

BAN HỌC TẬP KHOA CÔNG NGHỆ PHẦN MỀM

CHUỖI TRAINING GIỮA HỌC KÌ 2 NĂM HỌC 2021 - 2022



Sharing is learning



Ban học tập

Khoa Công Nghệ Phần Mềm
Trường ĐH Công Nghệ Thông Tin
ĐHQG Hồ Chí Minh



Email / Group

bht.cnpm.uit@gmail.com
fb.com/groups/bht.cnpm.uit

Training

Cấu trúc dữ liệu và Giải thuật

🕒 Thời gian training: 04/04/2022

📍 Phòng: MS Teams

👤 Trainer: Mai Hiếu Hiền – KHCL2020

Phan Phạm Trọng Phúc – ATTN2021



Sharing is learning



Sharing is learning

Nội dung Training



Sharing is learning

- I. Danh sách liên kết đơn, ngăn xếp, hàng đợi**
- II. Giải thuật tìm kiếm.**
- III. Giải đề tham khảo.**



Sharing is learning

Danh sách liên kết đơn



I. Danh sách liên kết đơn, ngăn xếp, hàng đợi

Note: (cần nắm vững trước khi học DSLK)

- Hàm?
- Mảng 1 chiều? (tìm kiếm, sắp xếp, thêm, xóa)
- Con trỏ?
- Kiểu cấu trúc? (Struct)



Sharing is learning

Danh sách liên kết đơn



Sharing is learning

**** Cài đặt danh sách liên kết đơn:**

1. Khai báo node
2. Khai báo danh sách (List)
3. Khởi tạo node có giá trị bằng x
4. Tạo danh sách liên kết rỗng
5. Thêm node vào DSLKĐ
6. Xóa node trong DSLKĐ



Sharing is learning

Danh sách liên kết đơn



Sharing is learning

1. Khai báo node:



Data: phần chứa dữ liệu của node

Next: con trỏ lưu trữ địa chỉ của node phía sau ??



