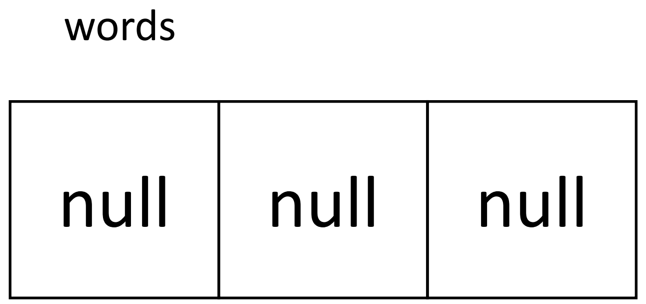
Accessing indices out of range results in a **runtime error**!

Arrays - default values

```
int[] numbers = new int[3];
```

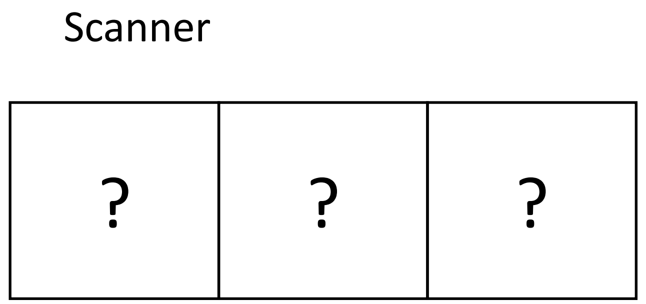


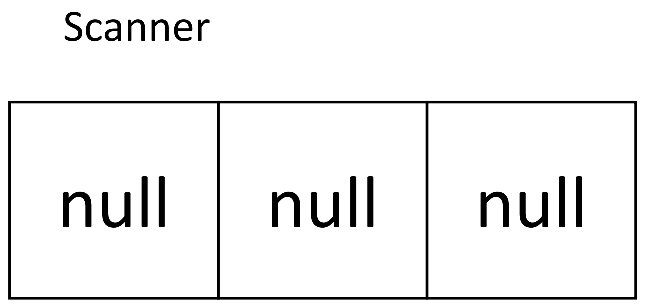
```
String[] words = new String[3];
```



Arrays - default values

```
Scanner[] words = new Scanner[3];
```





Arrays

```
int[] x = {2, 7, 5};
System.out.println(x.length); //what will this print?
```

.length field tells us how many elements are in the array

Once an array is full, you cannot add more elements to it!

Arrays

Implement a method `calculateSum` that takes an int array as a parameter and returns the sum of its elements

assume the array has 5 elements

Printing an Array

Let’s test our calculateSum method

Array Comparison

Strings and arrays are **NOT** primitives

They are objects

Explains why we can’t use “==“ to compare Strings

“==“ checks if two objects are the same

not if the two values are the same

Recursion Example – Boolean Array Negation

Implement a recursive method called `boolNeg` that takes a boolean array as input and returns a new array with each boolean value negated (e.g., `true` becomes `false` and vice versa).

Loops

Exercise

`calculateSum` with an unknown number of elements in arr

Loops

- Easy way to repeat some computation

- Two kinds of loops:
  - **While**
  - For

- Loops repeat block of code until the condition becomes false

While loop

While a condition is true, run a block of code

```
while(condition) {
  //run the code in this block
}
```

Tracing Loops

```
int sum = 1;
int count = 0;
while (count < 3) {
    sum = sum + 2;
    count = count + 1;
}
```

Count < 3	count	sum

Tracing Loops

```
int sum = 1;
int count = 0;
while (count < 3) {
    sum = sum + 2;
    count = count + 1;
}
```

Count < 3	count	sum
T	0	1

Tracing Loops

```
int sum = 1;
int count = 0;
while (count < 3) {
    sum = sum + 2;
    count = count + 1;
}
```

Count < 3	count	sum
T	0	1
T	1	3

Tracing Loops

```
int sum = 1;
int count = 0;
while (count < 3) {
    sum = sum + 2;
    count = count + 1;
}
```

Count < 3	count	sum
T	0	1
T	1	3
T	2	5

Tracing Loops

```
int sum = 1;
int count = 0;
while (count < 3) {
    sum = sum + 2;
    count = count + 1;
}
```

Count < 3	count	sum
T	0	1
T	1	3
T	2	5
T	3	7

Tracing Loops

```
int sum = 1;
int count = 0;
while (count < 3) {
    sum = sum + 2;
    count = count + 1;
}
```

Count < 3	count	sum
T	0	1
T	1	3
T	2	5
T	3	7
F	3	7

Example

rewrite `calculateSum` with a loop

Example

rewrite `ArrayEq` with a loop

Exercise: Tracing loops

```
int sum = 10;
int count = 0;
while (count < 6) {
    sum = sum - 1;
    count = count + 2;
}
```

Count < 6	count	sum

Exercise: Tracing loops

```
int sum = 10;
int count = 0;
while (count < 6) {
    sum = sum - 1;
    count = count + 2;
}
```

Count < 6	count	sum
T	0	10
T	2	9
T	4	8
T	6	7
F		

Accumulator pattern

Idea: Repeatedly update a variable (typically in a loop)

Pattern:

1. Initialize accumulator variable
2. Loop until done
    1. Update the accumulator variable

Convenient Assignment Syntax

Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

AssignSyntax.java

Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

sum = sum + 2	
count = count + 1	
count = count - 1	
product = product * 2	
divisor = divisor / 2	
message = message + " lol"	

Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

sum = sum + 2	sum += 2
count = count + 1	
count = count - 1	
product = product * 2	
divisor = divisor / 2	
message = message + “ lol”	

Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

sum = sum + 2	sum += 2
count = count + 1	count += 1
count = count - 1	
product = product * 2	
divisor = divisor / 2	
message = message + “ lol”	

Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

sum = sum + 2	sum += 2
count = count + 1	count += 1
count = count - 1	count -= 1
product = product * 2	
divisor = divisor / 2	
message = message + “ lol”	

Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

sum = sum + 2	sum += 2
count = count + 1	count += 1
count = count - 1	count -= 1
product = product * 2	product *= 2
divisor = divisor / 2	
message = message + “ lol”	

Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

sum = sum + 2	sum += 2
count = count + 1	count += 1
count = count - 1	count -= 1
product = product * 2	product *= 2
divisor = divisor / 2	divisor /= 2
message = message + “ lol”	

Convenience syntax: Assignment

Because updating variable values is so common, language such as Java provide shorthand syntax for it

- Analogy: contractions in English

sum = sum + 2	sum += 2
count = count + 1	count += 1
count = count - 1	count -= 1
product = product * 2	product *= 2
divisor = divisor / 2	divisor /= 2
message = message + “ lol”	message += “ lol”

Exercise: Write a program that computes powers of 2

Write a program, LoopPow2.java, that computes powers of twos. For example,

```
$ java LoopPow2
Enter an exponent: 0
2 to the power of 0 is 1

$ java LoopPow
Enter an exponent: 1
2 to the power of 1 is 2

$ java LoopPow
Enter an exponent: 4
2 to the power of 4 is 16
```

Exercise: Non-recursive blast off

take a number from the user, count down from that number to 0 and then print “BLAST OFF!”

Exercise: Non-recursive Factorial