WELCOME TO
CSCE 221: DATA STRUCTURES
REVIEW

COMPUTING, DATA, AND MEMORY
COMPUTER SCIENCE

• Study of algorithms
• Study of computing tools

• It is not just:
  • Programming
  • Electronics
  • Etc.

• In this class, we formalize this study of algorithms through the basics of data structures – a bread-and-butter component of almost all algorithms
PSEUDOCODE

• High-level description of an algorithm
• More structured than English prose
• Less detailed than a program
• Preferred notation for describing algorithms
• Hides program design issues

• Basic rundown
  • Use common math notations
  • Use ← vs = for assignment
  • Do not use (), {}, ;, ;, etc.
  • Let indenting denote scope
  • Use objects and functions without having to define them

• Look at my website LaTex tutorial for more info
1. One variable per data element – does not associate data together and can be very verbose
2. Arrays – group a large amount of data all of the same type
3. Objects – group a large amount of data all of different types
• **Memory** is storage for data and programs

• We will pretend that memory is an infinitely long piece of tape separated into different cells

• Each cell has an address, i.e., a location, and a value

• In the computer these values are represented in binary (0s and 1s) and addresses are located in hexadecimal (base 16, 0x)
• We will review arrays in Java later today

• **Arrays** are a sequential piece of memory all of the same type

---

**Array**

<table>
<thead>
<tr>
<th>0xA</th>
<th>0xB</th>
<th>100</th>
<th>...</th>
<th>x</th>
<th>y</th>
<th>...</th>
<th>z</th>
<th>...</th>
</tr>
</thead>
</table>

**Array A**

<table>
<thead>
<tr>
<th>0xB</th>
<th>0xB+99</th>
</tr>
</thead>
</table>

• **Pointer** (e.g., Java reference) – a variable that stores a memory location
Memory

Objects

• We will review objects in Java and learn new concepts/syntax about objects tomorrow.

• Objects are entities in your program. Another way to think about them is that they are collections of data of unassociated types.

• Objects are stored as pointers in Java, always.
BASIC COMPUTER ORGANIZATION

Central Processing Unit (CPU)
- Processes commands as 0’s and 1’s
- Performs arithmetic
- Requests (reads) and writes to/from memory

Input
- Files
- Keyboard
- Mouse
- Etc.

Output
- Monitor
- Force feedback
- Files
- Etc.

Memory
- Data encoded as 0s and 1s
- Cache
- Random Access Memory (RAM)
- Hard drive

Bus
TAKEAWAYS ABOUT MEMORY

• Programs can operate more efficiently when data is close together, e.g., arrays. This is called locality of data. The reason it works better is the cache.

• Pointers are not usually located close to the data. They hurt locality.
CH3.
FUNDAMENTAL DATA STRUCTURES

ACKNOWLEDGEMENT: THESE SLIDES ARE ADAPTED FROM SLIDES PROVIDED WITH DATA STRUCTURES AND ALGORITHMS IN JAVA, GOODRICH, TAMASSIA AND GOLDWASSER (WILEY 2016)
CH 3.1 ARRAYS
ARRAY DEFINITION

• An array is a sequenced collection of variables all of the same type. Each variable, or cell, in an array has an index, which uniquely refers to the value stored in that cell. The cells of an array, $A$, are numbered 0, 1, 2, and so on.

• Each value stored in an array is often called an element of that array.
ARRAY LENGTH AND CAPACITY

• Since the length of an array determines the maximum number of things that can be stored in the array, we will sometimes refer to the length of an array as its capacity.

• In Java, the length of an array named `a` can be accessed using the syntax `a.length`. Thus, the cells of an array, `a`, are numbered 0, 1, 2, and so on, up through `a.length-1`, and the cell with index `k` can be accessed with syntax `a[k]`. 
DECLARING ARRAYS (FIRST WAY)

• The first way to create an array is to use an assignment to a literal form when initially declaring the array, using a syntax as:

```
ElementType[] arrayName =
    {initialValue₀, initialValue₁, ..., initialValueₙ₋₁};
```

• The `ElementType` can be any Java base type or class name, and `arrayName` can be any valid Java identifier. The initial values must be of the same type as the array.
DECLARING ARRAYS (SECOND WAY)

- The second way to create an array is to use the `new` operator.
  - However, because an array is not an instance of a class, we do not use a typical constructor. Instead, we use the syntax:
    ```
    new ElementType[length]
    ```
  - `length` is a positive integer denoting the length of the new array.
  - The `new` operator returns a reference to the new array, and typically this would be assigned to an array variable.
ARRAYS OF OBJECTS

- Recall an array of objects is an array of pointers to objects.
ADDING AN ENTRY

• To add an entry $e$ into array $A$ at index $i$, we need to make room for it by shifting forward the $n - i$ entries $A[i], ..., A[n-1]$

Add
Input: Array $A$, index $i$, element $e$
1. for $k \leftarrow n, n-1, ..., i + 1$
2. $A[k] \leftarrow A[k-1]$
3. $A[i] \leftarrow e$
4. $n \leftarrow n + 1$
REMOVING AN ENTRY

To remove the entry \( e \) at index \( i \), we need to fill the hole left by \( e \) by shifting backward the \( n - i - 1 \) elements \( A[i + 1], ..., A[n - 1] \)

Remove

**Input:** Array \( A \), index \( i \), element \( e \)

1. for \( k \leftarrow i + 1, ..., n - 1 \)
2. \( A[k - 1] \leftarrow A[k] \)
3. \( A[n - 1] \leftarrow \text{null} \)
4. \( n \leftarrow n - 1 \)
EXERCISE

• With a partner, write an algorithm in pseudocode to compare the equality of two arrays $A$ and $B$. Use '==' for equality checking in pseudocode, not '=='. 