Data Structures and Object Oriented Programming

Lecture 8

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

Object-Oriented Programming in C++

Linked List

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()
{
    LinkedList list;

    list.insert_start(1);
    list.insert_end(2);
    list.insert_end(4);
    list.insert_after(2,3);

    list.printList();

    return 0;
}
```

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);

    void delete_start();
    void delete_end();
    void delete_after(int n);

};
```

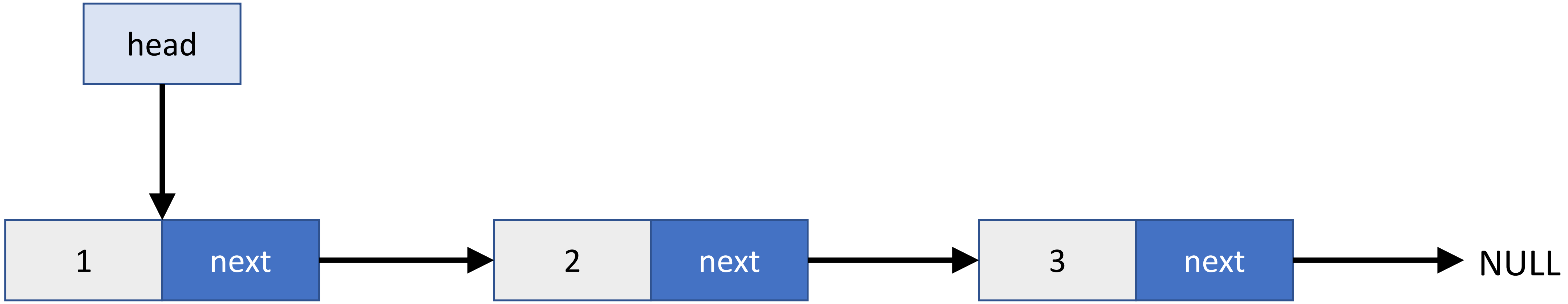
```
int main()
{
    LinkedList list;

    list.insert_start(1);
    list.insert_end(2);
    list.insert_end(4);
    list.insert_after(2,3);

    list.printList();

    return 0;
}
```

Linked List

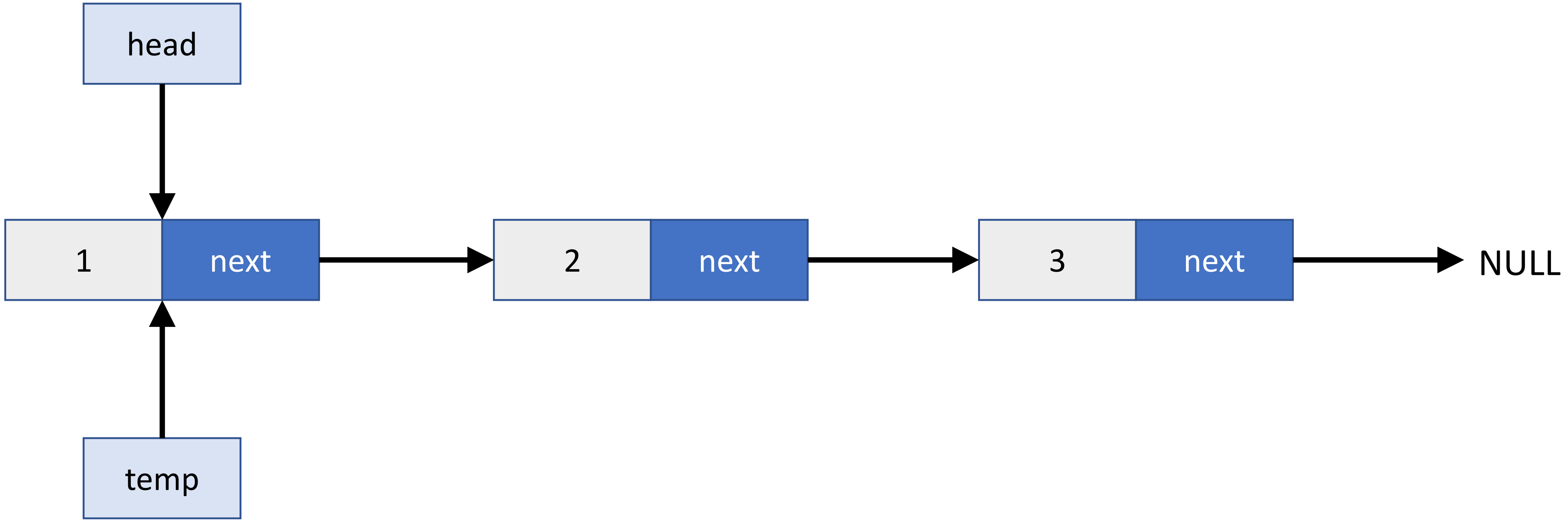
```cpp
struct Node {
    int data;
    Node* next = NULL;
};

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);

    void delete_start();
    void delete_end();
    void delete_after(int n);

};
```

```cpp
int main()
{
    LinkedList list;

    list.insert_start(1);
    list.insert_end(2);
    list.insert_end(4);
    list.insert_after(2,3);

    list.printList();

    return 0;
}
```