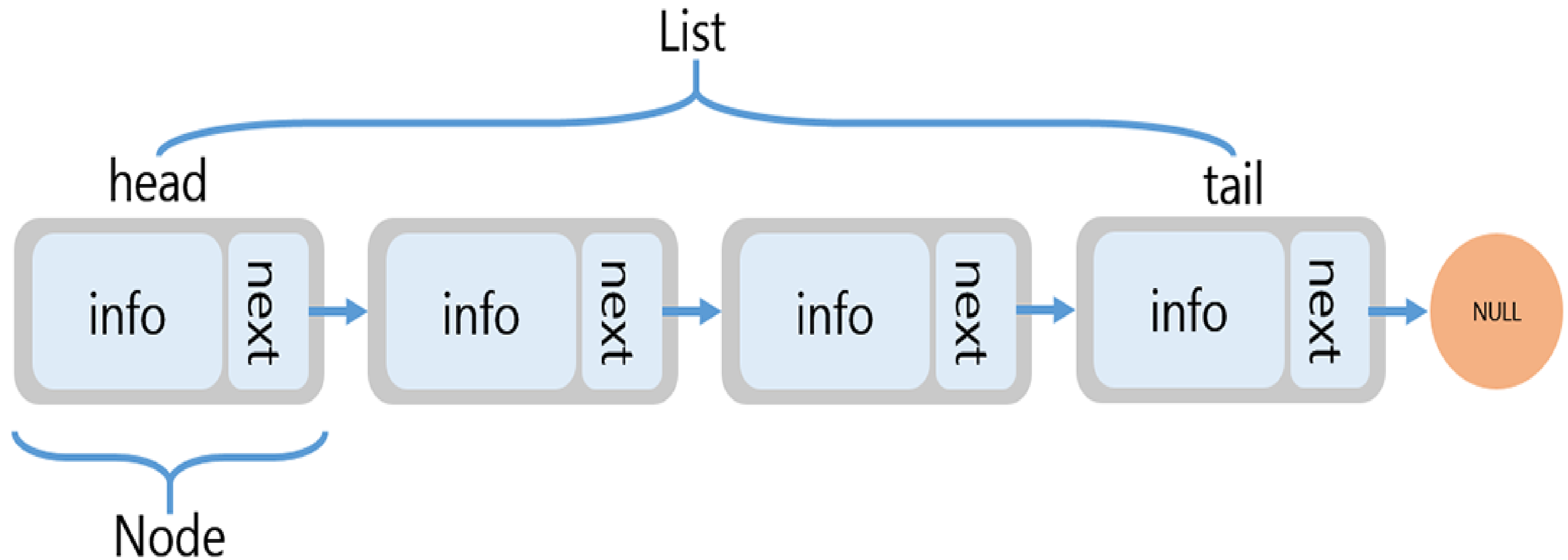
Sharing is learning

Danh sách liên kết đơn



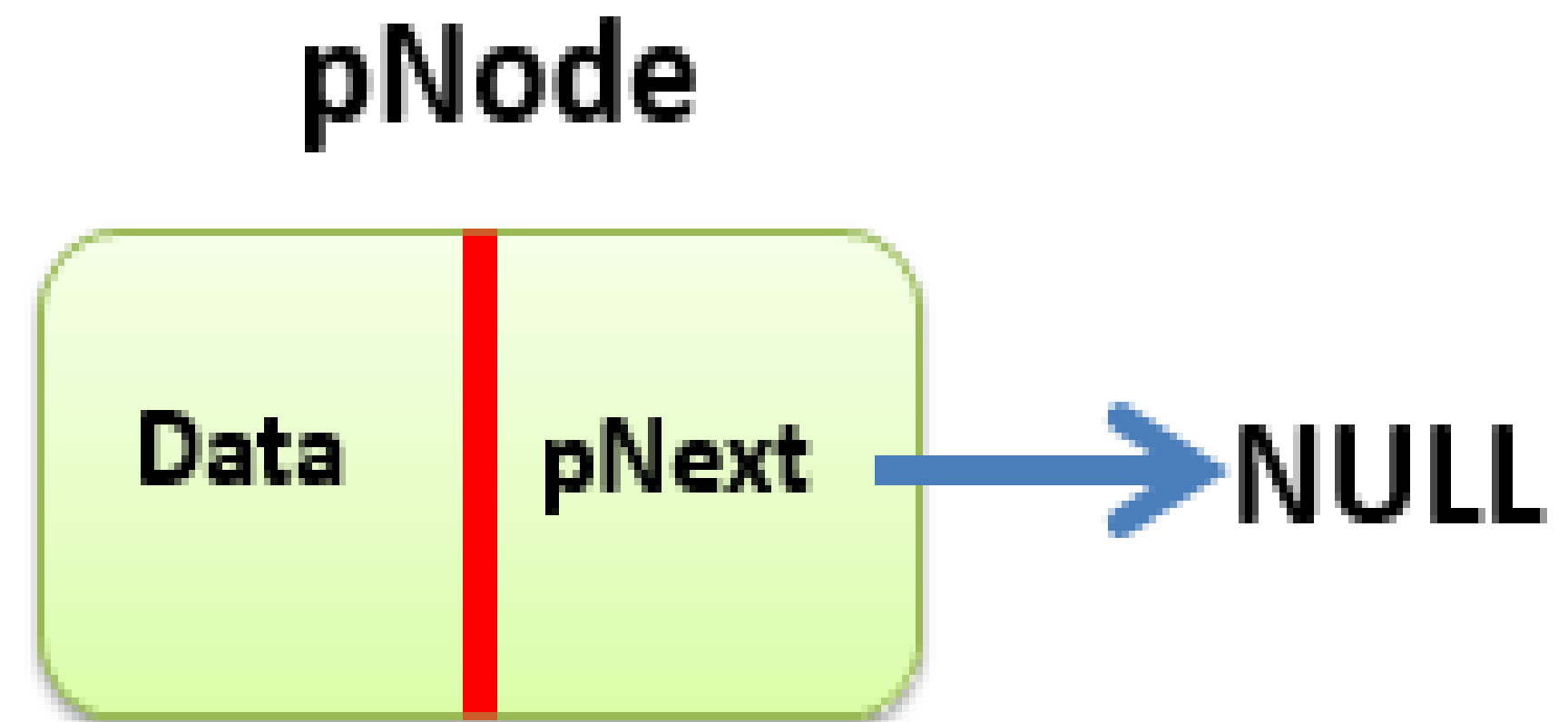
Sharing is learning

2. Khai báo danh sách liên kết đơn



3. Khởi tạo một node mới có giá trị bằng x

```
15  NODE* CreateNODE(int x)
16  {
17      NODE* p = new NODE;
18      if (p == NULL)
19          return NULL;
20      p->data = x;
21      p->pNext = NULL;
22      return p;
23  }
```



Danh sách liên kết đơn



Sharing is learning

4. Khởi tạo danh sách liên kết rỗng

```
void Createlist(List& l) {  
    l.head = NULL;  
    l.tail = NULL;  
}
```

5. Thêm 1 node vào danh sách liên kết

- Thêm 1 node vào đầu DSLK
- Thêm 1 node vào cuối DSLK
- Thêm 1 node p vào sau một node q trong DSLK



