CHƯƠNG 3. DANH SÁCH LIÊN KẾT

Danh sách liên kết đơn (Linked List)

ThS. Nguyễn Chí Hiếu

2021

NỘI DUNG

NỘI DUNG

Giới thiệu cấu trúc dữ liệu động

- Kiểu dữ liệu tĩnh
  - Là kiểu dữ liệu có kích thước *(số phần tử)* xác định *(không thay đổi trong vòng đời/chu kỳ sống)* như: số nguyên, số thực, ký tự, mảng, ...
  - Sử dụng phương pháp truy xuất trực tiếp *(direct access)* để truy xuất hay sửa một phần tử trong mảng.
  - Không có thao tác thêm và xóa một phần tử trên mảng.

Giới thiệu cấu trúc dữ liệu động

- Kiểu dữ liệu động
  - Là kiểu dữ liệu có kích thước thay đổi.
  - Sử dụng phương pháp truy xuất tuần tự (sequential access) để thực hiện các thao tác thêm, sửa, xóa một phần tử.
  - Trong ngôn ngữ lập trình C và C++, kiểu dữ liệu con trỏ thường được dùng để cấp phát động một kiểu dữ liệu, mảng, cấu trúc, đối tượng.

NỘI DUNG

Giới thiệu danh sách liên kết

- Là một dãy các nút (phần tử) được liên kết với nhau thông qua con trỏ liên kết.
- Các nút không cần lưu trữ liên tiếp nhau trong bộ nhớ.
- Kích thước của dãy có thể mở rộng!
- Thao tác thêm/xóa một nút không cần dịch chuyển các nút.

Giới thiệu danh sách liên kết

- Cấu trúc dữ liệu của một nút gồm
  - Thành phần dữ liệu.
  - Thành phần liên kết: con trỏ liên kết với nút kế tiếp (pNext) hoặc `NULL` nếu là nút cuối danh sách.

```
1 public class Node
2 {
3   int Info;
4   Node  pNext;
5 }
```

```
1 public class Node
2 {
3   Student Info;
4   Node  pNext;
5 }
```

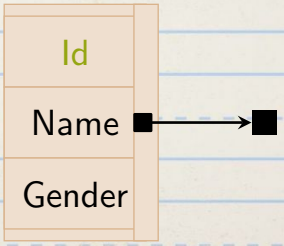
Giải thích

- Dòng 4: Data là một kiểu dữ liệu: int, double, $\cdots$, hay là kiểu dữ liệu cấu trúc *(struct)* tự định nghĩa.

Giới thiệu danh sách liên kết

- Trong thực tế, thành phần dữ liệu *(Data)* thường là kiểu cấu trúc *(struct)*

```
1  public class Student
2  {
3     public string Id;
4     public string Name;
5     public bool Gender;
6  }
7  public class Node
8  {
9     public Student  Info;
10    public Node pNext;
11 }
```

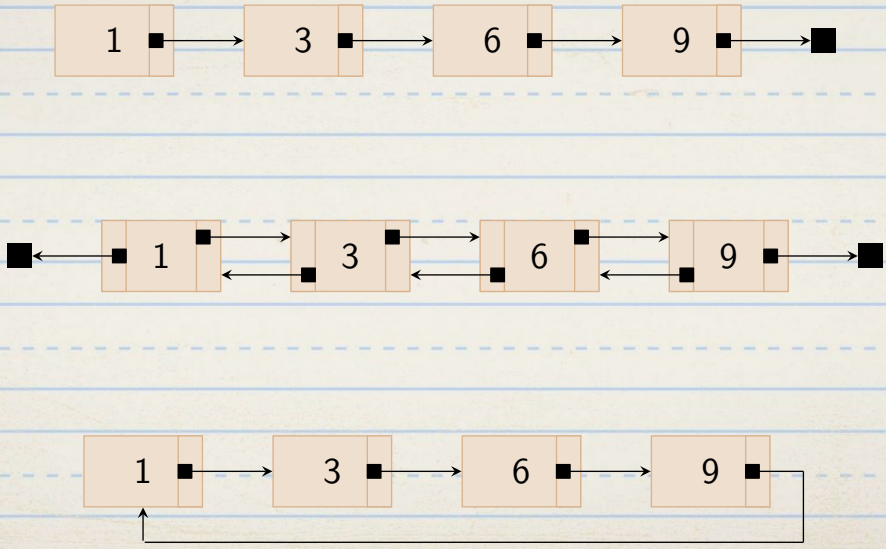


Giới thiệu danh sách liên kết

Các loại danh sách liên kết

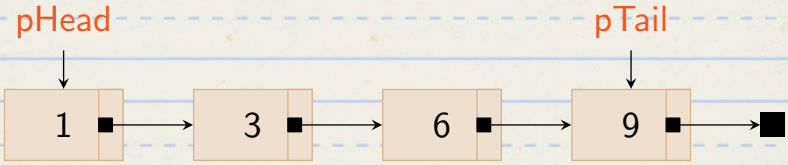
- Danh sách liên kết đơn *(Single - Linked list)*: mỗi nút chỉ có 1 con trỏ liên kết (pNext).
- Danh sách liên kết đôi *(Double - Linked list)*: mỗi nút có 2 con trỏ liên kết (pPrev, pNext).
- Danh sách liên kết vòng *(Circular - Linked list)*: liên kết ở nút cuối cùng của danh sách chỉ đến nút đầu tiên trong danh sách.

