

Bài 6: KDLTТ danh sách cài đặt bằng danh sách liên kết

Giảng viên: Hoàng Thị Điệp

Khoa Công nghệ Thông tin – Đại học Công Nghệ

Nội dung chính

- Thư viện khuôn mẫu chuẩn STL
- KDLTT danh sách
- Khái niệm DSLK
- Các phép toán trên DSLK



Thư viện khuôn mẫu chuẩn STL

- `<array>`
- `<vector>`
- `<deque>`
- `<forward_list>`
- `<list>`
- `<stack>`
- `<queue>`
- `<priority_queue>`
- `<set>`
- `<multiset>`
- `<map>`
- `<multimap>`
- `<unordered_set>`
- `<unordered_multiset>`
- `<unordered_map>`
- `<unordered_multimap>`
- `<bitset>`
- `<valarray>`

Ví dụ thư viện <list>:: push_front()

```
// list::push_front
#include <iostream>
#include <conio.h>
#include <list>
using namespace std;
int main(){
    list<int> mylist(2,100); // 2 bien nguyen gia tri 100
    mylist.push_front(200);
    mylist.push_front(300);
    cout << "mylist chua day so:";
    list<int>::iterator it;
    for (it=mylist.begin(); it!=mylist.end(); ++it)
        cout << ' ' << *it;
    cout << '\n';
    getch();
    return 0;
}
```

Ví dụ thư viện <list>:: pop_front()

```
// list::pop_front
#include <iostream>
#include <conio.h>
#include <list>
using namespace std;
int main(){
    list<int> mylist;
    mylist.push_back(100);
    mylist.push_back(200);
    mylist.push_back(300);
    cout << "Thực hiện pop_front các phần tử trong mylist:";
    while(!mylist.empty()){
        cout << ' ' << mylist.front();
        mylist.pop_front();
    }
    cout << "\nKích thước cuối cùng của mylist: "
        << mylist.size() << '\n';
    getch();
    return 0;
}
```

Tra cứu tương tự

- Trên <http://www.cplusplus.com/reference/list/list/>
- Các phép toán
 - push_back
 - pop_back
 - insert
 - erase

3 cách để liên kết dữ liệu

- Mảng: tập hợp các phần tử cùng kiểu
- struct/class: tập hợp các thành phần có kiểu (có thể) khác nhau
- Con trỏ

Các KDLTT đã học

- KDLTT danh sách

- Phép toán

- insert
 - erase
 - append
 - at
 - length
 - empty

- Cài đặt

- mảng tĩnh
 - mảng động

- KDLTT tập động

- Phép toán

- insert
 - erase
 - search
 - max
 - min
 - length
 - empty

- Cài đặt

- mảng động không được sắp
 - mảng động được sắp

KDLTT danh sách

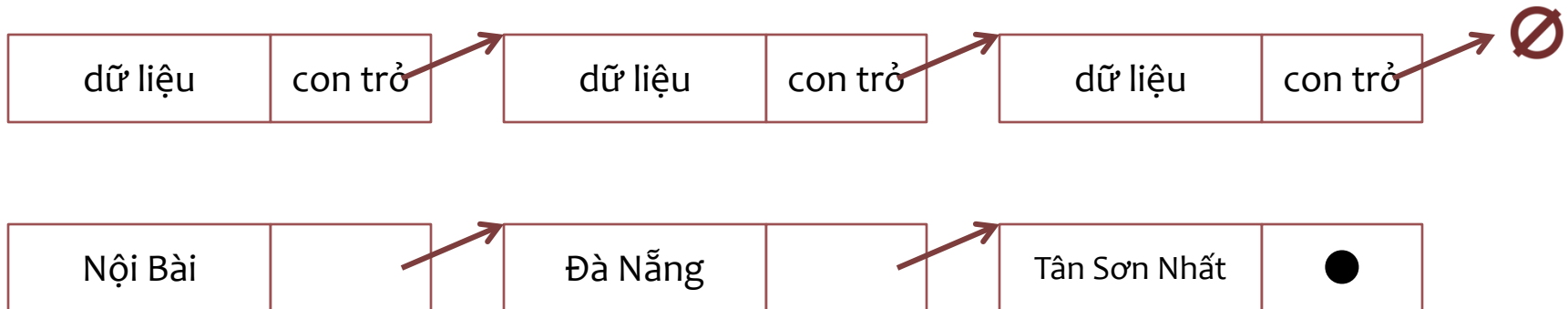
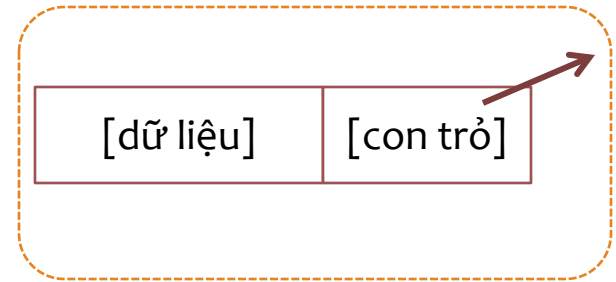
- Cài bằng mảng
 - truy cập & cập nhật, at: $O(1)$
 - xen, insert: $O(N)$
 - xóa, erase: $O(N)$
- Cài bằng danh sách liên kết
 - insert và erase hiệu quả hơn