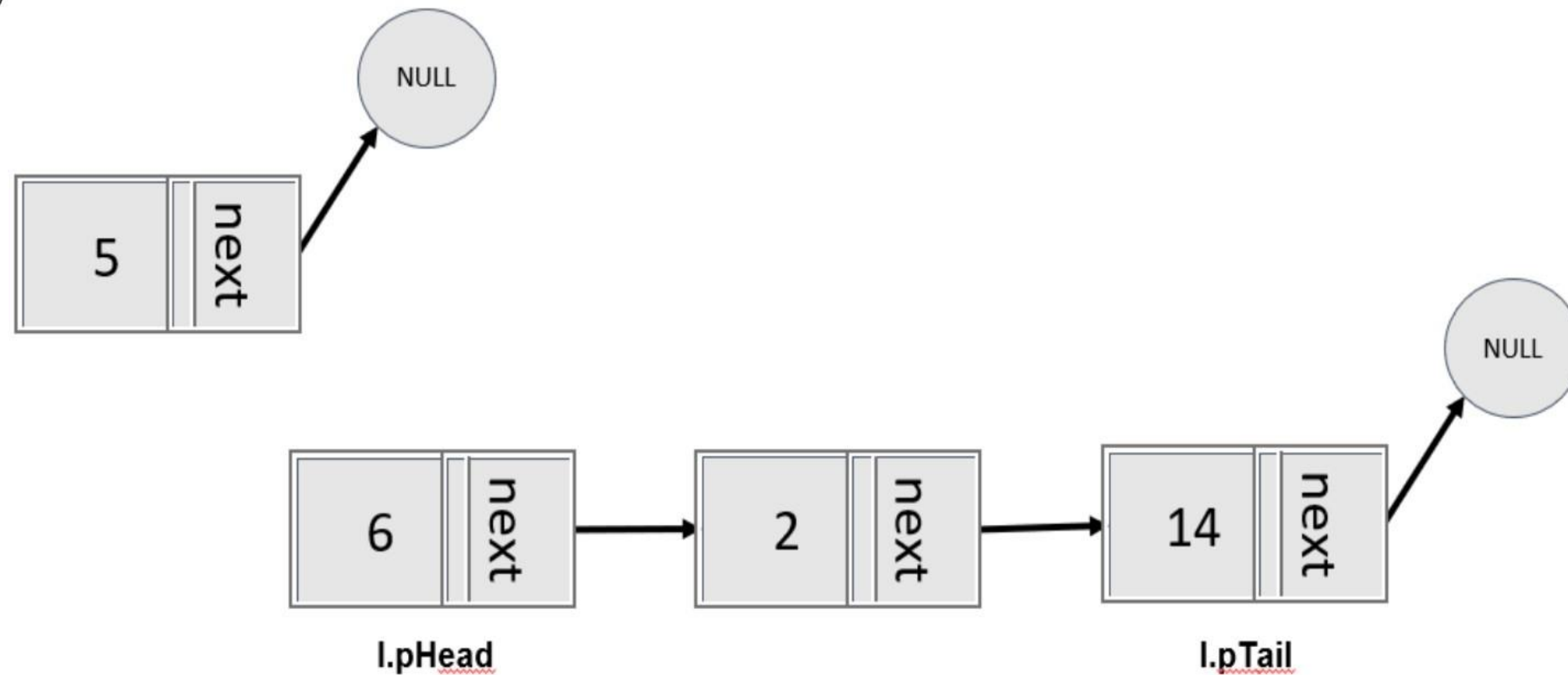
Sharing is learning

Thêm 1 node vào DSLKĐ



Sharing is learning

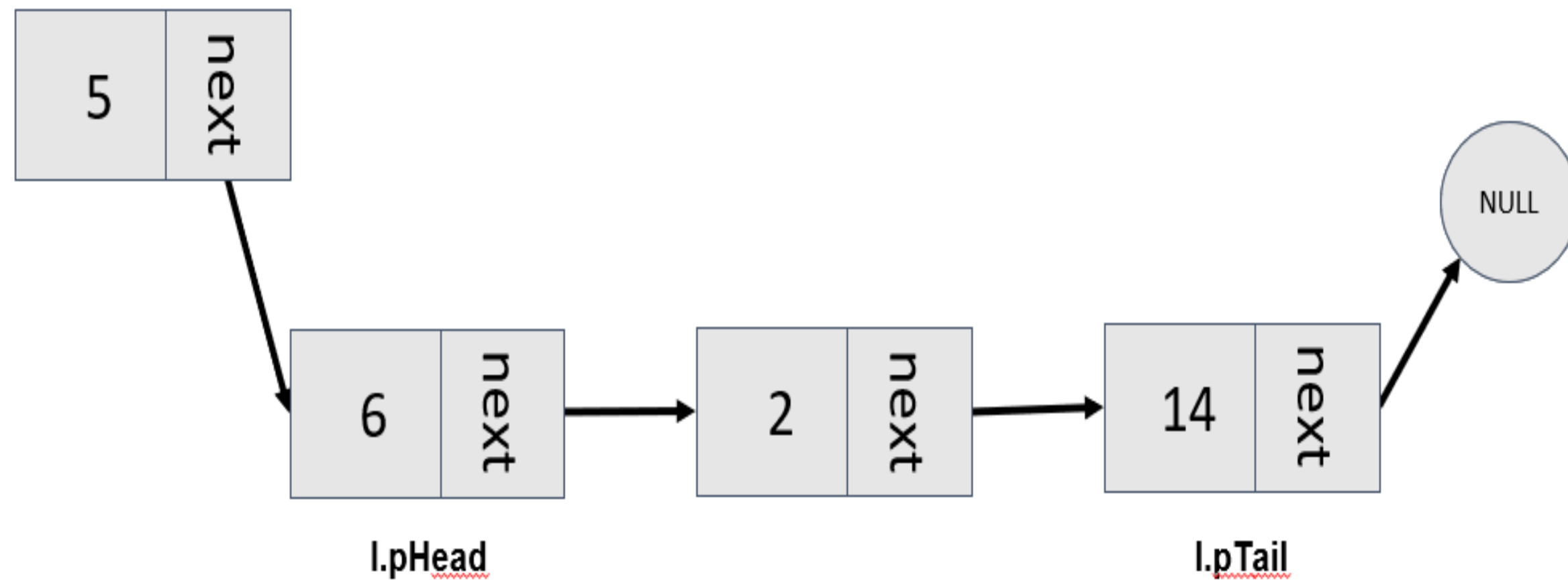
5a. Thêm node p vào đầu DSLKĐ (AddHead)