Giới thiệu danh sách liên kết



Giới thiệu danh sách liên kết

- Một danh sách được quản lý bởi con trỏ đầu *(pHead)* lưu trữ địa chỉ nút đầu tiên.
- Trong thực tế, có trường hợp cần truy xuất nút cuối danh sách nên có thể sử dụng thêm con trỏ cuối *(pTail)* để quản lý địa chỉ nút cuối.
- pHead và pTail không phải là một nút mà chỉ là con trỏ trỏ đến một nút.




```
1 public class List
2 {
3   public Node pHead;
4   public Node pTail;
5 }
```

NỘI DUNG

Các thao tác với danh sách liên kết đơn

Một danh sách liên kết thường có những thao tác sau:

- Thao tác khởi tạo
- Thao tác thêm phần tử
- Thao tác xóa phần tử
- Thao tác duyệt

Thao tác khởi tạo

Thao tác khởi tạo một nút

- Khai báo một biến con trỏ *trỏ đến kiểu dữ liệu* danh sách đã được định nghĩa.
- Gán *thành phần dữ liệu* và *thành phần liên kết* cho biến này.

Thao tác khởi tạo

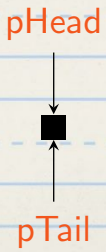
```
1 public static Node InitNode(Student std)
2 {
3    Node p = new Node();
4    p.Info = std;
5    p.pNext = null;
6    return p;
7 }
```

Thao tác khởi tạo

Thao tác khởi tạo danh sách

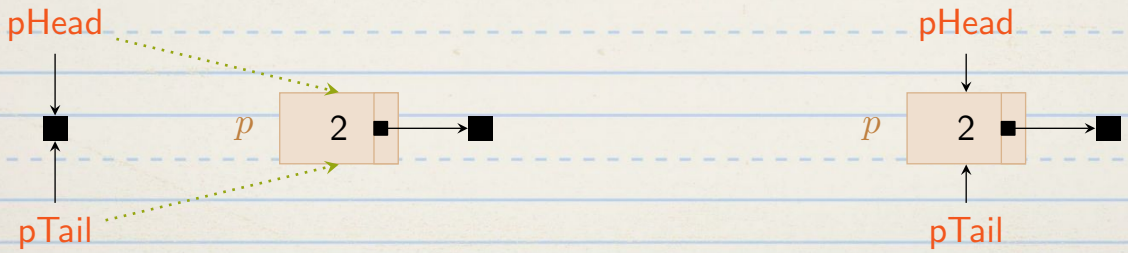
- Gán hai con trỏ pHead và pTail đến *NULL*.

```
1  public static void InitList()
2  {
3     pHead = null;
4     pTail = null;
5  }
```

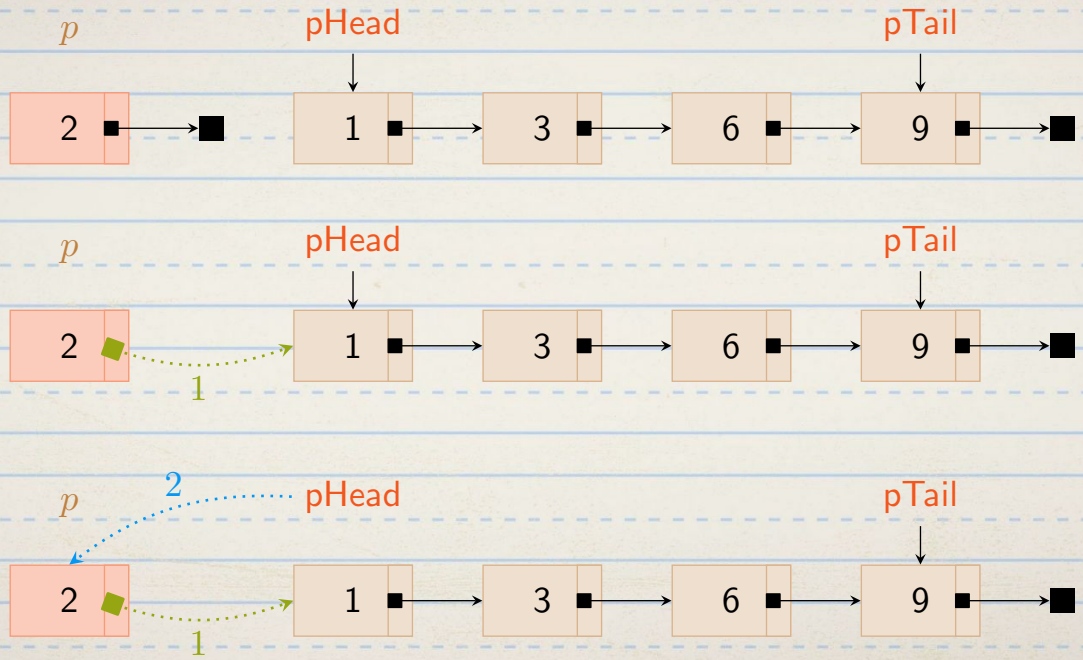


Thao tác thêm đầu

Trường hợp danh sách rỗng: con trỏ đầu và cuối sẽ trỏ đến nút p chứa giá trị x.



