HCMUTE

CHƯƠNG 4 : KIỂU DỮ LIỆU DANH SÁCH

HCMUTE

Nội dung

1. Khái niệm danh sách
2. Phân loại danh sách
3. Linked List ADT
4. So sánh Linked List và Arra
5. Cài đặt danh sách bằng Array
6. Cài đặt Linked List

HCMUTE

1. Khái niệm danh sách

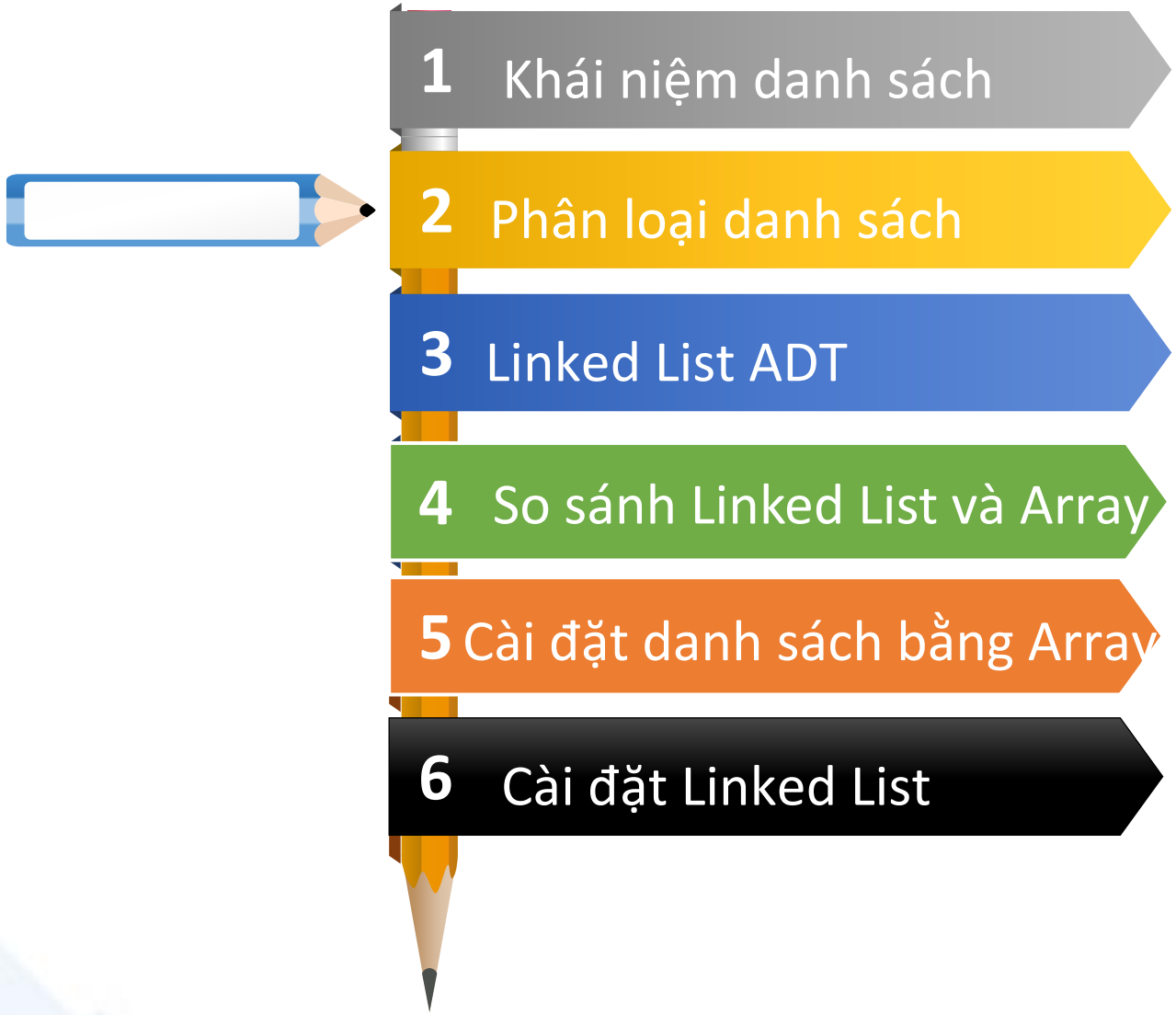
- Danh sách là một tập hợp hữu hạn các phần tử có cùng một kiểu. Danh sách là một dãy các phần tử của nó: $a_1, a_2, \ldots, a_n$ với $n \geq 0$. Nếu n=0 ta nói danh sách rỗng (empty list).
- Nếu $n > 0$ ta gọi a_1 là phần tử đầu tiên và a_n là phần tử cuối cùng của danh sách.
- Số phần tử của danh sách được gọi là độ dài của danh sách.
- Các phần tử của danh sách có thứ tự tuyến tính theo vị trí (position) xuất hiện của các phần tử.
- Ta nói a_i đứng trước a_{i+1}, với i từ 1 đến n-1

1. Khái niệm danh sách

Ví dụ thực tế thường gặp những danh sách sau:

- Danh sách học sinh
- Danh mục sách trong thư viện
- Danh bạ điện thoại
- Danh sách các nhân viên trong công ty
- Danh sách các cuộc gọi video đang chờ được xử lý
- ...

Nội dung

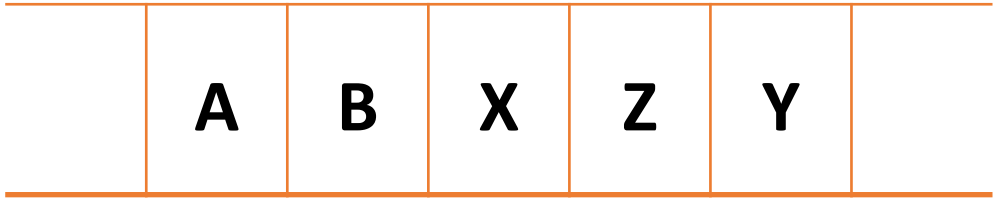

1 Khái niệm danh sách
2 Phân loại danh sách
3 Linked List ADT
4 So sánh Linked List và Array
5 Cài đặt danh sách bằng Array
6 Cài đặt Linked List

2. Phân loại danh sách

- Có 2 loại danh sách tuyến tính:
  - ✓ Danh sách đặc (mảng các phần tử)
  - ✓ Danh sách liên kết (linked list)

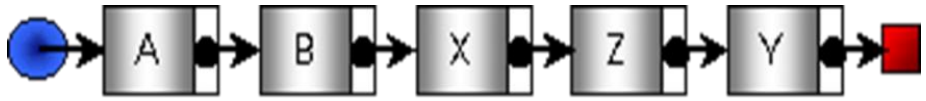
2. Phân loại danh sách

- **Danh sách đặc (array)**: Các phần tử được bố trí nằm liên tục với nhau, là dãy các ô nhớ liên tiếp nên được gọi là danh sách đặc



2. Phân loại danh sách

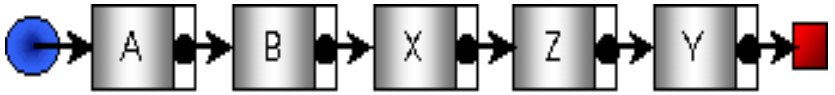
- **Danh sách liên kết (linked list)**:



  - ✓ Các phần tử trong danh sách liên kết tự quản lý nhau bằng cách sử dụng con trỏ.
  - ✓ Thành phần con trỏ của phần tử cuối cùng bằng NULL (dấu hiệu kết thúc danh sách).
  - ✓ Có thể thay đổi kích thước (gia tăng phần tử hoặc giảm phần tử) trong quá trình thực thi
  - ✓ Không lãng phí không gian bộ nhớ (nhưng với mỗi phần tử phải lưu trữ thêm con trỏ)

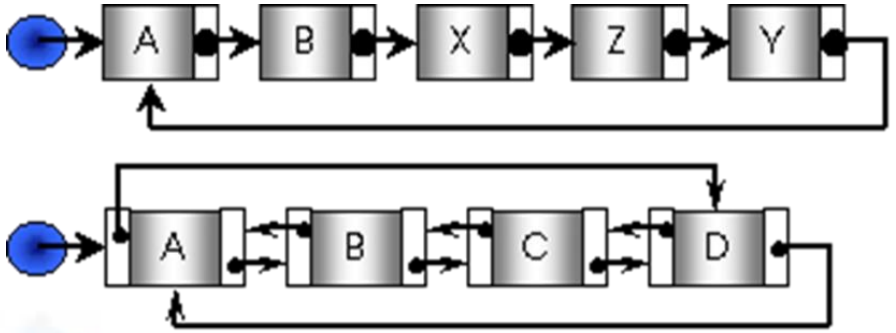
Phân loại danh sách

- Tùy thuộc vào mức độ và cách thức kết nối mà danh sách liên kết có thể chia ra nhiều loại khác nhau:
  - ✓ **Danh sách liên kết đơn**