Sharing is learning

Thêm 1 node vào DSLKĐ

5a. Thêm node p vào đầu DSLKĐ (AddHead)



Sharing is learning



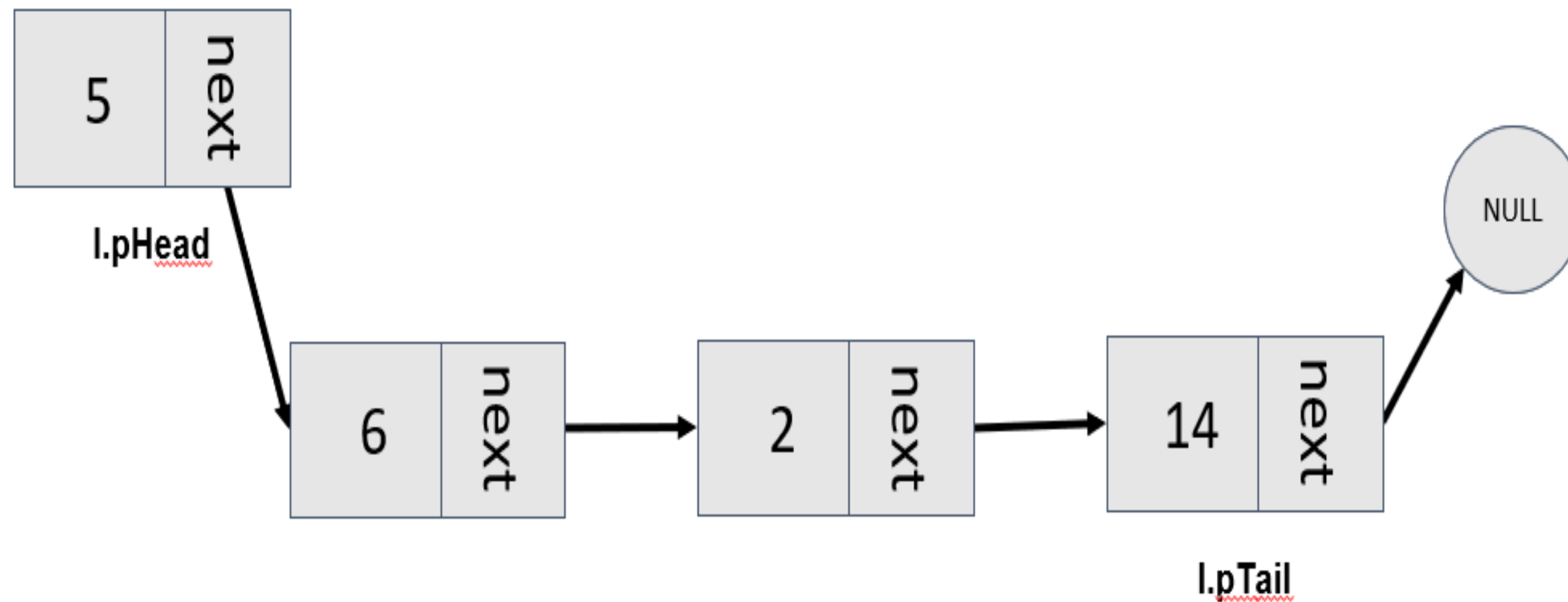
Sharing is learning

Thêm 1 node vào DSLKĐ



Sharing is learning

5a. Thêm node p vào đầu DSLKĐ (AddHead)



