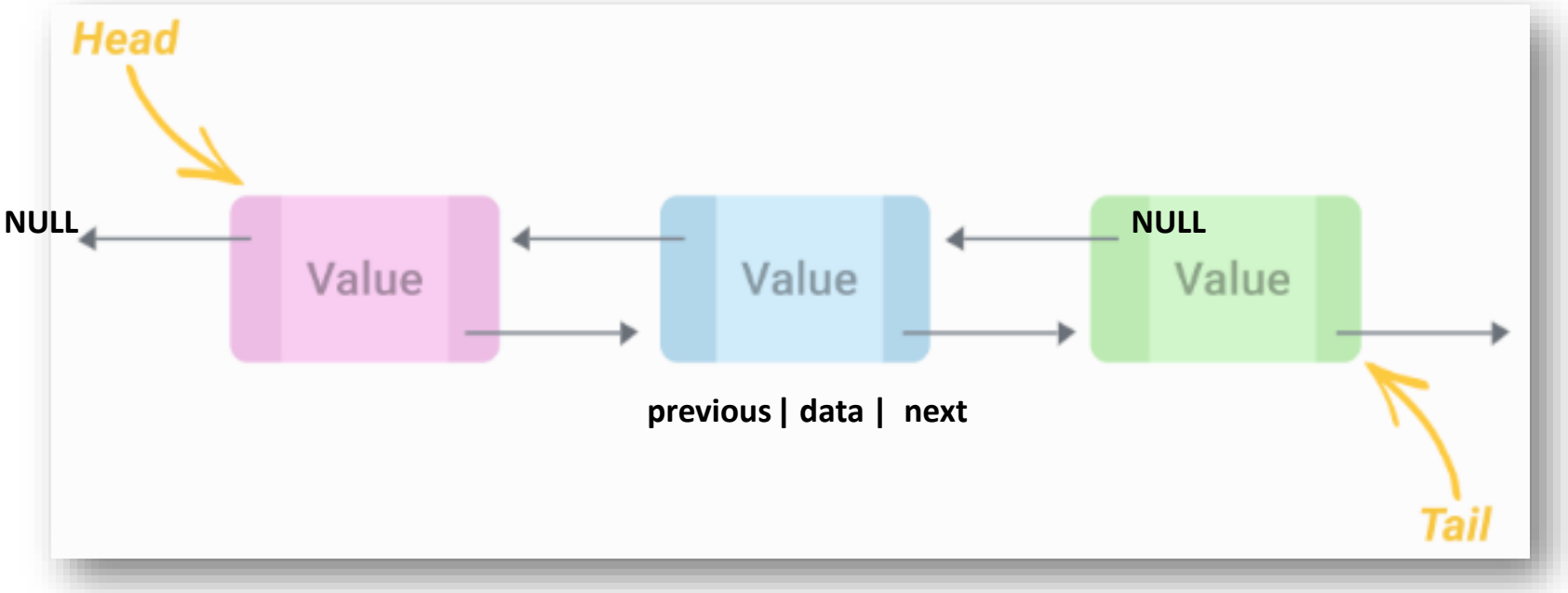
Double Linked List (DLL)



What is double linked list (DLL)

A doubly linked list is a type of linked list in which each node/element contains 3 parts:

1. **Data**: The actual value or information stored in the node.
2. **Next Pointer**: A reference to the next node in the list.
3. **Previous Pointer**: A reference to the previous node in the list.



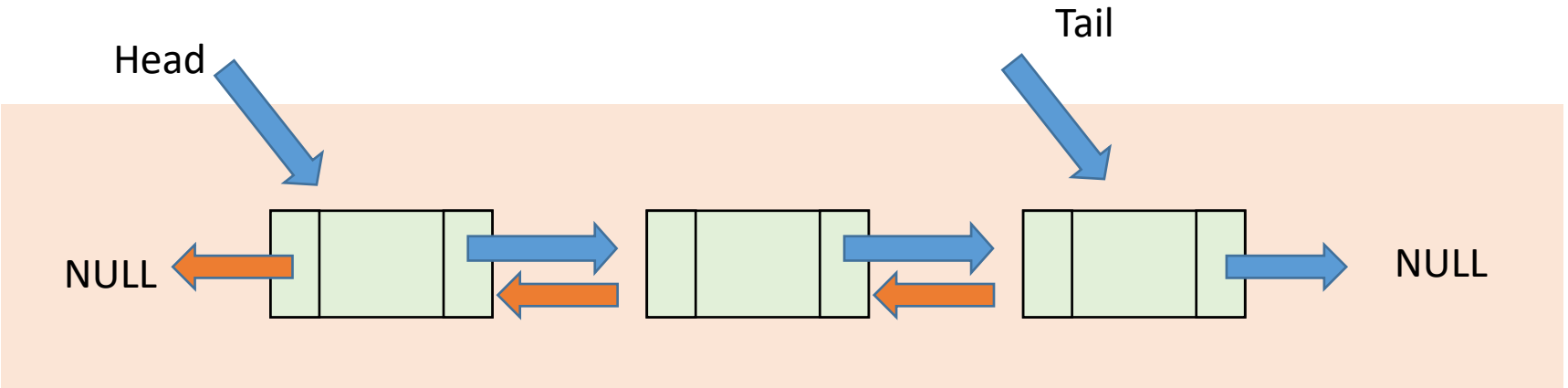

```
struct Element{
      int data;
      Element *next;
      Element *previous;
}
```

Advantage of Double linked list

```
struct Element{
      int data;
      Element *next;
      Element *previous;
}
```

Since the double liked list has two pointers: one pointing to the next node and another pointing to the previous node.

