Doubly and Circular Linked Lists

Lesson 4.2

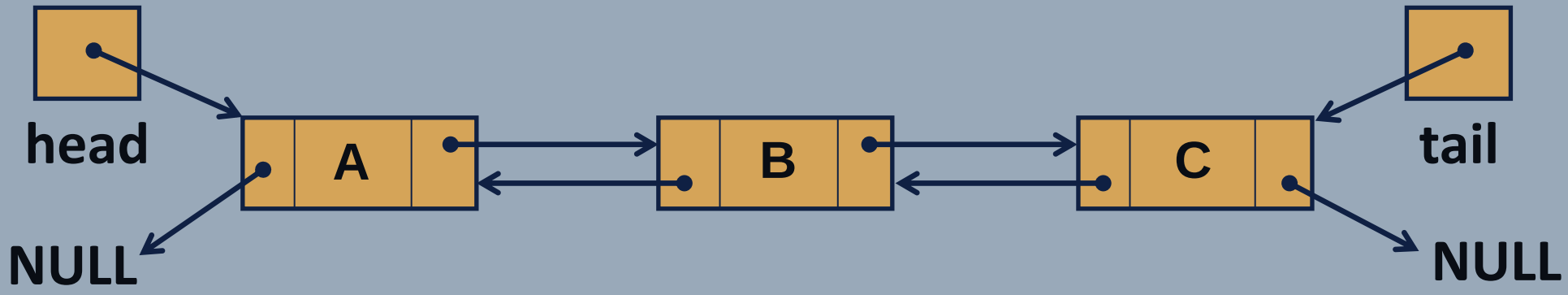
Learning Objectives

LO 4.2.1 **Create** and **initialize** doubly and circular linked lists

LO 4.2.2 **Perform** insertion and deletion of elements in a doubly and circular linked lists

Doubly Linked List

- A **doubly linked list** is a linked list that contains 2 pointer in its basic unit and is pointing to the next and previous node in the sequence of nodes

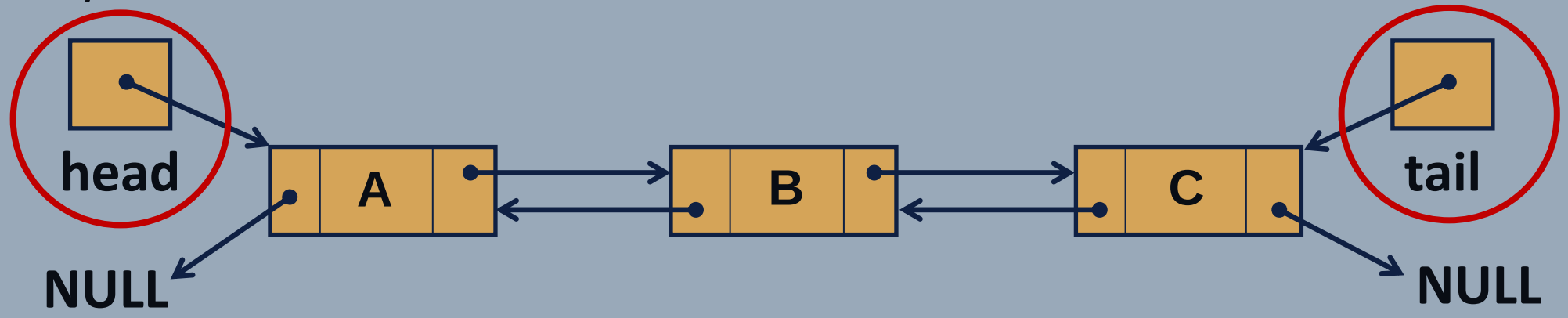


Basic Units of a Doubly Linked List

- A *doubly linked list node* contains 3 parts:
  - —The **data** which stores the value of that node; and
  - —The **next** pointer that points to the address of the next node in the list
  - —The **prev** pointer that points to the address of the previous node in the list