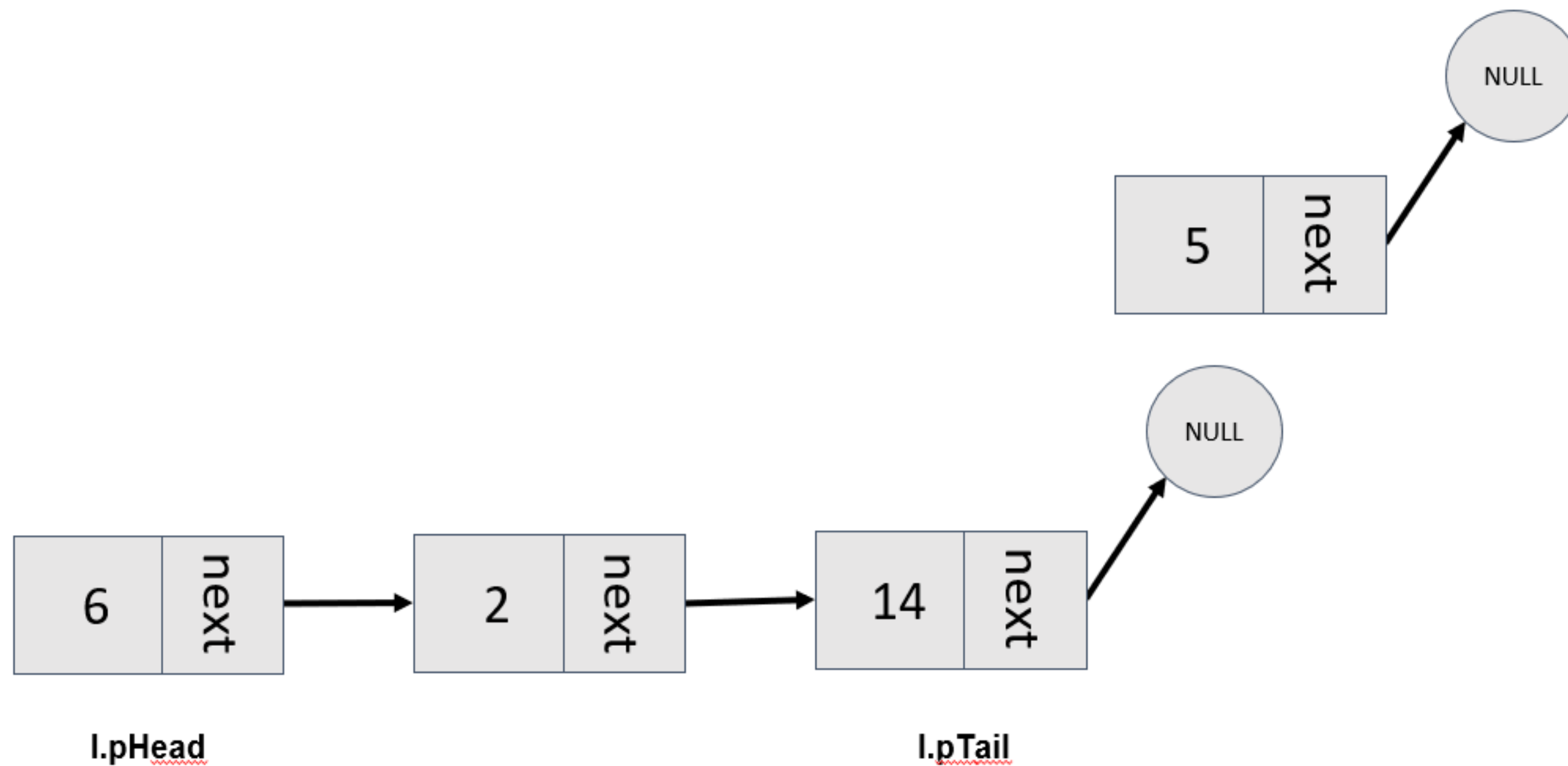
Sharing is learning

Thêm 1 node vào DSLKĐ



Sharing is learning

5b. Thêm node p vào cuối DSLKĐ (AddTail)