So sánh

	Mảng	Danh sách liên kết
Giống	Các phần tử có cùng kiểu và có thứ tự	
Khác	Bố cục logic giống với bố cục vật lý trong bộ nhớ máy tính	Bố cục logic không cần phải giống với bố cục vật lý

Khái niệm

- DSLK được tạo thành từ các nút
 - mỗi nút gồm 2 phần
 - phần dữ liệu: chứa phần tử dữ liệu
 - phần con trỏ: chứa 1 địa chỉ
 - các nút liên kết với nhau thông qua con trỏ

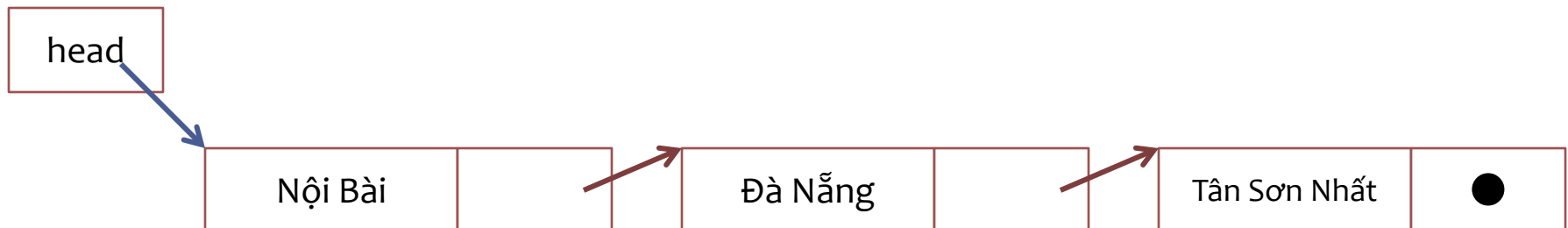


DSLK bằng C++

- Mỗi nút là một biến Node
- Nút cuối cùng có giá trị next bằng NULL
- Xác định DSLK bằng địa chỉ của nút đầu tiên trong danh sách
 - gọi biến lưu địa chỉ này là con trỏ đầu **head**
 - khởi tạo danh sách rỗng:

```
struct Node{  
    Item data;  
    Node * next;  
};
```

```
Node * head = NULL;
```