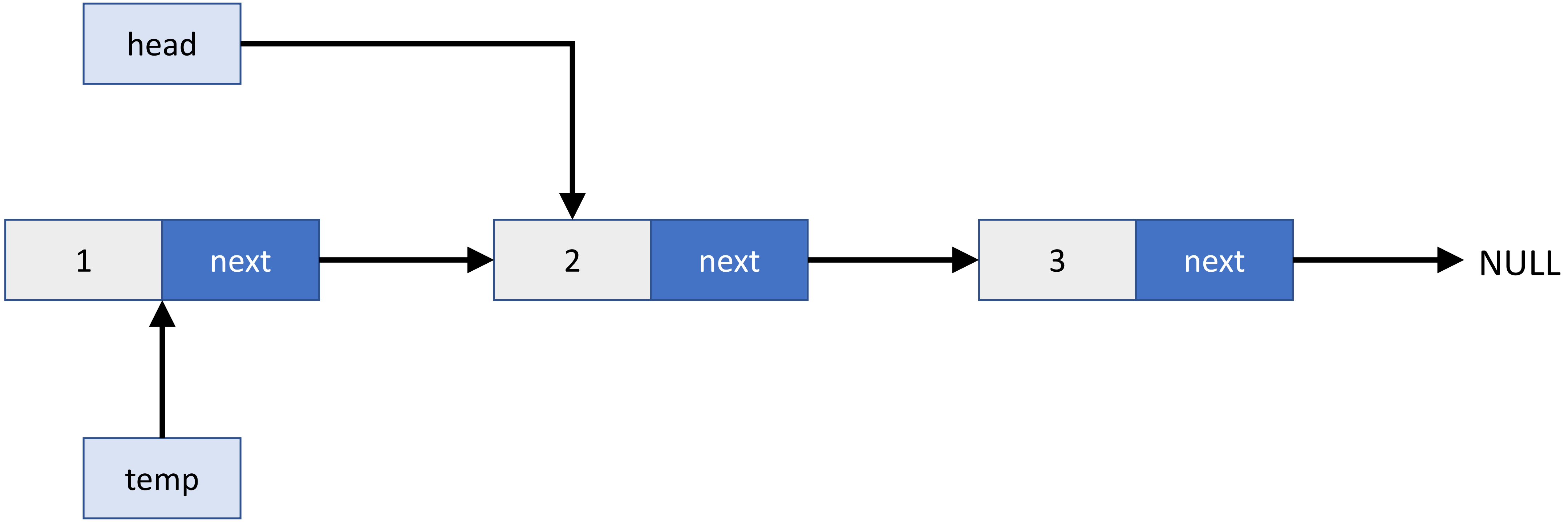
Linked List (`delete_start`)



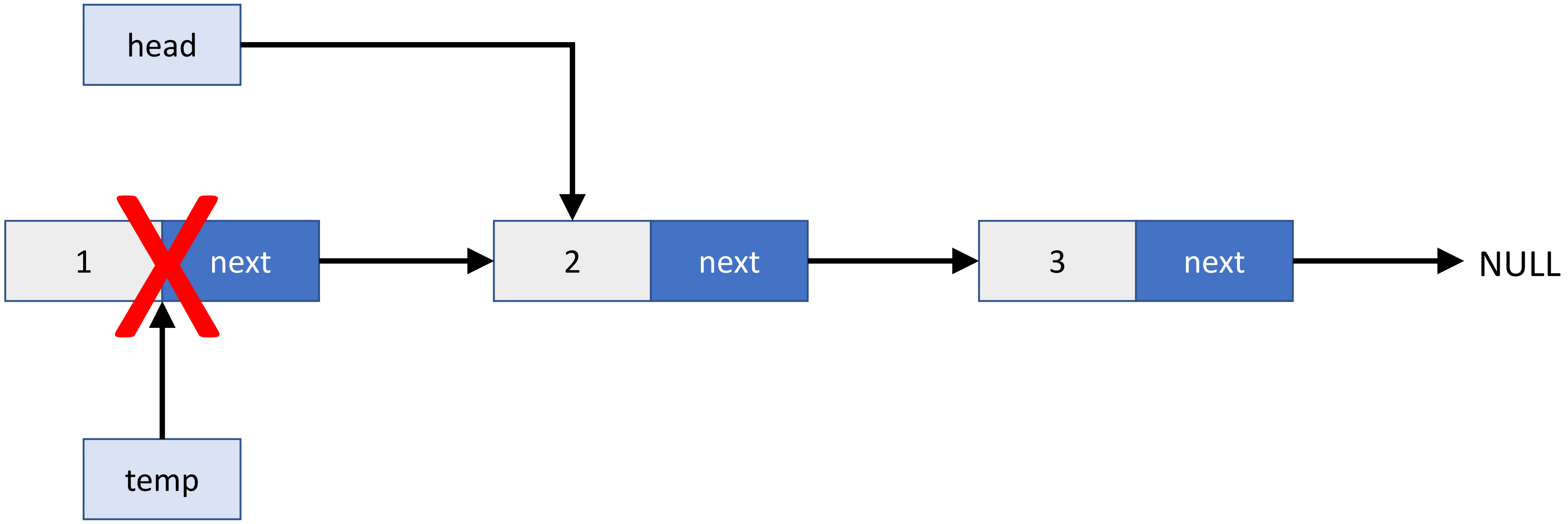
Linked List (`delete_start`)