=> It allows **traversal in both directions**: *forward* and *backward*. Meaning that, we can access data easily in any directions.



Double linked list

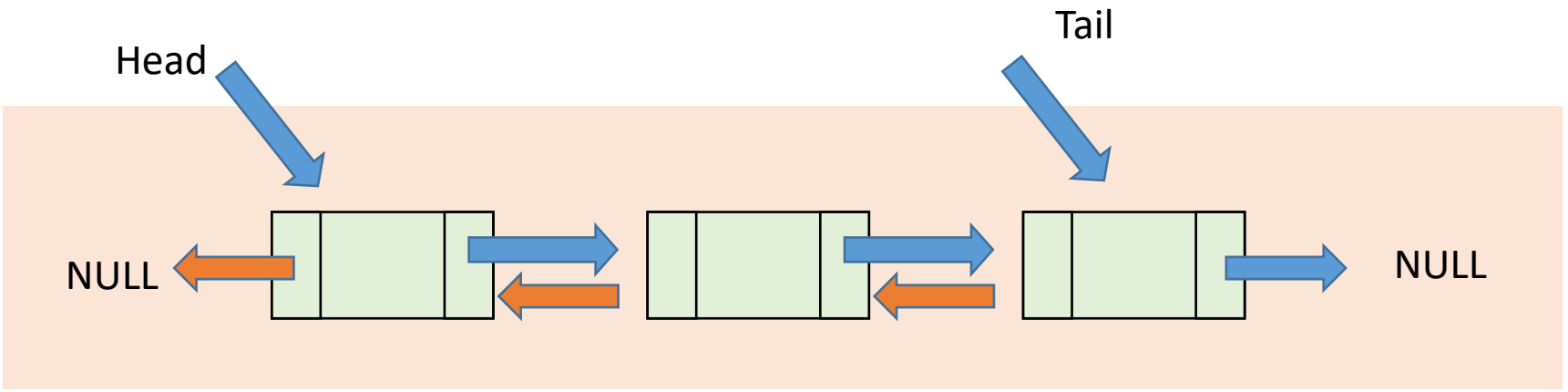
```
struct Element{
        int data;
        Element *next;
    Element *previous;
};
```

```
struct List{
        Element *head;
    Element *tail;
    int n;
};
```

- Each element contains
  - ✓ **Data**
  - ✓ A link pointer to its **next** element (successor)
  - ✓ A link pointer to its **previous** element (predecessor)
    - They are created as a pointer

Element
Var data: integer Var *next: Element Var *previous: Element

List
Var *head: Element Var *tail: Element Var n: integer





Implementation:

Double Linked List (DLL)

C++

DLL

```
1  #include<iostream>
2  using namespace std;
3  struct Element{
4      int data;
5      Element *next;
6  };   Element *previous;
7
8
9  struct List{
10     int n;  //number of elements
11     Element *head;
12     Element *tail;
13 };
14
```

```
17 //A function to create an empty list
18 List* createList(){
19     List *ls;
20
21     ls = new List(); //allocate memor
22     //ls.n = 0; //error
23     ls->n = 0;
24     ls->head = NULL;
25     ls->tail = NULL;
26
27     return ls;
28 }
```

```
31 void insert_begin(List *ls, int newData){
32     //Create new element
33     Element *e;
34     e = new Element();
35     e->data = newData;
36     e->previous = NULL;
37     e->next = ls->head;
38
39     //Update tail
40     if(ls->n == 0){
41         ls->tail = e;
42     }
43     //Update head
44     if(ls->n != 0){
45         ls->head->previous = e;
46     }
47     ls->head = e;
48     ls->n = ls->n + 1;
49 }
```

```
51 void displayList(List *ls){
52     Element *tmp;
53
54     tmp = ls->head;
55     while(tmp!=NULL){
56         cout<<tmp->data<<" ";
57         tmp = tmp->next;
58     }
59     cout<<endl;
60 }
```

```
149 int main(){
150
151     List *L;
152     L = createList();
153
154     insert_begin(L, 3);
155     insert_begin(L, 1);
156     insert_begin(L, 0);
157     insert_begin(L, -4);
158
159     displayList(L);
160     displayList2(L);
161 }
```

```
-4 0 1 3
3 1 0 -4
```

```
62 void displayList2(List *ls){
63     Element *tmp;
64
65     tmp = ls->tail;
66     while(tmp!=NULL){
67         cout<<tmp->data<<" ";
68         tmp = tmp->previous;
69     }
70     cout<<endl;
71 }
```

Q and A

Practice on Implementation of Linked List using C++

❑ Exercises

1. Create a double linked list that can store scores of students. Then perform these operations below
   a. Create two data structure (Element and List) then create an empty singly linked list
   b. Add the number 70.5 to the begin of the list
   c. Add the number 90 to the end of the list
   d. Add 89.5 to the begin of the list
   e. Remove the first element of the list
   f. Add 0 to the end of the list
   g. Display all numbers in the list
2. Create a doubly linked list that stores the English alphabet (A-Z). Then display the list from A to Z and also display from Z to A.
3. Store data of 5 students in a double linked list. They display all student from start to end (forward display). Also, display from end to start (backward display).