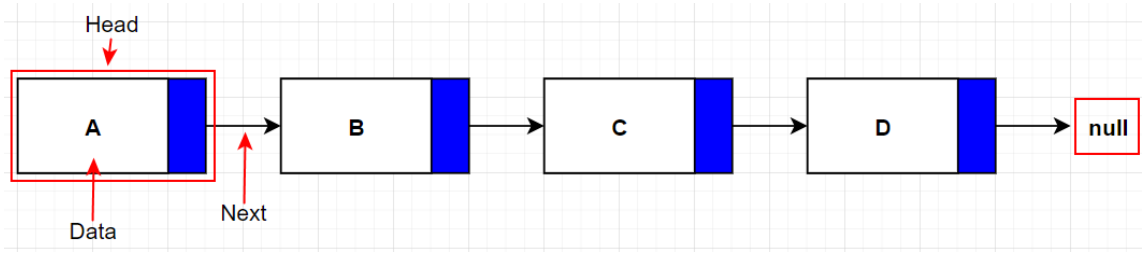
Bài 10.4: Giới thiệu danh sách liên kết

- ✓ Khái niệm
- ✓ Mục đích sử dụng
- ✓ Tạo node, danh sách LK
- ✓ Các thao tác chèn, duyệt danh sách
- ✓ Ví dụ minh họa & bài tập

Khái niệm

- ✓ Danh sách liên kết là kiểu cấu trúc dữ liệu tuyến tính trong đó các phần tử của nó không bắt buộc lưu trữ ở các vùng nhớ liền kề nhau.
- ✓ Các phần tử của danh sách liên kết được kết nối với nhau qua con trỏ.
- ✓ Hay nói cách khác, một danh sách liên kết bao gồm các node trong đó mỗi node chứa dữ liệu của nó và tham chiếu đến node liên quan.



Mục đích sử dụng

- ✓ Danh sách liên kết được sử dụng để lưu trữ tập dữ liệu có kích thước lớn, tổng số phần tử trong tập là chưa biết trước và có thể thay đổi.
- ✓ Sử dụng danh sách liên kết khi không cần truy cập ngẫu nhiên vào bất kì phần tử nào trong tập hợp.
- ✓ Sử dụng danh sách liên kết để cho phép ta chèn, xóa phần tử tại vị trí bất kì trong tập hợp.
- ✓ Danh sách liên kết khác với mảng: chỉ phù hợp với tập dữ liệu kích thước nhỏ, cố định và kích thước đã được biết trước.

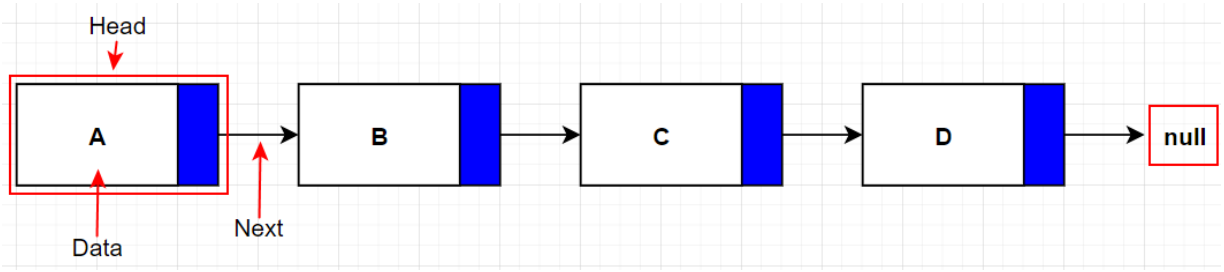
Phân loại

- ✓ Danh sách liên kết đơn.
- ✓ Danh sách liên kết đôi.
- ✓ Danh sách liên kết đơn vòng.
- ✓ Danh sách liên kết đôi vòng.

Danh sách liên kết đơn

- ✓ Khi nói về danh sách liên kết, mặc định ta đang nói về danh sách liên kết đơn.
- ✓ Cấu trúc dữ liệu này bao gồm nhiều phần tử được nối với nhau theo một chiều từ đầu đến cuối.
- ✓ Mỗi phần tử trong danh sách liên kết được gọi là một node.
- ✓ Mỗi node gồm 2 thành phần chính: dữ liệu của node và địa chỉ của node kế tiếp.
- ✓ Phần dữ liệu của node có thể là dữ liệu đơn: giá trị số, kí tự.. Hoặc giá trị phức tạp như thông tin sinh viên…
- ✓ Phần địa chỉ của node kế tiếp được lưu trữ trong một con trỏ cùng kiểu với kiểu của node hiện tại.

Danh sách liên kết đơn & tạo node



✓ Biểu diễn node của danh sách liên kết đơn:

```
8 references
class Node<T>
{
    3 references
    public T Data { get; set; } // phần dữ liệu của node
    8 references
    public Node<T> Next { get; set; } // tham chiếu next

    0 references
    public Node() { } // phương thức khởi tạo không tham số

    3 references
    public Node(T data) // phương thức khởi tạo có tham số
    {
        Data = data;
        Next = null;
    }
}
```

Tạo danh sách liên kết

Tiếp theo ta tạo danh sách liên kết đơn với các tùy chọn:

- ✓ Tạo danh sách liên kết chỉ gồm node head.
- ✓ Tạo danh sách liên kết bao gồm hai node head, tail.
- ✓ Trong khóa học này sẽ tạo danh sách liên kết theo cả hai cách. Ưu điểm của cách thứ nhất là đơn giản. Nhược điểm là phải thực hiện nhiều thao tác để chèn node vào cuối danh sách liên kết.
- ✓ Với cách thứ 2, ưu điểm là dễ dàng chèn thêm node mới vào cuối danh sách. Nhược điểm là tốn thêm chi phí bộ nhớ cho node tail.

