



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Con trỏ (P.2)

Kỹ thuật Lập trình (CO1027)

Ngày 10 tháng 4 năm 2021

ThS. Trần Ngọc Bảo Duy

Khoa Khoa học và Kỹ thuật Máy tính

Trường Đại học Bách Khoa, ĐHQG-HCM



Truyền con trỏ như tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiên thực danh sách
bằng mảng

Danh sách liên kết

① Truyền con trỏ như tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

② Hiên thực danh sách bằng mảng

③ Danh sách liên kết



TRUYỀN CON TRỎ NHƯ THAM SỔ

Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Truyền con trỏ như tham số

Trong C, cách **truyền tham số bằng tham chiếu** (pass by reference) không tồn tại. Vì vậy, các hàm muốn thay đổi giá trị của đối số, người ta phải sử dụng **con trỏ**.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Truyền con trỏ như tham số

Trong C, cách **truyền tham số bằng tham chiếu** (pass by reference) không tồn tại. Vì vậy, các hàm muốn thay đổi giá trị của đối số, người ta phải sử dụng **con trỏ**.

```
1  #include <iostream>
2  using namespace std;
3
4  void inc(int* p2) {
5      *p2 = *p2 + 1;
6  }
7
8  int main(){
9      int a = 5;
10     int *p1 = &a;
11     inc(p1);
12     cout << a;
13 }
```



Truyền con trỏ như tham số

Trong C, cách **truyền tham số bằng tham chiếu** (pass by reference) không tồn tại. Vì vậy, các hàm muốn thay đổi giá trị của đối số, người ta phải sử dụng **con trỏ**.

```
1  #include <iostream>
2  using namespace std;
3
4  void inc(int* p2) {
5      *p2 = *p2 + 1;
6  }
7
8  int main(){
9      int a = 5;
10     int *p1 = &a;
11     inc(p1);
12     cout << a;
13 }
```

Chương trình trên in ra giá trị: 6



Bản chất của việc truyền con trỏ như tham số

a

5

0x1234FFFF

p1

0x1234FFFF

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Bản chất của việc truyền con trỏ như tham số

a



0x1234FFFF

p1



Khi gọi `inc(p1)`:

p2



Bản chất của việc truyền con trỏ như tham số

a

5

0x1234FFFF

p1

0x1234FFFF

Khi gọi `inc(p1)`:

p2

0x1234FFFF

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiển thực danh sách
bằng mảng

Danh sách liên kết

Bản chất của việc truyền con trỏ như tham số

a

5

0x1234FFFF

Khi gọi `inc(p1)`:

p1

0x1234FFFF

p2

0x1234FFFF

Việc sử dụng địa chỉ mà p2 chứa rồi lấy tham khảo dẫn đến việc thay đổi giá trị mà a chứa.



Truyền tham chiếu một con trỏ

```
1  #include <iostream>
2  using namespace std;
3
4  int gV = 10;
5
6  void foo(int* p) {
7      p = &gV;
8  }
9
10 int main(){
11     int a = 5;
12     int *p1 = &a;
13     foo(p1);
14     cout << *p1;
15 }
```

Chương trình trên in ra giá trị: 5



Truyền con trỏ như
tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Truyền tham chiếu một con trỏ

```
1  #include <iostream>
2  using namespace std;
3
4  int gV = 10;
5
6  void foo(int*& p) {
7      p = &gV;
8  }
9
10 int main(){
11     int a = 5;
12     int *p1 = &a;
13     foo(p1);
14     cout << *p1;
15 }
```

Chương trình trên in ra giá trị: 10

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Truyền mảng bằng con trỏ

NHẮC LẠI: Con trỏ và mảng đều giữ địa chỉ của ô nhớ.

- **Con trỏ:** giữ địa chỉ của ô nhớ nào đó (của biến khác, của bộ nhớ động).
- **Mảng:** giữ địa chỉ của phần tử đầu tiên.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Truyền mảng bằng con trỏ

NHẮC LẠI: Con trỏ và mảng đều giữ địa chỉ của ô nhớ.

- **Con trỏ:** giữ địa chỉ của ô nhớ nào đó (của biến khác, của bộ nhớ động).
- **Mảng:** giữ địa chỉ của phần tử đầu tiên.

Các prototype sau là tương đương:

```
int foo(int arr[], int n);  
int foo(int* arr, int n);
```





NHẮC LẠI: Con trỏ và mảng đều giữ địa chỉ của ô nhớ.

- **Con trỏ:** giữ địa chỉ của ô nhớ nào đó (của biến khác, của bộ nhớ động).
- **Mảng:** giữ địa chỉ của phần tử đầu tiên.

Các prototype sau là tương đương:

```
int foo(int arr[], int n);  
int foo(int* arr, int n);
```

Vì vậy, bản chất của việc truyền mảng là truyền địa chỉ của phần tử đầu tiên, chứ **KHÔNG** truyền thông tin đó là mảng, trong thân hàm ta không thể lấy được kích thước của mảng truyền vào.



