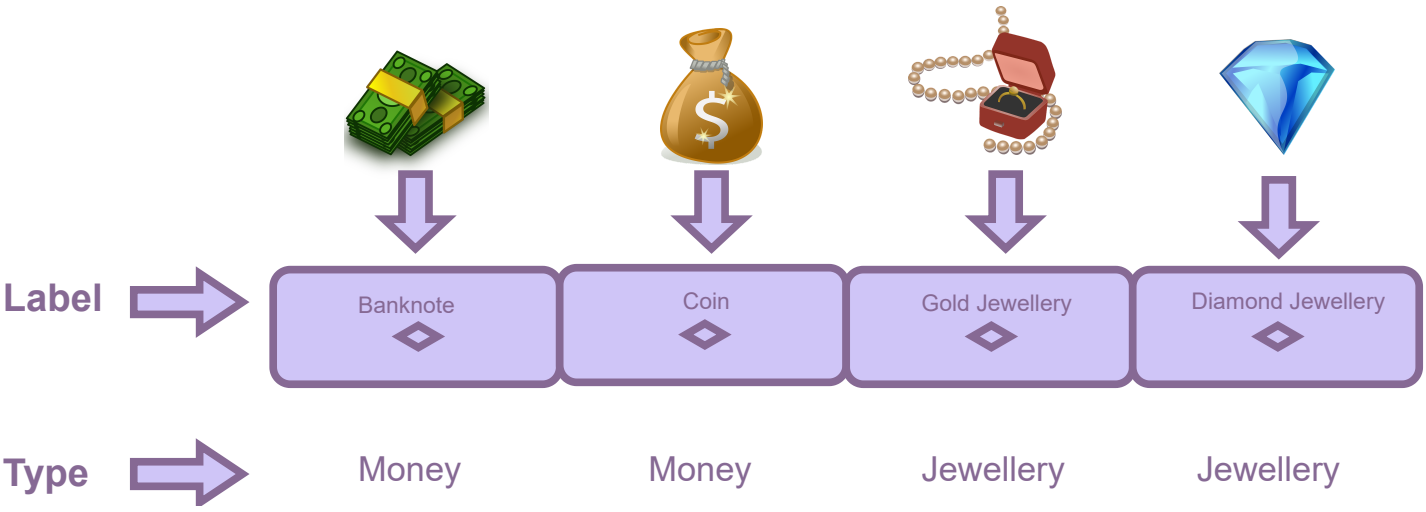
Introduction to Arrays

In this lesson, you will be introduced to arrays.

We'll cover the following

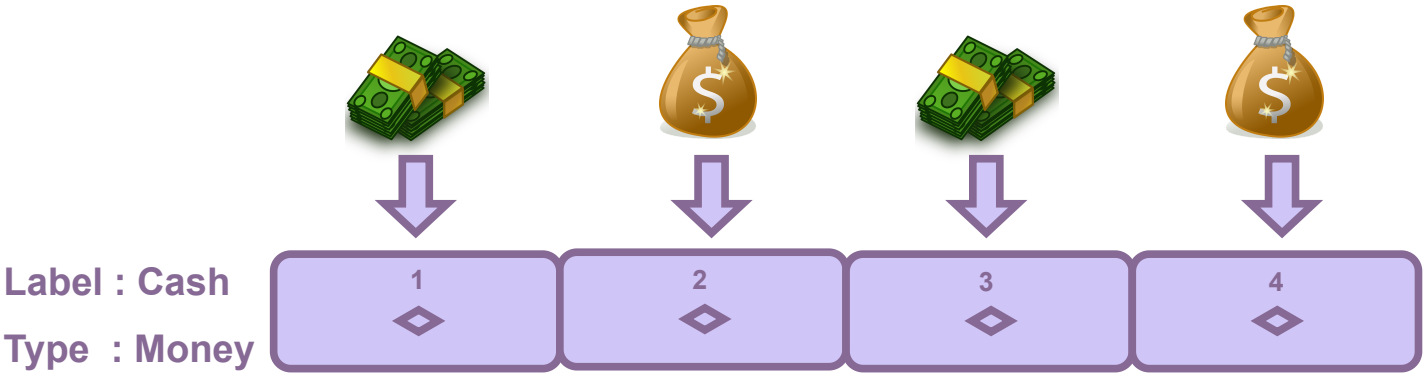
What is an array?

In the variable lesson, we saw that a variable is just like a cabinet that can store one item only. To store the item in the cabinet, we must decide its type (analogous to the data type) and put a unique label on it (analogous to variable name).



If we have to store a lot of items of the same type, then putting a label on each cabinet is quite a tedious task. So, can't we just store the items of the same type under the same label?

Yes, we can. Here is where arrays come in.



Array analogy

In the above figure, we have stored the items of the same type under a single label `Cash`.

---

*An **array** is a sequential collection of values of the same data type under the same name.*

> **i** An array is a derived data type.

Basic terms

Let's introduce the basic terms associated with an array.

Element

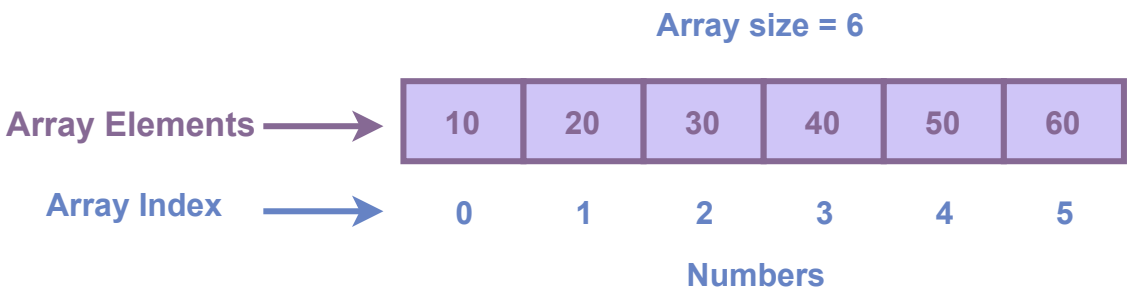
The array element is a **value stored in an array**. Elements in an array are stored at neighboring memory locations.

Index

An array index **identifies the position of an element** in an array. It starts from 0 and increments by one for each element added in an array.

Size

The size of an array is the **total number of elements** stored in an array.



Array representation

In the above figure, you can see an array with 6 elements. The name of an array is `Number`, and its size is **6**.

- The first element, **10**, is stored at index `0`.
- The second element, **20**, is stored at index `1`.
- The third element, **30**, is stored at index `2`.
- The fourth element, **40**, is stored at index `3`.
- The fifth element, **50**, is stored at index `4`.
- The sixth element, **60**, is stored at index `5`.

Why use arrays?

We have seen fundamental data types such as **int**, **long**, **char**, etc. The limitation of these data types is that they can store one value at a time. When we have large volumes of data, we need a data type that can store and access different amounts of data under a single name.

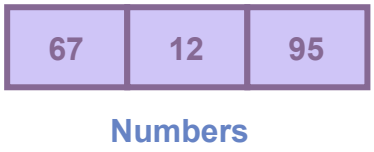
For example

Suppose there are **100** students in a class, and you want to store their roll numbers. Declaring **100** variables and then storing the roll number of each student is quite an impractical approach. Here, arrays come in handy!

Quiz

Q

In the figure given below, array `Number` can store **3** elements. What is the value of the element stored at index **1**?



Retake Quiz

In the upcoming lesson, you will see how to create arrays in C++.