Data Structures and Object Oriented Programming

Lecture 7

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

Object-Oriented Programming in C++

Linked List

Linked List

```
struct Node {
    int data;
    Node* next = NULL;
};
```

```
int main()
{
    Node* head;
    Node* second;
    Node* third;

    head = new Node;
    second = new Node;
    third = new Node;

    head->data = 1;       // assign data in first node
    head->next = second; // Link first node with second

    second->data = 2;     // assign data to second node
    second->next = third;// Link first node with second

    third->data = 3; // assign data to third node

    return 0;
}
```

Linked List

```
struct Node {
    int data;
    Node* next = NULL;
};


class LinkedList {

Node *head=NULL;

};
```

```
int main()
{









}
```

Linked List

```
struct Node {
    int data;
    Node* next = NULL;
};


class LinkedList {

Node *head=NULL;

public:
    void printList();




};
```

```
int main()
{




}
```

Linked List

```
struct Node {
    int data;
    Node* next = NULL;
};


class LinkedList {

Node *head=NULL;

public:
    void printList();
    void insert_start(int value);
    void insert_end(int value);
    void insert_after(int n,int value);


};
```

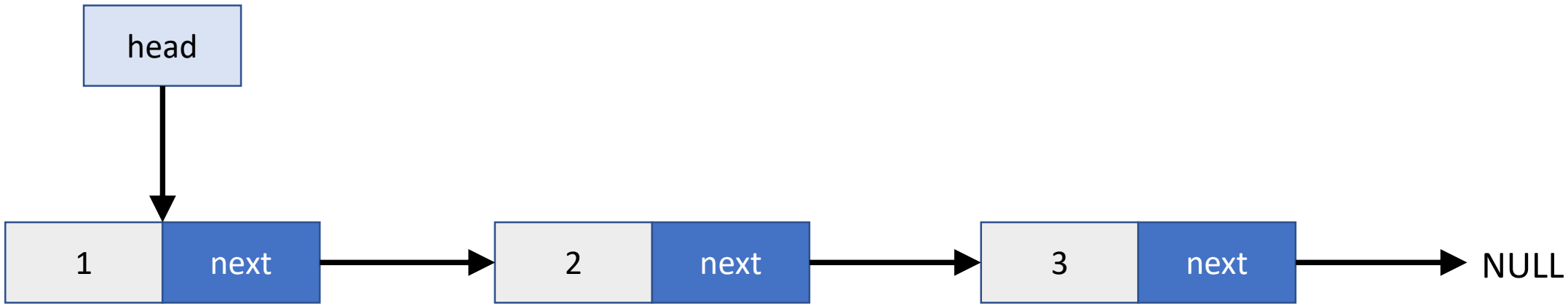
```
int main()
{



}
```

Linked List

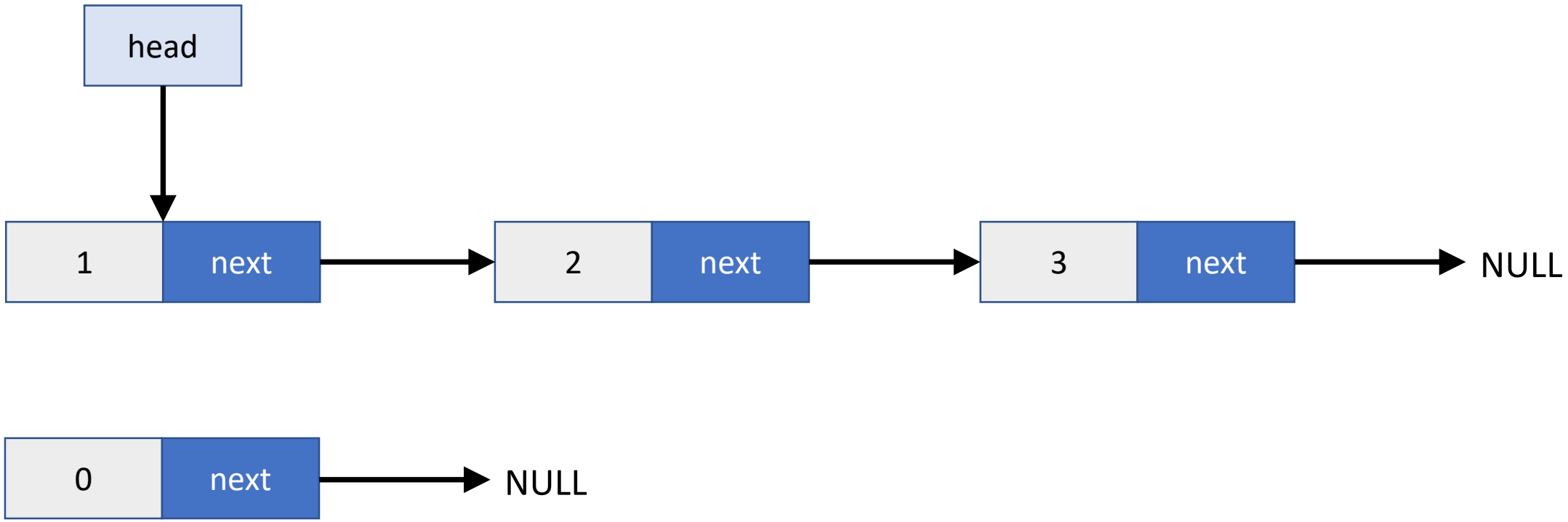
```
struct Node {
    int data;
    Node* next = NULL;
};


class LinkedList {

Node *head=NULL;

public:
    void printList();
    void insert_start(int value);
    void insert_end(int value);
    void insert_after(int n,int value);


};
```

```
int main()
{




}
```

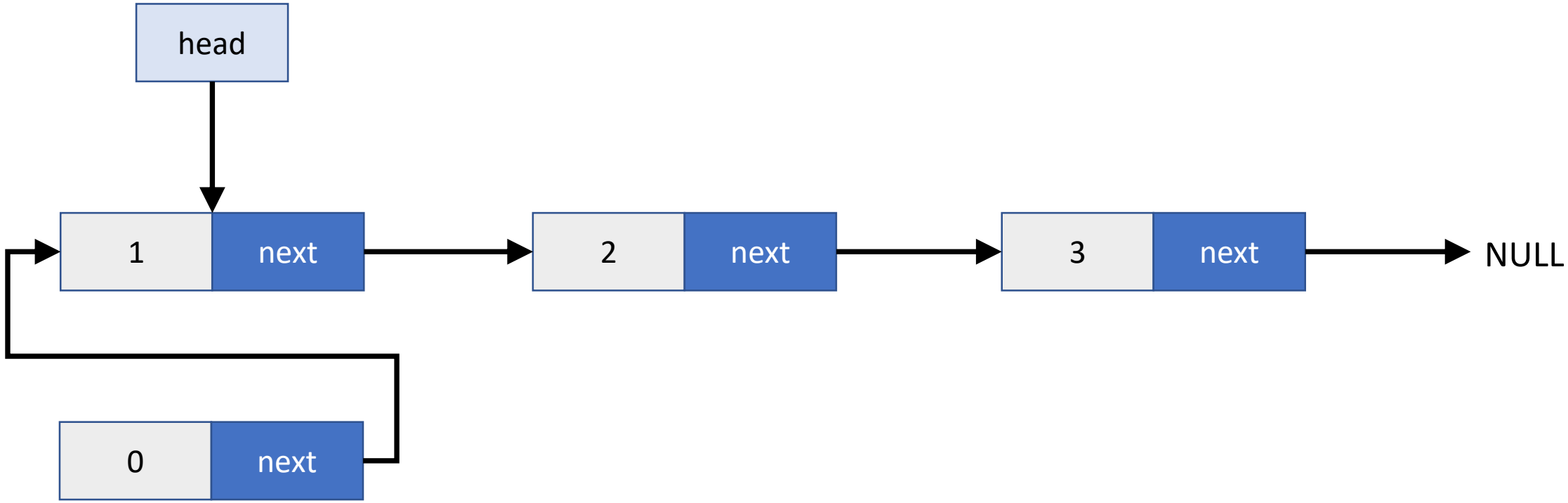
Linked List (`insert_start`)