Linked List (`delete_start`)



Linked List (`delete_start`)



Linked List (delete_start)

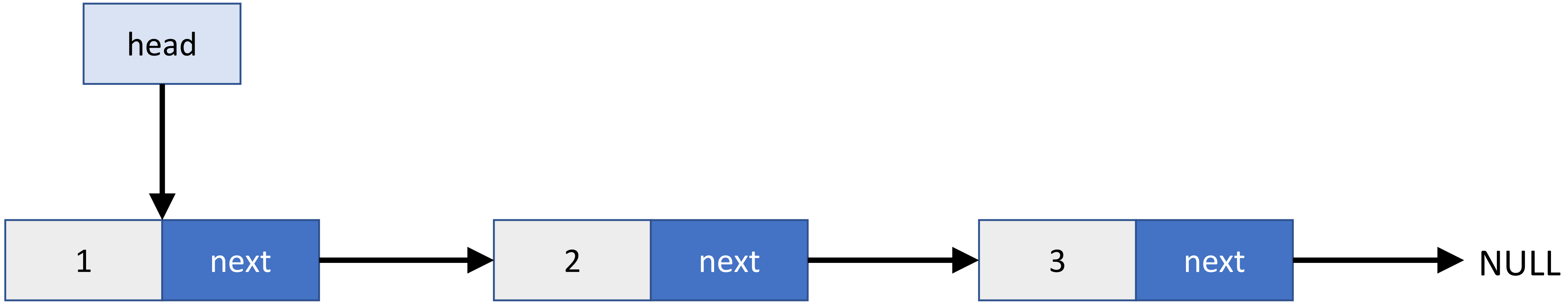
```
void LinkedList::delete_start()
{
    if (head == NULL)
    {
        return;
    }

    else
    {
        Node* temp = head;
        head = temp->next;
        delete temp;
    }

}
```

Linked List

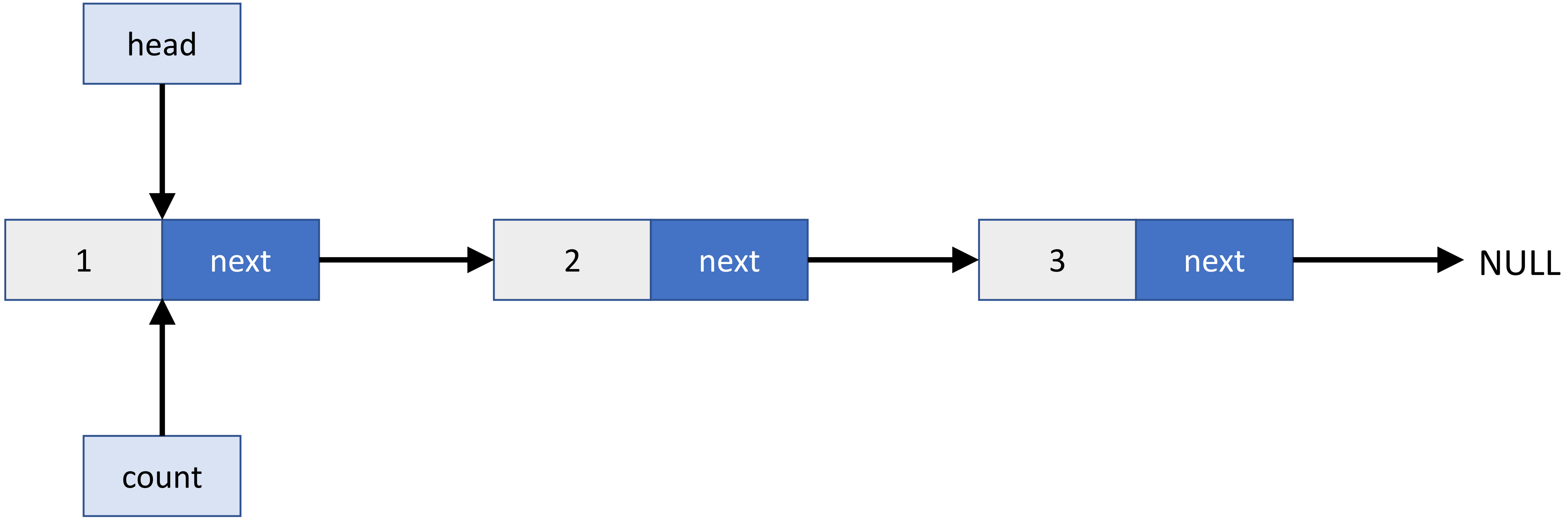
```
struct Node {
    int data;
    Node* next = NULL;
};

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);

    void delete_start();
    void delete_end();
    void delete_after(int n);

};
```

```
int main()
{
    LinkedList list;

    list.insert_start(1);
    list.insert_end(2);
    list.insert_end(4);
    list.insert_after(2,3);

    list.printList();

    return 0;
}
```

Linked List (`delete_end`)