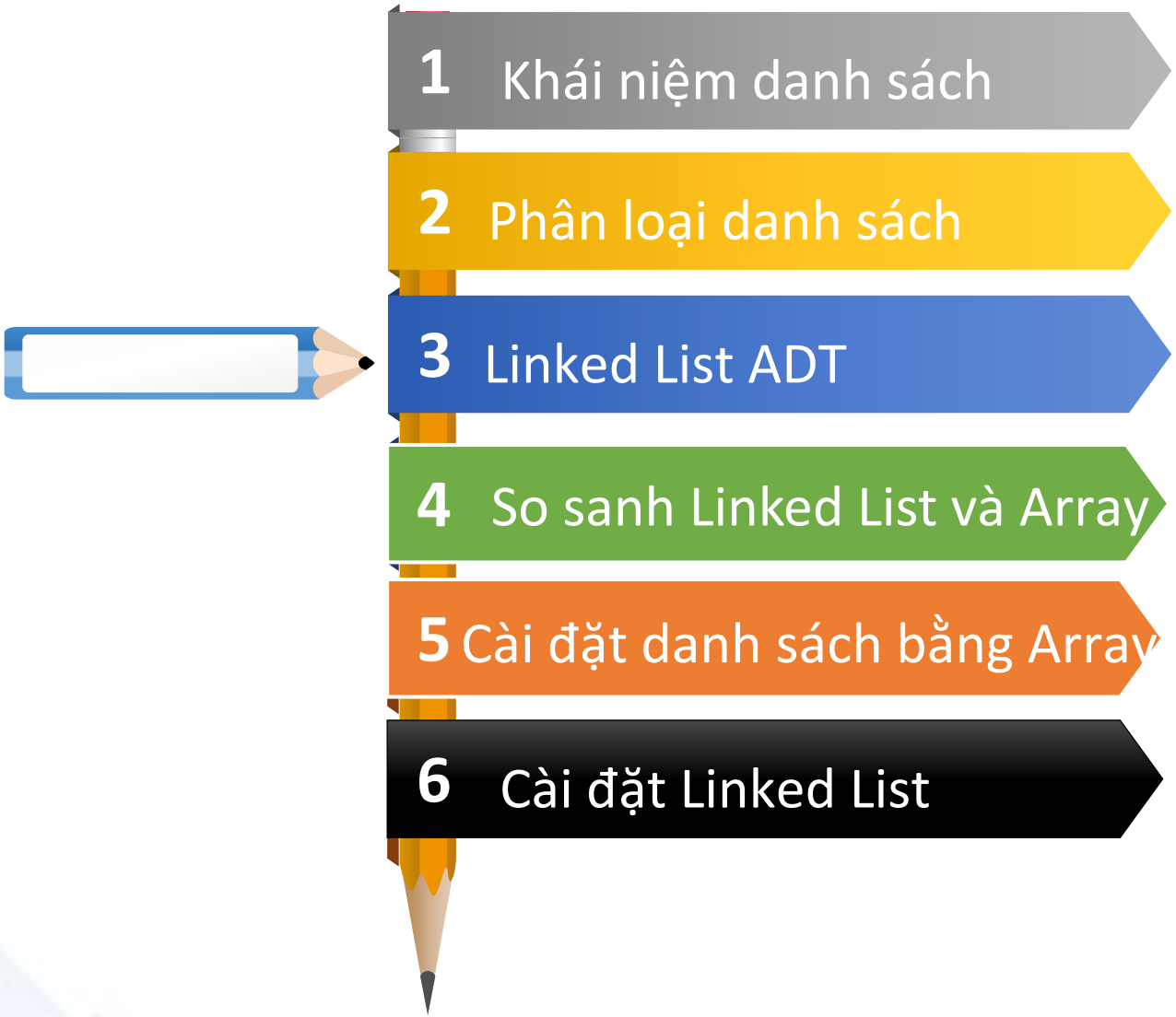


  - ✓ **Danh sách liên kết đôi/kép**

  - ✓ **Danh sách liên kết vòng (vòng đơn, vòng đôi).**



Nội dung


1 Khái niệm danh sách
2 Phân loại danh sách
3 Linked List ADT
4 So sanh Linked List và Array
5 Cài đặt danh sách bằng Array
6 Cài đặt Linked List

3. Linked List ADT

- **Một kiểu dữ liệu** được **trang bị tập thao tác** trên nó thì được **gọi là kiểu dữ liệu trừu tượng ADT** (abstract data type).
- Linked list ADT sẽ có các thao tác sau:

Thao tác chính:

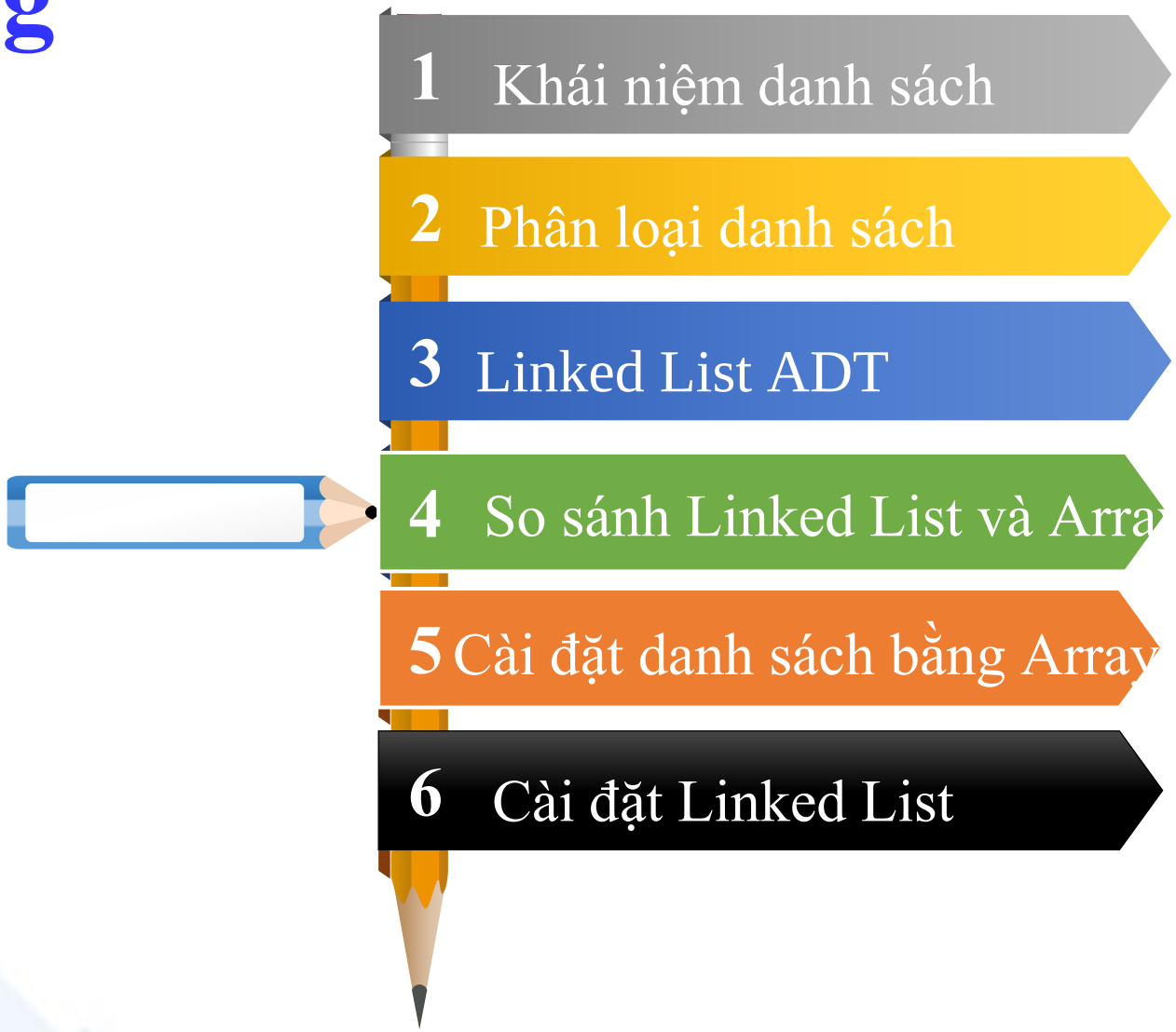
- ✓ Chèn (insert): Chèn một phần tử vào danh sách
- ✓ Xóa (delete): Xóa phần tử ra khỏi danh sách

Thao tác bổ trợ:

- ✓ Xóa danh sách (delete List): Xóa toàn bộ phần tử của danh sách
- ✓ Đếm: Trả về số phần tử có trong danh sách
- ✓ Tìm: Tìm phần tử (node) thứ n của danh sách
- ✓ ...

HCMUTE

Nội dung



4. So sánh Array và Linked List

Array

- Là 1 khối ô nhớ dày đặc.
- Truy cập phần tử trực tiếp, nhanh chóng.
- Phải khai báo trước kích thước. Khi muốn mở rộng thì phải tạo mảng mới và sao chép từ mảng cũ qua→ Nếu cấp phát nhiều thì gây lãng phí.

Linked List

- Các phần tử của danh sách không nhất thiết phải liên tục.
- Truy cập phần tử chi phí lớn. Xấu nhất là O(n).
- Danh sách liên kết có thể được mở rộng rất nhanh. Cấp phát từng phần tử một.

HCMUTE

4. So sánh Array và Linked List

Array

- Việc cấp phát vùng nhớ cho mảng đôi khi sẽ không thực hiện được khi kích thước lớn.
- Thao tác chèn, xóa phức tạp.

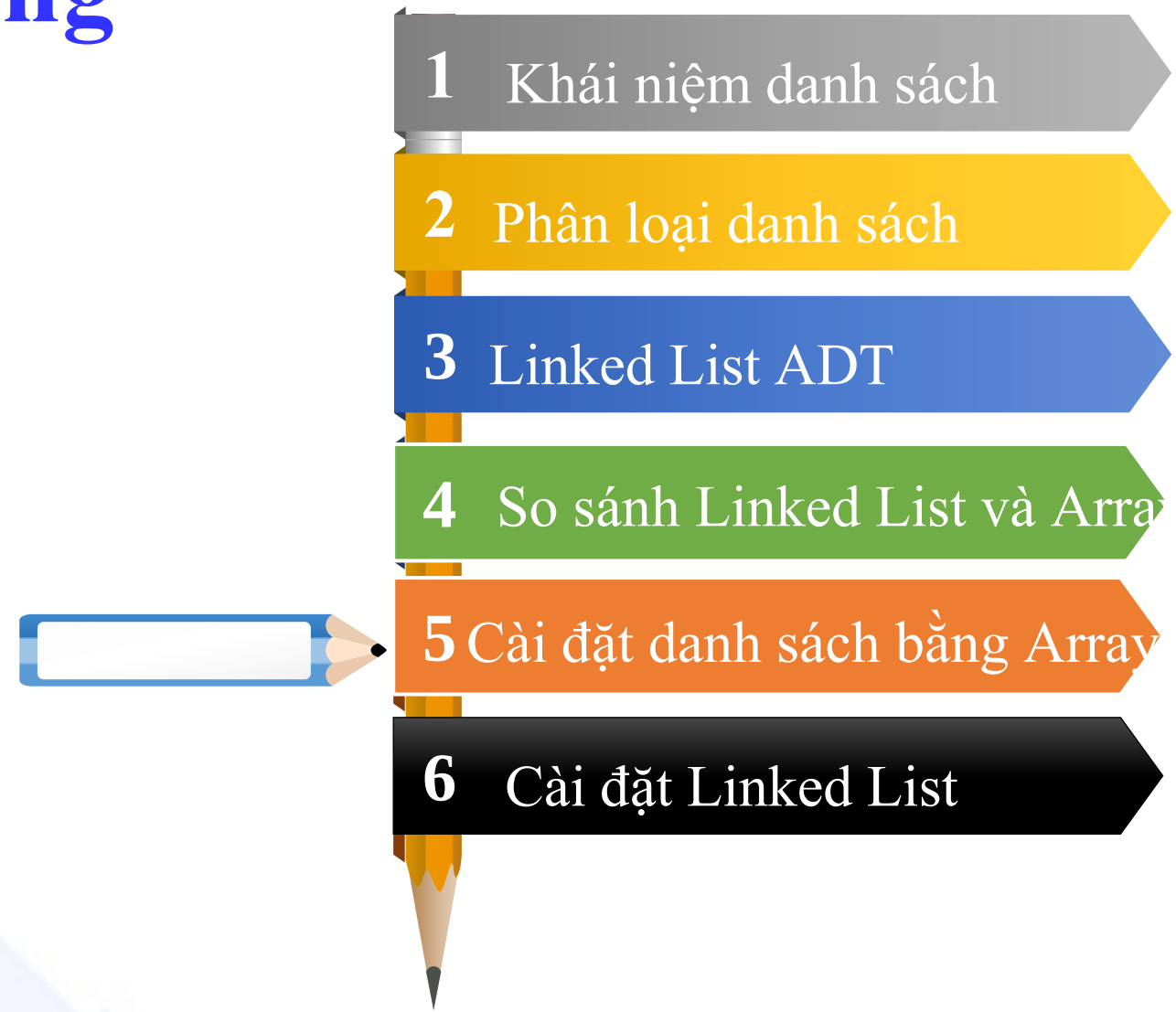
Linked List

- Mỗi phần tử phải mất thêm một vùng nhớ để lưu trữ thành phần con trỏ.