Truyền con trở như
tham số

Truyền con trở như tham số

Truyền mảng bằng con trở

Hiện thực danh sách
bằng mảng

Danh sách liên kết

HIỆN THỰC DANH SÁCH BẰNG MẢNG

Sự khác biệt giữa mảng và danh sách

- Trong C/C++, **mảng** (array) là một **vùng nhớ liên tục** có kích thước cố định bao gồm nhiều phần tử cùng kiểu. Trong vùng nhớ đó, mỗi ô nhớ chứa một phần tử của mảng.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Sự khác biệt giữa mảng và danh sách

- Trong C/C++, **mảng** (array) là một **vùng nhớ liên tục** có kích thước cố định bao gồm nhiều phần tử cùng kiểu. Trong vùng nhớ đó, mỗi ô nhớ chứa một phần tử của mảng.
- **Danh sách**, nói chung, là một kiểu dữ liệu có thể chứa nhiều phần tử nhưng **kích thước không cố định**. Trong các kiểu dữ liệu cơ bản (primitive type) của C/C++, không có kiểu dữ liệu cho danh sách. Vì vậy, ta phải tự hiện thực nó hoặc sử dụng các thư viện có sẵn trong C/C++.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Sự khác biệt giữa mảng và danh sách

- Trong C/C++, **mảng** (array) là một **vùng nhớ liên tục** có kích thước cố định bao gồm nhiều phần tử cùng kiểu. Trong vùng nhớ đó, mỗi ô nhớ chứa một phần tử của mảng.
- **Danh sách**, nói chung, là một kiểu dữ liệu có thể chứa nhiều phần tử nhưng **kích thước không cố định**. Trong các kiểu dữ liệu cơ bản (primitive type) của C/C++, không có kiểu dữ liệu cho danh sách. Vì vậy, ta phải tự hiện thực nó hoặc sử dụng các thư viện có sẵn trong C/C++.

Một số thư viện có sẵn: `<vector>`, `<list>`.



Hiện thực danh sách bằng mảng

Ý tưởng

- 1 Cấp phát một mảng có kích thước lớn (sức chứa của danh sách).

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Hiện thực danh sách bằng mảng

Ý tưởng

- 1 Cấp phát một mảng có kích thước lớn (sức chứa của danh sách).
- 2 Sử dụng mảng trên cho đến khi sức chứa của danh sách không đáp ứng được số lượng phần tử của danh sách.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Ý tưởng

- ❶ Cấp phát một mảng có kích thước lớn (sức chứa của danh sách).
- ❷ Sử dụng mảng trên cho đến khi sức chứa của danh sách không đáp ứng được số lượng phần tử của danh sách.
- ❸ Cấp phát một vùng mảng mới có kích thước lớn hơn và chép các phần tử cũ qua.



Hiện thực danh sách bằng mảng

Ý tưởng

- ❶ Cấp phát một mảng có kích thước lớn (sức chứa của danh sách).
- ❷ Sử dụng mảng trên cho đến khi sức chứa của danh sách không đáp ứng được số lượng phần tử của danh sách.
- ❸ Cấp phát một vùng mảng mới có kích thước lớn hơn và chép các phần tử cũ qua.

⇒ Muốn vậy, mảng trên **PHẢI** được cấp phát động.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Hiện thực danh sách bằng mảng

Ý tưởng

- 1 Cấp phát một mảng có kích thước lớn (sức chứa của danh sách).
- 2 Sử dụng mảng trên cho đến khi sức chứa của danh sách không đáp ứng được số lượng phần tử của danh sách.
- 3 Cấp phát một vùng mảng mới có kích thước lớn hơn và chép các phần tử cũ qua.

⇒ Muốn vậy, mảng trên **PHẢI** được cấp phát động.

0	1	2	3	4	5	6	7	8	9
13	6	42	43	30	9				



Tạo ra danh sách

```
1 void createList(int*& arr, int cap = 10) {  
2     arr = new int[cap];  
3 }
```

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Tạo ra danh sách

```
1 void createList(int*& arr, int cap = 10) {  
2     arr = new int[cap];  
3 }
```

Hàm `createList` có hai tham số:

- 1 `arr`: con trỏ dùng để nhận lại địa chỉ của mảng sau khi cấp phát.
- 2 `cap`: sức chứa của danh sách, hay số lượng phần tử tối đa của mảng.



Thêm một phần tử vào danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	42	43	30	9				

Chèn giá trị **15** tại vị trí **2**:

- ❶ Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang phải **1** đơn vị.

Thêm một phần tử vào danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	42	43	30	9	9			

Chèn giá trị **15** tại vị trí **2**:

- ❶ Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang phải **1** đơn vị.

Thêm một phần tử vào danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	42	43	30	30	9			

Chèn giá trị **15** tại vị trí **2**:

- ❶ Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang phải **1** đơn vị.

Thêm một phần tử vào danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	42	43	43	30	9			

Chèn giá trị **15** tại vị trí **2**:

- ❶ Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang phải **1** đơn vị.

Thêm một phần tử vào danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	42	42	43	30	9			

Chèn giá trị **15** tại vị trí **2**:

- ❶ Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang phải **1** đơn vị.

Thêm một phần tử vào danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	15	42	43	30	9			

Chèn giá trị **15** tại vị trí **2**:

- ① Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang phải **1** đơn vị.
- ② Gán giá trị **15** vào vị trí **2**.

Thêm một phần tử vào danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	15	42	43	30	9			

Chèn giá trị **15** tại vị trí **2**:

- ① Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang phải **1** đơn vị.
- ② Gán giá trị **15** vào vị trí **2**.
- ③ Tăng kích thước của danh sách lên 1.

Thêm một phần tử vào danh sách

0	1	2	3	4	5	6	7	8	9
13	6	15	42	43	30	9			

Chèn giá trị **15** tại vị trí **2**:

- ❶ Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang phải **1** đơn vị.
- ❷ Gán giá trị **15** vào vị trí **2**.
- ❸ Tăng kích thước của danh sách lên **1**.