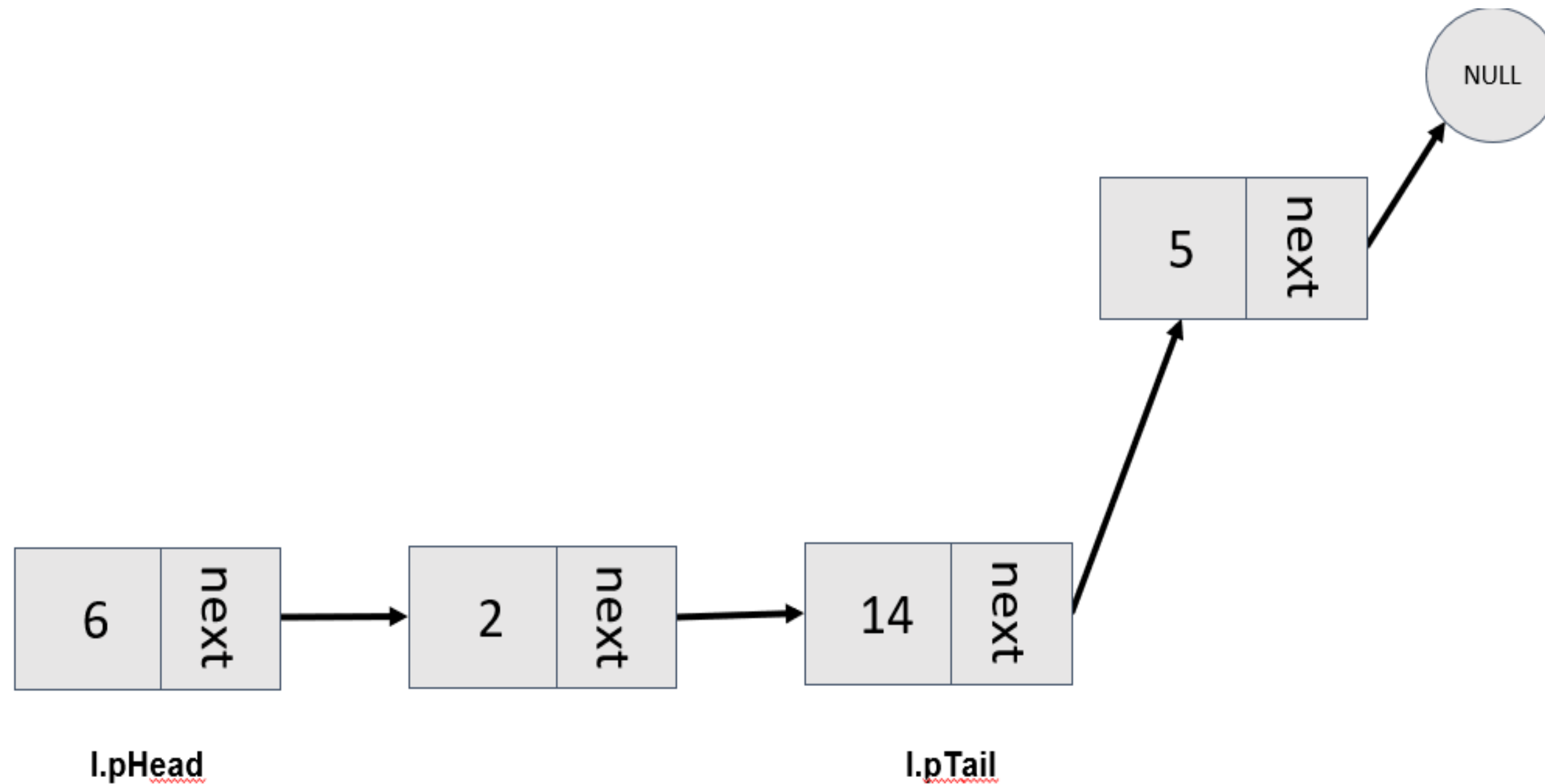
Sharing is learning

Thêm 1 node vào DSLKĐ



Sharing is learning

5b. Thêm node p vào cuối DSLKĐ (AddTail)