Tiêu chí	Linked List	Array
Truy cập	O(n)	O(1)
Chèn/Xóa ở đầu	O(1)	O(n), nếu mảng không đầy để có thể di chuyển được phần tử
Chèn ở cuối	O(n)	O(1), nếu mảng không đầy
Xóa ở cuối	O(n)	O(1)
Chèn ở giữa	O(n)	O(n), nếu mảng không đầy để có thể di chuyển phần tử
Xóa ở giữa	O(n)	O(n), nếu mảng không đầy để có thể di chuyển phần tử

HCMUTE

Nội dung



5. Cài đặt danh sách bằng mảng

- KHAI BÁO CẤU TRÚC DỮ LIỆU

```
class ArrayList
{
    private int[] _items; //mảng chứa các phần tử của danh sách
    private int _capacity;//số phần tử tối đa danh sách có thể chứa
    private int _size;    //số phần tử hiện có trong danh sách
    /*
    * Thuộc tính Count trả về số phần tử hiện có trong danh sách
    */
    public int Count
    {
        get
        {
            return _size;
        }
    }
    ...
}
```

5. Cài đặt danh sách bằng mảng

- KHỞI TẠO DANH SÁCH

```
/*
* Hàm tạo và khởi tạo ban đầu cho 1 danh sách
*/
public ArrayList(int maxSize)
{
    _capacity = maxSize;
    _size = 0;
    _items = new int[_capacity];
}
```

5. Cài đặt danh sách bằng mảng

- CÁC THAO TÁC

```
/*
* Kiểm tra danh sách có rỗng hay không
* return: true nếu danh sách rỗng, false nếu danh sách khác rỗng
*/
public bool IsEmpty()
{
    if (_size == 0)
        return true;
    return false;
}
/*
* Kiểm tra danh sách có đầy hay không
* return: true nếu danh sách đầy, flase nếu danh sách chưa đầy
*/
public bool IsFull()
{
    if (_size == _capacity)
        return true;
    return false;
}
```

5. Cài đặt danh sách bằng mảng

- CÁC THAO TÁC

```
/*
* Xóa một phần tử ra khỏi danh sách
* Tham số position: là vị trí cần xóa
*/
public void RemoveAt(int position)
{
    if (position >= 0 && position < _size && !IsEmpty())
    {
        for (int i = position + 1; i < _size; i++)
        {
            _items[i - 1] = _items[i];
        }
        _size--;
    }
    else throw new InvalidOperationException();
}
```

5. Cài đặt danh sách bằng mảng

- CÁC THAO TÁC

```
/*
* Chèn thêm một phần tử vào danh sách
* Tham số:
*         position: là vị trí cần chèn
*         data: dữ liệu cần chèn
*/
public void Insert(int position, int data)
{
    if (position >= 0 && position <= _size && !IsFull())
    {
        for (int i = _size; i > position; i--)
        {
            _items[i] = _items[i - 1];
        }

        _items[position] = data;
        _size++;
    }
    else throw new InvalidOperationException();
}
```

5. Cài đặt danh sách bằng mảng

- CÁC THAO TÁC

```
/*
    * Thêm một phần tử vào cuối danh sách
    * Tham số data: dữ liệu cần thêm
    */
public void Add(int data)
{
    if (!IsFull())
    {
        _items[_size] = data;
        _size++;
    }
    else throw new InvalidOperationException();
}
```

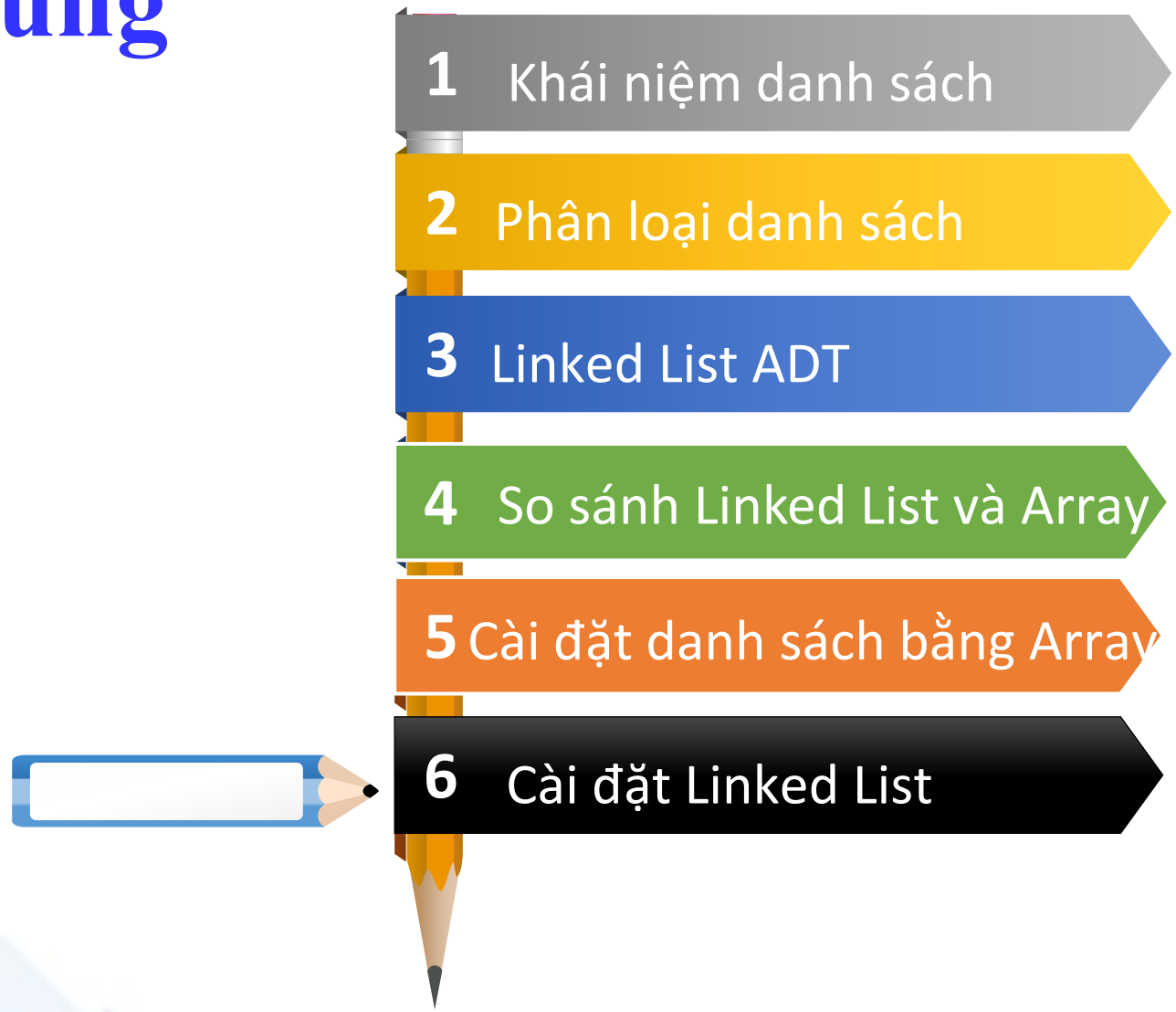
5. Cài đặt danh sách bằng mảng

- CÁC THAO TÁC

```
/*
* In danh sách
*/
public void PrintList()
{
    for (int i = 0; i < _size; i++)
    {
        Console.Write(_items[i] + " ");
    }
}
```

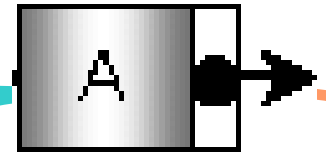
HCMUTE

Nội dung



Khai báo cấu trúc dữ liệu

```
Class Node
{
        internal Data_type Data;
        internal Node Next;
};
```



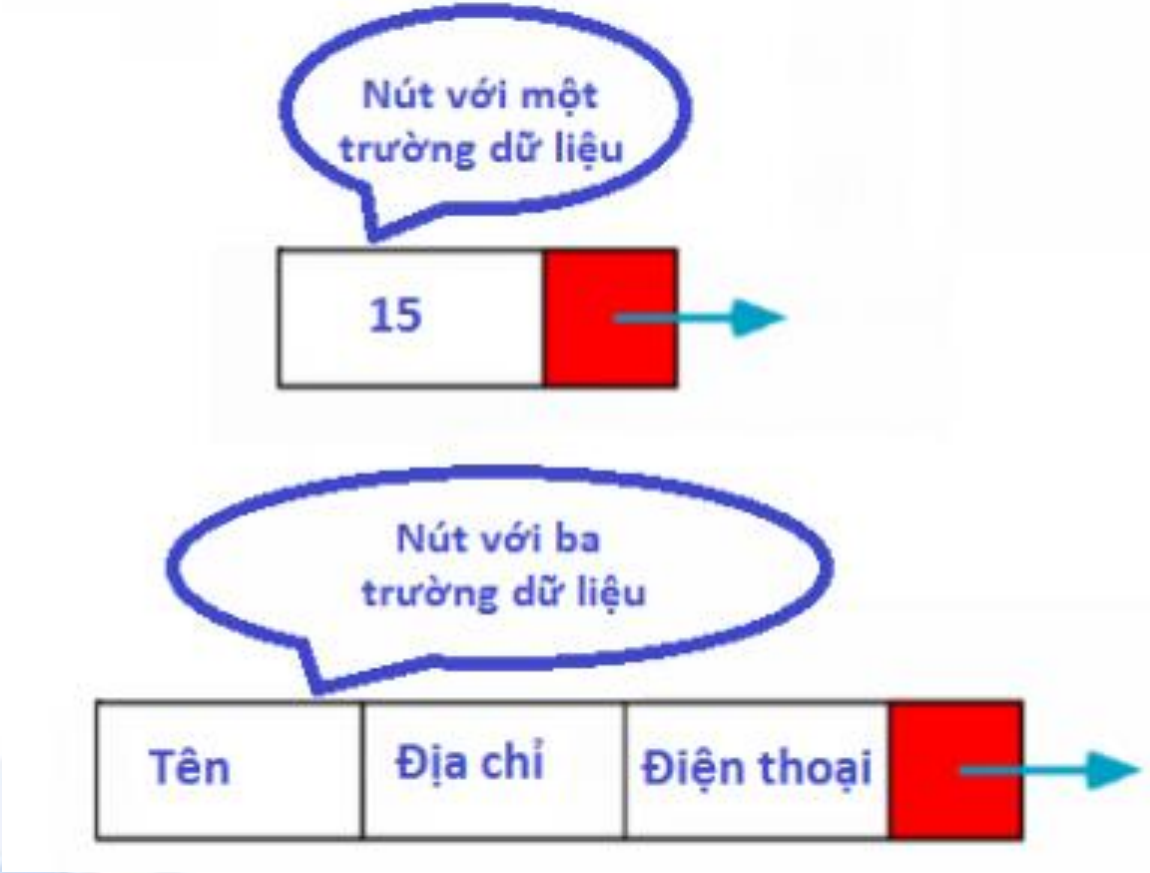
Thành phần dữ liệu:

❖Lưu trữ các thông tin về bản thân phần tử

Thành phần mối liên kết

❖Lưu trữ địa chỉ của phần tử kế tiếp trong danh sách, hoặc lưu trữ giá trị NULL nếu là phần tử cuối danh sách.