Prototype:

```
void add(int*& arr, int& len, int& cap, int ele, int pos = 0);
```



Xóa một phần tử ra khỏi danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	42	43	30	9				

Chèn giá trị **15** tại vị trí **2**:

- ❶ Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang trái **1** đơn vị.

Xóa một phần tử ra khỏi danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	43	43	30	9				

Chèn giá trị **15** tại vị trí **2**:

- ❶ Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang trái **1** đơn vị.

Xóa một phần tử ra khỏi danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	43	30	30	9				

Chèn giá trị **15** tại vị trí **2**:

- ❶ Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang trái **1** đơn vị.

Xóa một phần tử ra khỏi danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	43	30	9	9				

Chèn giá trị **15** tại vị trí **2**:

- ❶ Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang trái **1** đơn vị.
- ❷ Giảm kích thước của danh sách xuống 1.

Xóa một phần tử ra khỏi danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	43	30	9					

Chèn giá trị **15** tại vị trí **2**:

- ① Dịch chuyển toàn bộ các phần tử từ vị trí **2** sang trái **1** đơn vị.
- ② Giảm kích thước của danh sách xuống 1.

Prototype:

```
int remove(int*& arr, int& len, int& cap);
```

Xử lý khi danh sách vượt sức chứa

0	1	2	3	4
13	6	43	30	9

Khi danh sách vượt sức chứa:

- 1 Cấp phát một danh sách mới có sức chứa lớn hơn.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Xử lý khi danh sách vượt sức chứa

0 1 2 3 4 5 6 7 8 9

--	--	--	--	--	--	--	--	--	--

0 1 2 3 4

13	6	43	30	9
----	---	----	----	---

Khi danh sách vượt sức chứa:

- 1 Cấp phát một danh sách mới có sức chứa lớn hơn.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Xử lý khi danh sách vượt sức chứa

0	1	2	3	4	5	6	7	8	9
13									

0	1	2	3	4
13	6	43	30	9

Khi danh sách vượt sức chứa:

- ① Cấp phát một danh sách mới có sức chứa lớn hơn.
- ② Chép từng phần tử trong danh sách cũ lên danh sách trên theo đúng vị trí.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Xử lý khi danh sách vượt sức chứa

0	1	2	3	4	5	6	7	8	9
13	6								

0	1	2	3	4
13	6	43	30	9

Khi danh sách vượt sức chứa:

- ① Cấp phát một danh sách mới có sức chứa lớn hơn.
- ② Chép từng phần tử trong danh sách cũ lên danh sách trên theo đúng vị trí.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Xử lý khi danh sách vượt sức chứa

0	1	2	3	4	5	6	7	8	9
13	6	43							

0	1	2	3	4
13	6	43	30	9

Khi danh sách vượt sức chứa:

- ① Cấp phát một danh sách mới có sức chứa lớn hơn.
- ② Chép từng phần tử trong danh sách cũ lên danh sách trên theo đúng vị trí.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Xử lý khi danh sách vượt sức chứa

0	1	2	3	4	5	6	7	8	9
13	6	43	30						

0	1	2	3	4
13	6	43	30	9

Khi danh sách vượt sức chứa:

- ❶ Cấp phát một danh sách mới có sức chứa lớn hơn.
- ❷ Chép từng phần tử trong danh sách cũ lên danh sách trên theo đúng vị trí.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Xử lý khi danh sách vượt sức chứa

0	1	2	3	4	5	6	7	8	9
13	6	43	30	9					

0	1	2	3	4
13	6	43	30	9

Khi danh sách vượt sức chứa:

- ① Cấp phát một danh sách mới có sức chứa lớn hơn.
- ② Chép từng phần tử trong danh sách cũ lên danh sách trên theo đúng vị trí.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Xử lý khi danh sách vượt sức chứa

0	1	2	3	4	5	6	7	8	9
13	6	43	30	9					

0	1	2	3	4
13	6	43	30	9

Khi danh sách vượt sức chứa:

- 1 Cấp phát một danh sách mới có sức chứa lớn hơn.
- 2 Chép từng phần tử trong danh sách cũ lên danh sách trên theo đúng vị trí.
- 3 Chuyển con trỏ sang chỉ cho mảng mới, tăng sức chứa.



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Xử lý khi danh sách vượt sức chứa

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

0	1	2	3	4	5	6	7	8	9
13	6	43	30	9					

Khi danh sách vượt sức chứa:

- 1 Cấp phát một danh sách mới có sức chứa lớn hơn.
- 2 Chép từng phần tử trong danh sách cũ lên danh sách trên theo đúng vị trí.
- 3 Chuyển con trỏ sang chỉ cho mảng mới, tăng sức chứa.
- 4 Thu hồi mảng cũ đã cấp phát.

Xử lý danh sách vượt sức chứa

0	1	2	3	4	5	6	7	8	9
13	6	43	30	9					

Khi danh sách vượt sức chứa:

- ❶ Cấp phát một danh sách mới có sức chứa lớn hơn.
- ❷ Chép từng phần tử trong danh sách cũ lên danh sách trên theo đúng vị trí.
- ❸ Chuyển con trỏ sang chỉ cho mảng mới, tăng sức chứa.
- ❹ Thu hồi mảng cũ đã cấp phát.