Thao tác thêm đầu



Thao tác thêm đầu

Thuật toán 1: `InsertHead(l, x)`

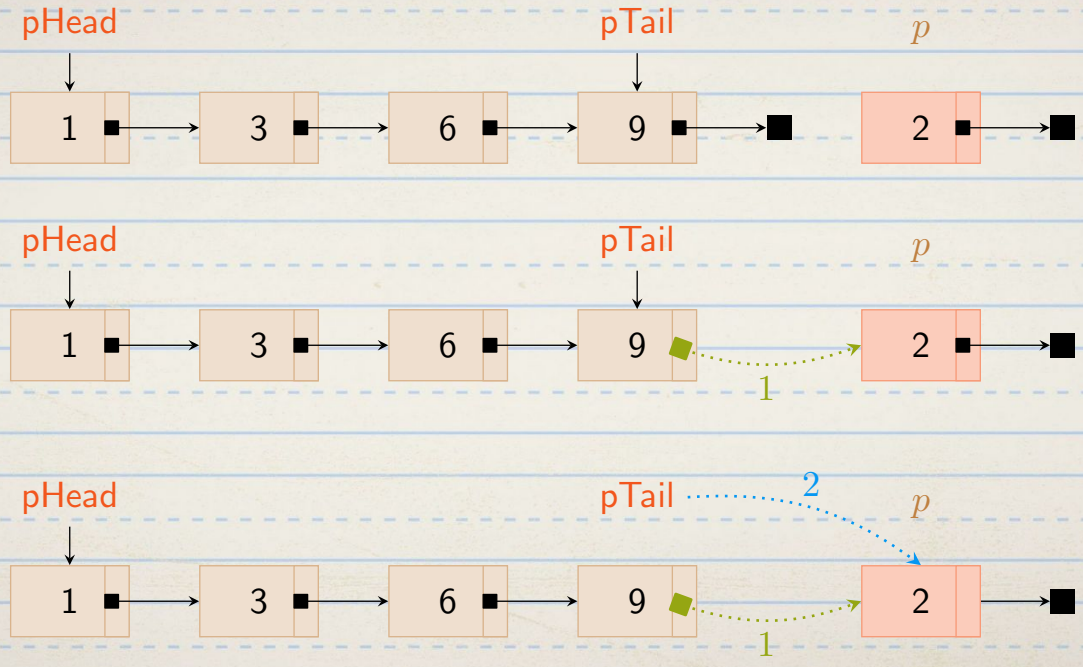
- Đầu vào: danh sách l và giá trị nút cần thêm.
- Đầu ra: danh sách l sau khi thêm đầu.

```
1 Khởi tạo nút p có giá trị x
2 if danh sách rỗng
3   pHead trỏ đến p
4   pTail trỏ đến p
5 else
6   p->pNext trỏ đến pHead
7   Cập nhật pHead
```

Thao tác thêm đầu

```
1  public void InsertHead(int x)
2  {
3    Node p = Node.InitNode(x);
4
5    if (pHead == null)
6    {
7      pHead = p;
8      pTail = p;
9    }
10   else
11   {
12     p.pNext = pHead;
13     pHead = p;
14   }
15 }
```


Thao tác thêm cuối



Thao tác thêm cuối

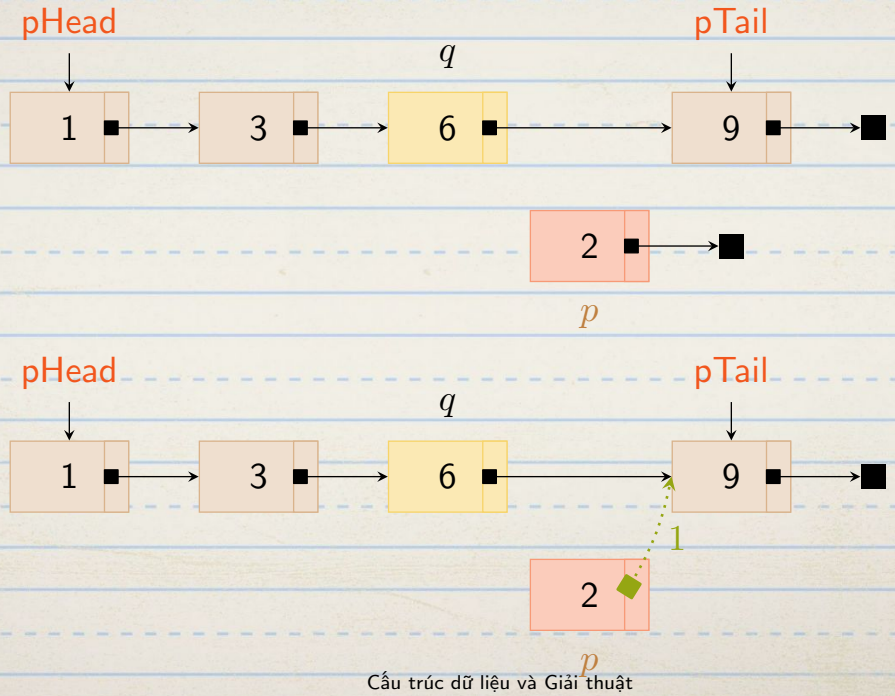
Thuật toán 2: InsertTail(l, x)
- Đầu vào: danh sách l và giá trị nút cần thêm.
- Đầu ra: danh sách l sau khi thêm cuối.

```
1  Khởi tạo nút p có giá trị x
2  if danh sách rỗng
3    pHead trỏ đến p
4    pTail trỏ đến p
5  else
6    Thêm p vào pTail->pNext
7    Cập nhật pTail
```