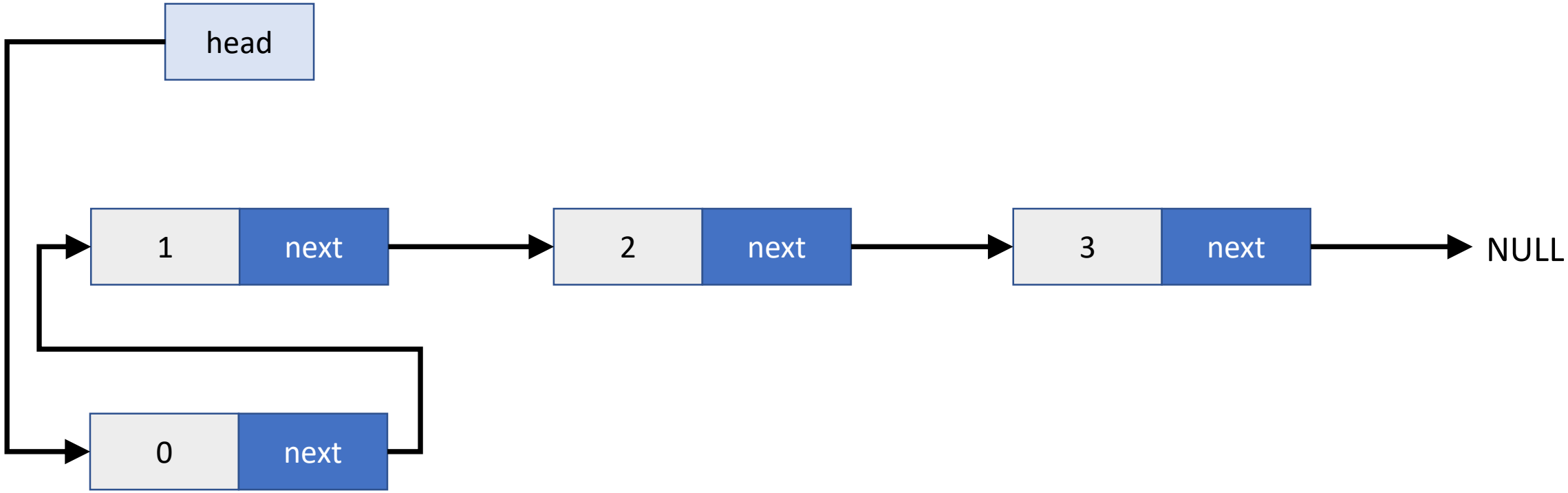
Linked List (insert_start)



Linked List (insert_start)



Linked List (insert_start)



Linked List (insert_start)