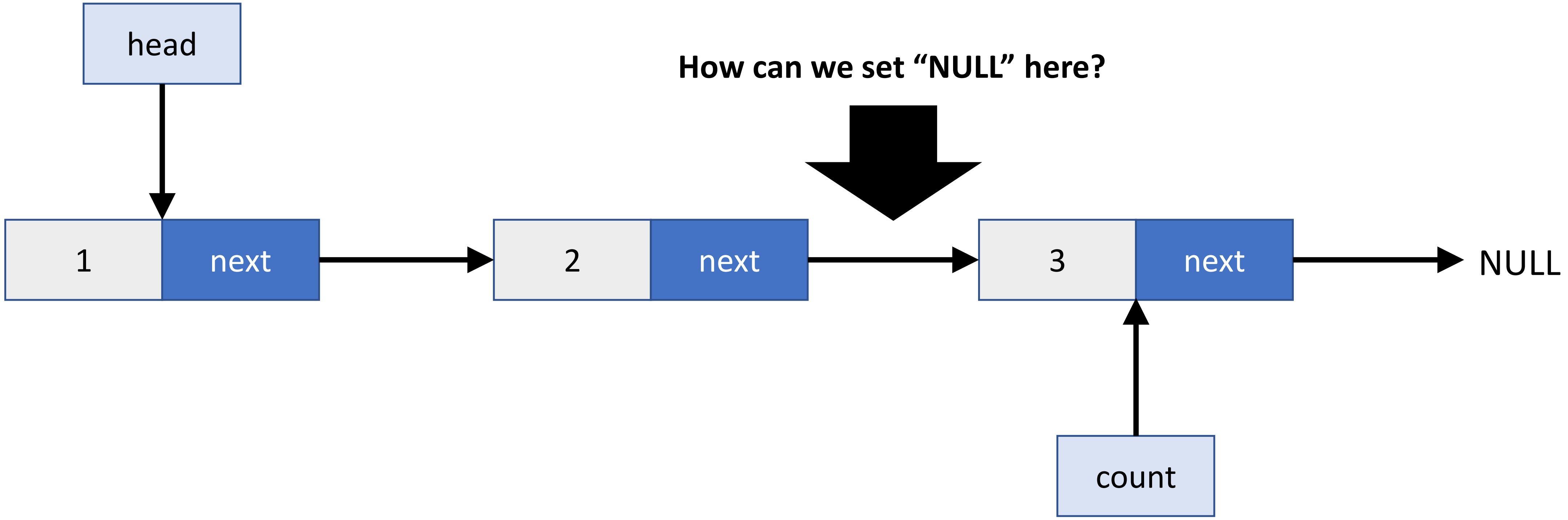


Linked List (`delete_end`)

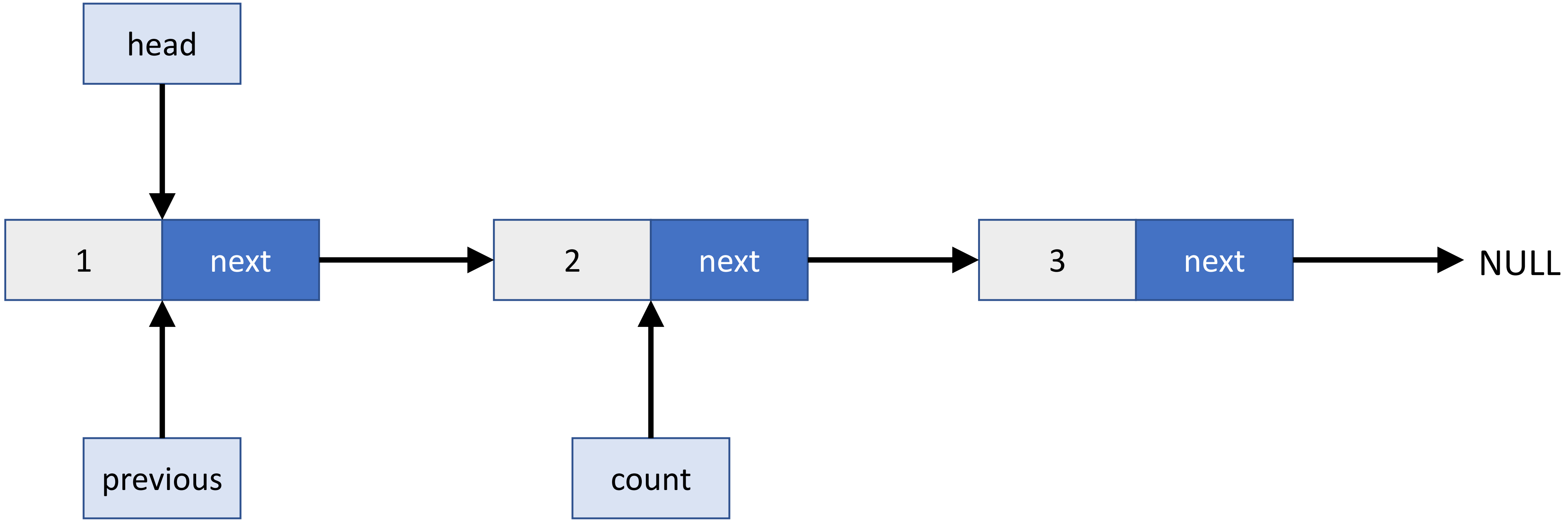


Node* count = head;

Linked List (`delete_end`)