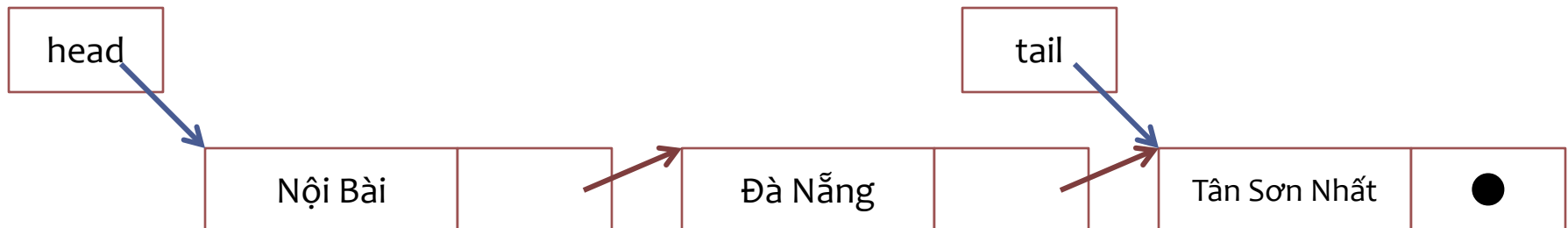


DSLK bằng C++

- Có thể sử dụng thêm con trỏ đuôi **tail** để các thao tác trên DSLK được thuận lợi
 - danh sách rỗng

```
struct Node{  
    Item data;  
    Node * next;  
};  
Node * head;  
Node * tail;
```

```
head = tail = NULL;
```



```
typedef int Item;

struct Node{
    Item data;
    Node * next;
};

struct SList{
    Node * head;
    Node * tail;
    long size;

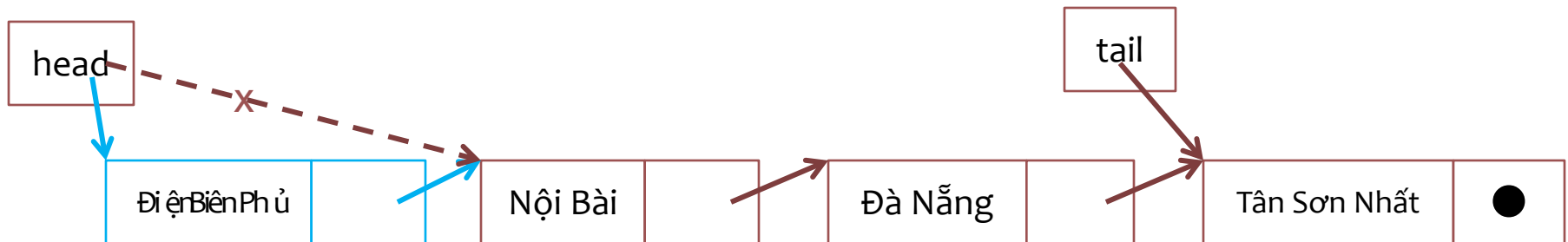
    SList();
    ~SList();

    Node* findPrevious(Node* p);
    Node* addFirst(const Item& v);
    Node* addLast(const Item& v);
    Node* insertAfter(Node* p, const Item& v);
    Node* insertBefore(Node* p, const Item& v);
    void removeFirst();
    void remove(Node*& p);
    void print();
};
```

Các phép toán trên DSLK

- Thêm dữ liệu vào danh sách
 - Thêm vào đầu danh sách: `addFirst(v)`
 - Thêm vào sau nút p: `insertAfter(p, v)`
 - Thêm vào trước nút p: `insertBefore(p, v)`
 - Thêm vào cuối danh sách: `addLast(v)`
- Xóa dữ liệu khỏi danh sách
 - Xóa nút đầu danh sách: `removeFirst()`
 - Xóa nút cuối danh sách: `removeLast()` ?
 - Xóa nút không phải đầu danh sách: `remove(p)`
- Duyệt danh sách: `traverse()`

Thêm vào đầu danh sách



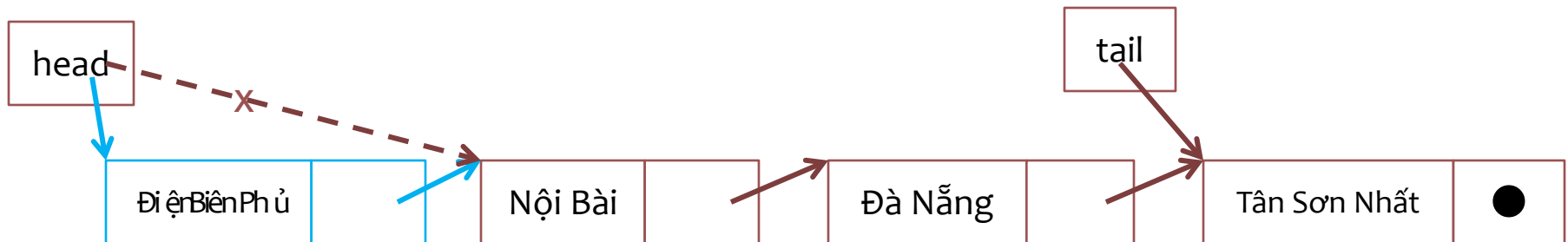
addFirst(v)