Khai báo cấu trúc dữ liệu



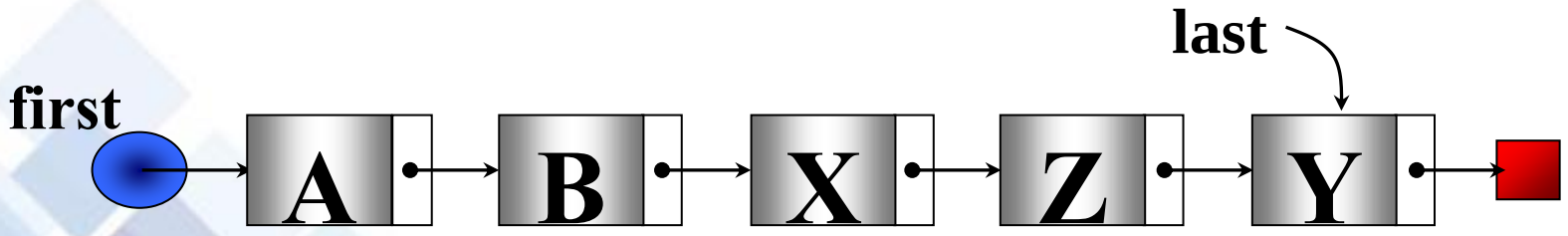
Khai báo cấu trúc dữ liệu

- Để quản lý một xâu đơn chỉ cần biết địa chỉ phần tử đầu xâu.
- Con trỏ **first** sẽ được dùng để lưu trữ địa chỉ phần tử đầu xâu, ta gọi **first** là đầu xâu. Ta có khai báo :

```
Node _first;
```

- Để tiện lợi, có thể sử dụng thêm một con trỏ **last** trỏ đến phần tử cuối xâu. Khai báo **last** như sau :

```
Node _last;
```



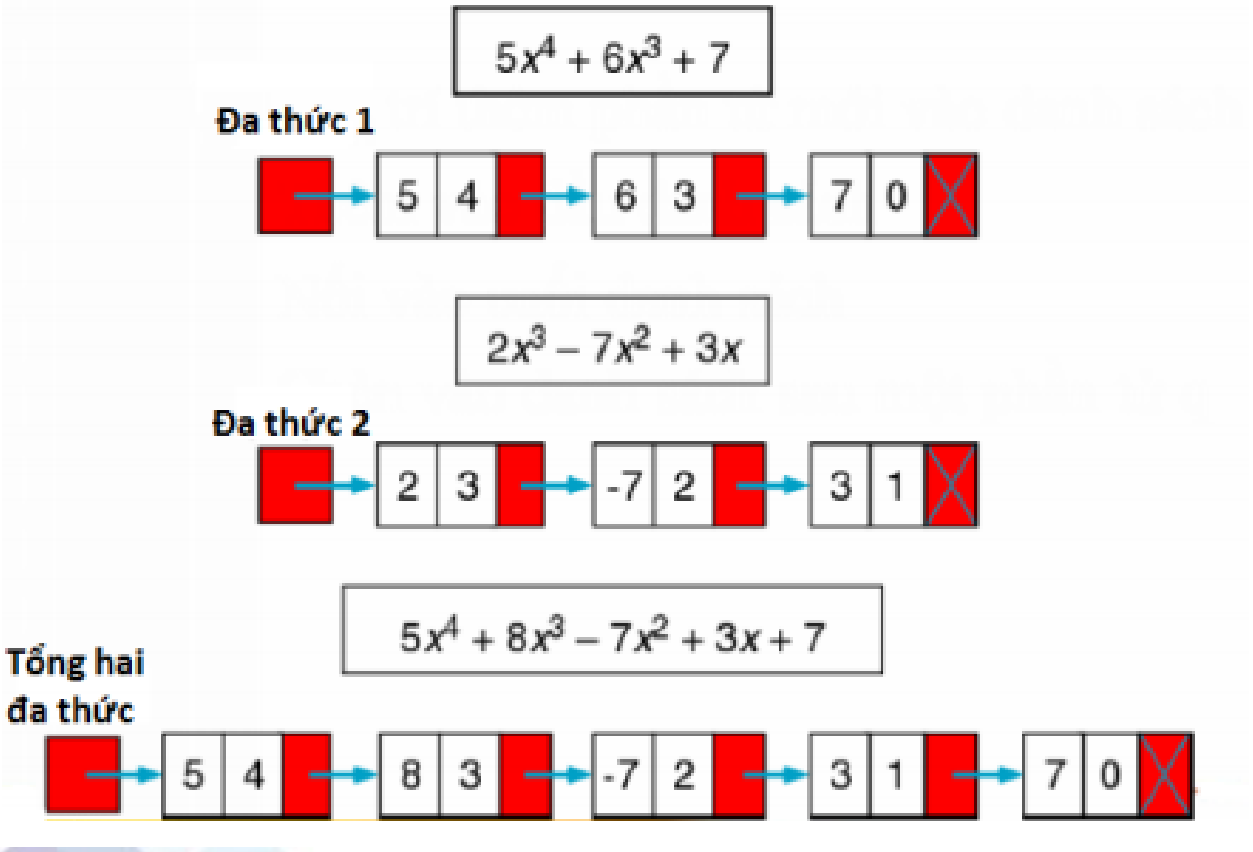
Khai báo cấu trúc dữ liệu

```
public class Node
{
    internal int Data;
    internal Node Next;
    public Node(int d)
    {
        Data = d;
        Next = null;
    }
}
```

```
public class LinkedList
{
    private Node _first;
    private Node _last;
    private int _size;
    public int Count
    {
        get
        {
            return _size;
        }
    }
    public LinkedList()
    {
        _first = null;
        _last = null;
        _size = 0;
    }
}
```

Khai báo cấu trúc dữ liệu

- Hãy khai báo cấu trúc dữ liệu cho một phần tử của danh sách liên kết sau:

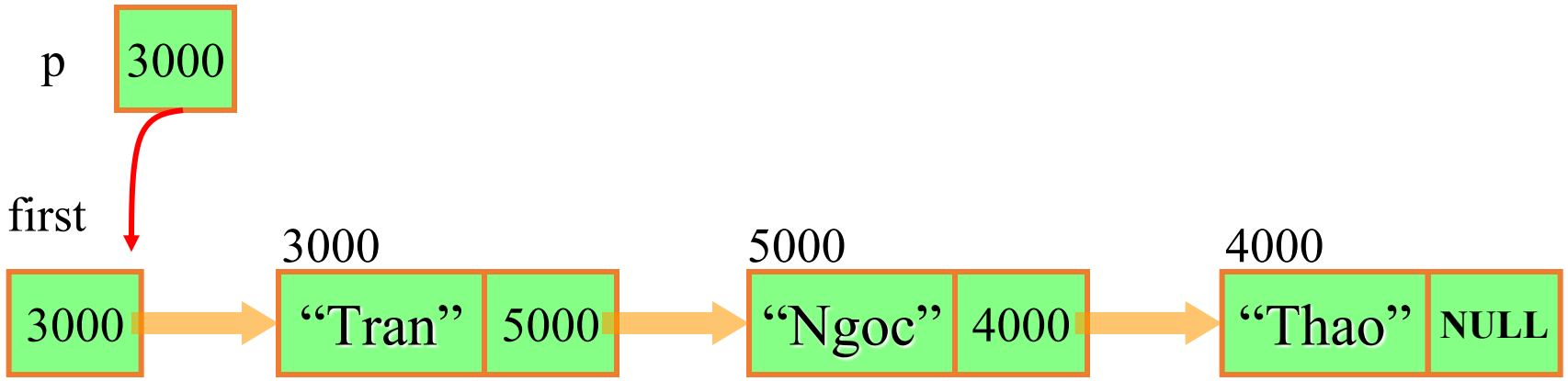


HCMUTE

Các thao tác trên Linked List

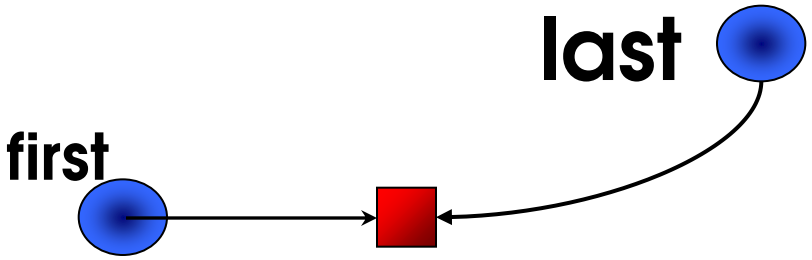
- ✓ Duyệt danh sách
- ✓ Khởi tạo danh sách
- ✓ Thêm một phần tử vào danh sách
- ✓Xóa một phần tử ra khỏi danh sách
- ✓ Tìm kiếm
- ✓ Sắp xếp

Duyệt danh sách




```
/*
* Xuat toan bo thong tin danh sach ra man hinh
*/
public void PrintList()
{
    Node p = _first;
    while (p != null)
    {
        Console.Write(p.Data + " ");
        p = p.Next;
    }
}
```