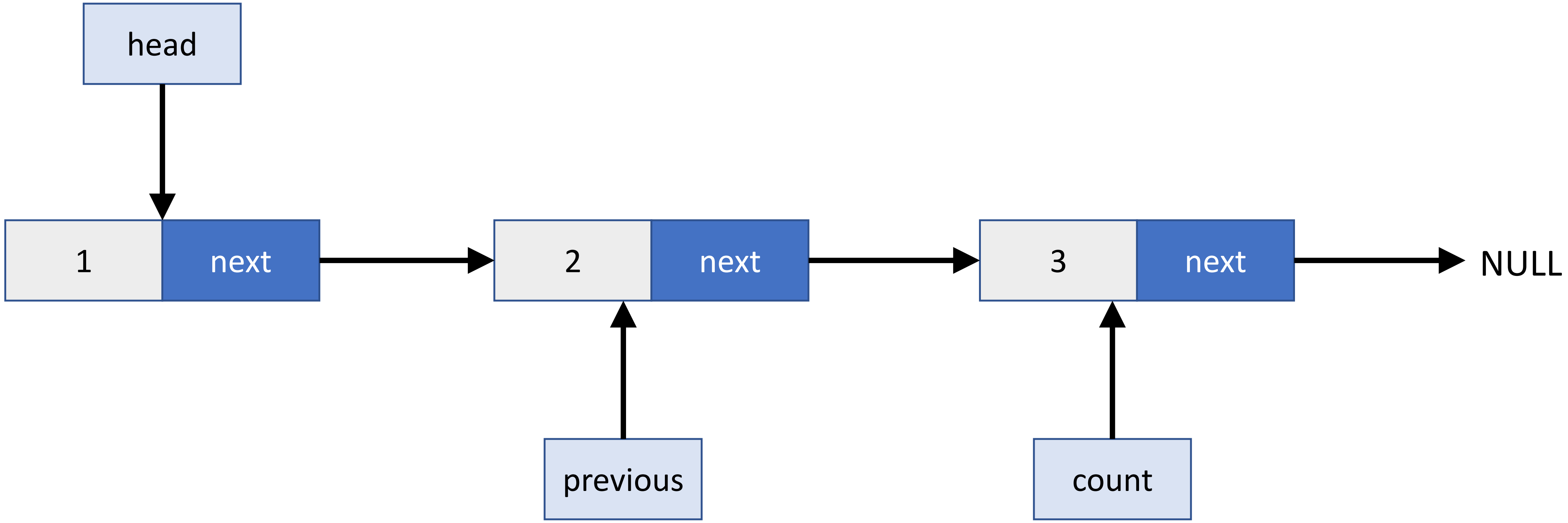
Linked List (`delete_end`)



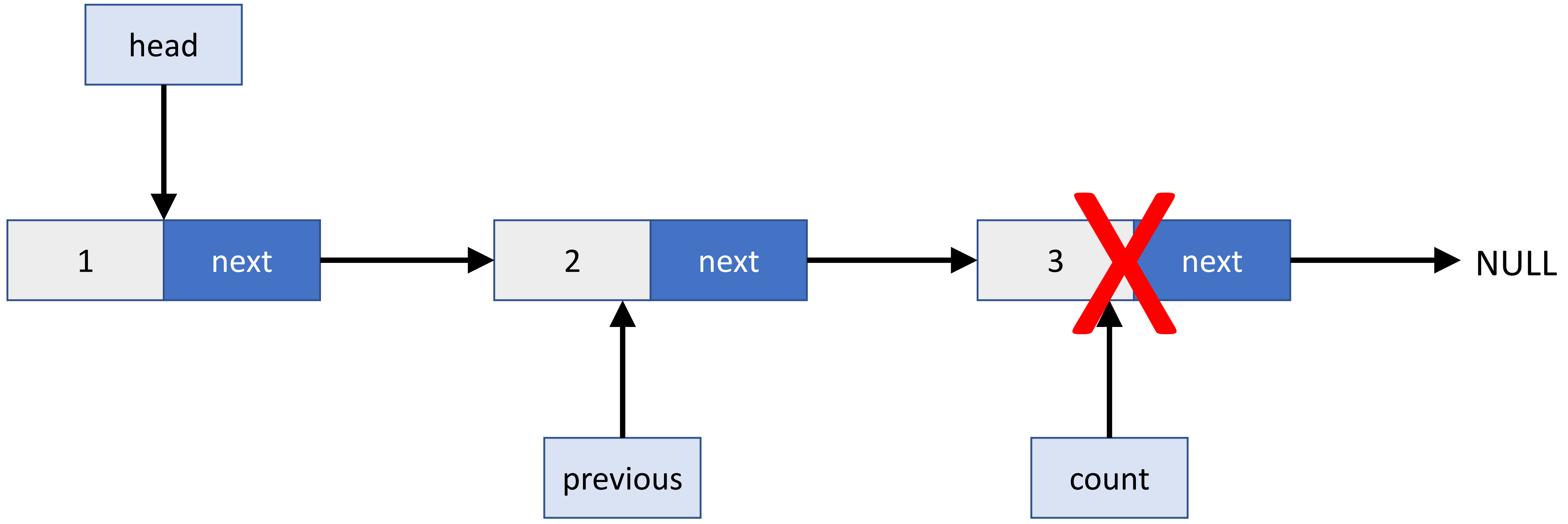
Node* previous = head;

Node* count = head->next;

Linked List (`delete_end`)