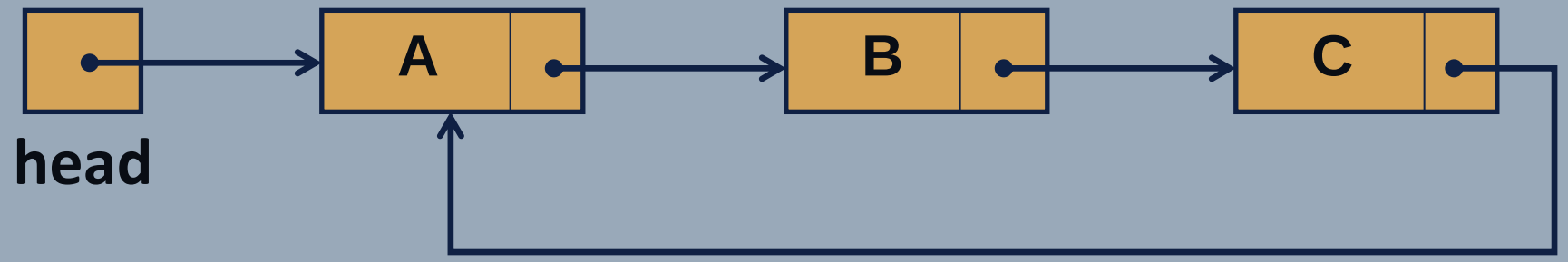
Basic Units of a Doubly Linked List

- To make a list of doubly linked list nodes, the nodes must be interconnected by linking the next pointers sequentially forward and the prev pointers sequentially backward of the nodes
- The first node must be pointed by a node pointer named head and the last node must be pointed by a node pointer named tail to facilitate the reference of the list when it will be utilized from the first/last node of the list.



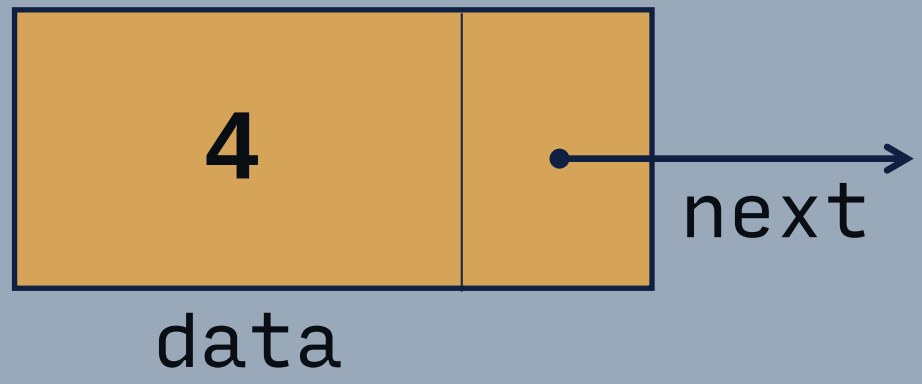
Circular Linked List

- A **circular linked list** is like a singly linked list in which the nodes only point to the next node. The difference between a singly linked list and a circular linked list is that the last node doesn't point to *NULL*, instead it points to the first node of the list.