Algorithm *addFirst(v)*

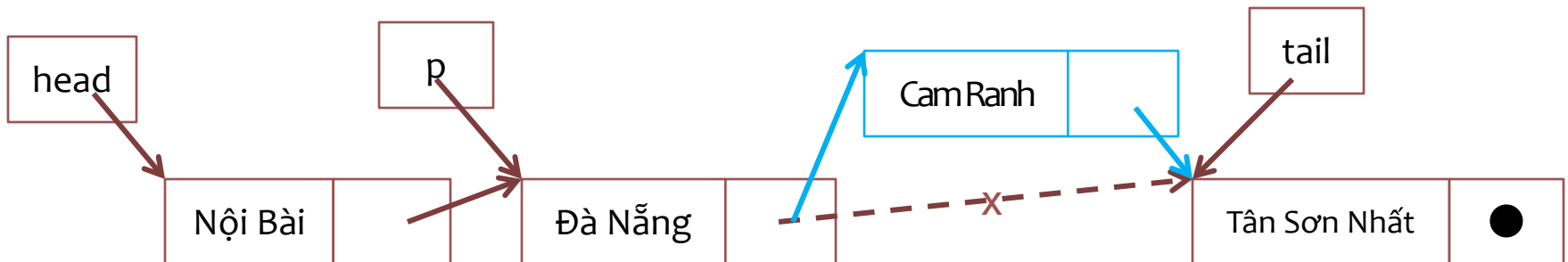
Input dữ liệu v cần thêm vào đầu danh sách

Output

$q \leftarrow \text{new Node}()$ {tạo nút mới}
 $(*q).data \leftarrow v$ {nút mới chứa dữ liệu v }
 $(*q).next \leftarrow head$ {nút mới chứa con trỏ đến nút head cũ}
 $head \leftarrow q$ {trỏ head đến nút mới}
 $size \leftarrow size + 1$ {tăng biến đếm nút}
cập nhật tail



Thêm vào sau nút p



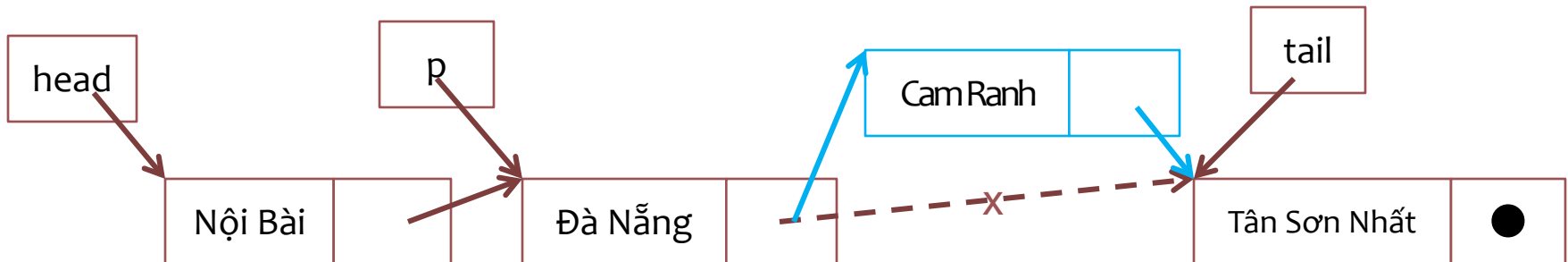
insertAfter(p, v)

Algorithm *insertAfter(p, v)*

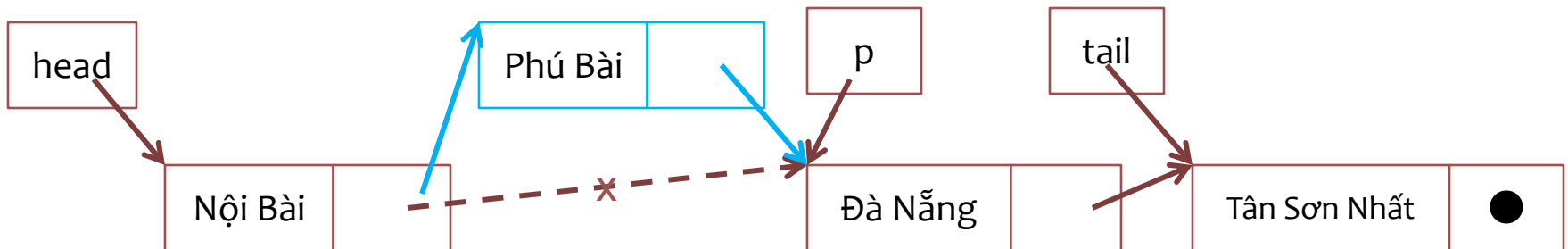
Input dữ liệu v cần thêm vào sau nút p trong danh sách