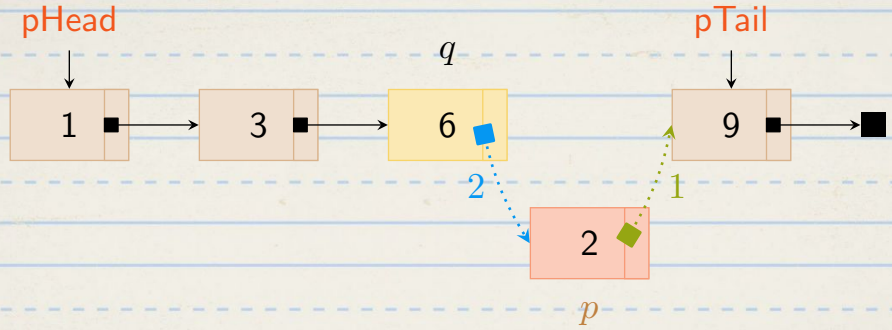

Thao tác thêm cuối

```
1  public void InsertTail(int x)
2  {
3    Node p = Node.InitNode(x);
4
5    if (pHead == null)
6    {
7      pHead = p;
8      pTail = p;
9    }
10   else
11   {
12     pTail.pNext = p;
13     pTail = p;
14   }
15 }
```

Thao tác thêm sau một nút khác



Thao tác thêm sau một nút khác



✍ *Chú ý trường hợp: nút q là nút cuối của danh sách.*

Thao tác thêm sau một nút khác

Thuật toán 3: InsertAfter(l, q, x)

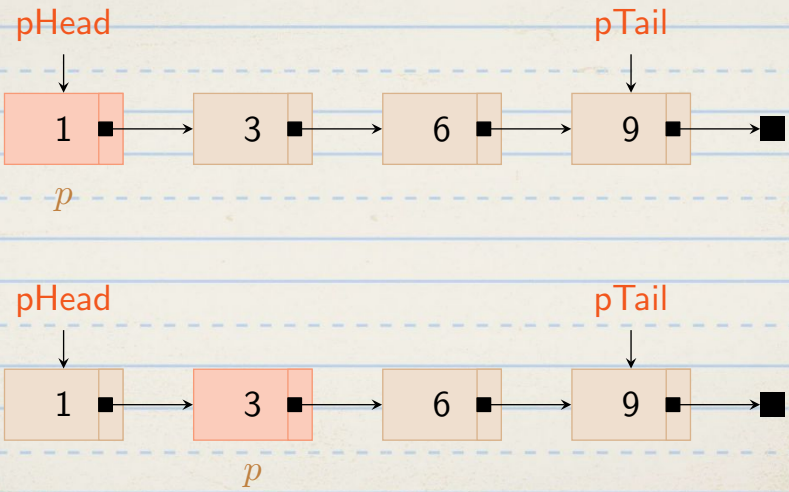
- Đầu vào: danh sách l, nút q và giá trị nút cần thêm.
- Đầu ra: danh sách l sau khi thêm một nút sau nút q.

```
1  Khởi tạo nút p có giá trị x
2  if q ≠ null
3    p->pNext trỏ đến q->pNext
4    q->pnext trỏ đến p
5    if q là nút cuối
6      Cập nhật pTail
```

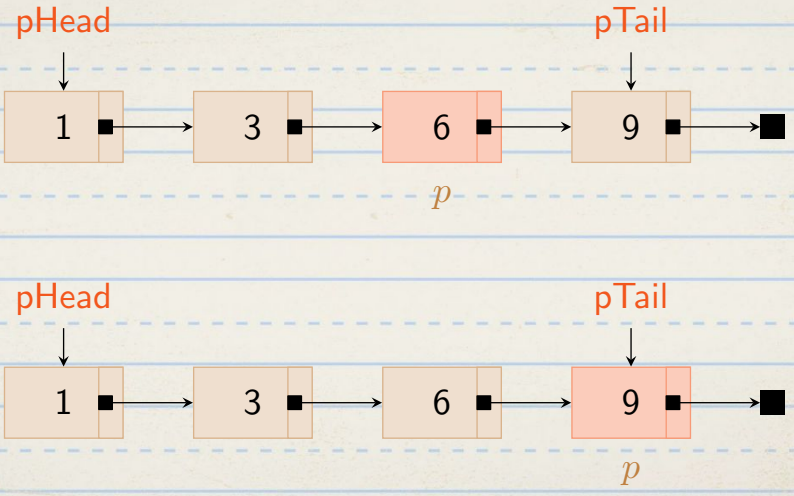

Thao tác thêm sau một nút khác

```
1  public void InsertAfter(Node q, int x)
2  {
3    Node p = Node.InitNode(x);
4
5    if (q != null)
6    {
7      p.pNext = q.pNext;
8      q.pNext = p;
9      if (q == pTail)
10     {
11       pTail = p;
12     }
13   }
14 }
```