Khởi tạo danh sách

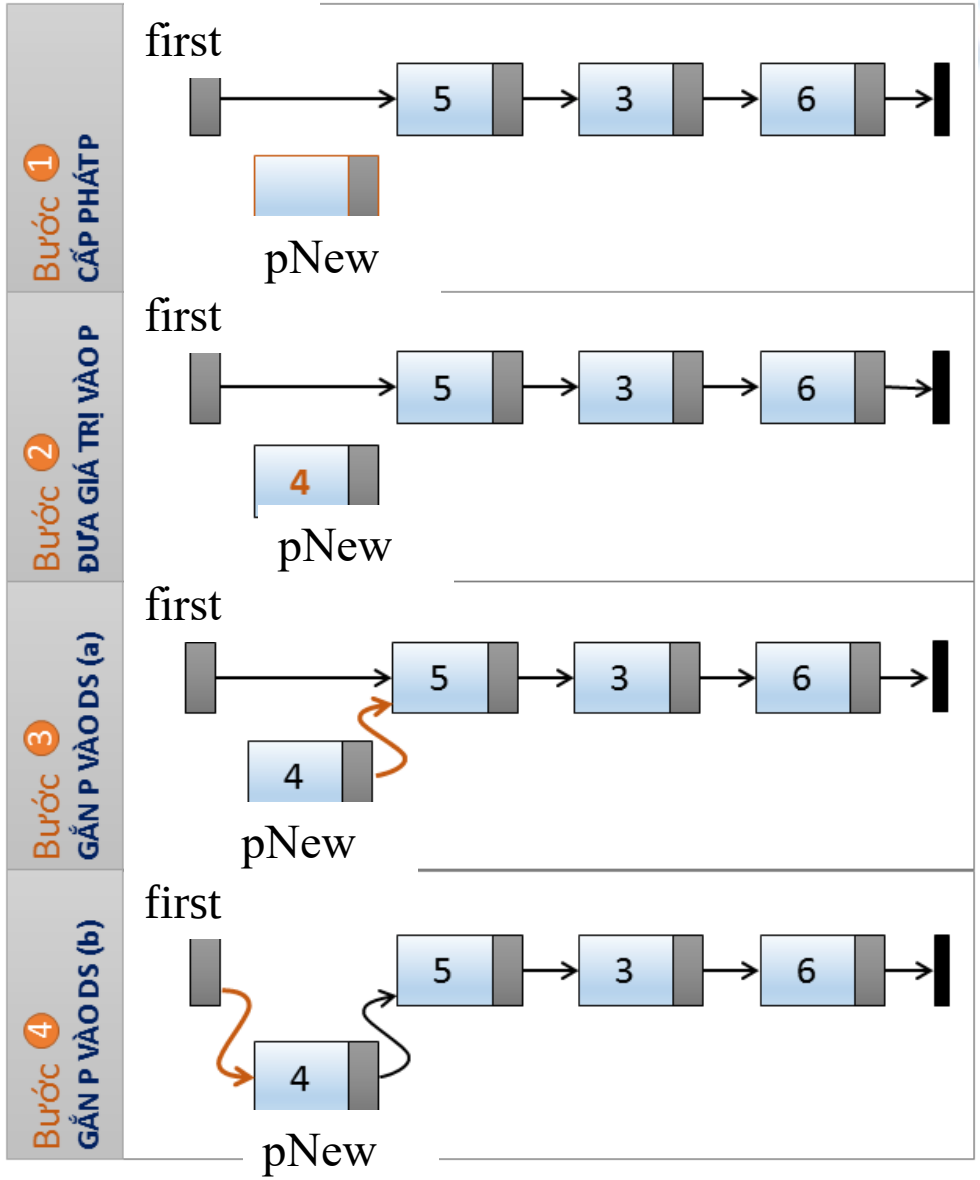



```
public LinkedList()
{
    _first = null;
    _last = null;
    _size = 0;
}
```

Thêm một phần tử danh sách

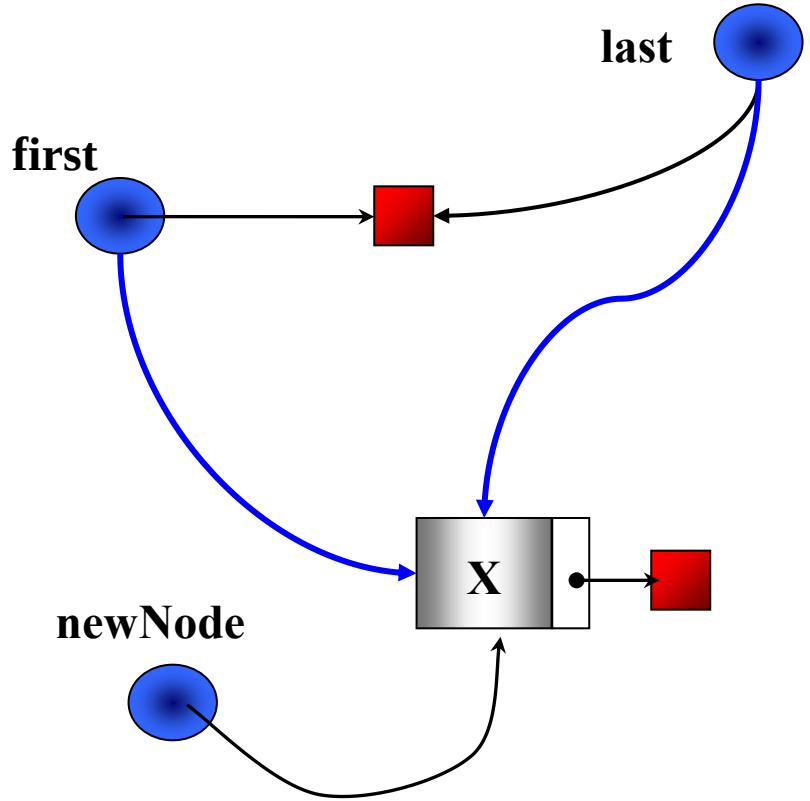
- Để thêm một phần tử vào danh sách chúng ta có thể:
  - ❖ Thêm vào đầu danh sách
  - ❖ Thêm vào cuối danh sách
  - ❖ Thêm vào giữa danh sách

Quy trình thêm



Thêm vào đầu danh sách

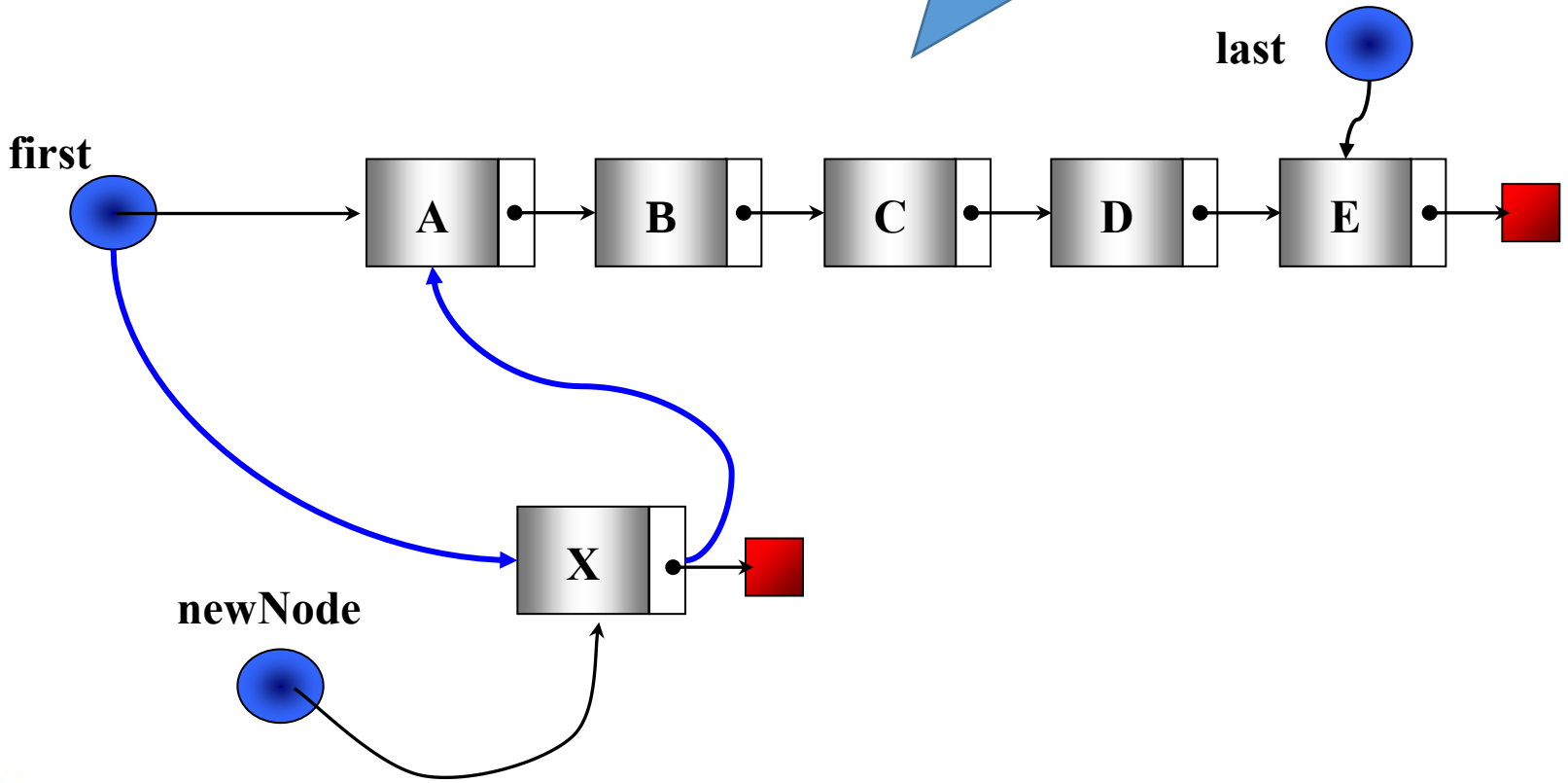
- Trường hợp danh sách rỗng



Thêm vào đầu danh sách

Dựa vào mô phỏng hãy cho biết con trỏ nào thay đổi?
Câu lệnh tương ứng là gì?

- Trường hợp danh sách khác rỗng

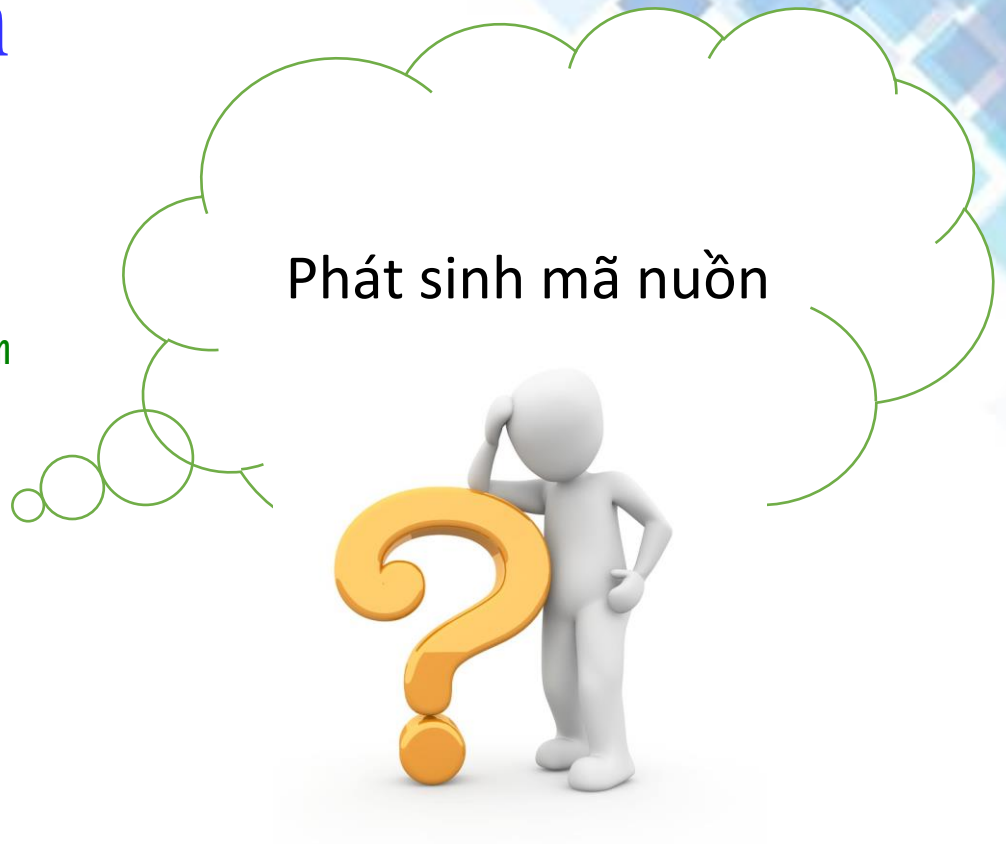


Thêm vào đầu danh sách

- Nếu Danh sách rỗng Thì
  - first = newNode;
  - last = newNode;
- Ngược lại
  - newNode.Next = first;
  - first = newNode;