Output

$q \leftarrow \text{new Node}()$ {tạo nút mới}
 $(*q).data \leftarrow v$ {nút mới chứa dữ liệu v }
 $(*q).next \leftarrow (*p).next$ {nút mới chứa con trỏ đến nút sau p }
 $(*p).next \leftarrow q$ {nút p chứa con trỏ đến nút mới}
 $size \leftarrow size + 1$ {tăng biến đếm nút}
cập nhật *tail*



Thêm vào trước nút p



insertBefore(p, v)

Algorithm *insertBefore(p, v)*

Input dữ liệu v cần thêm vào trước nút p trong danh sách

Output

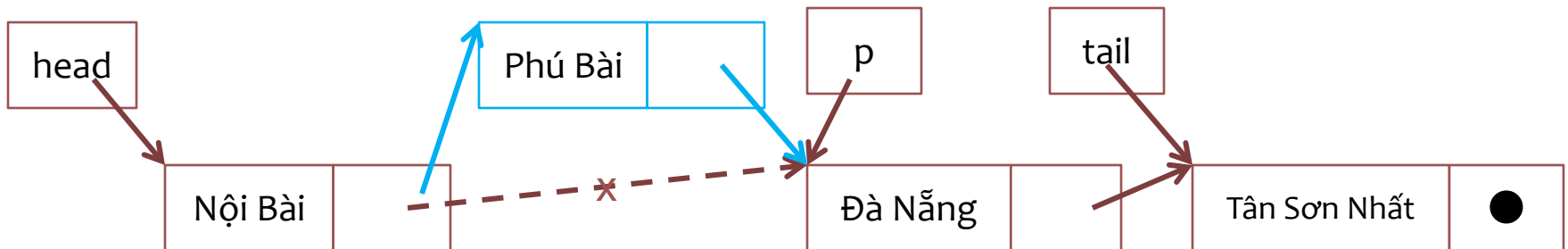
if $p = \text{head}$ then

$\text{addFirst}(v)$

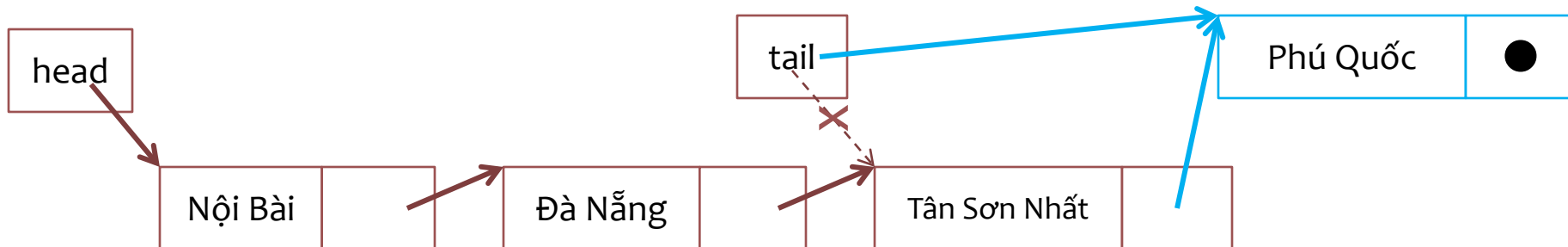
else

 tìm nút pre liền trước p

$\text{insertAfter}(\text{pre}, v)$



Thêm vào cuối danh sách



addLast(v)

