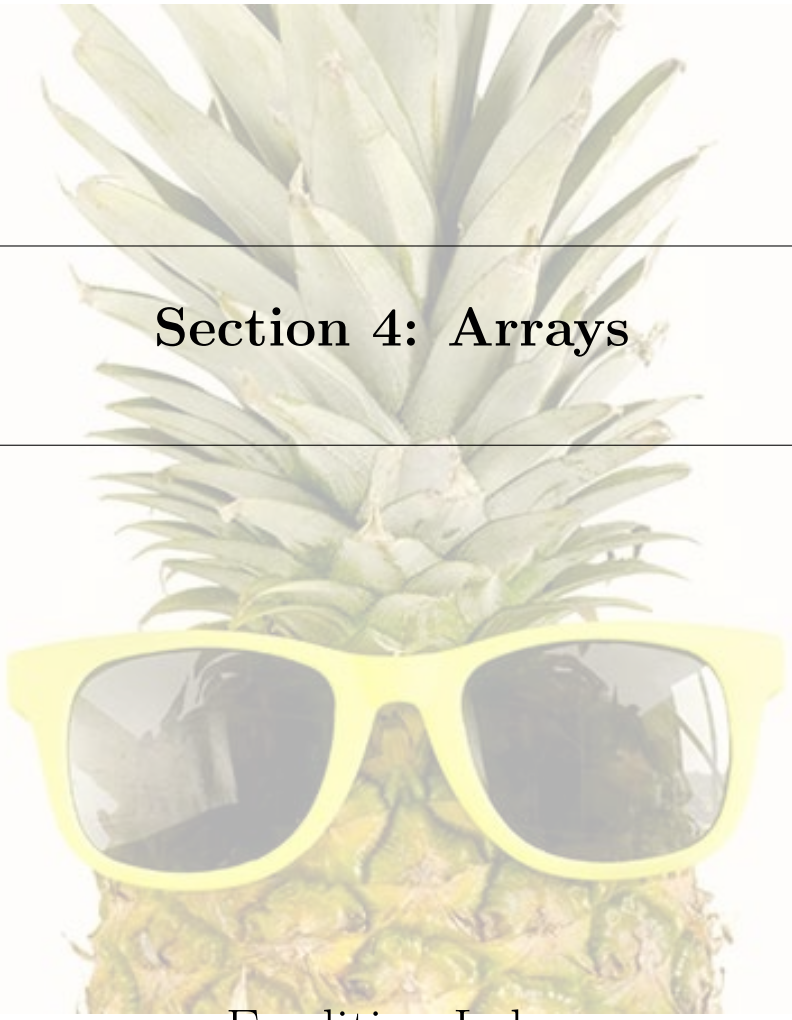
Section 4: Arrays



Erudition Labs

Computer Science 101: Introduction to Java and Algorithms

May 8, 2019

Contents

1 Terminology

- ***Memory*** – Memory is where all the action happens. Many people often refer to it as storage space although that is not entirely true. When programmers talk of memory, we are talking about addressable memory. This means memory that computers can create addresses for and thus use. Memory is NOT hard drive space. It is RAM. The amount of memory you have depends on two things. The amount of RAM you have and the operating systems you are using (eg. 32 bit windows vs 64 bit windows). With 32 bit systems, you are limited to 4 GB of ram where as 64 bit machines are (theoretically since we haven't done it) limited to 16.8 million terabytes of RAM. Everything happens in your memory. When you start up your computer, windows (or mac, linux, android, IOS...whatever) gets loaded into memory. Whenever you launch a program, it gets loaded into memory. Pretty much whenever you do anything on a computer, it's done in memory. As a programmer, you generally deal with two types of memory, heap and stack. Refer to subsection 2.1 to lean more about heap vs stack.
- ***Data Structure*** – Data structures is basically a term to describe a way of organizing data. We create certain rules for how we store and organize data so that we can take advantage of its organization when performing operations like insertion, retrieval, and searching. Some example data structures are arrays, linked lists, binary trees, hash maps and vectors. Feel free to look any up if you are curious. One of the simplest to understand is the linked list.
- ***Collections*** – In many programming languages we often come across the terms collections and containers. In Java, we only deal with collections. A collection is a way of grouping data together under the same name. I would recommend `<https://www.geeksforgeeks.org/collections-in-java-2/>` if you are curious about more.
- ***Array*** – A collection of data elements of the same type stored in a contiguous chunk of memory.
- ***Elements of an array*** – We use this phrase to talk about all of the pieces of data stored in an array.
- ***Iterate over an array (loop over an array)*** – We use these phrases when we are talking about using one of the loops, like a for or while loop, to look at each element in an array (one each iteration of the loop).
- ***Random access*** – This means we can access an arbitrary piece of memory. In reference to arrays, it means that we can access any arbitrary element in an array as long as it exists. You won't hear this term too much until you learn about algorithm efficiency.
- ***Index*** – The position of something, often in an array-like collection. In reference to arrays, the index is the position that an element is stored at in the array.
- ***Index into an array*** – We are using the index (aka the position of an element in an array) to get the value of the element at that position in the array.

2 Pre-Chapter

2.1 Heap and Stack Memory(Over Simplified)

2.2 The “new“ keyword