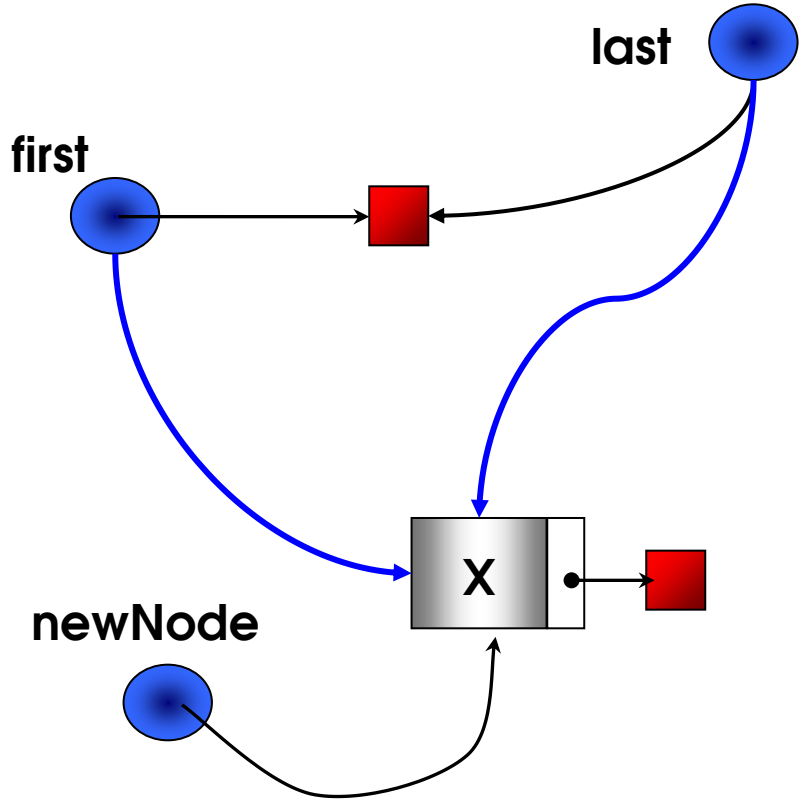
Thêm vào đầu danh sách

```
/*
* Them mot phan tu vao dau danh sach
* Tham so: newData la du lieu cua phan tu can them
*/
public void AddFirst(int newData)
{
    Node newNode = new Node(newData);
    if (_first == null)
    {
        _first = newNode;
        _last = newNode;
    }
    else
    {
        newNode.Next = _first;
        _first = newNode;
    }
    _size++;
}
```