Algorithm *addLast(v)*

Input dữ liệu v cần thêm vào cuối trong danh sách

Output

$q \leftarrow \text{new Node}()$

{tạo nút mới}

$(*q).data \leftarrow v$

{nút mới chứa dữ liệu v }

$(*q).next \leftarrow \text{NULL}$

{nút mới chứa con trỏ NULL}

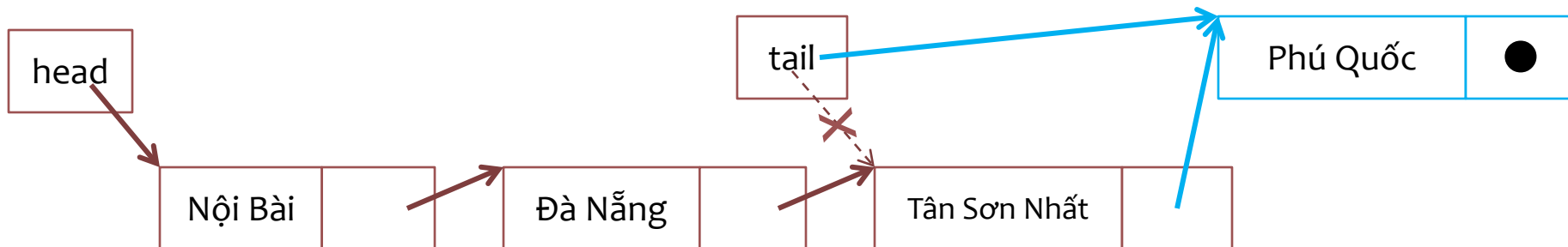
$(*tail).next \leftarrow q$

{nút tail cũ chứa con trỏ đến nút mới}

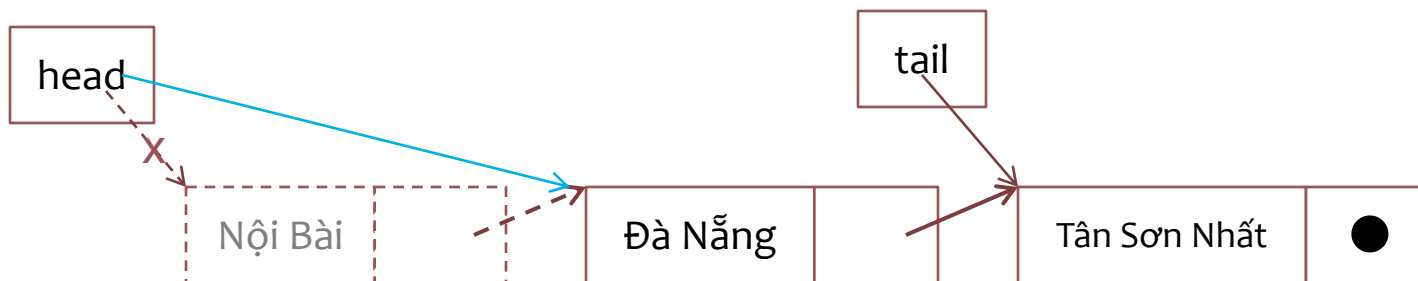
$tail \leftarrow q$

{tail trỏ đến nút mới}

$size \leftarrow size + 1$ {tăng biến đếm nút}



Xóa nút đầu danh sách



removeFirst()