Basic Units of a Circular Linked List

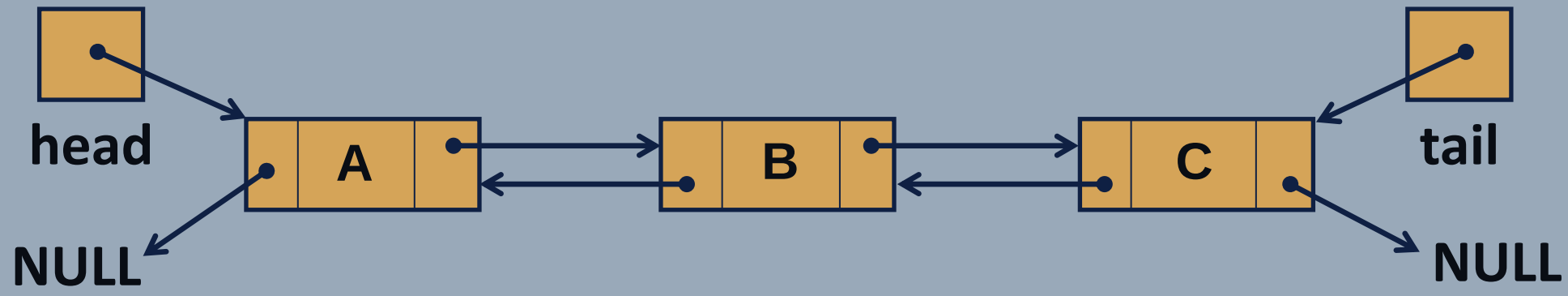
- A *circular linked list node* and a *singly linked list node* are of the same structure



Strengthening the Learning Objectives

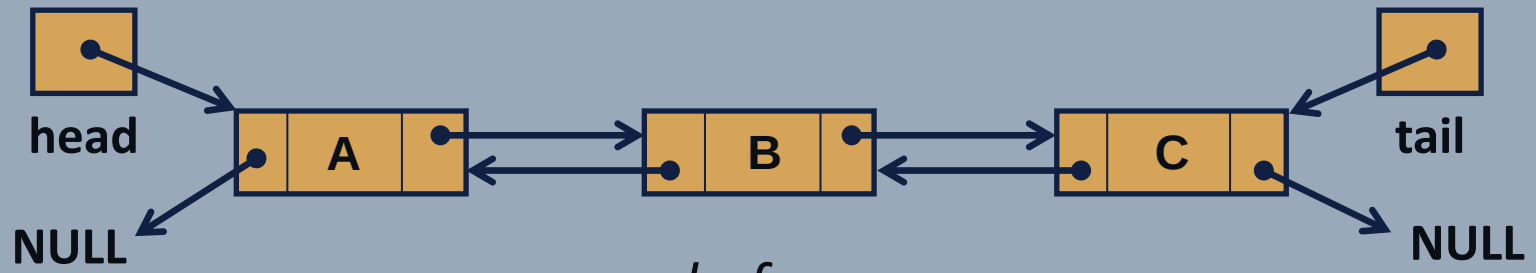
LO 4.2.1 Create and initialize doubly and circular linked lists

Create a doubly linked list with the same configuration below using C structures then display the contents of the linked list by traversing the list from the head and tail by using only 1 pointer variable.