Thao tác duyệt



Thao tác duyệt



Thao tác duyệt

Thuật toán 4: Traverse(l)
- Đầu vào: danh sách l.
- Đầu ra:

```
1 Khai báo nút p trỏ đến pHead
2 while chưa duyệt hết danh sách
3   // ...
4   p trỏ đến p->pNext
```


Thao tác duyệt

```
1  public void Traverse()
2  {
3    Node p = pHead;
4    while (p != null)
5    {
6      // In, tim gia tri
7      // ...
8
9      // Chuyen den nut ke tiep
10     p = p.pNext;
11   }
12 }
```

Giải thích

- Dòng 7: mã nguồn tương ứng với thao tác: tìm, in, ... các phần tử trong một danh sách.

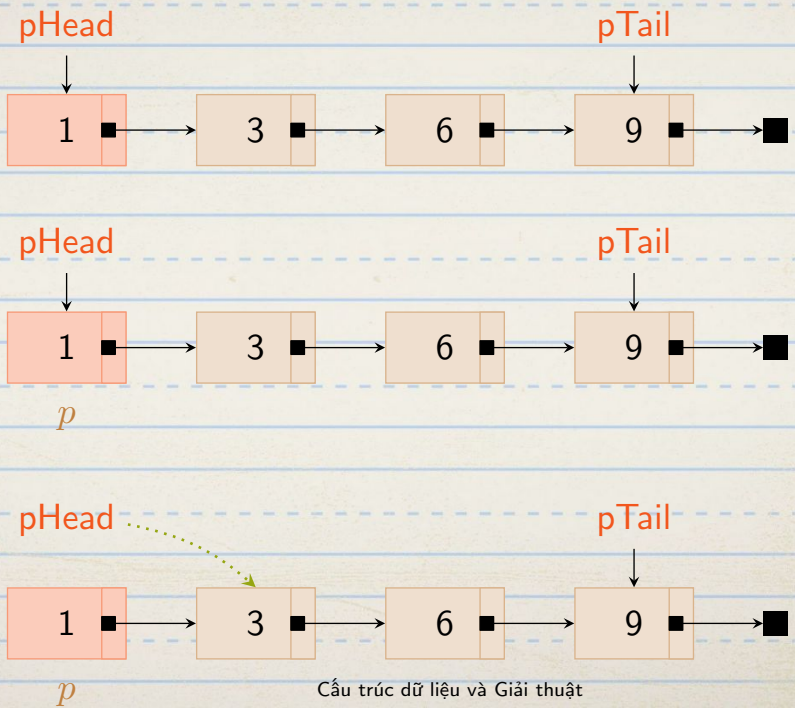
Thao tác tìm kiếm

```
1  public Node Search(int x)
2  {
3    Node p = pHead;
4
5    while (p != null && p.Info != x)
6    {
7      p = p.pNext;
8    }
9    return p;
10 }
```

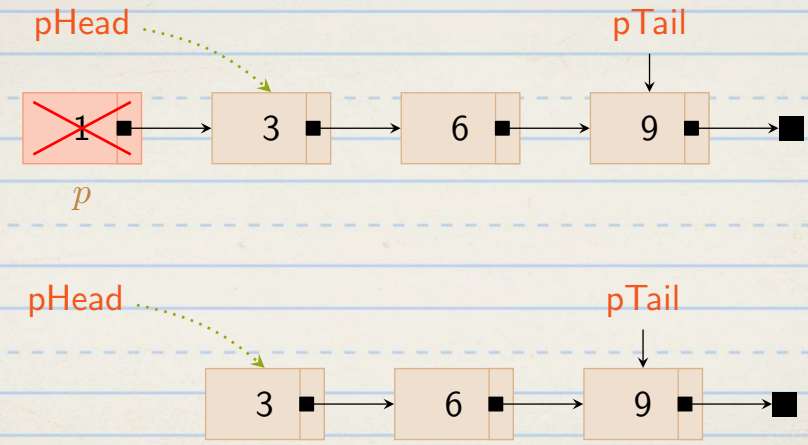
Giải thích

- Dòng 5: tương tự thao tác duyệt danh sách. Mỗi lần duyệt kiểm tra giá trị x với thành phần dữ liệu của nút đang xét.

Thao tác xóa đầu



Thao tác xóa đầu



✍ *Chú ý hai trường hợp: danh sách rỗng và danh sách chỉ chứa một nút.*

Thao tác xóa đầu

Thuật toán 5: `RemoveHead(l)`
- Đầu vào: danh sách `l`.
- Đầu ra: danh sách `l` sau khi xóa nút đầu.

