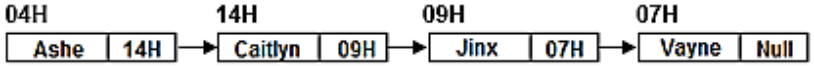
Linked Lists

Linked List Basics

- A **linked list** is used for storing a collection of data where each element is a separate object.
- Elements in a linked list are called **nodes**.
- The parts of a node are the following:

Data	Pointer

  - **Data field** – This contains the value of the element.
  - **Pointer field** (link or reference) – This contains the address (random memory location) of the next node.
- The next node in the list is referred to as the **successor**.
- The first node in the list is called **head**.
- The last node points to **null** since there are no more successive elements.



- A linked list is illustrated by:
  - Placing the address of the node above its data field
  - Placing the address of the next node in the node's pointer field
  - Indicating *null* in the pointer field of the last node
  - Connecting the previous node to the next node using an arrow to the right.
- Operations of a linked list:
  - **Display** – shows the elements in the list
  - **Insert** – adds an element into the list
  - **Delete** – removes a specific element or all the elements from the list
  - **Search** – finds a specific element in the list
  - **Count** – returns the number of elements in the list

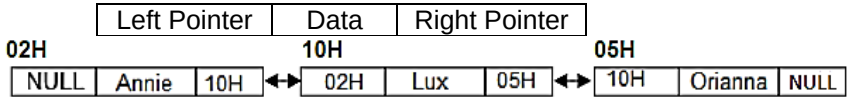
Linked List versus Array

- **Iteration** is the process of repeating a set of instructions. This is also known as "**looping**."

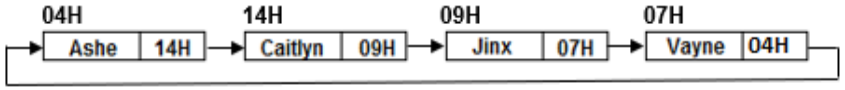
Linked List	Array
The number of elements can expand.	The number of elements is fixed upon creating the array.
It can grow and shrink during program execution.	The array size is specified during declaration.
The position of the elements is allocated during runtime.	The position of the elements is allocated during compilation.
Elements are sequentially accessed.	Elements are randomly accessed.
It utilizes memory efficiently.	Memory utilization is ineffective.

Types of Linked List

- **Singly linked list** – the basic linked list
- **Doubly linked list** – contains an extra pointer to connect to the previous node in the sequence. The **left pointer** contains the address of the preceding node called "**predecessor**."



  - A doubly linked list is illustrated by:
    - Placing the address of the node above its data field
    - Placing the address of the preceding node in the node's left pointer field
    - Placing the address of the next node in the node's right pointer field
    - Indicating *null* in the left pointer field of the first node and in the right pointer field of the last node.
- **Circular linked list** is a linked list in which the last node's right pointer contains the address of the first node.



References:
Karumanchi, N. (2017). *Data structures and algorithms made easy.* Hyderabad: CareerMonk Publications.
TechDifferences (n.d.). *Differences between array and linked list.* Retrieved from https://techdifferences.com/difference-between-array-and-linked-list.html