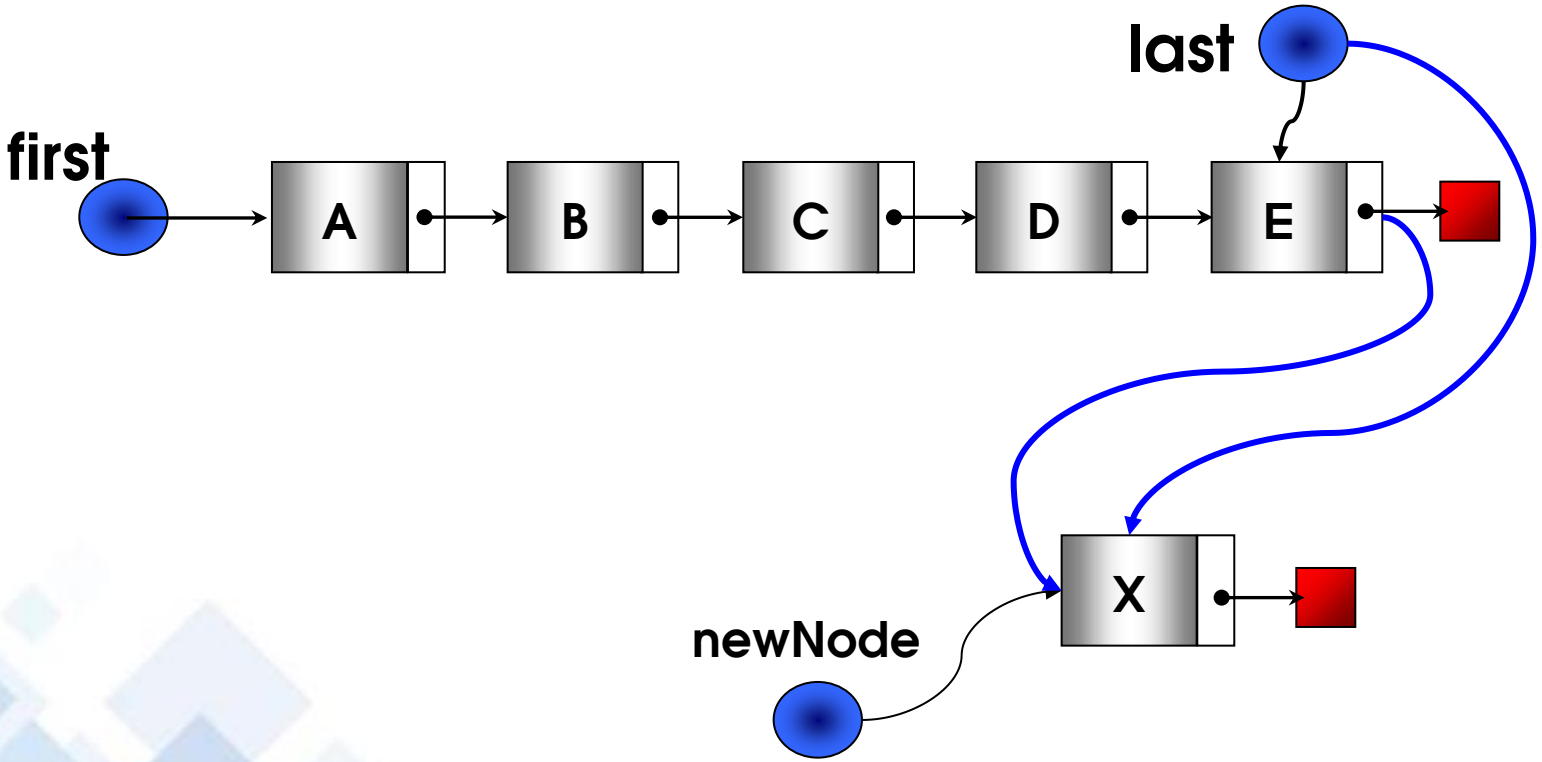
Thêm một phần tử vào cuối

- Trường hợp nếu danh sách rỗng



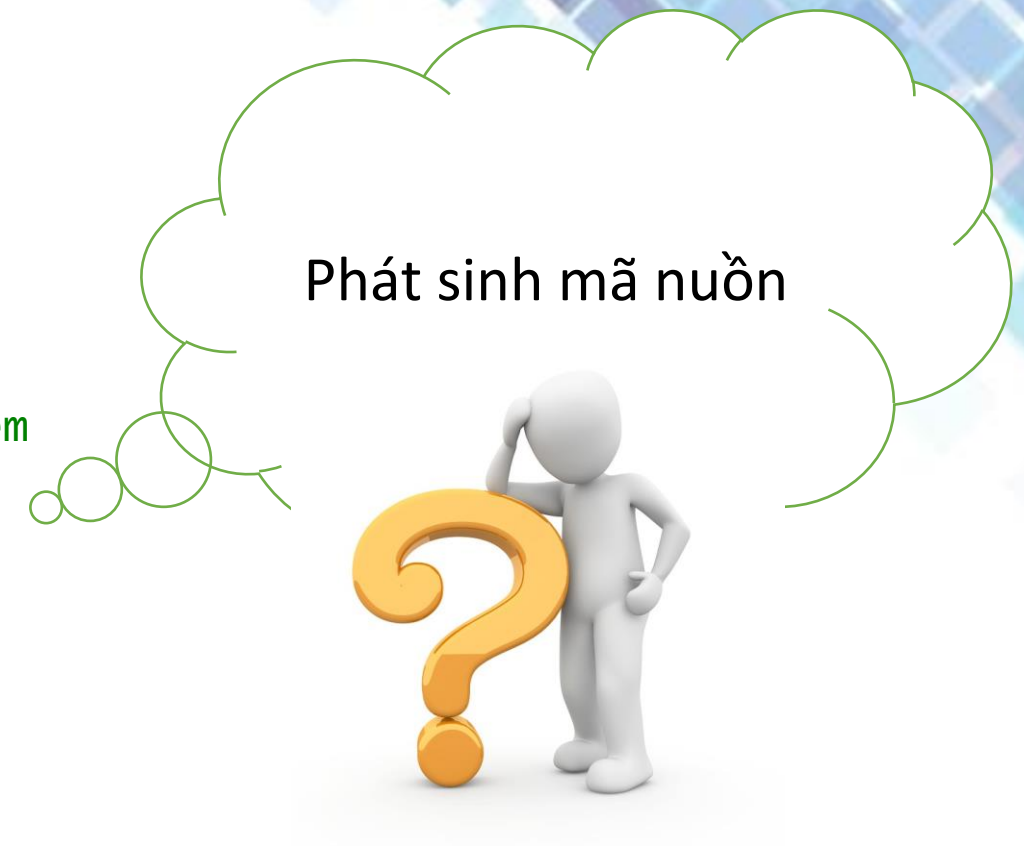
Thêm một phần tử vào cuối

- Nếu danh sách khác rỗng



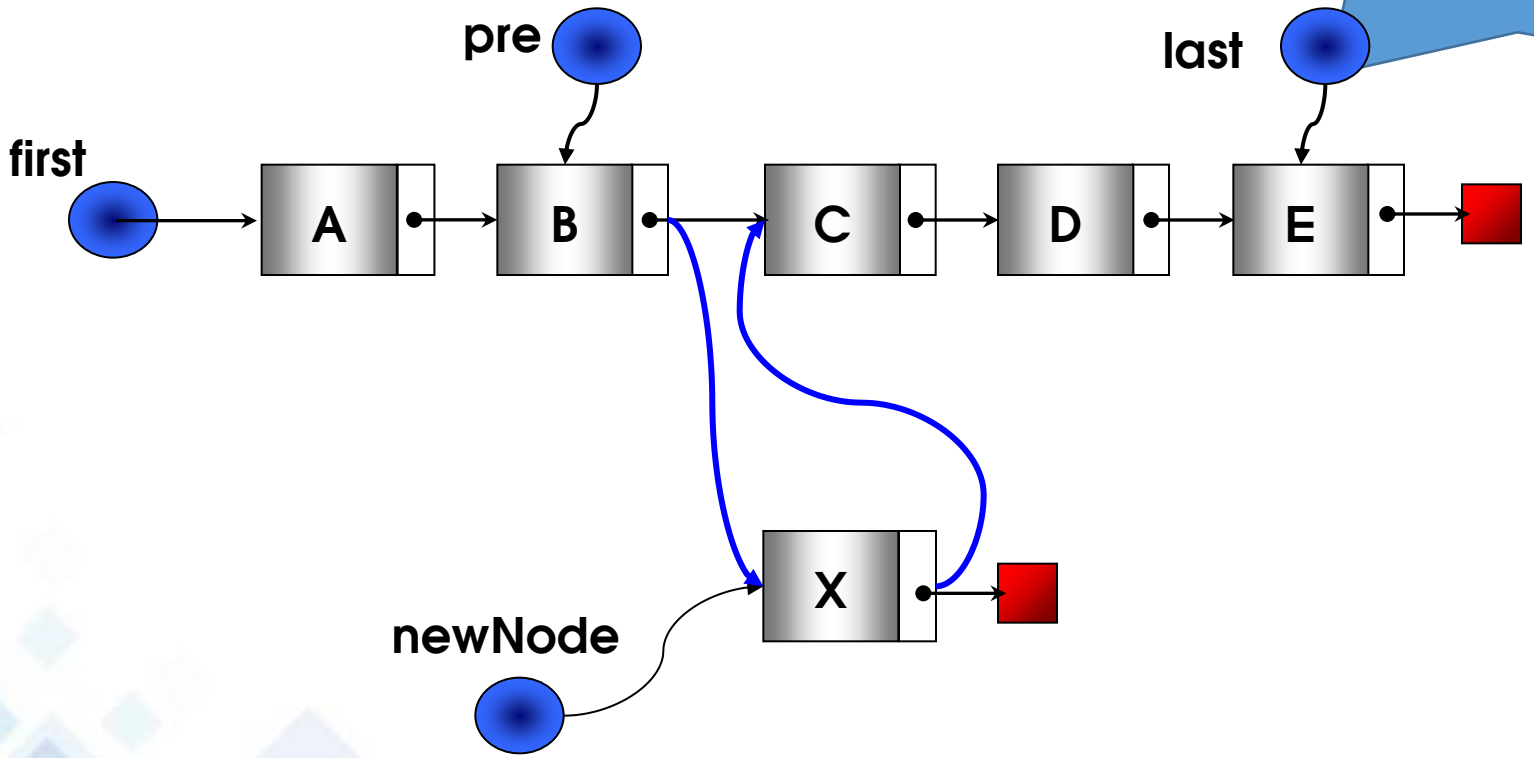
Thêm một phần tử vào cuối

```
/*
* Them mot phan tu vao cuoi danh sach
* Tham so: newData la du lieu cua phan tu can them
*/
public void AddLast(int newData)
{
    Node newNode = new Node(newData);
    if (_first == null)
    {
        _first = newNode;
        _last = newNode;
    }
    else
    {
        _last.Next = newNode;
        _last = newNode;
    }
    _size++;
}
```



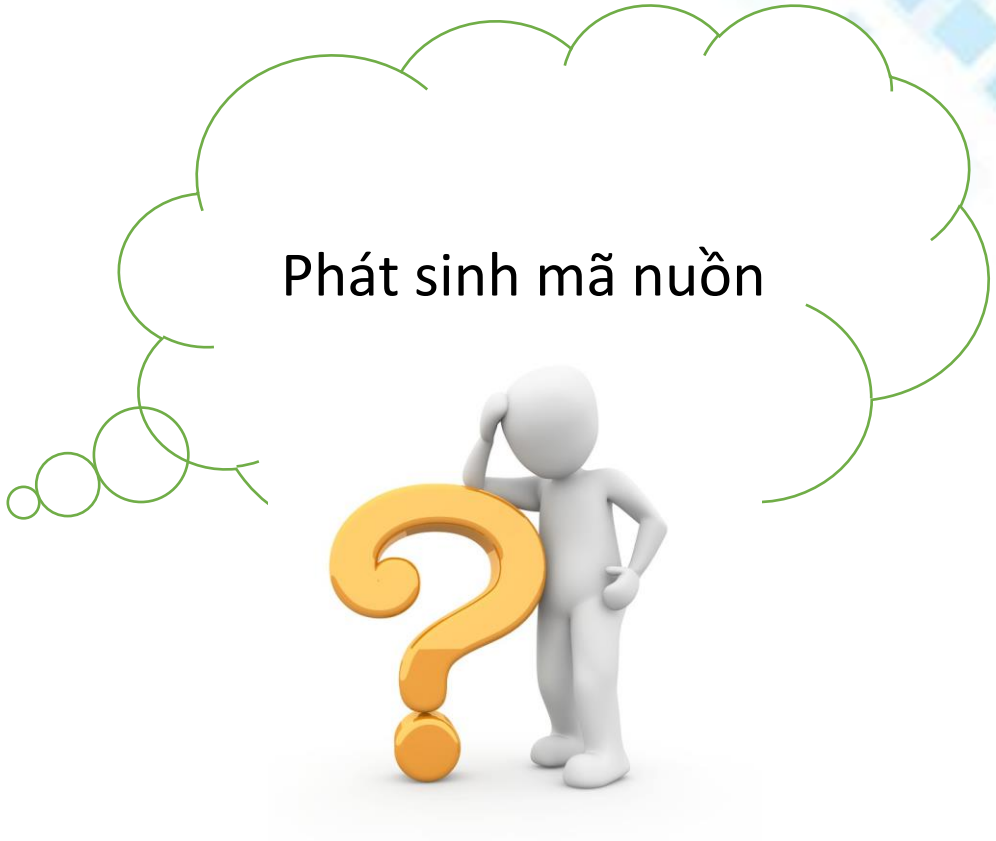
Chèn một phần tử vào sau

Dựa vào mô phỏng hãy cho biết con tro nào thay đổi? Câu lệnh tương ứng là gì?



Chèn một phần tử vào sau

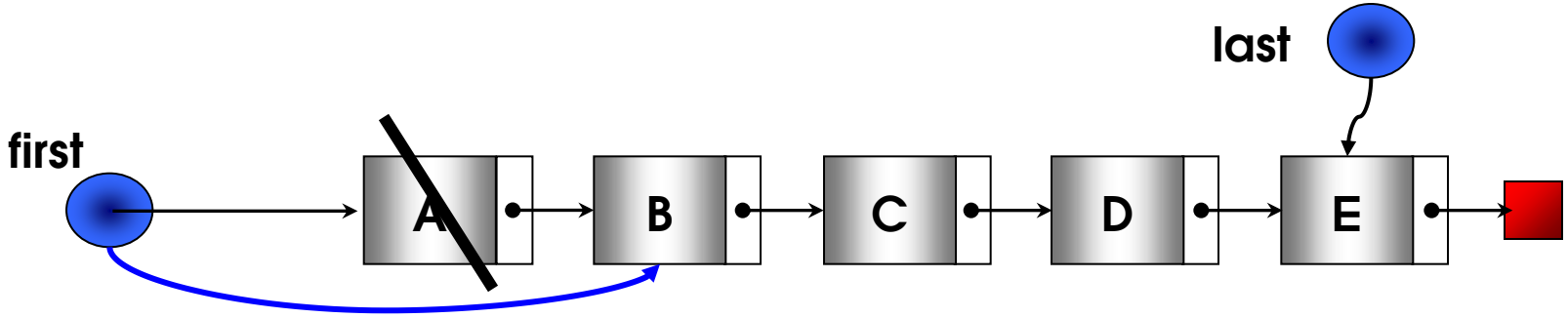
```
/*
* Them mot phan tu sau mot phan tu khac
* Tham so:
*       pre: chi vao phan tu dung truoc
*       newData: du lieu cua phan tu moi can them
*/
public void AddAfter(Node pre, int newData)
{
    if (pre != null)
    {
        Node newNode = new Node(newData);
        newNode.Next = pre.Next;
        pre.Next = newNode;
        if (pre == _last)
            _last = newNode;
        _size++;
    }
}
```



Xóa một phần tử trong danh sách

- Xóa một phần tử ở đầu danh sách
- Xóa một phần tử ở cuối danh sách
- Xóa một phần tử sau một phần tử khác
- Xóa một phần tử có khóa là x bất kỳ