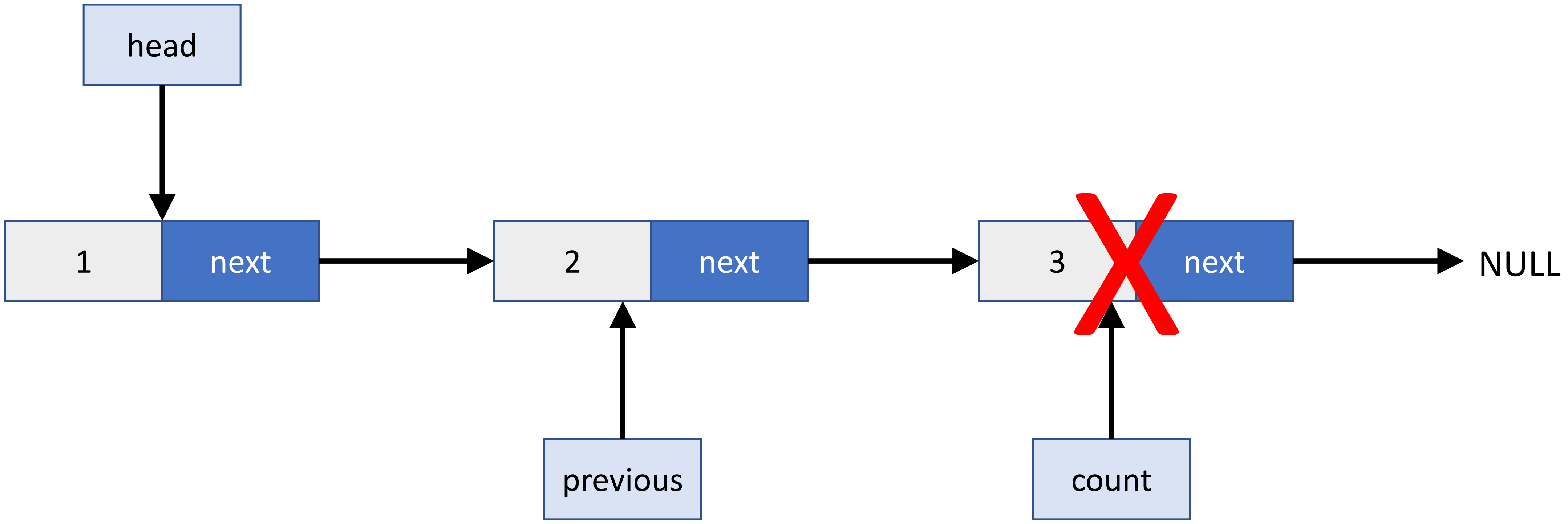


Linked List (`delete_end`)



delete count;

Linked List (`delete_end`)



previous->next= NULL;

delete count;

Linked List (**delete_end**)

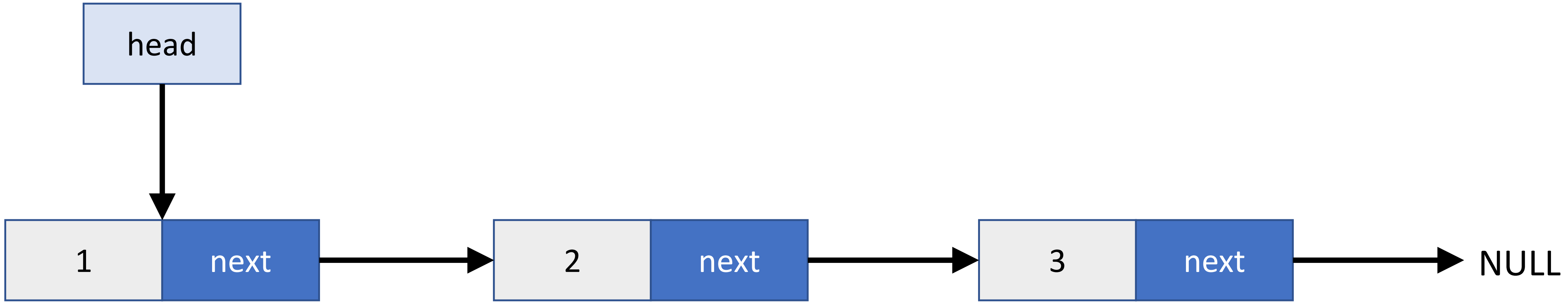
```
void LinkedList::delete_end()
{

    if (head == NULL)
    {
        return;
    }

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

Linked List

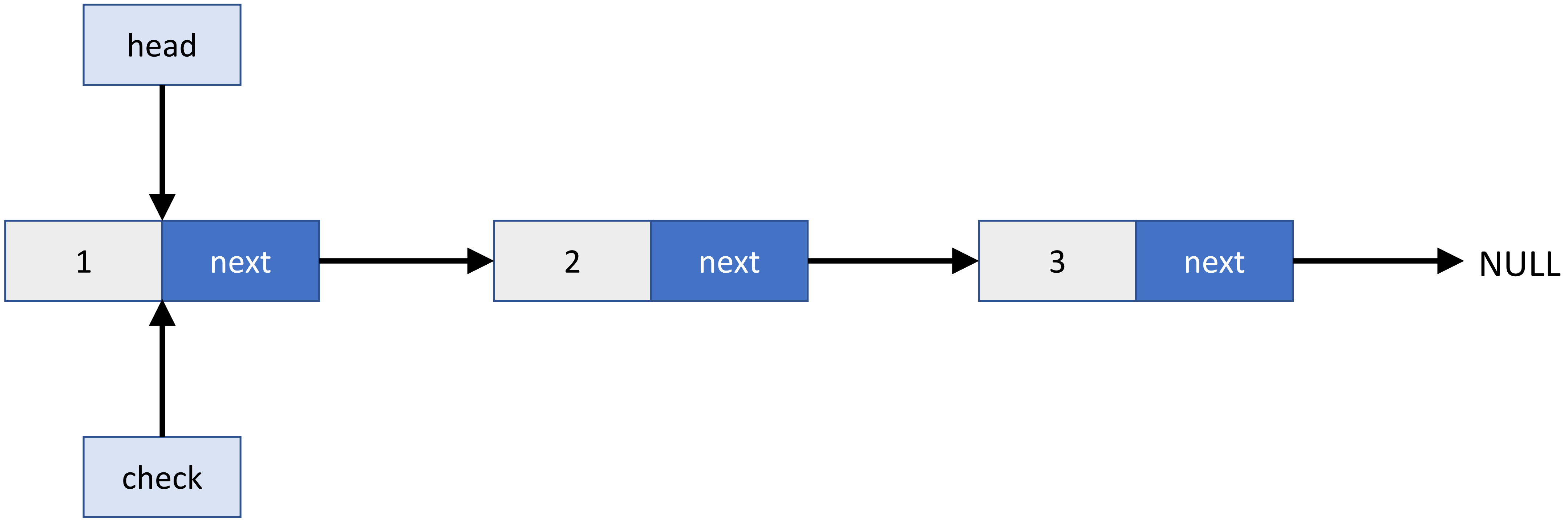
```
struct Node {
    int data;
    Node* next = NULL;
};