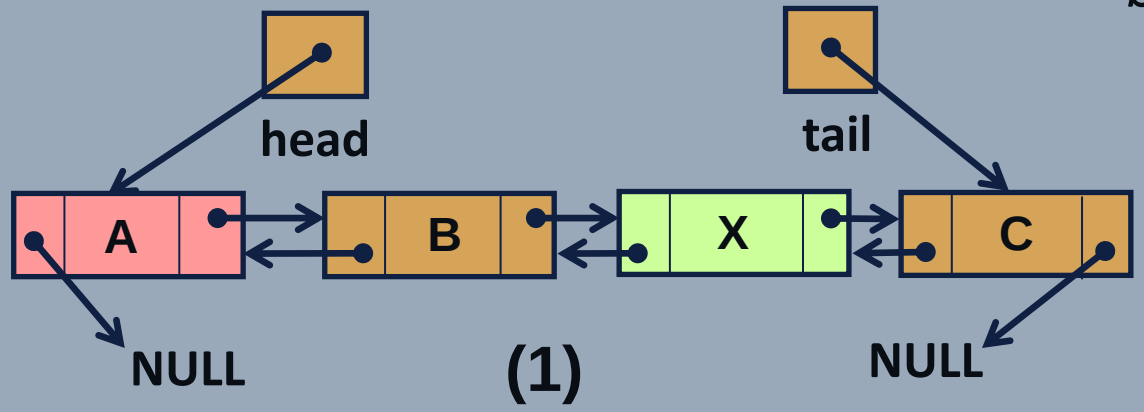


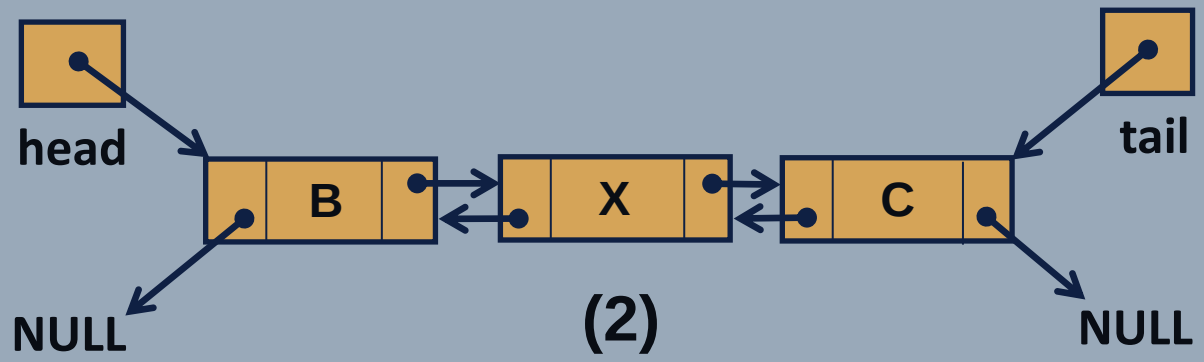
LO 4.2.2 Perform insertion and deletion of elements in a doubly and circular linked lists

On the doubly linked list you have just created, insert the data X between B and C, then delete A.



before

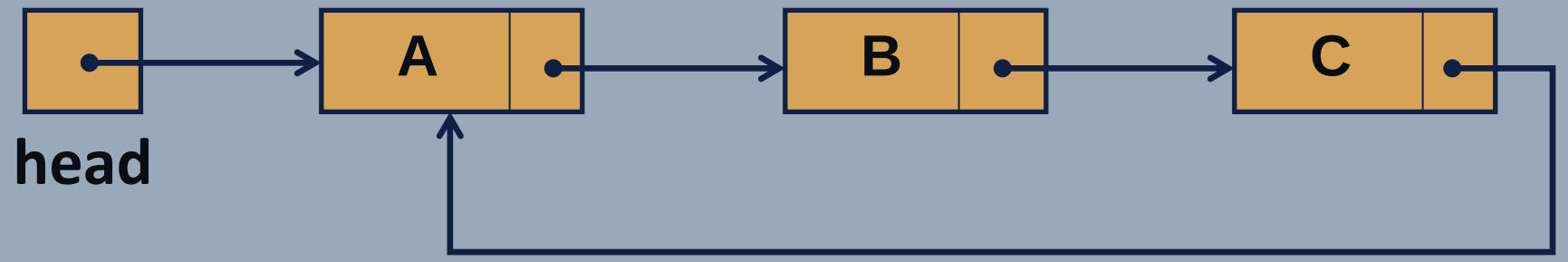




after

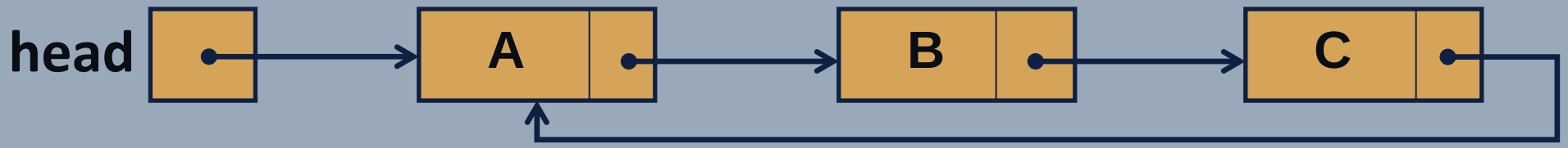
LO 4.2.1 Create and initialize doubly and circular linked lists

Create a circular linked list with the same configuration below using C structures then display the contents of the linked list by traversing the list from the head using only 1 pointer variable.