Prototype:

```
void ensureCap(int*& arr, int& cap, int newCap);
```



Khai báo cấu trúc cho danh sách

```
1 struct ArrayList {  
2     int* data;  
3     int cap;  
4     int len;  
5 };
```

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Khai báo cấu trúc cho danh sách

```
1 struct ArrayList {  
2     int* data;  
3     int cap;  
4     int len;  
5 };
```

Prototype:

```
void add(ArrayList& list, int ele, int pos = 0);  
    int remove(ArrayList& list, int pos = 0);  
    void ensureCap(ArrayList& list, int newCap);
```

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết



Truyền con trở như
tham số

Truyền con trở như tham số

Truyền mảng bằng con trở

Hiện thực danh sách
bằng mảng

Danh sách liên kết

DANH SÁCH LIÊN KẾT

Nhược điểm của hiện thực danh sách bằng mảng

- 1 Mảng đòi hỏi phải cấp phát một vùng nhớ liên tục.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Nhược điểm của hiện thực danh sách bằng mảng

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

- ❶ Mảng đòi hỏi phải cấp phát một vùng nhớ liên tục.
- ❷ Khi thêm hoặc xóa phần tử, cần phải dịch chuyển một khối lớn nhiều phần tử sang trái hoặc sang phải.

Ý tưởng của danh sách liên kết

Ý tưởng chung

- Các phần tử nằm rời rạc trong vùng nhớ máy tính để khắc phục các nhược điểm nêu trên.
- Các phần tử phải liên kết với nhau bằng việc chứa thêm thông tin địa chỉ của các phần tử khác theo một thứ tự nào đó.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Ý tưởng của danh sách liên kết

Ý tưởng chung

- Các phần tử nằm rời rạc trong vùng nhớ máy tính để khắc phục các nhược điểm nêu trên.
- Các phần tử phải liên kết với nhau bằng việc chứa thêm thông tin địa chỉ của các phần tử khác theo một thứ tự nào đó.

Danh sách liên kết (DSLK) đơn

DSLK đơn là danh sách mà ở đó mỗi phần tử (node - nút):

- Chứa phần dữ liệu của mỗi phần tử tương tự như mảng.
- Chứa địa chỉ của phần tử tiếp theo trong danh sách trong đó, phần tử cuối cùng không giữ địa chỉ của bất kỳ phần tử nào.

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Ý tưởng của danh sách liên kết

Ý tưởng chung

- Các phần tử nằm rời rạc trong vùng nhớ máy tính để khắc phục các nhược điểm nêu trên.
- Các phần tử phải liên kết với nhau bằng việc chứa thêm thông tin địa chỉ của các phần tử khác theo một thứ tự nào đó.

Danh sách liên kết (DSLK) đơn

DSLK đơn là danh sách mà ở đó mỗi phần tử (node - nút):

- Chứa phần dữ liệu của mỗi phần tử tương tự như mảng.
- Chứa địa chỉ của phần tử tiếp theo trong danh sách trong đó, phần tử cuối cùng không giữ địa chỉ của bất kỳ phần tử nào.

NHƯ VẬY, DSLK đơn chỉ cần giữ địa chỉ của phần tử đầu tiên (head).

Pointer

ThS.
Trần Ngọc Bảo Duy



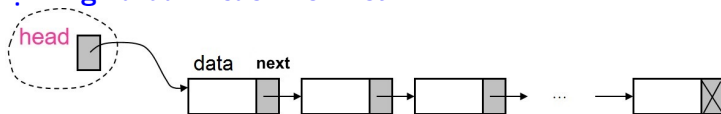
Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Định nghĩa danh sách liên kết



Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

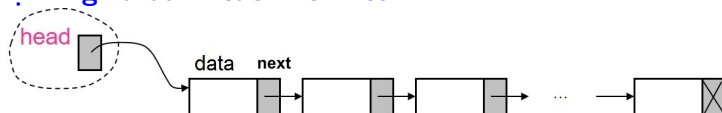
Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiên thực danh sách
bằng mảng

Danh sách liên kết

Định nghĩa danh sách liên kết

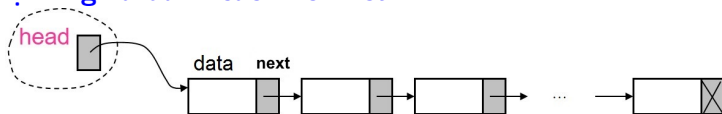


Để tạo ra danh sách liên kết mà mỗi nút chứa một dữ liệu kiểu T:

```
1 struct Node {  
2     T data;  
3     Node* next;  
4 };  
5  
6 struct LList {  
7     Node* head;  
8 };
```



Định nghĩa danh sách liên kết



Để tạo ra danh sách liên kết mà mỗi nút chứa một dữ liệu kiểu **T**:

```
1 struct Node {  
2     T data;  
3     Node* next;  
4 };  
5  
6 struct LList {  
7     Node* head;  
8 };
```

trong đó, cấu trúc **Node** dùng để chứa một nút trong danh sách liên kết, còn **LList** là cấu trúc đại diện của một danh sách liên kết.

Kiểu **T** trên có thể là kiểu số nguyên (**int**), kiểu số thực (**double**), kiểu Sinh viên tự định nghĩa (**Student**) ...