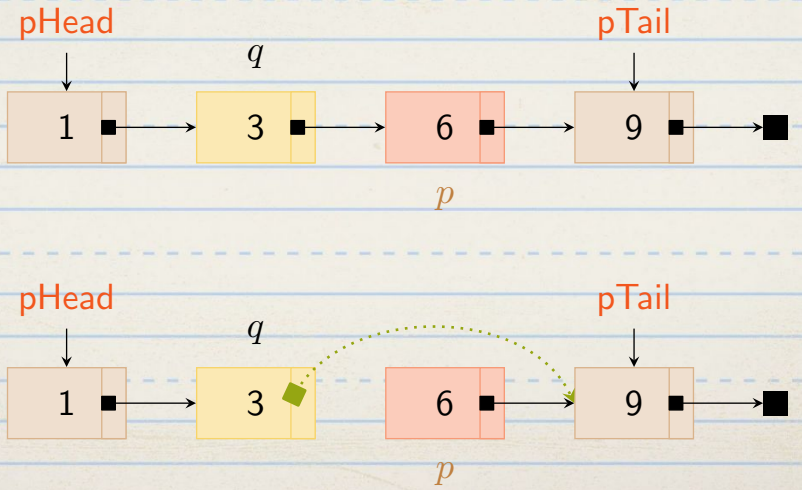
```
1 if danh sách khác rỗng
2    p trỏ đến pHead
3    pHead trỏ đến pHead->pNext
4    Xóa nút p
5    if danh sách rỗng
6       Cập nhật pTail
```

Thao tác xóa đầu

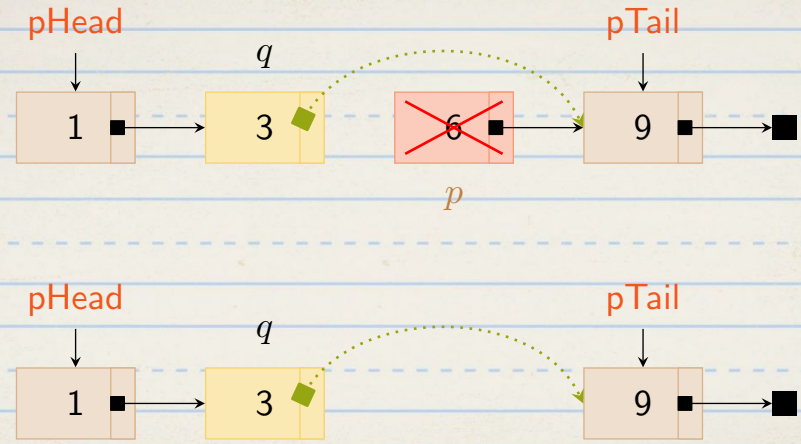
```
1  public void RemoveHead()
2  {
3    Node p = new Node();
4    if (pHead != null) // TH. Danh sach khac rong
5    {
6      p = pHead;
7      pHead = pHead.pNext;
8      p = null;
9      if (pHead == null)
10     {
11       pTail = null;
12     }
13   }
14 }
```


Thao tác xóa sau một nút khác

✍ Giả sử nút xóa nút p sau nút q.



Thao tác xóa sau một nút khác



✍ *Chú ý trường hợp: nút q là nút kế cuối (nút p là nút cuối) của danh sách (cần cập nhật lại con trỏ pTail).*

Thao tác xóa sau một nút khác

Thuật toán 6: `RemoveAfter(l, q)`

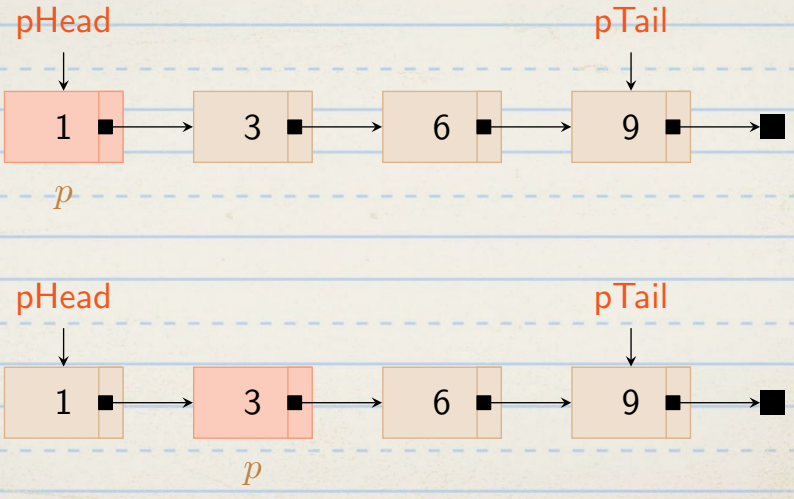
- Đầu vào: danh sách l và nút q.
- Đầu ra: danh sách l sau khi xóa nút.

```
1  if q ≠ null
2    p trỏ đến q->pNext
3    if p ≠ null
4      // Nếu p là nút cuối danh sách
5      if p là phần tử cuối của danh sách
6        Cập nhật pTail trước khi xóa p
7      // Ngược lại, p không là nút cuối danh sách
8      q->pNext trỏ đến p->pNext
9      Xóa nút p
```