Tạo danh sách liên kết

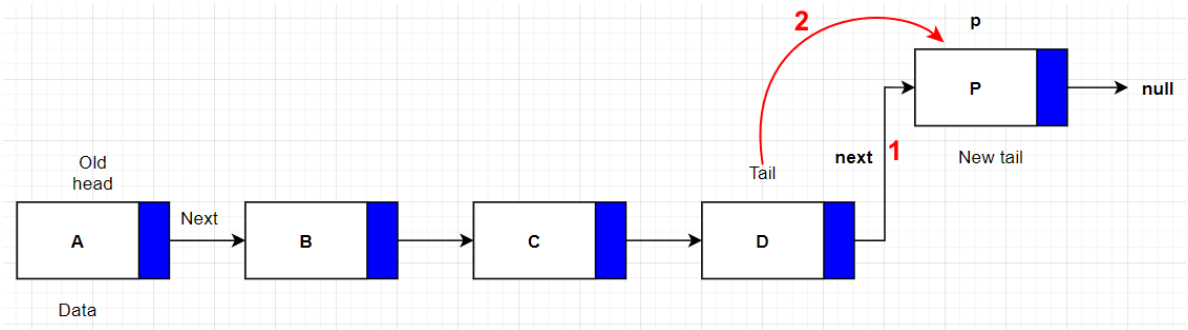
```
3 references
class LinkedList<T>
{
    8 references
    public Node<T> First { get; set; }
    6 references
    public Node<T> Last { get; set; }

    1 reference
    public LinkedList()...
    // thêm node vào đầu danh sách
    0 references
    public void AddFirst(T data)...
    // thêm node vào cuối danh sách
    4 references
    public void AddLast(T data)...
    // thêm node vào sau node x
    1 reference
    public void AddAfterX(T data, T x)...
    // kiểm tra rỗng
    4 references
    public bool IsEmpty()...
    // hiển thị các node trong danh sách
    1 reference
    public void ShowNodes()...
}
```

Chèn vào đầu danh sách

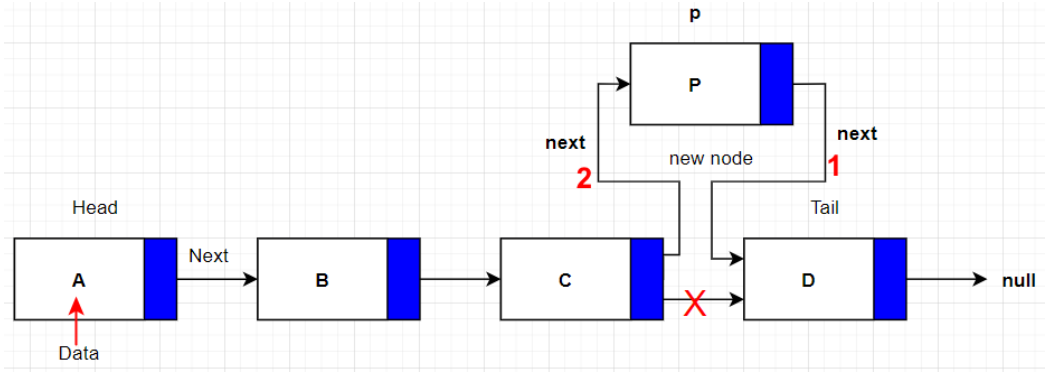
```
// thêm node vào đầu danh sách
0 references
public void AddFirst(T data)
{
    var newNode = new Node<T>(data);
    if (IsEmpty())
    {
        First = newNode;
        Last = newNode;
    } else
    {
        newNode.Next = First;
        First = newNode;
    }
}
```

Chèn vào cuối danh sách




```
// thêm node vào cuối danh sách
4 references
public void AddLast(T data)
{
    var newNode = new Node<T>(data);
    if (IsEmpty())
    {
        First = newNode;
        Last = newNode;
    }
    else
    {
        Last.Next = newNode;
        Last = newNode;
    }
}
```

Chèn vào sau node x trong danh sách




```
// thêm node vào sau node x
1 reference
public void AddAfterX(T data, T x)
{
    if(!IsEmpty())
    {
        var currentNode = First;
        while(currentNode != null)
        {
            if(currentNode.Data.Equals(x))
            {
                var newNode = new Node<T>(data);
                newNode.Next = currentNode.Next;
                currentNode.Next = newNode;
            }
            currentNode = currentNode.Next;
        }
    }
}
```

Duyệt danh sách LK

```
// kiểm tra rỗng
4 references
public bool IsEmpty()
{
    return First == null && Last == null;
}
// hiển thị các node trong danh sách
1 reference
public void ShowNodes()
{
    if(!IsEmpty())
    {
        var x = First;
        while (x != null)
        {
            Console.Write($"{x.Data} -> ");
            x = x.Next;
        }
        Console.WriteLine("null");
    }
}
```

Nội dung tiếp theo

Danh sách liên kết đôi