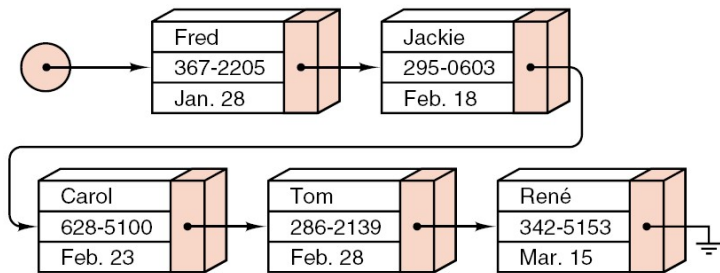
Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Ví dụ về danh sách liên kết



Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Ví dụ về danh sách liên kết

Pointer

**ThS.
Trần Ngọc Bảo Duy**



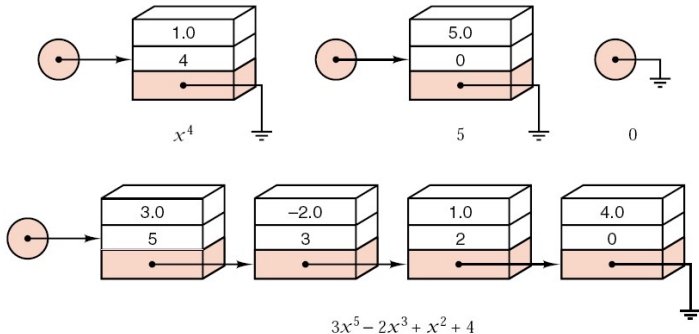
Truyền con trở như
tham số

Truyền con trở như tham số

Truyền mạng bằng con trỏ

Hiện thực danh sách bằng mạng

Danh sách liên kết



Ví dụ về tạo danh sách liên kết

```
1  #include <iostream>
2  using namespace std;
3
4  struct Node { int data; Node* next; };
5  struct LList { Node* head; };
6
7  int main() {
8      LList l;
9      l.head = new Node(10, NULL);
10     l.head->next = new Node(20, NULL);
11     l.head->next->next = new Node(30, NULL);
12 }
```

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Thêm một phần tử vào DSLK đơn

Khi thêm một phần tử vào DSLK đơn, ta cần quan tâm đến các trường hợp sau:

- 1 Trường hợp 1: DSLK rỗng.
- 2 Trường hợp 2: DSLK không rỗng.
 - Thêm vào đầu danh sách.
 - Thêm vào vị trí bất kỳ.



Thêm một phần tử vào DSLK đơn

Khi thêm một phần tử vào DSLK đơn, ta cần quan tâm đến các trường hợp sau:

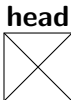
- 1 Trường hợp 1: DSLK rỗng.
- 2 Trường hợp 2: DSLK không rỗng.
 - Thêm vào đầu danh sách.
 - Thêm vào vị trí bất kỳ.

Prototype:

```
bool insert(LList& list, int ele, int pos = 0);
```



Thêm vào danh sách rỗng



Khi thêm phần tử 18 vào DSLK:



Truyền con trỏ như
tham số

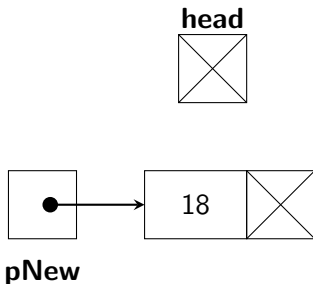
Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Thêm vào danh sách rỗng

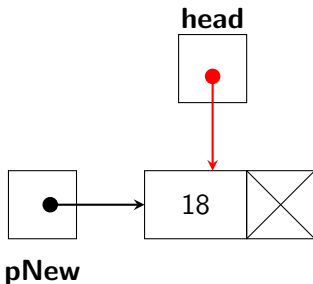


Khi thêm phần tử 18 vào DSLK:

- 1 Cấp phát ra một nút chứa số 18, và được giữ bởi con trỏ **pNew**.



Thêm vào danh sách rỗng

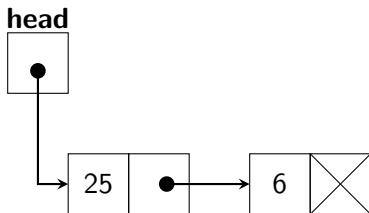


Khi thêm phần tử 18 vào DSLK:

- 1 Cấp phát ra một nút chứa số 18, và được giữ bởi con trỏ **pNew**.
- 2 Trỏ con trỏ **head** đến nút mới đó: **head = pNew**



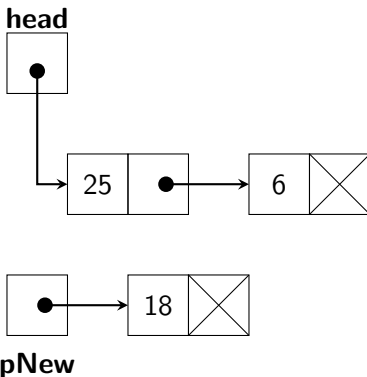
Thêm vào đầu danh sách không rỗng



Khi thêm phần tử số 18 vào đầu danh sách không rỗng:



Thêm vào đầu danh sách không rỗng

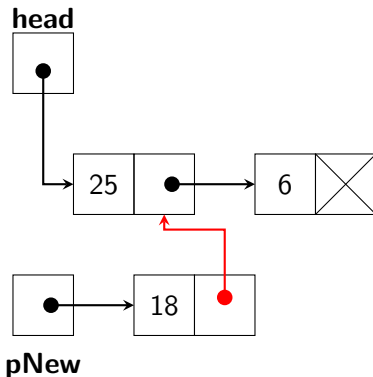


Khi thêm phần tử số 18 vào đầu danh sách không rỗng:

- 1 Cấp phát một nút mới chứa số 18 và được giữ bởi con trỏ pNew.