Algorithm *removeFirst()*

Input

Output

if *head* = null then

báo lỗi: danh sách rỗng

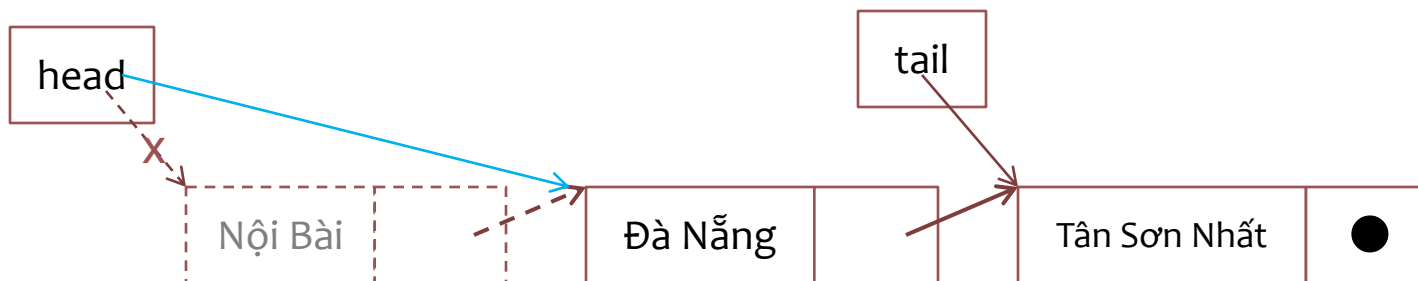
t ← *head*

head ← (**head*).next {trở *head* đến nút sau nó}

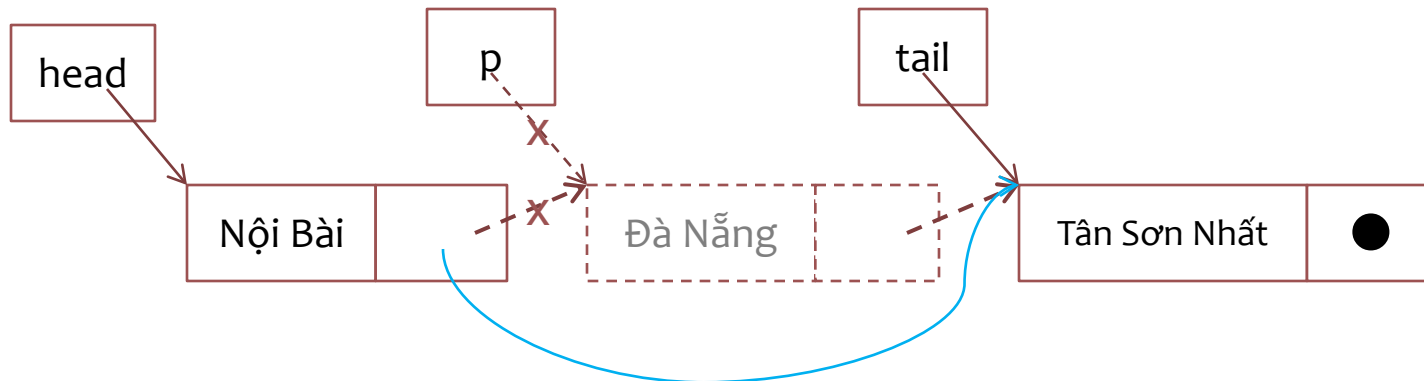
delete *t* {giải phóng vùng nhớ trở bởi *head* cũ}

size ← *size* - 1 {giảm biến đếm nút}

cập nhật *tail*



Xóa nút không phải đầu danh sách



remove(p)

Algorithm **remove(p)**

Input nút *p* cần xóa khỏi danh sách

Output

if *p=head* then

removeFirst()

