**C++ vs. Java Arrays**

* Java itself is based on the C++ language but has several distinct differences. How arrays are implemented is one such difference.

**Initialization**

* A Java array is quite different then a C++ stack-allocated (fixed) array.
* It is, however, essentially the same as a C++ dynamic array (pointer to an array allocated on the ***heap****)*.

That is,

int[] a = new int[100]; // Java initialization

is not the same as

int a[100]; // C++ fixed array initialization

but rather, it is equivalent to

int\* a = new int[100]; // C++ dynamic array initalization

in other words,

Java Array == C++ Dynamic Array

int[] a = new int[100]; == int\* a = new int[100];

**Implementation**

**C++ Arrays Are Primitives**

* An array in C++ is basically a pointer to the address of a variable.
* An array itself is a sequence of contiguous blocks of allocated memory.
* Access to subsequent elements is done through pointer arithmetic.

**Java Arrays Are Objects**

* An array in Java is an object.
* It has instance variables and functions.
* Since an array in Java is an object, they must be dynamically declared.
* You cannot allocate Arrays onto the stack because it is an object, and all objects must be dynamically declared onto the heap in Java.
* Furthermore, there is no pointer arithmetic—you can’t increment a to point to the next element in the array. This follows suit, because C++ array use pointer arithmetic to implement the functionality of arrays—whereas Java arrays are higher-level objects.

**Size**

**Java Array Size**

* Since an array in Java is an object, it may have instance fields.
* Every Java array has a field defined as **public int length** that stores the size of (number of elements in) an array.

A picture containing text

Description automatically generated

**C++ Array Size**

* C++ programmers must maintain a separate, distinct variable to track the size of an array.

**Bounds Checking**

* Arrays, in both the C++ and Java programming languages, are zero-indexed, meaning that legal index values range from 0 to size - 1 (where size is the maximum number of elements in the array).

**C++ Bounds Checking**

* C++ offers no bounds checking.
* If you were to index an out-of-bounds element, C++ would do nothing to stop you.
* This of course, would result in undefined behavior.

**Java Bounds Checking**

* If a program attempts to index an out-of-bounds element, the Java virtual machine (JVM) does and aborts the program by throwing an exception, **IndexOutOfBoundsException**.
* While bounds checking in Java makes your code less error prone, it does come with its share of overhead.
* Therefore, C++ programs will always outperform similar Java programs in this regard.

**Storage Allocation**

**C++ Storage Allocation**

* In C++, storage for a static array is allocated at compile time
* In C++, storage for a dynamic array is allocated on the heap at run time

C++

int A[10]; // A is an array of length 10

A[0] = 5; // set the 1st element of array A

**Java Storage Allocation**

* Unlike C++, Java arrays cannot be allocated onto the stack.
* Java arrays are a special type of object, hence they can only be dynamically allocated via "new” operator, and are therefore allocated on the heap at runtime.
* In Java, when you declare an array, you are really only declaring a pointer to an array. Storage for the array itself is not allocated until you use "new".

JAVA

int [] A; // A is a pointer to an array

A = new int [10]; // now A points to an array of length 10

A[0] = 5; // set the 1st element of the array pointed to by A

**Arrays of Objects**

In C++ it is possible to create an array of fully constructed objects in a single operation; Java requires multiple operations to create an array of objects

Sources:

<https://people.eecs.ku.edu/~jrmiller/Courses/JavaToC++/StackAllocatedArrays.html>

<https://pages.cs.wisc.edu/~hasti/cs368/JavaTutorial/NOTES/Java_vs.html#arrays>

<https://icarus.cs.weber.edu/~dab/cs1410/textbook/7.Arrays/cpp_v_java.html>

<https://docs.oracle.com/javase/specs/jls/se7/html/jls-10.html>