Integers

An integer is used to hold numeric values. An integer can either be signed (holds both positive and negative numbers), or unsigned (will only hold positive numbers). Integers are represented in binary and an essential representation will need 4 bytes.

In this reading, you will learn about the integer data structure, such as how integers are represented, memory allocation for integers and the integer wrapper class in Java.

Integer representation

There are several ways to represent integers in binary. One typical example is sign-magnitude. If integers are represented in binary, how can they distinguish between a positive and negative value? Sign-magnitude proposes using an indicator on the far left of the binary number to denote polarity.

Integer	Sign-magnitude representation
2	0010
1	0001
+0	0000
-0	1000
-1	1001
-2	1010

An integer cannot represent fractions. For this, one would use decimal or float. A fixed number of bytes is used when representing integers, the size of which can be specified in some languages. There is a standardized approach for representing numbers called the IEEE 754 standard, which outlines a common set of standards for representing all numbers.

Some high-level languages like Python and JavaScript subscribe to this approach and encapsulate the initialization of integers to this fixed representation. This makes working with numbers easy in these high-level dynamic languages but removes the ability to customize your approach to enable memory optimization.

Memory allocation

Other statically typed languages like C++, Rust, Ada and C allow you to customize the size in memory. C++ allows you to customize your integer. Using `unsigned short int` will only take 2 bytes. This savings can mount up if your application includes many positive numbers that do not require precision to the degree of fractions.

Rust goes one better and enables the instantiation of unsigned 1-byte integers. This limited `int` can hold integers from 0 – 255. While that may not seem like a massive range to deal with, if you are working with pixels, you would be working with a tuple of values in this range. Image processing is a highly CPU-intensive process and being able to store 8 pixels for the standard price of 1 is a massive saving.

Wrapper classes

In addition to primitive integers denoted by `int`, Java allows you to wrap the value of the primitive into a wrapper class `Integer`. This allows several methods for dealing with integers, such as converting from a string to a double, comparisons, maximum and minimum size and so on. The integer class is immutable, which makes it thread-safe. The extra functionality and safety incur a memory cost, and `Integer` object will take 16 bytes of memory to store.

Conclusion

There is enormous freedom in controlling how your figures are stored in memory. While a high-level dynamically typed language gets up and running, it may have limited returns if your application becomes resource heavy. Suppose you are working with Arduinos or other hobby electronics. In that case, space becomes a premium, and the freedom to customize may be the difference between a functional and non-functional application. However, if you are developing an application that requires substantial amounts of available memory, using integers quickly and not having to worry about the detail may be the way to go.

In this reading, you learned about the integer data structure, such as how integers are represented, memory allocation for integers and the integer wrapper class.