else

tìm nút pre liền trước p

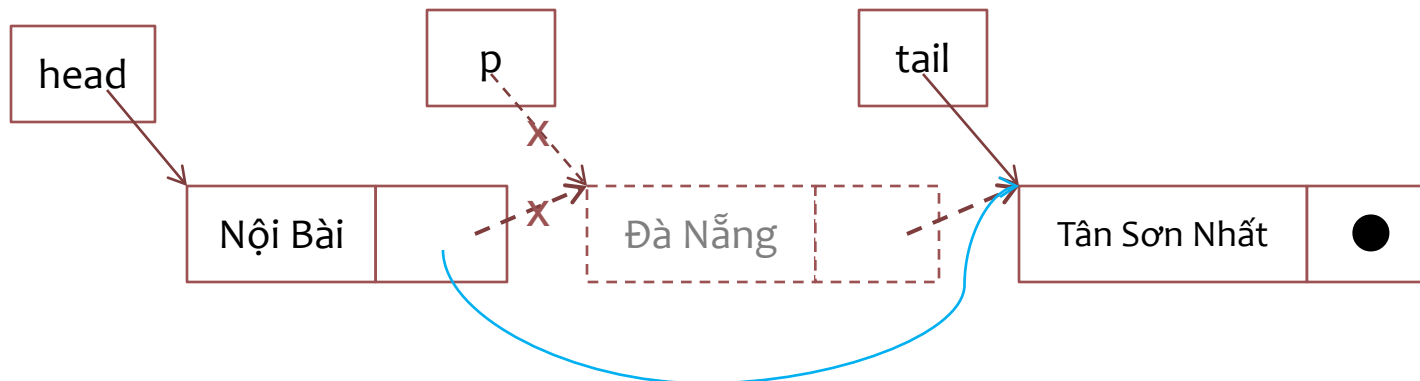
$(*pre).next \leftarrow (*p).next$

delete p

size \leftarrow *size* - 1

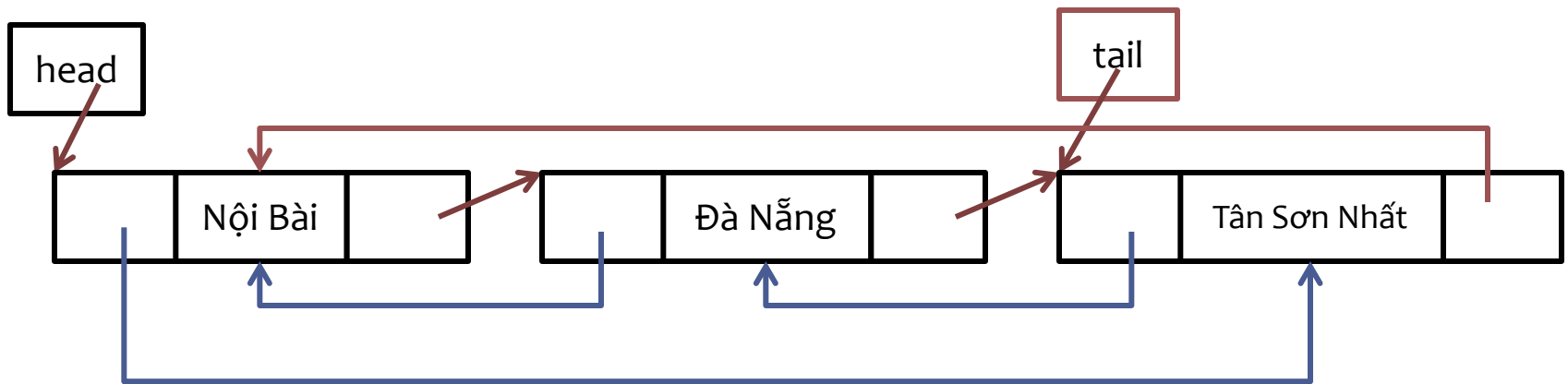
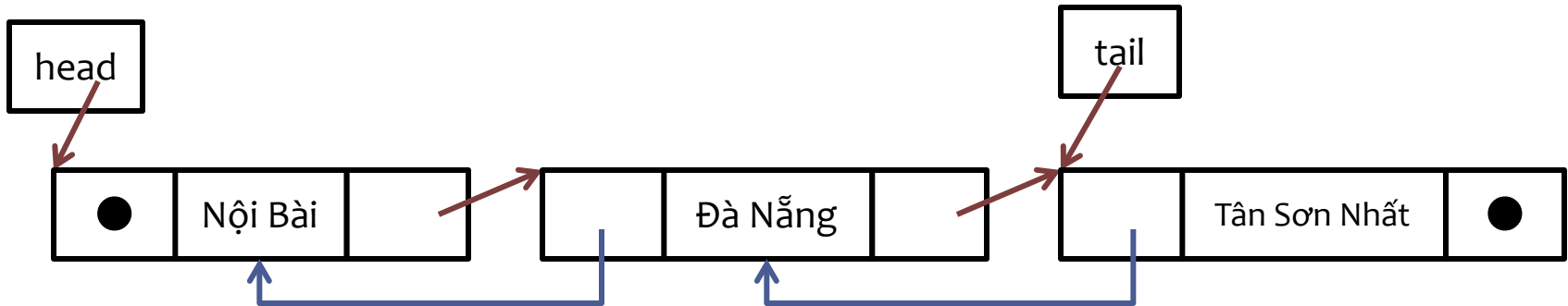
cập nhật tail

{pre chứa con trỏ đến nút sau p}
{giải phóng vùng nhớ trỏ bởi p}
{giảm biến đếm nút}



Các dạng DSLK

- DSLK đơn
 - singly linked list, uni-directional list, one-way list
- DSLK kép
 - doubly linked list, bi-directional list
- DSLK vòng tròn
 - ring list



Cài đặt danh sách bởi DSLK

- Sinh viên tự nghiên cứu chương 5.4, 5.5.

Câu hỏi

- Hãy mô tả cấu trúc của DSLK được định nghĩa bởi đoạn mã sau.

```
typedef struct Node ListNode;
struct Node{
    int data;
    ListNode *next;
}

typedef struct FirstNode *LinkedList;
struct FirstNode{
    ListNode *first;
}
```

```
typedef struct Node ListNode;
struct Node{
    int data;
    ListNode* next;
}

typedef struct FirstNode* LinkedList;
struct FirstNode{
    ListNode* first;
}
```

Chuẩn bị tuần tới

- Thực hành: Cài đặt các phép toán trên DSLK
- Lý thuyết: Đọc chương 6, 7 giáo trình.