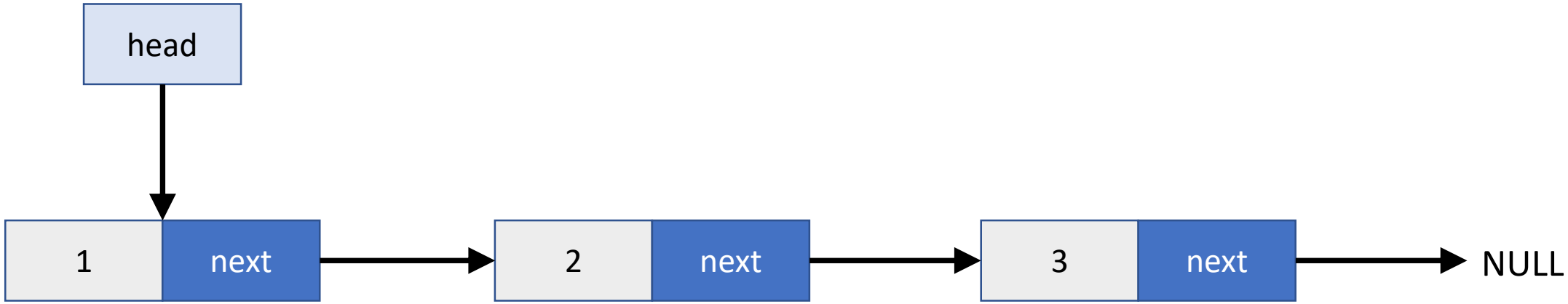
```
void LinkedList::insert_start(int value)
{
    Node* temp = new Node;
    temp->data = value;

    temp->next = head;
    head = temp;
    }
}
```

Linked List (insert_end)

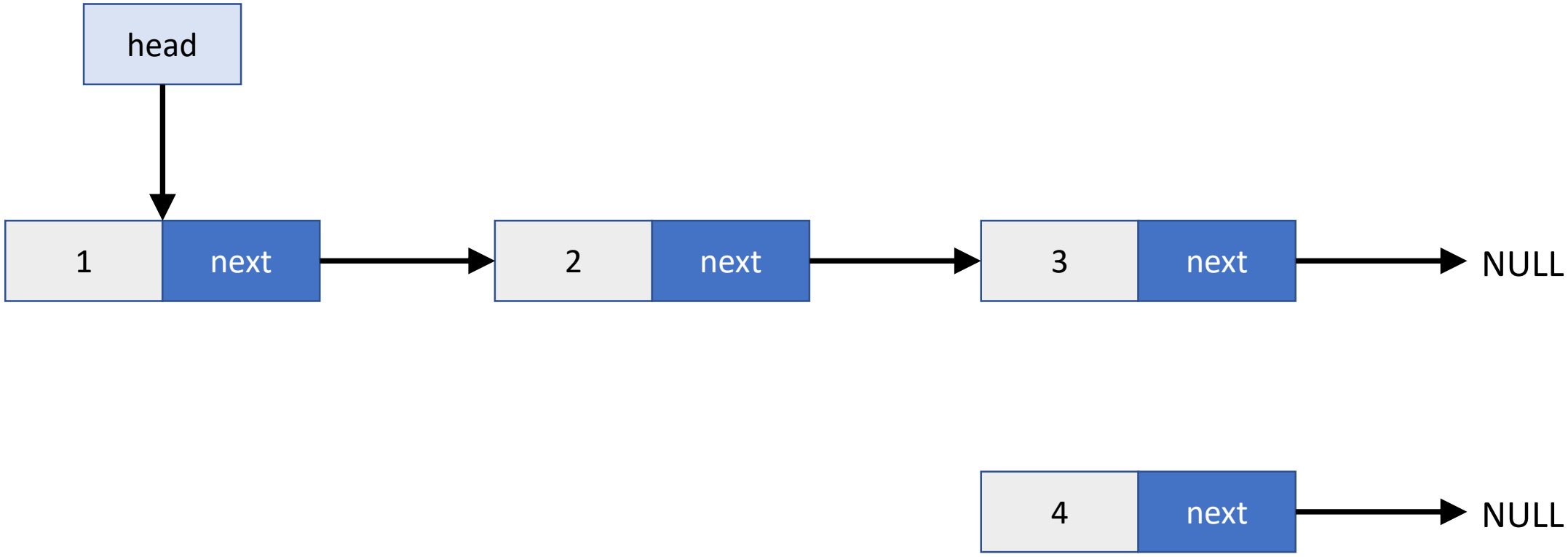
```
struct Node {
    int data;
    Node* next = NULL;
};


class LinkedList {

Node *head=NULL;

public:
    void printList();
    void insert_start(int value);
    void insert_end(int value);
    void insert_after(int n,int value);


};
```

```
int main()
{









}
```

