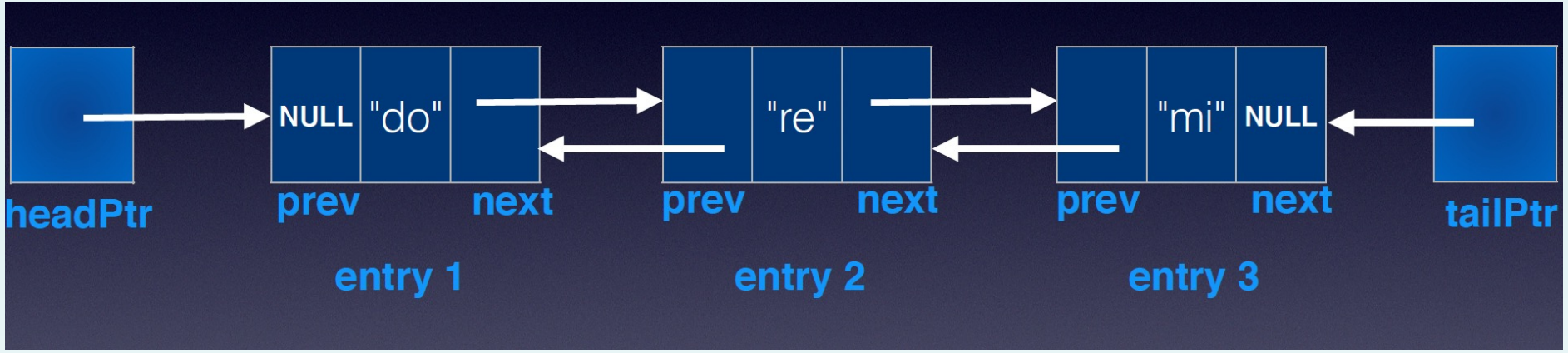
Doubly Linked Lists

By: Anita Rathi

Doubly Linked List

- A Doubly Linked List (DLL) contains
  1. previous pointer to point to previous node
  2. next pointer to point to next node
  3. Data

- It can be traversed in both ways –
  1. From first node to last node using the next pointer
  2. From last node to first node using the previous pointer



Doubly Linked List

- Sample node definition will go as follows:

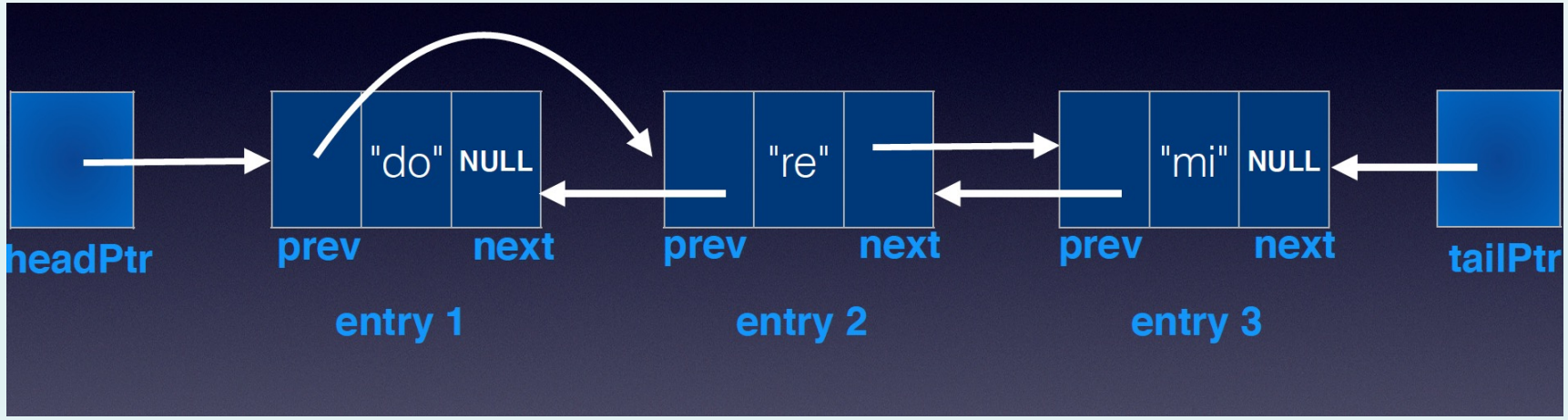
```
class Node {
public:
   int data;

   // Pointer to next node
   Node* nextPtr;

   // Pointer to previous node
   Node* prevPtr;
};
```

Comparison between Array based and Linked based Implementation

operation	array-based list	linked list	advantage
insertion / deletion	slow - can require shifting many items	fast	linked list
insertion into full list	very slow - requires resizing of array	fast	linked list
random access	fast	slow - requires traversal of list	array-based list
sequential access	fast, can benefit from cache	can be slower (limited cache benefit)	array-based list
storage efficiency	only stores the data	pointer overhead plus data	array-based list