Thêm vào đầu danh sách không rỗng

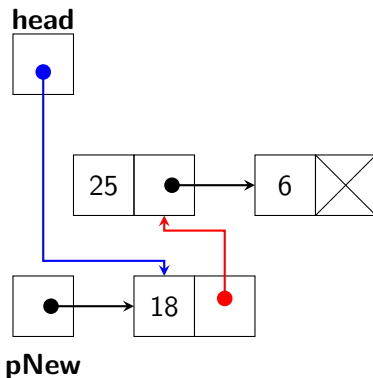


Khi thêm phần tử số 18 vào đầu danh sách không rỗng:

- 1 Cấp phát một nút mới chứa số 18 và được giữ bởi con trỏ pNew.
- 2 $pNew \rightarrow next = head$



Thêm vào đầu danh sách không rỗng



Khi thêm phần tử số 18 vào đầu danh sách không rỗng:

- ① Cấp phát một nút mới chứa số 18 và được giữ bởi con trỏ pNew.
- ② $pNew \rightarrow next = head$
- ③ $head = pNew$



Truyền con trỏ như tham số

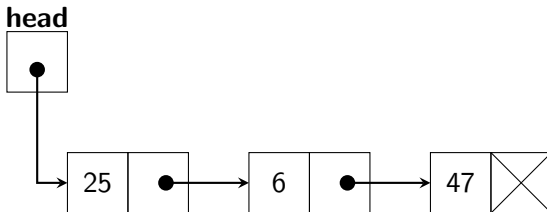
Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách bằng mảng

Danh sách liên kết

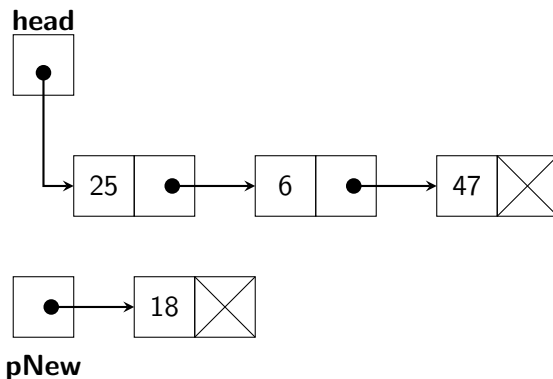
Thêm vào vị trí bất kỳ của danh sách không rỗng



Khi thêm phần tử số 18 vào vị trí **2** của DS không rỗng:



Thêm vào vị trí bất kỳ của danh sách không rỗng

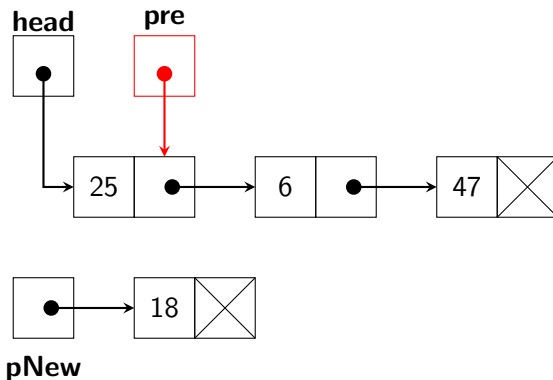


Khi thêm phần tử số 18 vào vị trí **2** của DS không rỗng:

- 1 Cấp phát một nút mới chứa số 18, giữ bởi con trỏ pNew.



Thêm vào vị trí bất kỳ của danh sách không rỗng

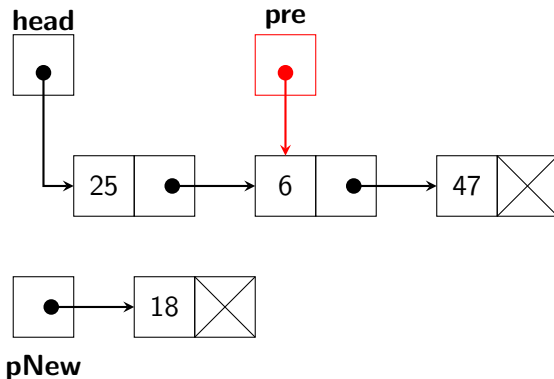


Khi thêm phần tử số 18 vào vị trí **2** của DS không rỗng:

- 1 Cấp phát một nút mới chứa số 18, giữ bởi con trỏ **pNew**.
- 2 Tìm ra địa chỉ (con trỏ) của nút tại vị trí **1**.



Thêm vào vị trí bất kỳ của danh sách không rỗng

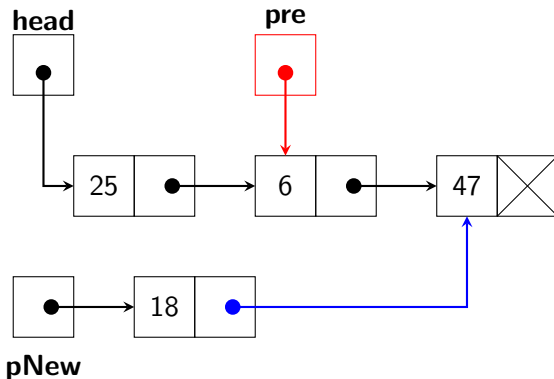


Khi thêm phần tử số 18 vào vị trí **2** của DS không rỗng:

- 1 Cấp phát một nút mới chứa số 18, giữ bởi con trỏ pNew.
- 2 Tìm ra địa chỉ (con trỏ) của nút tại vị trí **1**.