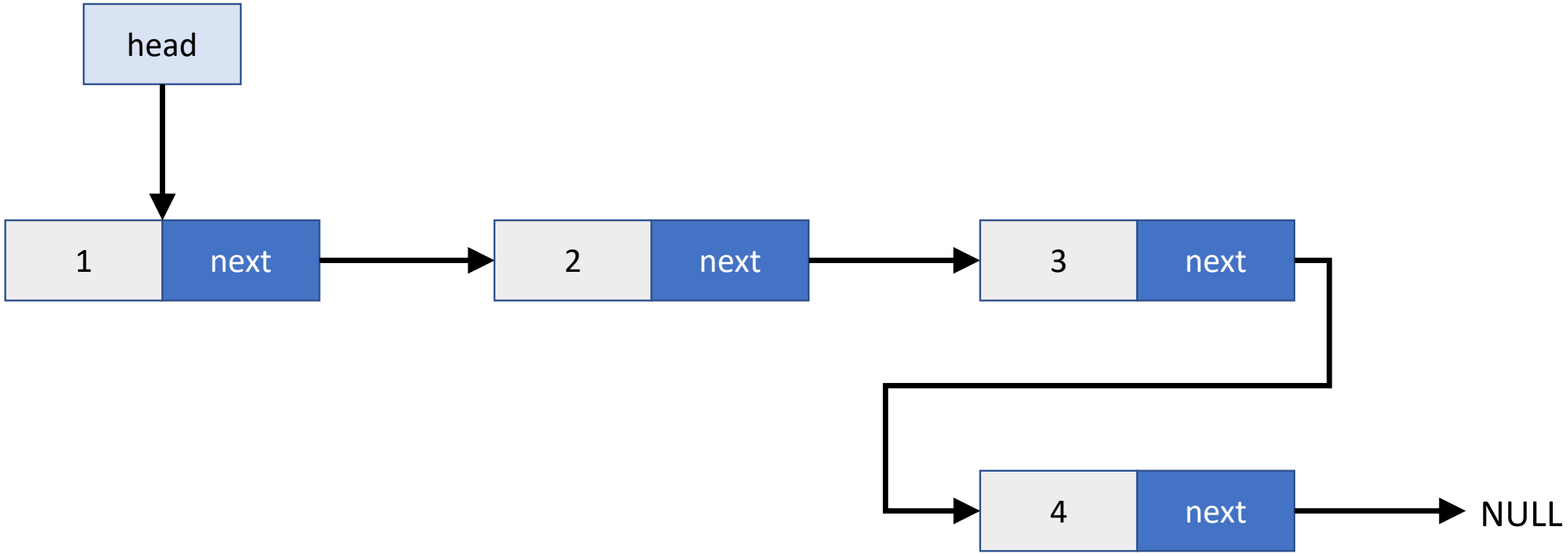
Linked List (`insert_end`)



Linked List (insert_end)



Linked List (insert_end)



Linked List (insert_end)

```
void LinkedList::insert_end(int value)
{
    Node* temp = new Node;
    temp->data = value;

    Node* count = head;
    while (count->next != NULL)
    {
        count = count->next;
    }
    count->next = temp;

}
```

Linked List (insert_end)

```
void LinkedList::insert_end(int value)
{
    Node* temp = new Node;
    temp->data = value;

    if (head == NULL)
    {
        head = temp;
    }

    else
    {
        Node* count = head;
        while (count->next != NULL)
        {
            count = count->next;
        }
        count->next = temp;
    }
}
```

Linked List (insert_end)

