Data Structures and Object Oriented Programming

Lecture 5

Dr. Naveed Anwar Bhatti

Webpage: naveedanwarbhatti.github.io

Object-Oriented Programming in C++

Linked List

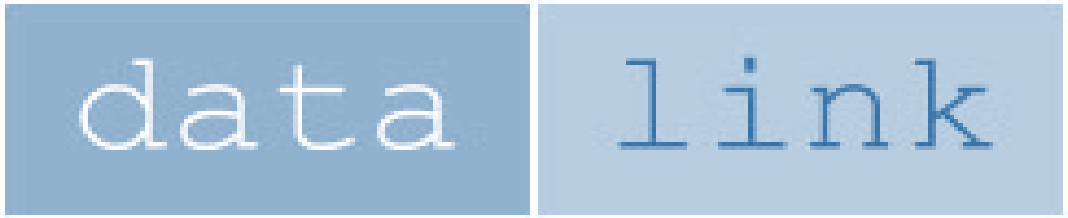
Why we need *Linked List*?

- Arrays are slower when it comes to dynamic data…we need something else

- **Problems with an array**
  - Array size is fixed (even when we dynamically allocate it)
  - Sorted array: insertion and deletion is slow because it requires data movement

Linked List

- **Linked list: a collection of items (nodes) containing two components:**
  - Data
  - Address (link) of the next node in the list



- Unlike arrays, linked list elements are not stored at a contiguous location; the elements are linked using pointers.

Linked List

- Example:
  - Link field in the last node is null

Linked List

- A node is declared as a **`class`** or **`struct`**
  - Data type of a node depends on the specific application
  - Link component of each node is a pointer

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

- Variable declaration:

```
Node* Head;
```

Linked List

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

```
int main()
{

}
```

Linked List

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

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

}
```

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;









}
```

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




}
```

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

}
```

Linked List

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

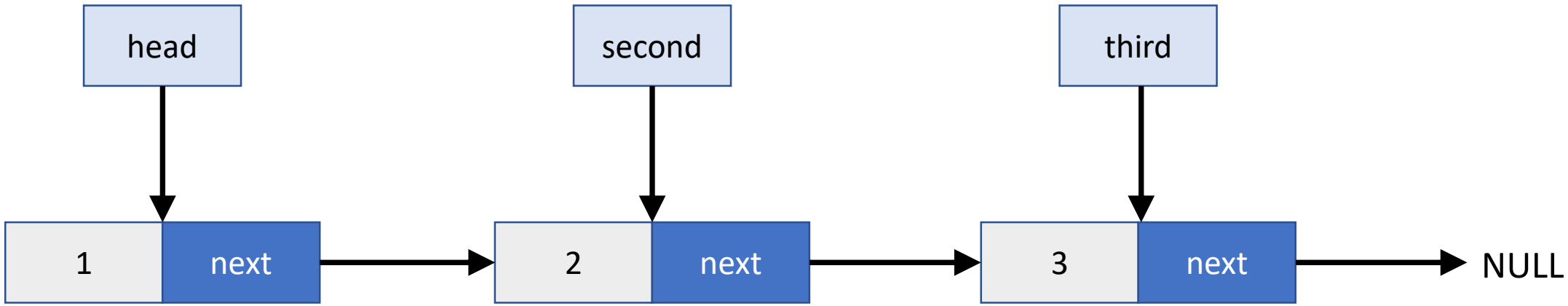
```
int main()
{
    Node* head;
    Node* second;
    Node* third;

    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



}
```

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



Linked List Traversal

```
void printList(Node* n)
{
   while (n != NULL) {
      cout << n->data;
      n = n->next;
   }
}
```

Linked List

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


void printList(Node* n)
{
    while (n != NULL) {
        cout << n->data;
        n = n->next;
    }
}
```

```
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

    printList(head);
    return 0;
}
```

Linked List

- **Exercise: Write a function which tells the total length of the linked list**

Thanks a lot

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