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);

    void delete_start();
    void delete_end();
    void delete_after(int n);

};
```

```
int main()
{
    LinkedList list;

    list.insert_start(1);
    list.insert_end(2);
    list.insert_end(4);
    list.insert_after(2,3);

    list.printList();

    return 0;
}
```

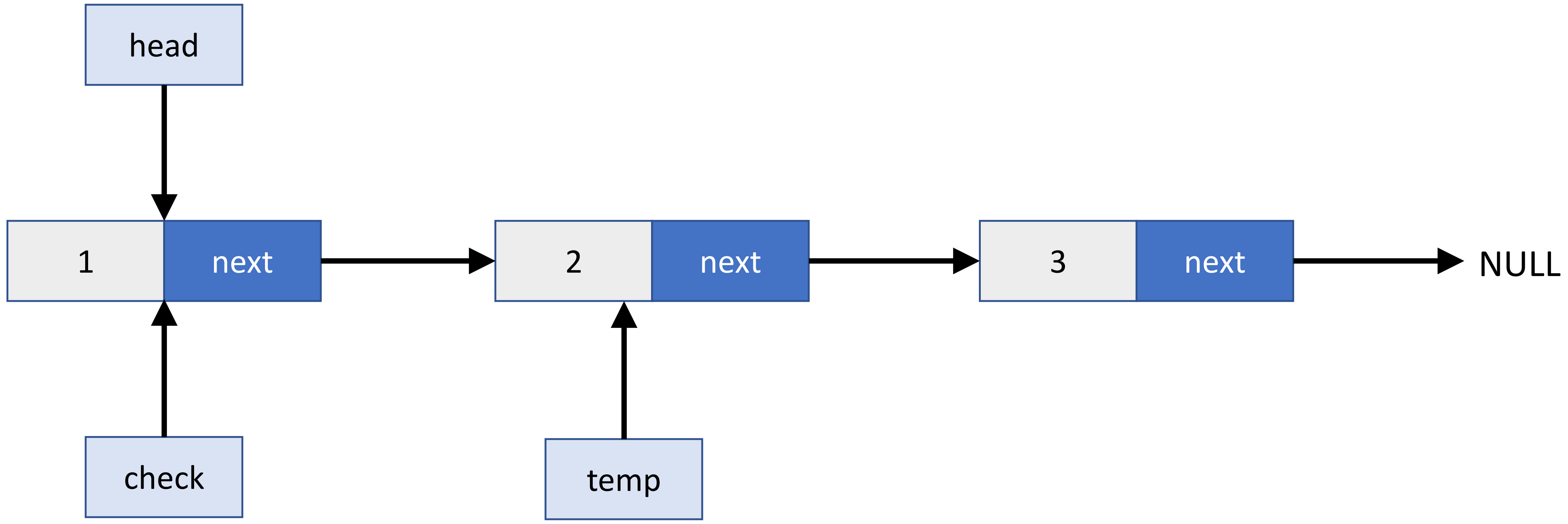
Linked List (`delete_after`)



Linked List (**`delete_after`**)

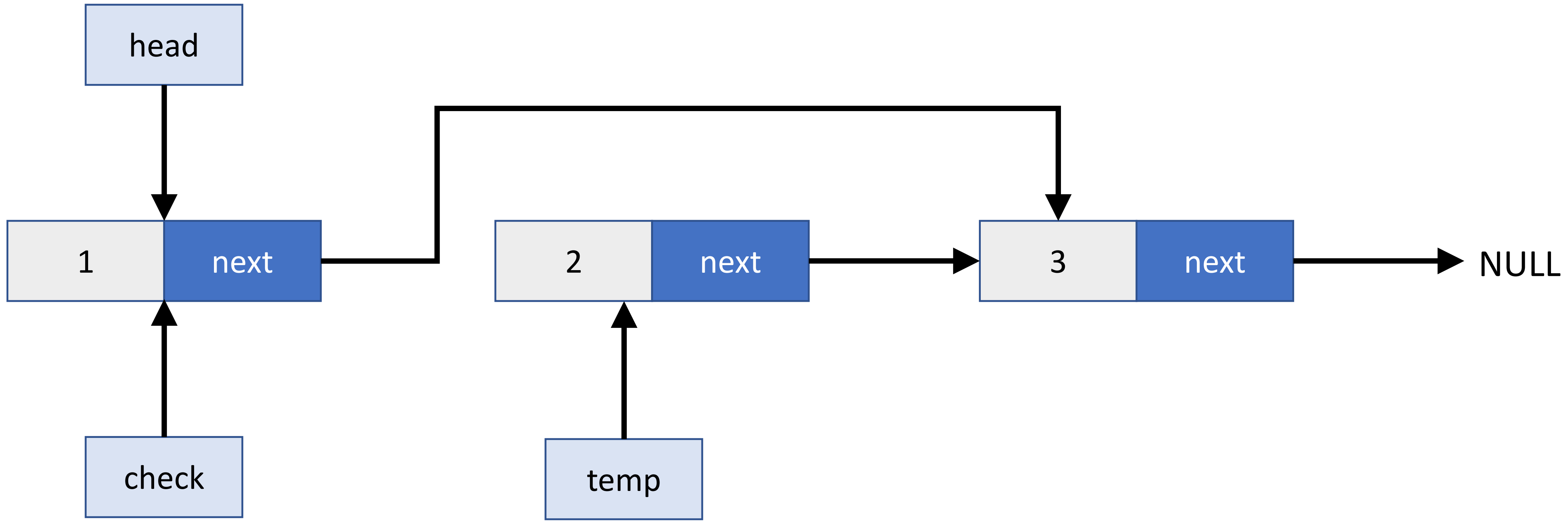


Linked List (`delete_after`)



