**Data Structures:**

A variable is a named location in memory which stores a data value.

A data structure is a collection of memory locations, which are joined together.

**Operations on Data Structure:**

* Appending – Add a new data value to the end of the data structure
* Inserting – Add a new data value somewhere else in the data structure
* Editing – changing the value of one data item
* Deleting – Removing a data value
* Traversing – visiting every data item in turn. This is done when searching for an item or printing out every item in the data structure.
* Sorting – rearrange the data items into alphabetical or numerical order.

**The Array:**

An array is a fixed series of memory locations. Each location can store a value or be empty.

The computer can access any of the locations in a single operation. The size of the array does not change.

**Array Operations:**

* Set up an empty array of *n* elements.
* Add a value to any of the empty slots.
* If you know the index number of an element, you can access it in a single operation.
* If you don’t know the index number of the value you are looking for, you will have to search the array. You can traverse the entire array with a ‘for loop.

**Array VS List:**

|  |  |  |
| --- | --- | --- |
| **Features:** | **Array:** | **List:** |
| **Size** | Fixed when the array is made | Size changes as you add or delete elements |
| **Elements:** | Each element always keeps the same index number | Index numbers can change as the list grows or shrinks |
| **Empty Spaces:** | Can have empty spaces | No empty spaces in the list |

**Advantages of an Array:**

* Compared to a stack or queue you can access any value, stored anywhere in the array.
* Once a value has been put into the array, it stays in that numbered memory location.
* The computer can access any value in the array in one operation if it knows the index number
* The computer can search and sort the array. This can take more operations.

**Disadvantages of an Array:**

* Every time you delete an element, you leave an empty space. This can lead to a lot gaps, which is a waste of space.
* When you add an element, the computer will look through for an empty slot, which is a waste of time.
* If you need to insert an element at a particular point in the array, you will have to move all the other elements along to make space. This is very slow.

**Visualising an Array:**

Make an Array:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|  |  |  |  |  |  |  |  |  |

Append:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| “A” | “B” |  |  |  |  |  |  |  |

Insert:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| “A” | “B” |  |  | “E” |  |  |  |  |

Delete:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| “A” |  |  |  | “E” |  |  |  |  |

**Inserting a value into an Array:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| “A” | “B” | “C” | “D” | “E” | “F” | “G” |  |  |

Let’s say you wanted to insert the value “X” at position array[0]

You wouldn’t be able to because array[0] is currently storing the value “A”

So you have to move every value up by one in order to make a free memory location for “X”

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
|  | “A” | “B” | “C” | “D” | “E” | “F” | “G” |  |

Now you can insert the new value:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| “X” | “A” | “B” | “C” | “D” | “E” | “F” | “G” |  |