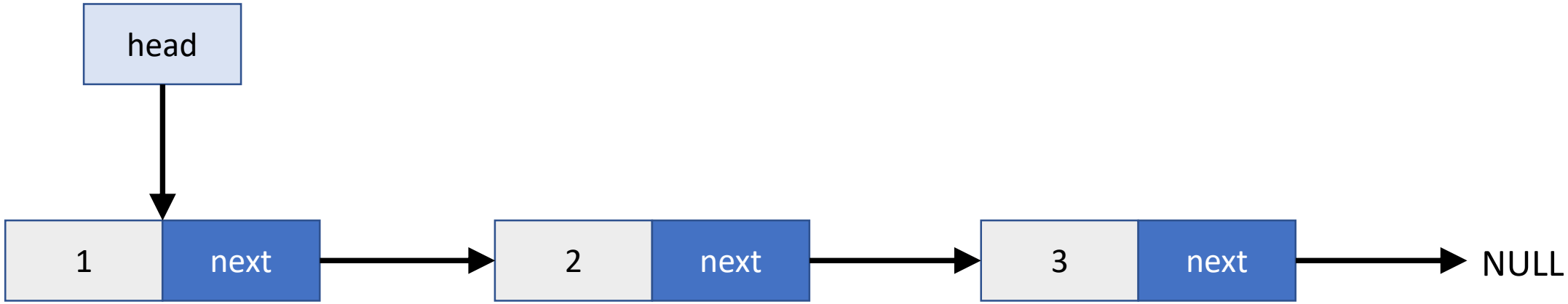
```
void LinkedList::insert_end(int value)
{
    Node* temp = new Node;
    temp->data = value;

    if (head == NULL)
    {
        head = temp;
    }

    else
    {
        Node* count = head;
        while (count->next != NULL)
        {
            count = count->next;
        }
        count->next = temp;
    }
}
```

Linked List (insert_after)

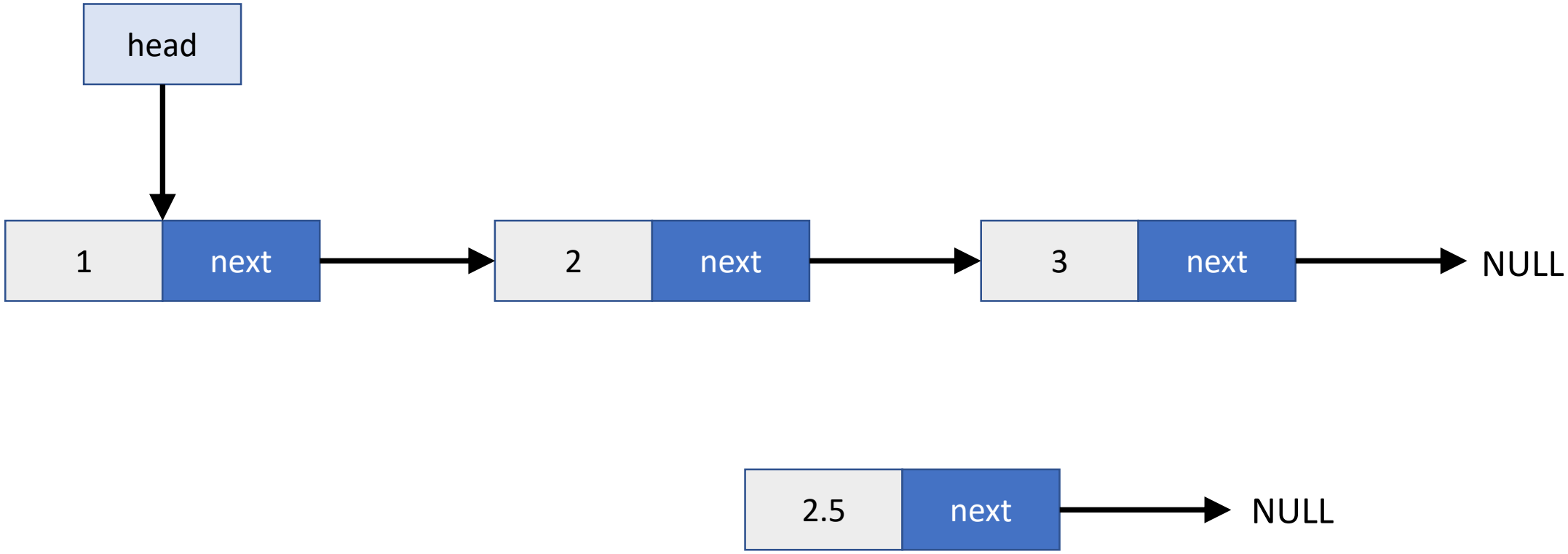
```
struct Node {
    int data;
    Node* next = NULL;
};


class LinkedList {

Node *head=NULL;

public:
    void printList();
    void insert_start(int value);
    void insert_end(int value);
    void insert_after(int n,int value);


};
```

```
int main()
{




}
```