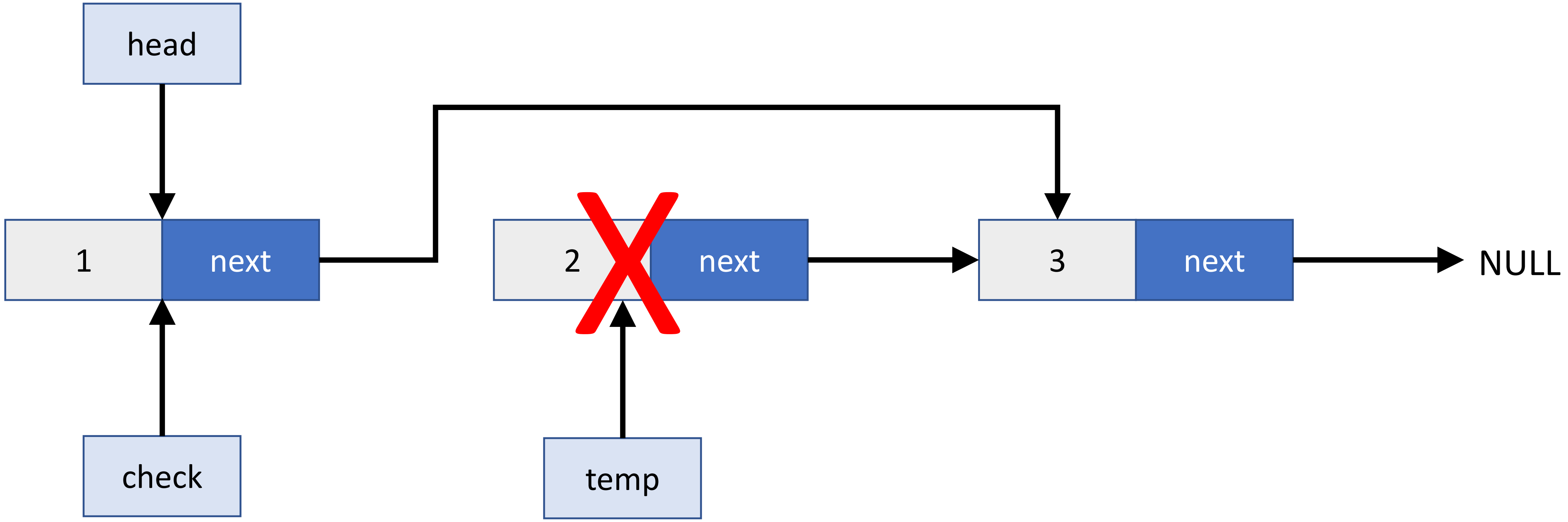
Node* temp=check->next;

Linked List (**`delete_after`**)



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

Linked List (`delete_after`)



delete temp;

Linked List (delete_after)

```
void LinkedList::delete_after(int n)
{
    if (head == NULL)
        return;

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

        Node* temp = check->next;
        check->next = check->next->next;
        delete temp;
    }
}
```

Thanks a lot

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