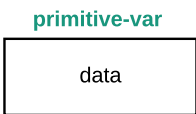


C# - Value and Reference Types

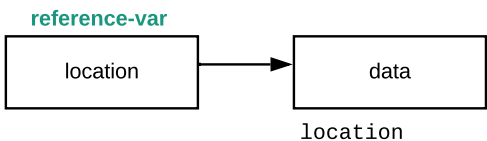
A **primitive type** is a variable whose name has the data assigned to it and are known as **value types**

int, double, bool, char, float, long, string (i.e. all standard C# data types) are example of primitive types

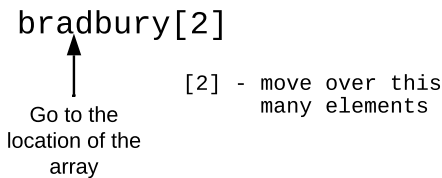
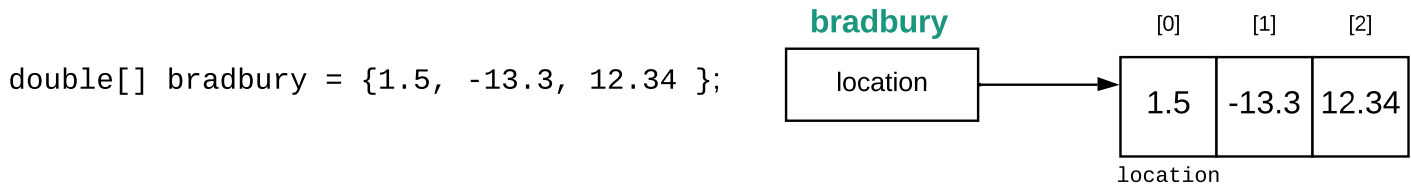
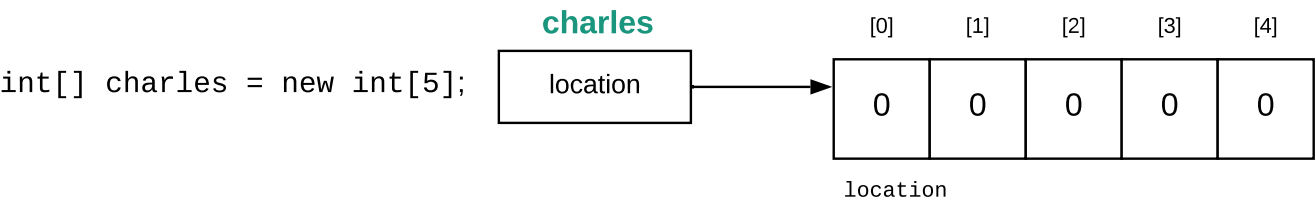
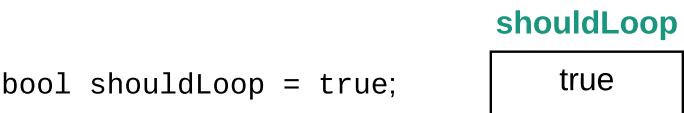
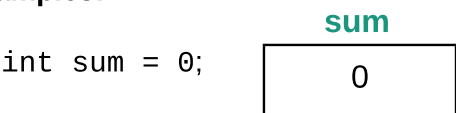


A **reference type** is a variable whose name contains the **location of the memory allocated for the data rather than the data itself**.

Arrays and **objects** are example of **reference types**



Examples:



Heap vs Stack

Stack - a memory area assigned to every process/app running in an operating system. the default size of the stack is determined by the language and the operating system the size of the stack can be changed (*but we don't show you how to do that*)

The Stack contains **non-reference variables, arguments for functions, return values, literals**, the **location part of a reference variable** and any **other temporary storage areas C# might need**.

Since the Stack is fixed in size when a process starts, it could become full and not be able hold anymore data. When that happens you get a Stack Overflow error.

C# has mechanism to clean up data in the Stack that is no longer needed in the process called "**Garbage Collection**".

When a C# program enters a block, primitive variables are created in the Stack. When the programs leaves the block the variables are removed from the Stack. **This is why you cannot reference variables outside the block in which they are defined.**

The concept of **Scope** is used to describe when a variable is available and accessible to a program statement.

A variable is in scope for block in which it is defined and any block that block is in.

The Heap - Any memory in the machine not already allocated to program code, stacks or operating system stuff.

Programmers and programming languages are responsible for managing memory in The Heap.

If memory in The Heap is no longer needed and its not released by the programmer or language, it cannot be reused == **Memory Leak**

Memory Leaks waste memory, slows down processing, generally not good.

C# **Garbage Collection** handles this for us.

Reference variable data is placed in The Heap.

```
double[] bradbury = {1.5, -13.3, 12.34 };
```