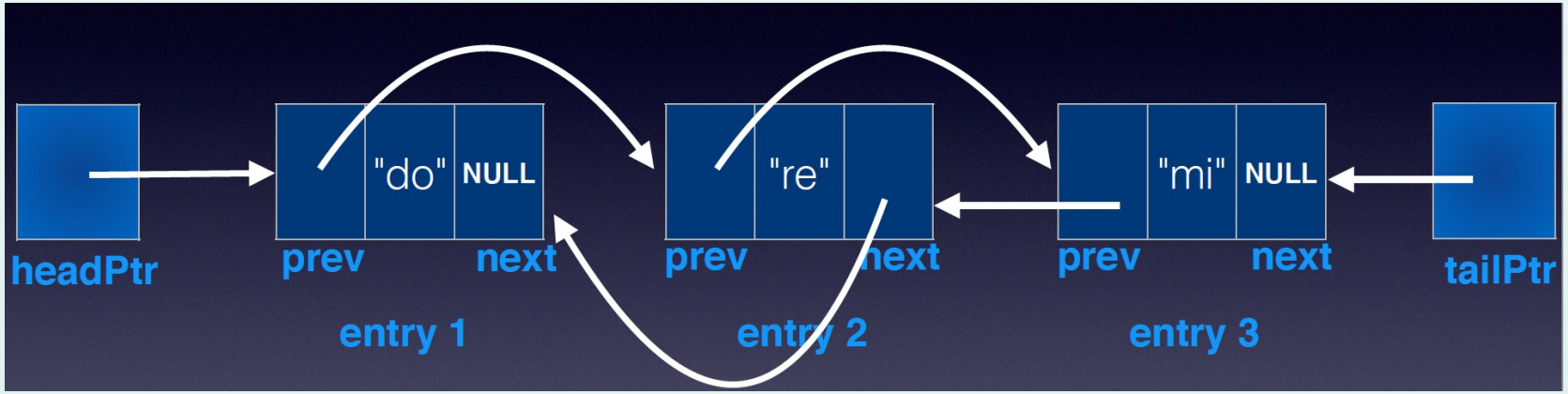
Reversing a Doubly Linked List

- Reversing a doubly linked list is a simple process.
- Swap the prevPtr and nextPtr of all nodes.
- Then swap headPtr and TailPtr.



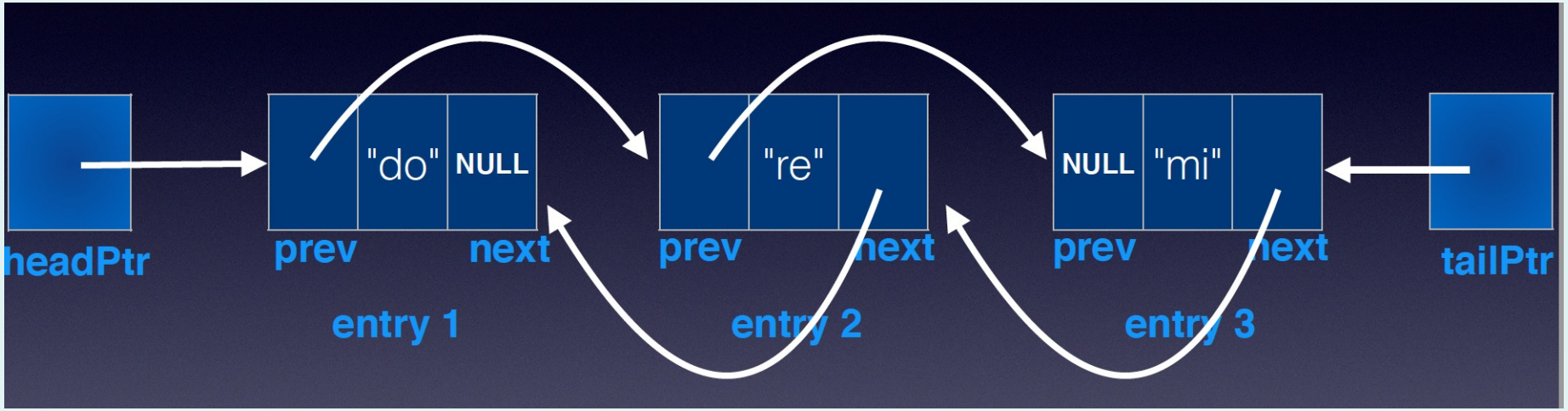
Reversing a Doubly Linked List

- Reversing a doubly linked list is a simple process.
- Swap the prevPtr and nextPtr of all nodes.
- Then swap headPtr and TailPtr.



Reversing a Doubly Linked List

- Reversing a doubly linked list is a simple process.
- Swap the prevPtr and nextPtr of all nodes.
- Then swap headPtr and TailPtr.



Reversing a Doubly Linked List

- Reversing a doubly linked list is a simple process.
- Swap the prevPtr and nextPtr of all nodes.
- Then swap headPtr and TailPtr.