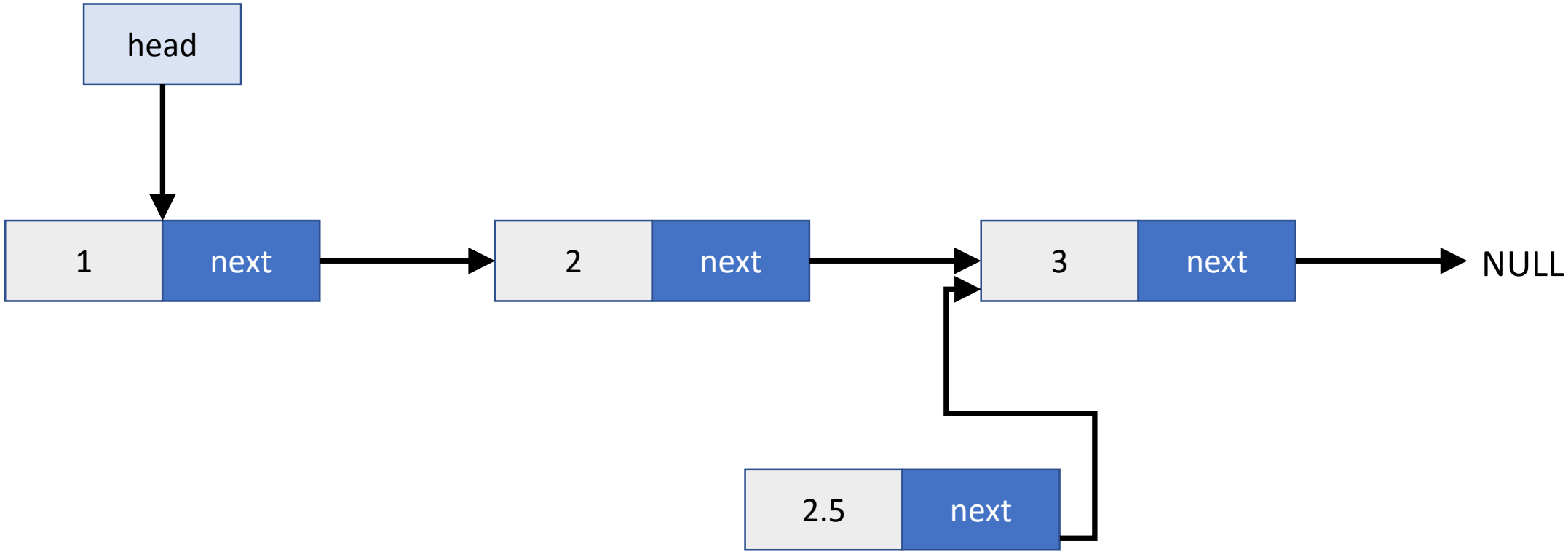
Linked List (`insert_after`)



