ARRAYS

Recall that one variable has three aspects: its name, the variable itself which is a storage facility, and its current value. We also saw that each variable has a type.

An array is several different variables with the same type, where each one is named by a common array name combined with an integer number that is either zero or positive. Each of the different variables is called an element and the integer number that helps name a single element is called the element’s index or subscript. People often say and think that the elements are in the array.

When you want a program to make the computer do similar things with many different variables, it is very convenient to use one array of variables instead of many different separately named variables. That one array will actually be comprised of many different variables.

The first novel feature of an array is that the number used for an index or subscript can be derived from user input or be computed when the program is run. Unlike a simple variable’s name, the numeric index or subscript value does not have to be written explicitly into the code by the programmer! The index or subscript can come from a loop variable so the loop body accesses a different array element during each iteration of the loop.

Example: Code that computes the sum of an already existing array of doubles:

```java
double Sum = 0.0;
for( int i = 0; i < myArray.length; i++)
{
    Sum = Sum + myArray[i];
}
//HA! Now Sum contains the sum of all the numbers in array myArray.
```

Thought question: Arrays can easily have millions of elements, and ones with billions are possible; even more on 64-bit computers! The above code can add up millions or billions of numbers. How much time will it take for you to write a program that adds up the numbers in a million separately named simple variables? For only 12 it begins to look ugly:

```java
double varOne, varTwo, varThree, varFour, varFive, varSix, varSeven,
    varEight, varNine, varTen, varEleven, varTwelve;
    // code to set 12 values omitted.
double Sum;
Sum = varOne + varTwo + varThree + varFour + varFive + varSix + varSeven +
    varEight + varNine + varTen + varEleven + varTwelve;
```

The second novel feature of an array is that the number of elements in it (like 12 for the months of the year) might be coded specifically or literally by you, the programmer, when you write the program, but it does not have to be. In many languages like Java, the number of elements in an array can depend on user input given when a person runs the program\(^2\). It can depend on any data that the program reads or calculates when it runs. However, we will first cover the simplest pattern in which the number of array elements is fixed by the programmer but the values (i.e. data) stored in each of the array elements varies while the program runs.

**To use an array in your program**

**First**, you decide what type of variable each array element will be. Let’s agree that for the first example, that type will be Java’s double (numbers expressed with a decimal point that might have a fractional part, like 3.25, with or an exponent like 2.99792458e8 which is the speed of light in meters/sec.)

**Second**, you must declare a variable that will refer to the whole array. You must choose a name for that. Let’s agree to use the name `myArray` for the first example.
The code that declares the variable named `myArray` to be capable of referring to an array of `doubles` is:

```java
double[] myArray;
```

Pitfall: In Java, it is illegal to specify size here. Don't write `double[12] myArray;`

At this point, the array reference variable `myArray` will contain the `null` value. That means that `myArray` does not yet refer to an array. It only has the potential to. (In C and C++, a array reference is called an array pointer, and the pointer's value might not be valid for use even when it is not `null` or `0`\(^1\). The Java system is designed so this situation cannot happen; but in Java runtime errors can and often do happen because an array reference variable contains the `null` value.)

**Third,** you must decide how much should be the exact number or count of the different `double` variables in the array. In simple examples like this one, that number is a constant. Suppose we want one array element for each of the 12 months of the year. So this exact number or count will be 12 in our example. This number of different variable in an array is often called the array's `size` or `length`.

**Fourth,** you must write code that makes the array you want to come into being. Warning: That is NOT the same as the second step above. Just read on..

We now illustrate the third and fourth step, having finished with the first and second. The code that makes our example of an array of 12 doubles come into being together with enabling other code to refer to it with the name `myArray` is:

```java
myArray = new double[12];
```

The operation `"new double[12]"` makes the array come into being. The professional way to say this is the operation `"new double[12]"` instantiates the array.

The operation `"myArray = "` stores into the variable named `myArray` the reference returned by the `"new double[12]"` operation. Both instantiating the array and storing the returned reference into an array reference variable are necessary if you want your program to use an array.

Now, if you are lazy about using your keyboard, you can make the computer accomplish all these feats with a single, shorter statement. For our example, it is:

```java
double[] myArray = new double[12];
```

In Java, the size or length of an array can be retrieved by accessing the `length` field through the reference variable. For example, the following line will print 12, the length of the array that had been instantiated by the operation `"new double[12]"

```java
System.out.println( myArray.length );
```

Pitfall: That's `myArray.length`, not `myArray.length()` ! Go complain to the developers of Java if you don't like this\(^4\).

**Complete Example:** Write an application that reads the 12 temperatures measured on the first day of each month of the year, and then prints (1) their average and (2) their median.

```java
import java.util.Scanner;
class monthlyTempAnalyzer
{
    public static void main(String[] args)
    {
        double myArray[] = new double[12];
        //First through fourth steps in one line.
        Scanner sc = new Scanner(System.in);  //Construct a Scanner for input.
```
for(int i = 0; i < 12; i++)
{
    System.out.print("Temp for month " + (i+1) + " please:");
    myArray[i] = sc.nextDouble();
}
double Sum = 0.0;
for(int i = 0; i < 12; i++)
{
    Sum = Sum + myArray[i];
}
double Average = Sum / 12;
System.out.println("The average is " + Average + " degrees.");
//Now, sort the array to help find the median.
//The algorithm implemented below is called selection sort.
for( int i = 0; i < 11; i++)
{
    for( int j = i+1; j < 12; j++ )
    {
        if(myArray[i] > myArray[j])
        {
            double temporary = myArray[i];
            myArray[i] = myArray[j];
            myArray[j] = temporary;
        }
    }
}
//Since 12 is even, the median is the average of the two numbers
//straddling the middle of the sorted sequence.
double median = (myArray[5] + myArray[6])/2.0;
System.out.println("The median is " + median + " degrees.");
}

1. The number of different variables of an array might be only one; and in some languages, even zero. Allowing these options without restriction is an example of the quality of generality. Generality is a quality of programming languages and other things in computer science that helps makes them easier to use.

2. An array whose size or length is myArray.length can be determined at run time like this is called a dynamically allocated array. In Java, all arrays are dynamically allocated. In C or C++, dynamically allocated arrays are provided by malloc, calloc or new (new is in C++ much as in Java), but other kinds of allocations are available (static and local).

3. Indeed, in C or C++, the above code declaring myArray, if it is inside a function body, will NOT guarantee that myArray initially contains the null value 0. In Java the compiler will detect if a variable like myArray is undefined when it is used.

4. Java arrays are built into the core Java language, they are not provided from a library of classes.
An array can be an array of (references to) objects

ABSTRACTION OF MEMORY

Beyond separately named variables, the array is the among the most important concepts for programmers to understand because the array is a quite realistic simplification of the memory in computers.

- Understanding the memory nature of arrays will make you understand why there are some restrictions on programming with arrays and how efficient some array operations can be, so that it will be much easier for you to remember important rules and techniques.
- Familiarity with array programming will translate into understanding important aspects about how the whole computer works so that you can anticipate the effects on computer speed of different choices you make in programming them.

We can say that arrays are a “thin abstraction” of computer memory.

Array allocations have fixed size.

Java keeps track of array sizes.

Expanding arrays