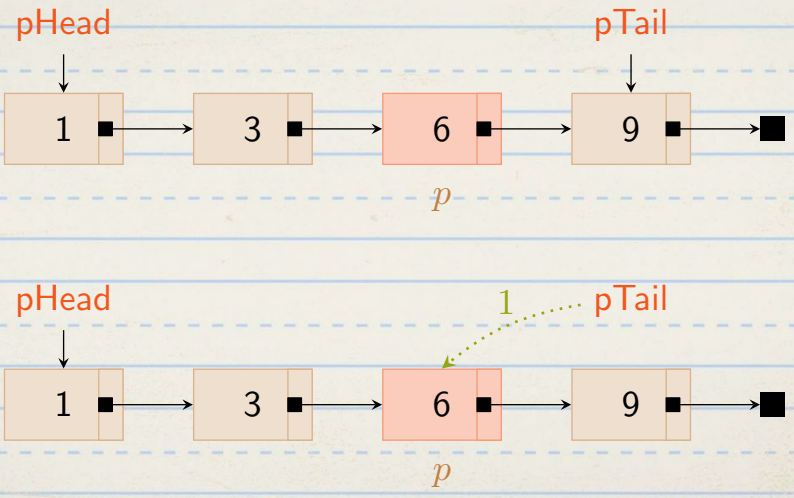
Thao tác xóa sau một nút khác

```
1  public void RemoveAfter(Node q)
2  {
3    Node p = new Node();
4    if (q != null)
5    {
6      p = q.pNext;
7      if (p != null)
8      {
9        if (p == pTail)
10       {
11         pTail = q;
12       }
13       q.pNext = p.pNext;
14       p = null;
15     }
16   }
17 }
```

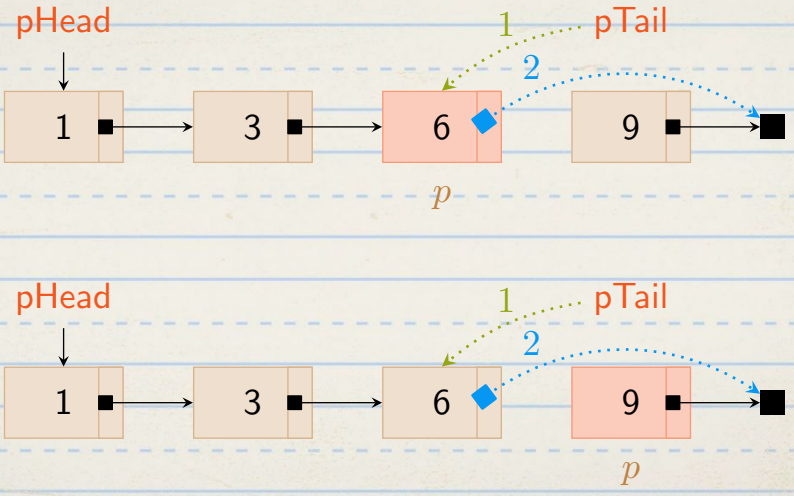

Thao tác xóa cuối



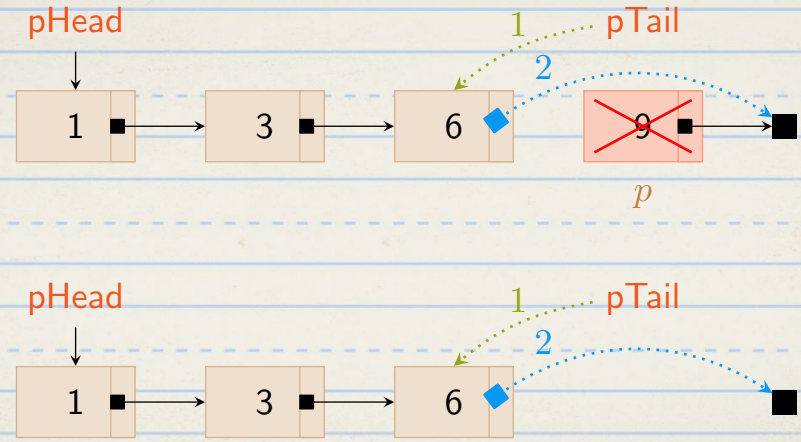
Thao tác xóa cuối



Thao tác xóa cuối



Thao tác xóa cuối



✍ *Chú ý hai trường hợp: danh sách rỗng và danh sách chỉ chứa một nút.*

Thao tác xóa cuối

Thuật toán 7: RemoveTail(l)
- Đầu vào: danh sách l.
- Đầu ra: danh sách l sau khi xóa cuối.

```
1  if danh sách khác rỗng
2    p trỏ đến pHead
3    if danh sách chỉ chứa 1 nút p
4      Xóa nút p ···
5      Kết thúc
6    while p chưa trỏ đến vị trí kế cuối của danh sách
7      p trỏ đến p->pNext
8    Cập nhật pTail
9    Xóa nút p ···
```

Thao tác xóa cuối

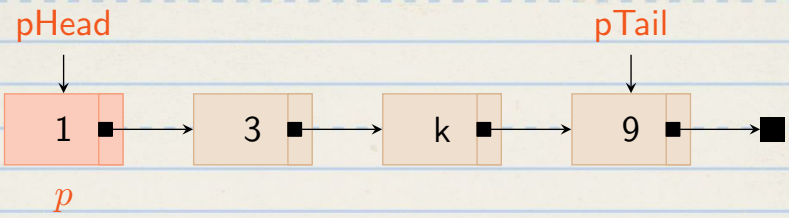
```
1  public void RemoveTail()
2  {
3  ··Node p = new Node();
4  ··if (pHead != null) // Danh sach khac rong
5  ··{
6  ····p = pHead;
7  ····if (p == pTail) // TH1: 1 nut
8  ····{
9  ····· p = null;
10 ····· pHead = null;
11 ····· pTail = null;
12 ····· return;
13 ····}
```