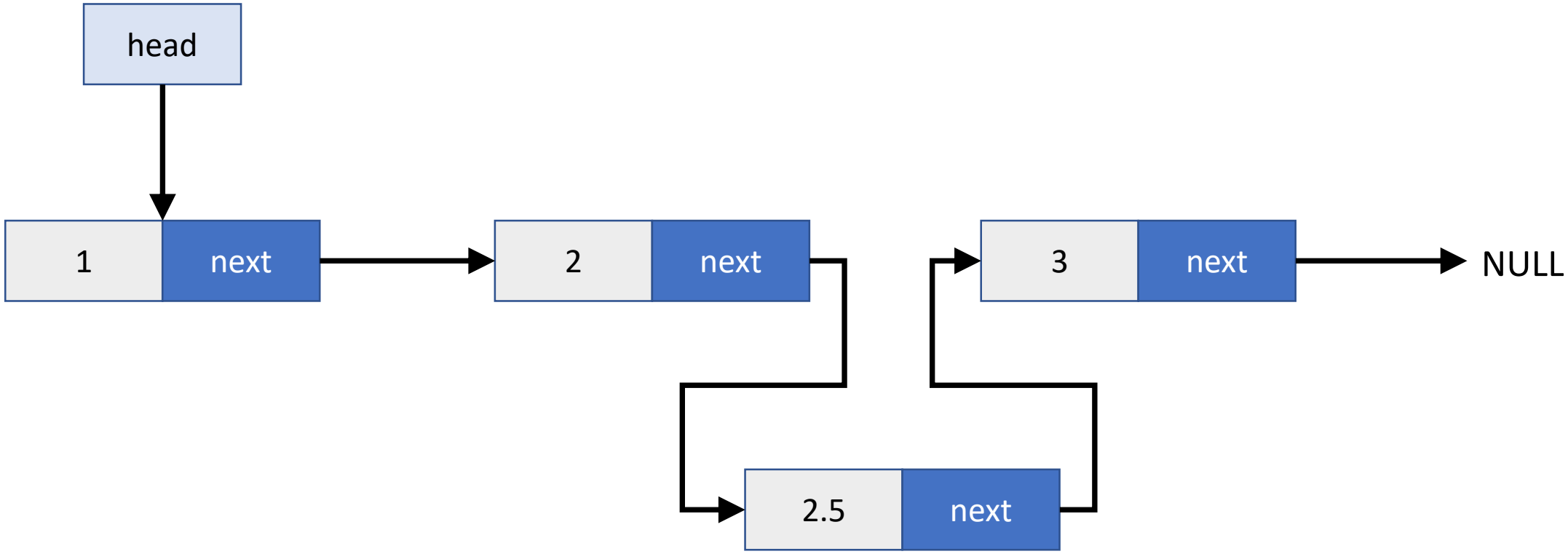
Linked List (`insert_after`)



Linked List (`insert_after`)



Linked List (`insert_after`)



Linked List (insert_after)

```
void LinkedList::insert_after(int n, int value)
{
    Node* temp = new Node;
    temp->data = value;

    Node* check = head;
    while (check->data != n)
    {
        check = check->next;
    }

    temp->next = check->next;
    check->next = temp;

}
```

Linked List (insert_after)

```
void LinkedList::insert_after(int n, int value)
{
    Node* temp = new Node;
    temp->data = value;

    if (head == NULL)
    {
        head=temp;
    }

    else
    {
        Node* check = head;
        while (check->data != n)
        {
            check = check->next;
        }
        temp->next = check->next;
        check->next = temp;
    }
}
```

Linked List

```
struct Node {
    int data;
    Node* next = NULL;
};


class LinkedList {

Node *head=NULL;

public:
    void printList();
    void insert_start(int value);
    void insert_end(int value);
    void insert_after(int n,int value);


};
```

```
int main()
{



}
```

- **Next Lecture: Deletion**

Thanks a lot

If you are taking a Nap, **wake up**.....Lecture Over