2.3 Declaration vs. Initialization

3 Arrays (Video Series Lecture 20 and 21)

This section introduces Arrays, which is a data structure that stores collections of elements. What is a data structure? Well a data structure is just a way organizing data by following certain rules when we store and retrieve it. By doing this, we can take advantage of the way the data is organized to make efficient algorithms. Some data structures are better at some things than others. It all depends on what you need to do. So when you take a data structures class (or look into it yourself) you will learn about different data structures (aka different ways of organizing data), their rules, their advantages, disadvantages and their best use cases.

For example, if your program is going to be doing a lot of retrieving, there are data structures that are great for that. If you need to do a lot of saving or insertions, we have great data structures for that too, but those might also not be the best for retrieving. There are often trade-offs when using them and it is up to the programmer to decide which ones are best for your task. For now, you don‘t really need to know what a data structure is other than it‘s a way if organizing data. If you are taking computer science classes, then there are entire classes dedicated to that topic. All you need to know right now is that an array is used to store a collection of things of the same type and are extremely good for random access (meaning I can get any arbitrary piece of data at any time efficiently as long as I know where it is at in the array).

3.1 What does an Array Look Like?

Arrays are contiguous chunks of memory. And they MUST be contiguous due to how we access the data. I will talk about that more later. Usually when we try to visualize and conceptualize what an array is, we draw it like this:

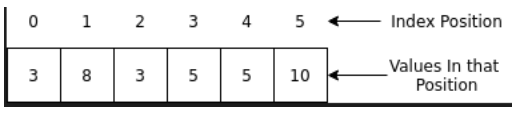


Figure 1: Image from Lecture

Imagine, if you will, having a bunch of variables of the same type under one name. This is essentially what

an array is. So if you need to store a collection of integers, you can declare one array to store them instead of declaring a bunch of different variables. If you look at the table, the top row corresponds to the index position of the array. The index is simply where that particular piece of data is stored at in the array. Also note that we refer to these "pieces of data" as elements. For example, if we look at the table above, we can see that 8 is an element of that array at index position 1.

Notice that we start counting from 0. The 0th element is actually the first element in the array. Next, you can see all the boxes (aka the elements). Think of each box as a variable that you don't have to name that stores a value. So in the image, we have an array of size 6 that stores 6 integers.

3.2 Declaring/Initializing an Array

We can declare an array just like we declare any other variable, in the form of

```
int myArray[];
```

4 Looping Over Arrays (Video Series Lecture 22 and 23)

5 2D Arrays (Video Series Lecture 24 and 25)

6 Sorting Arrays: Insertion Sort Algorithm (Video Series Lecture 26 and 27)

7 Gotchas with Arrays