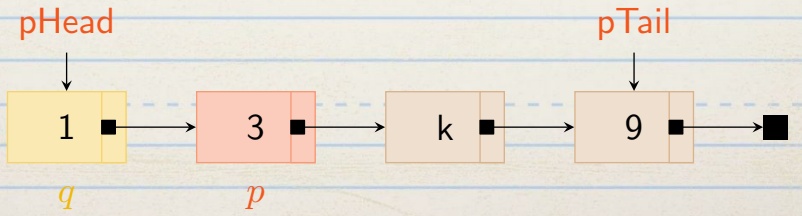

Thao tác xóa cuối

```
14 ····// TH2: > 1 nut
15 ····while (p.pNext != pTail)
16 ····{
17 ····· p = p.pNext;
18 ····}
19 ····pTail = p;
20 ····pTail.pNext = null;
21 ····p = p.pNext;
22 ····p = null;
23 ··}
24 }
```

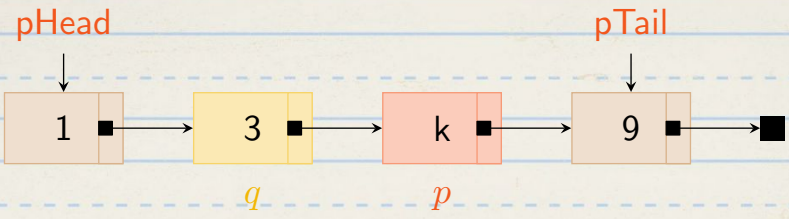
Thao tác xóa nút có khóa k

✍ Giả sử nút p là nút chứa khóa k.





Thao tác xóa nút có khóa k



✍ Chú ý trường hợp

- Nút có khóa k là nút đầu danh sách.
- Nút có khóa k là nút giữa danh sách.
- Nút có khóa k là nút cuối danh sách.

Thao tác xóa nút có khóa k

Thuật toán 8: RemoveNode(l, k)

- Đầu vào: danh sách l và giá trị k của nút cần xóa.
- Đầu ra: true hay false.

```
Lặp tìm nút p có giá trị k và nút q là nút trước của p ...
if p = null // TH1. Không tìm thấy p
  return false
if q = null // TH2. Tìm thấy p và p là nút đầu danh sách
  Thực hiện thao tác xóa đầu···
else q ≠ null // Tìm thấy p và q (q, p)
  if p là nút cuối danh sách // TH3. p là nút cuối danh sách···
    Cập nhật pTail
  // TH4. p là nút giữa danh sách···
  q->pNext trỏ đến p->pNext
  Xóa nút p
return true
```


Thao tác xóa nút có khóa k

```
public bool RemoveNode(int k)
{
    Node p = pHead;
    Node q = null;
    while (p != null)
    {
        if (p.Info == k)
        {
            break;
        }
        q = p;
        p = p.pNext;
    }
    if (p == null) // TH1. Khong tim thay p
    {
        return false;
    }
```

Thao tác xóa nút có khóa k

```
14  __if (q != null) // TH2: Tim thay p va p la nut dau danh sach
15  __{
16  ____pHead = p.pNext;
17  ____if (pHead == null)
18  ____{
19  ______pTail = null;
20  ____}
21  __}
```


Thao tác xóa nút có khóa k

```
20  __else // Tim thay p va q (q, p)
21  __{
22  ____if (p == pTail) // TH3. p la phan tu cuoi danh sach
23  ____{
24  ______pTail = q;
25  ____}
26  ____// TH4. p la phan tu giua danh sach
27  ____q.pNext = p.pNext;
28  ____p = null;
29  __}
30
31  __return true;
32  }
```

Bài tập

1. Xây dựng cấu trúc dữ liệu thích hợp để biểu diễn đa thức $P(x)$ có dạng như sau:

$$P(x) = c_1 x^{e_1} + c_2 x^{e_2} + \cdots + c_n x^{e_n}$$

với c_i là hệ số và e_i là số mũ, $1 \leq i \leq n$
Các thao tác:
   - Thêm đơn thức vào cuối đa thức.
   - In đa thức.
   - Tính giá trị đa thức với x cho trước.
2. Xây dựng cấu trúc dữ liệu thích hợp để quản lý danh sách sinh viên.
   - Dữ liệu mỗi sinh viên gồm các thông tin: MSSV, họ tên, giới tính, ngày sinh.
   - Các thao tác thực hiện với danh sách sinh viên gồm: thêm, xóa, tìm kiếm một sinh viên.

Tài liệu tham khảo


Donald E. Knuth.
The Art of Computer Programming, Volume 3.
Addison-Wesley, 1998.


Dương Anh Đức, Trần Hạnh Nhi.
Nhập môn Cấu trúc dữ liệu và Thuật toán.
Đại học Khoa học tự nhiên TP Hồ Chí Minh, 2003.


Niklaus Wirth.
Algorithms + Data Structures = Programs.
Prentice-Hall, 1976.


Robert Sedgewick.
Algorithms in C.
Addison-Wesley, 1990.