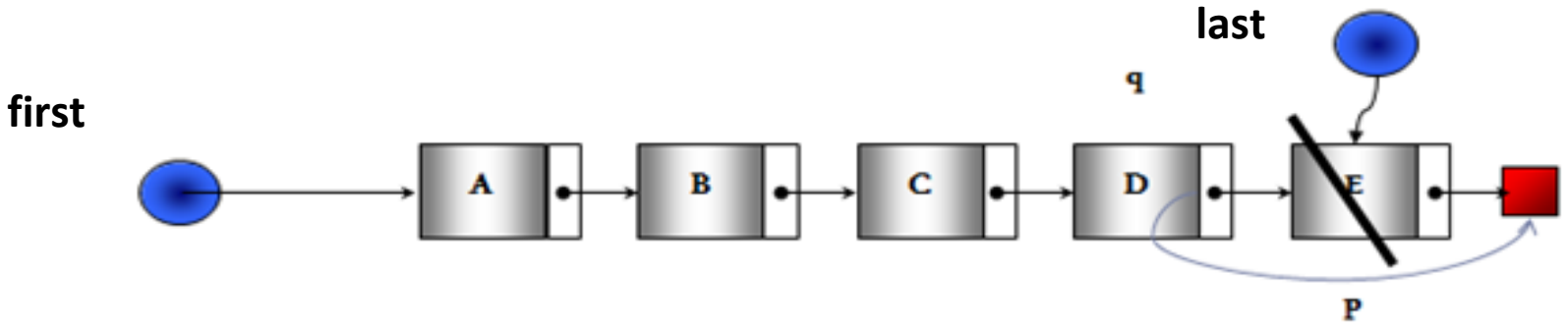
Xóa một phần tử đầu danh sách



Xóa một phần tử đầu danh sách

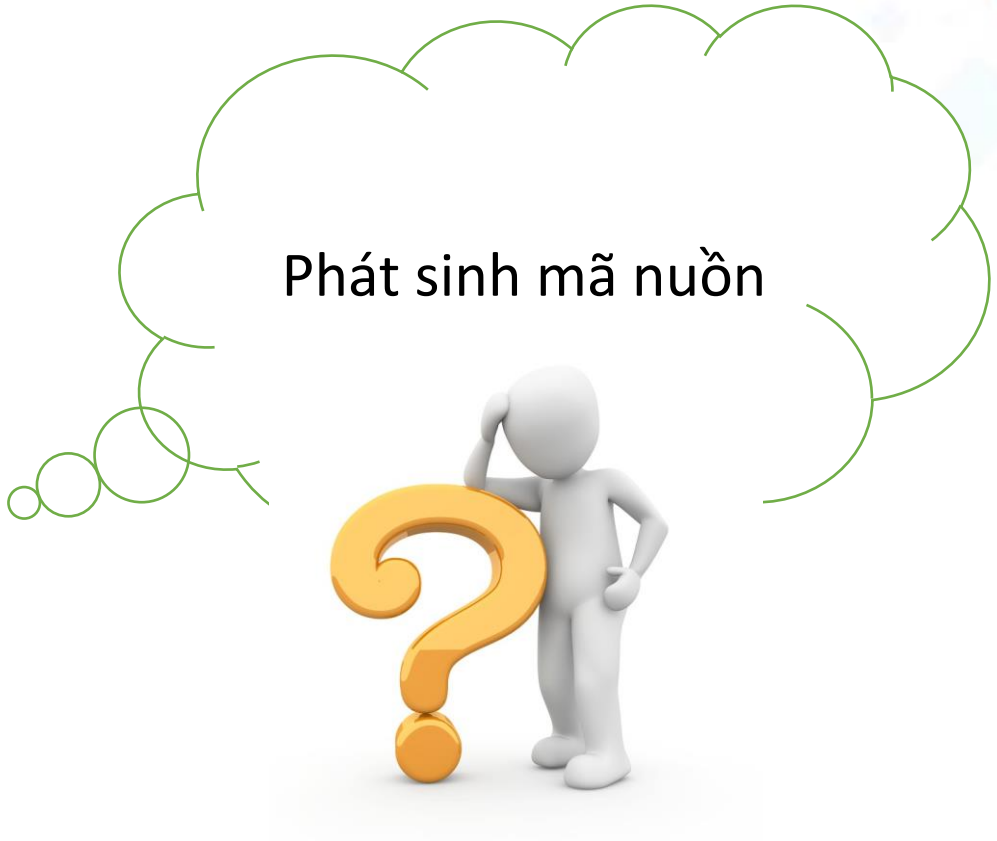
```
/*
* Xoa mot phan tu o dau danh sach
*/
public void RemoveFirst()
{
    Node p = _first;
    if (p != null)
    {
        _first = p.Next;
        if (_first == null)
            _last = null;
        _size--;
    }
}
```

Xóa một phần tử ở cuối danh sách

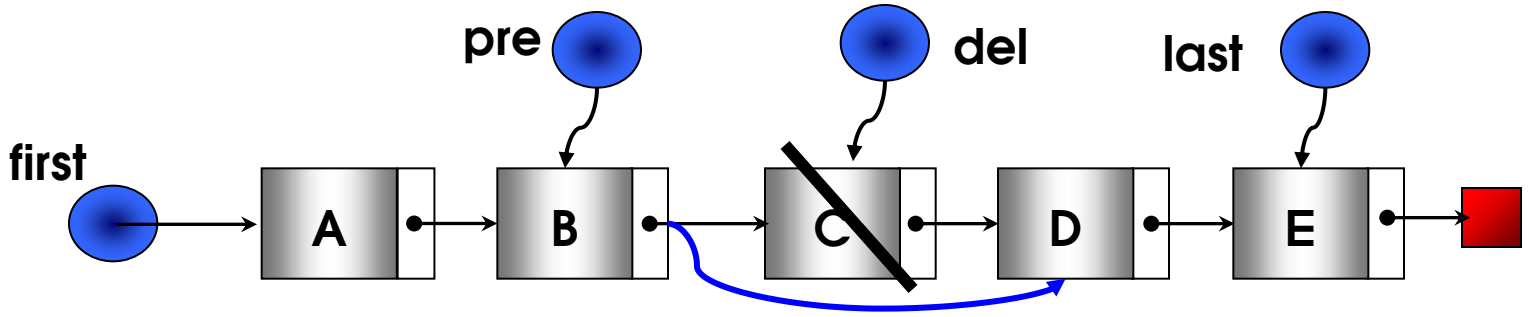


Xóa một phần tử ở cuối danh sách

```
public void RemoveLast()
{
    if (_first == _last)
    {
        _first = null;
        _last = null;
        _size = 0;
    }
    else
    {
        Node pre = _first;
        while (pre.Next != _last)
        {
            pre = pre.Next;
        }
        pre.Next = null;
        _last = pre;
        _size--;
    }
}
```

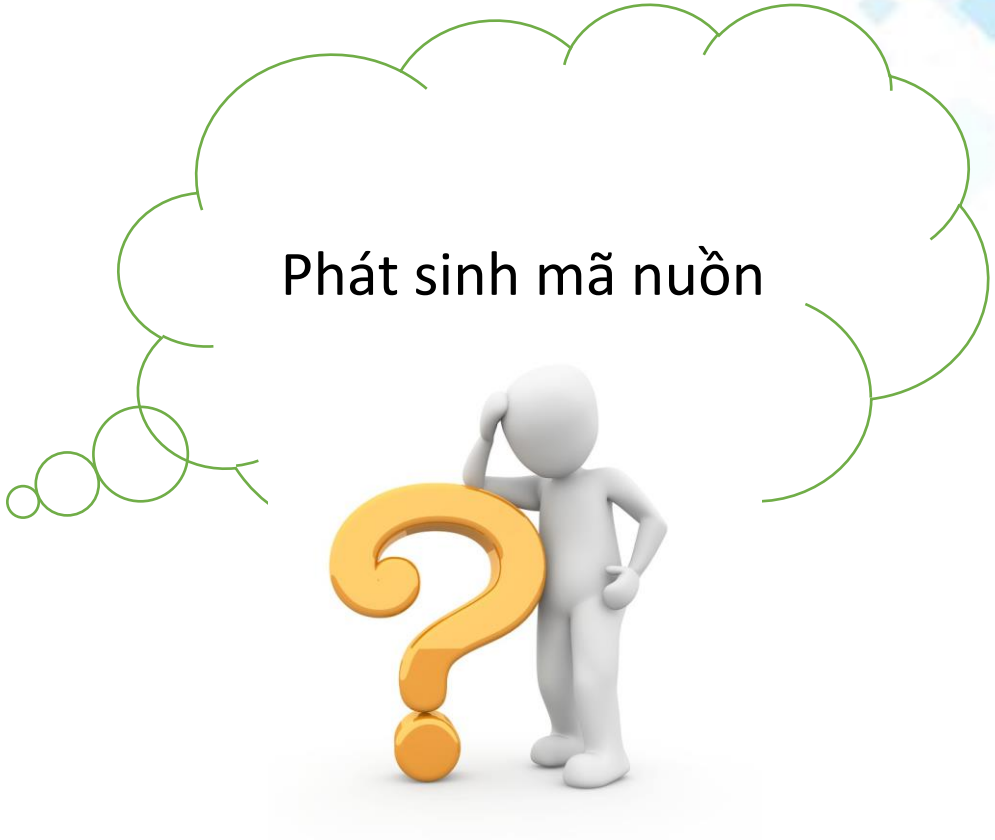


Xóa node sau node pre trong danh sách

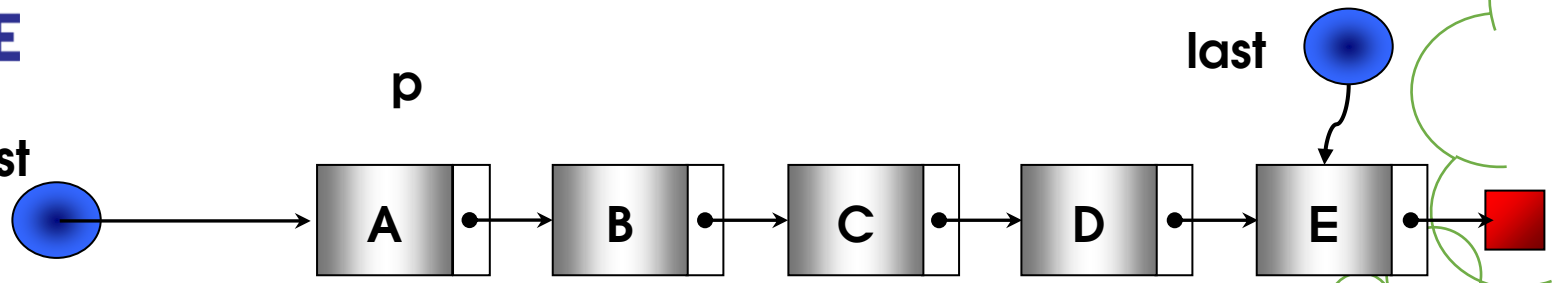


Xóa node sau node pre trong danh sách

```
/*
* Xoa mot phan tu sau mot phan tu khac
* Tham so pre: chi vao phan tu dung truoc
*/
public void RemoveAfter(Node pre)
{
    Node del;
    if (pre != null)
    {
        del = pre.Next;
        if (del != null)
        {
            pre.Next = del.Next;
            if (pre.Next == null)
                _last = pre;
            _size--;
        }
    }
}
```



Tìm kiếm tuần tự



Phát sinh mã nuồn

```
/*
* Tim kiem phan tu
* Tham so:
*          key: khoa can tim
* return: null neu khong tim thay, neu co tra ve node tim duoc
*/
public Node Find(int key)
{
    for (Node p = _first; p != _last; p = p.Next)
    {
        if (p.Data == key)
            return p;
    }
    return null;
}
```

HCMUTE

Sắp xếp danh sách

Cách tiếp cận:

- **Phương án 1:**
  Hoán vị nội dung các phần tử trong danh sách (thao tác trên vùng data).
- **Phương án 2 :**
  Thay đổi các mối liên kết (thao tác trên vùng link)

Phương án 1: Hoán vị nội dung các phần tử trong danh sách

- Điểm khác biệt duy nhất là cách thức truy xuất đến các phần tử trên danh sách liên kết thông qua liên kết thay vì chỉ số như trên mảng.
- Do thực hiện hoán vị nội dung của các phần tử nên đòi hỏi sử dụng thêm vùng nhớ trung gian $\Rightarrow$ chỉ thích hợp với các xâu có các phần tử có thành phần data kích thước nhỏ.
- Khi kích thước của trường data lớn, việc hoán vị giá trị của hai phân tử sẽ chiếm chi phí đáng kể.

Sắp xếp đổi chỗ trực tiếp (Interchange Sort)

```
public void InterchangeSort() // Sap xep danh sach tang dan
{
    for (Node p = _first; p != null; p = p.Next)
    {
        for (Node q = p.Next; q != null; q = q.Next)
        {
            if (q.Data < p.Data)
            {
                int temp = q.Data;
                q.Data = p.Data;
                p.Data = temp;
            }
        }
    }
}
```

Sắp xếp chọn trực tiếp (Selection sort)

```
public void SelectionSort() //Sap xep danh sach tang dan
{
    Node min;
    for (Node p = _first; p != null; p = p.Next){
        min = p;
        for (Node t = p.Next; t != null; t = t.Next){
            if (t.Data < min.Data){
                min = t;
            }
        }
        int temp = min.Data;
        min.Data = p.Data;
        p.Data = temp;
    }
}
```