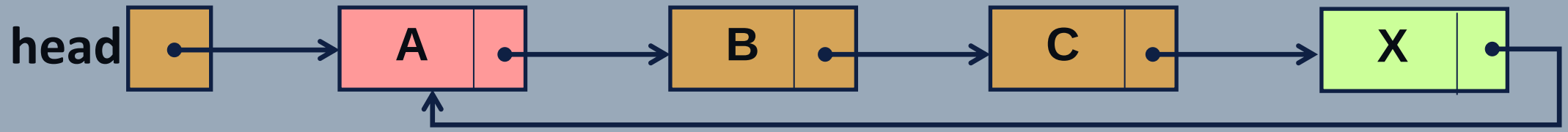


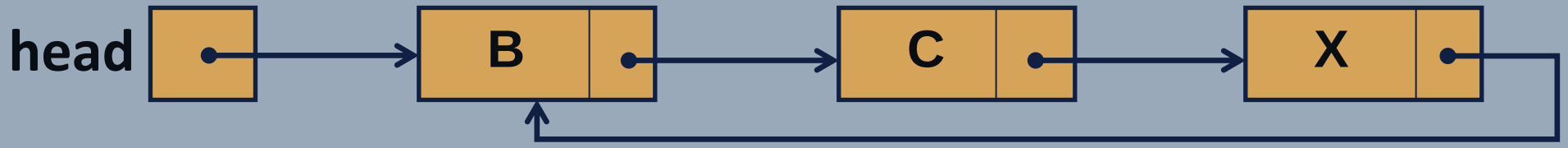
LO 4.2.2 Perform insertion and deletion of elements in a doubly and circular linked lists

On the circular linked list you have just created, insert the data X at the end of the list, then delete the first element of the list.



before





after

LO 4.2.1 Create and initialize doubly and circular linked lists

Create a function

```
DOUBLYLINKEDLIST createDList()
```

that will return a pointer pointing to a doubly linked list structure with no doubly linked list node.

LO 4.2.2 Perform insertion and deletion of elements in a doubly and circular linked lists

Create a function

```
void addFirst(DOUBLYLINKEDLIST list, int data)
```

where **`list`** is a pointer to an existing doubly linked list structure of integer data and **`data`** is to be inserted at the first/front part of the list.

LO 4.2.2 Perform insertion and deletion of elements in a doubly and circular linked lists

Create a function

```
void addLast(DOUBLYLINKEDLIST list, int data)
```

where **`list`** is a pointer to an existing doubly linked list structure of integer data and **`data`** is to be inserted at the last/rear part of the list.

LO 4.2.1 Create and initialize doubly and circular linked lists

Create a function

```
CIRCULARLINKEDLIST createCList()
```

that will return a pointer pointing to a circular linked list structure with no circular linked list node.

LO 4.2.2 Perform insertion and deletion of elements in a doubly and circular linked lists

Create a function

```
void addFirst(CIRCULARLINKEDLIST list, int data)
```

where **`list`** is a pointer to an existing circular linked list structure of integer data and **`data`** is to be inserted at the first/front part of the list.

LO 4.2.2 Perform insertion and deletion of elements in a doubly and circular linked lists

Create a function

```
void addLast(CIRCULARLINKEDLIST list, int data)
```

where **`list`** is a pointer to an existing circular linked list structure of integer data and **`data`** is to be inserted at the last/rear part of the list.