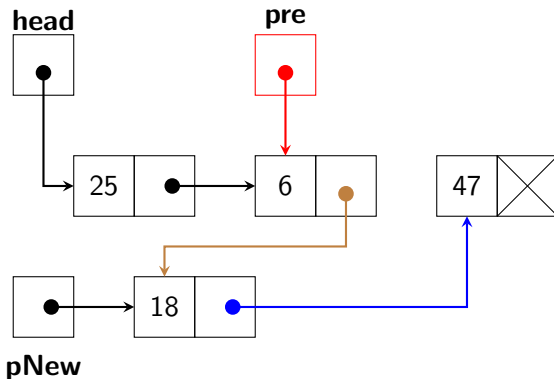
Thêm vào vị trí bất kỳ của danh sách không rỗng



Khi thêm phần tử số 18 vào vị trí **2** của DS không rỗng:

- 1 Cấp phát một nút mới chứa số 18, giữ bởi con trỏ pNew.
- 2 Tìm ra địa chỉ (con trỏ) của nút tại vị trí **1**.
- 3 $pNew \rightarrow next = pre \rightarrow next$

Thêm vào vị trí bất kỳ của danh sách không rỗng



Khi thêm phần tử số 18 vào vị trí **2** của DS không rỗng:

- 1 Cấp phát một nút mới chứa số 18, giữ bởi con trỏ pNew.
- 2 Tìm ra địa chỉ (con trỏ) của nút tại vị trí **1**.
- 3 $pNew \rightarrow next = pre \rightarrow next$
- 4 $pre \rightarrow next = pNew$

Xóa phần tử ra khỏi DSLK đơn

Khi xóa một phần tử ra khỏi DSLK đơn, ta cần quan tâm đến các trường hợp sau:

- ❶ Xóa phần tử ở đầu danh sách.
- ❷ Xóa phần tử ở vị trí bất kỳ trong danh sách.



Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

Xóa phần tử ra khỏi DSLK đơn

Khi xóa một phần tử ra khỏi DSLK đơn, ta cần quan tâm đến các trường hợp sau:

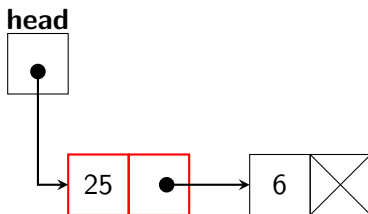
- ❶ Xóa phần tử ở đầu danh sách.
- ❷ Xóa phần tử ở vị trí bất kỳ trong danh sách.

Prototype:

```
bool remove(LList& list, int& rem, int pos = 0);
```



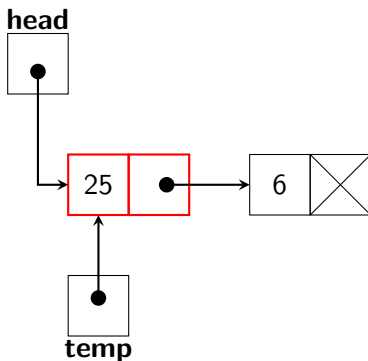
Xóa phần tử ở đầu danh sách



Xóa phần tử đầu tiên ra khỏi DS:



Xóa phần tử ở đầu danh sách

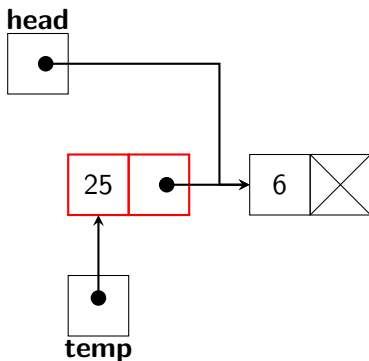


Xóa phần tử đầu tiên ra khỏi DS:

- 1 Đặt tên con trỏ tạm: $\text{temp} = \text{head}$



Xóa phần tử ở đầu danh sách

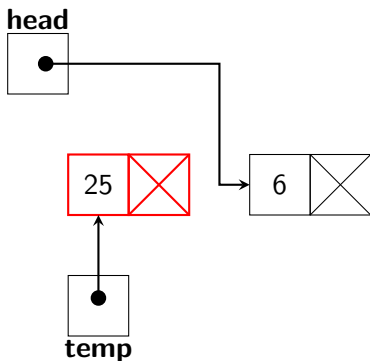


Xóa phần tử đầu tiên ra khỏi DS:

- 1 Đặt tên con trỏ tạm: $\text{temp} = \text{head}$
- 2 Chuyển địa chỉ mà head đang trỏ qua nút cạnh bên:
 $\text{head} = \text{head} \rightarrow \text{next}$



Xóa phần tử ở đầu danh sách

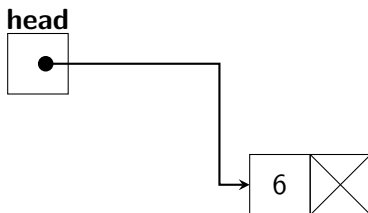


Xóa phần tử đầu tiên ra khỏi DS:

- 1 Đặt tên con trỏ tạm: $\text{temp} = \text{head}$
- 2 Chuyển địa chỉ mà head đang trỏ qua nút cạnh bên:
 $\text{head} = \text{head} \rightarrow \text{next}$
- 3 Cô lập temp: $\text{temp} \rightarrow \text{next} = \text{NULL}$



Xóa phần tử ở đầu danh sách



Xóa phần tử đầu tiên ra khỏi DS:

- 1 Đặt tên con trỏ tạm: `temp = head`
- 2 Chuyển địa chỉ mà head đang trỏ qua nút cạnh bên:
`head = head->next`
- 3 Cô lập temp: `temp->next = NULL`
- 4 Xóa bỏ nút mà temp đang trỏ tới: `delete temp`



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiên thực danh sách
bằng mảng

Danh sách liên kết

Tìm kiếm phần tử trong danh sách liên kết

Pointer

ThS.
Trần Ngọc Bảo Duy



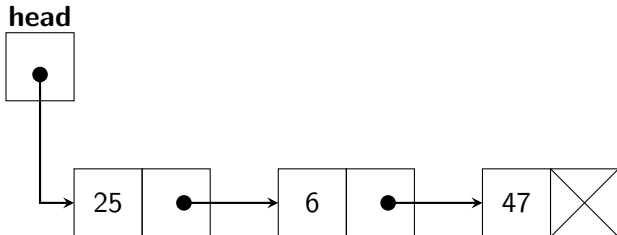
Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết



Tìm kiếm phần tử tại vị trí `index = 1`:

Tìm kiếm phần tử trong danh sách liên kết

Pointer

ThS.
Trần Ngọc Bảo Duy



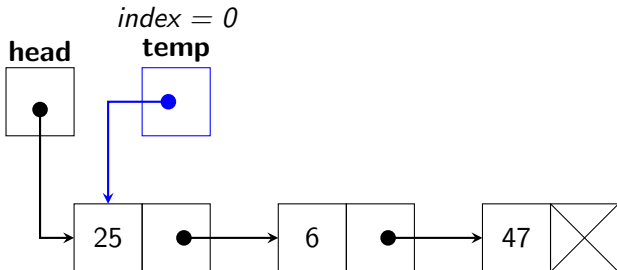
Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết



Tìm kiếm phần tử tại vị trí $index = 1$:

- 1 Đặt con trỏ tạm $temp = head$ và biến đếm $index = 0$ dùng để chạy.

Tìm kiếm phần tử trong danh sách liên kết

Pointer

ThS.
Trần Ngọc Bảo Duy



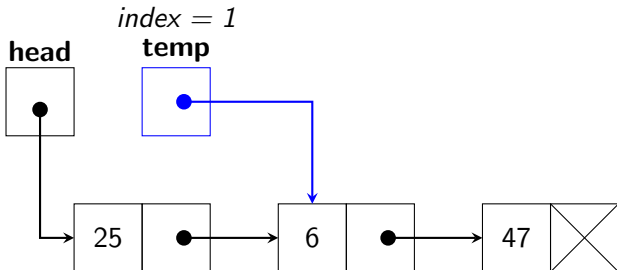
Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết



Tìm kiếm phần tử tại vị trí $index = 1$:

- 1 Đặt con trỏ tạm $temp = head$ và biến đếm $index = 0$ dùng để chạy.
- 2 Nếu $index$ đang mang giá trị cần tìm thì dừng lại.
- 3 Nếu không, ta trỏ $temp = temp \rightarrow next$ và tăng biến đếm lên 1 đơn vị.

Tìm kiếm phần tử trong danh sách liên kết

Pointer

ThS.
Trần Ngọc Bảo Duy



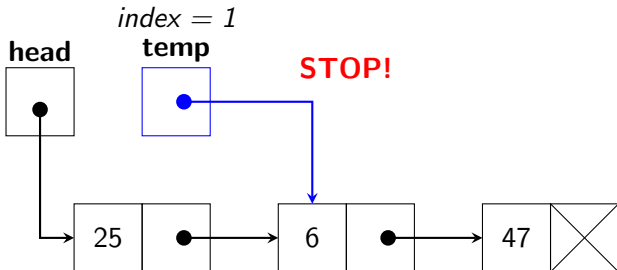
Truyền con trỏ như
tham số

Truyền con trỏ như tham số

Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết



Tìm kiếm phần tử tại vị trí $index = 1$:

- 1 Đặt con trỏ tạm $temp = head$ và biến đếm $index = 0$ dùng để chạy.
- 2 Nếu $index$ đang mang giá trị cần tìm thì dừng lại.
- 3 Nếu không, ta trở $temp = temp \rightarrow next$ và tăng biến đếm lên 1 đơn vị.
- 4 Quay lại bước 2.

So sánh giữa hai phương pháp hiện thực danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

① Hiện thực bằng mảng được ưu tiên sử dụng khi:

- số lượng phần tử nhỏ;
- kích thước danh sách nên được biết khi viết chương trình;
- số lượng thao tác thêm và xóa ít được diễn ra;
- thường xuyên truy cập ngẫu nhiên

So sánh giữa hai phương pháp hiện thực danh sách

Pointer

ThS.
Trần Ngọc Bảo Duy



Truyền con trỏ như
tham số

Truyền con trỏ như tham số
Truyền mảng bằng con trỏ

Hiện thực danh sách
bằng mảng

Danh sách liên kết

① Hiện thực bằng mảng được ưu tiên sử dụng khi:

- số lượng phần tử nhỏ;
- kích thước danh sách nên được biết khi viết chương trình;
- số lượng thao tác thêm và xóa ít được diễn ra;
- thường xuyên truy cập ngẫu nhiên

② Danh sách liên kết sẽ được ưu tiên khi:

- số lượng phần tử khá lớn;
- kích thước danh sách không được biết trước tại thời điểm viết;
- thao tác thêm, xóa, tái sắp xếp danh sách thường xuyên diễn ra.