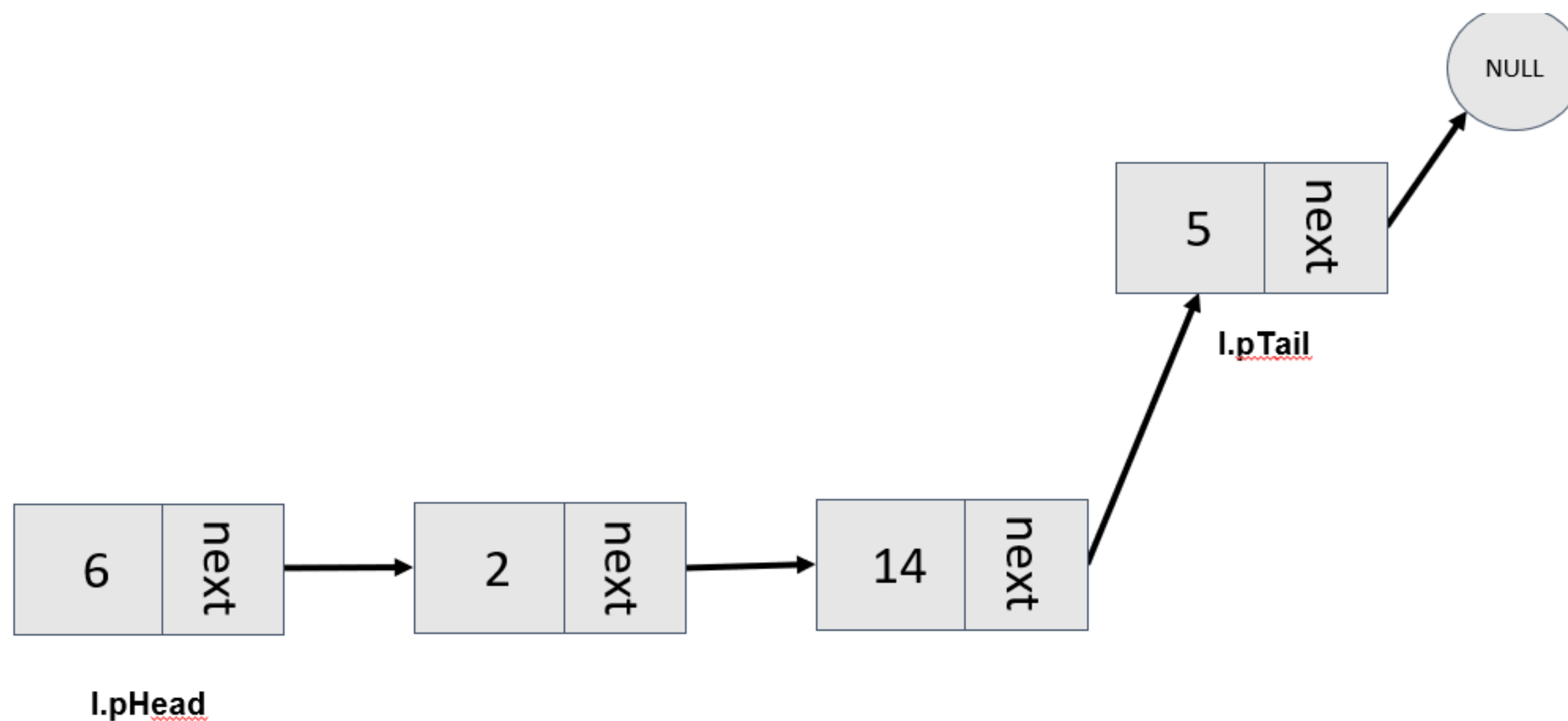
Sharing is learning

Thêm 1 node vào DSLKĐ

5b. Thêm node p vào cuối DSLKĐ (AddTail)



Sharing is learning

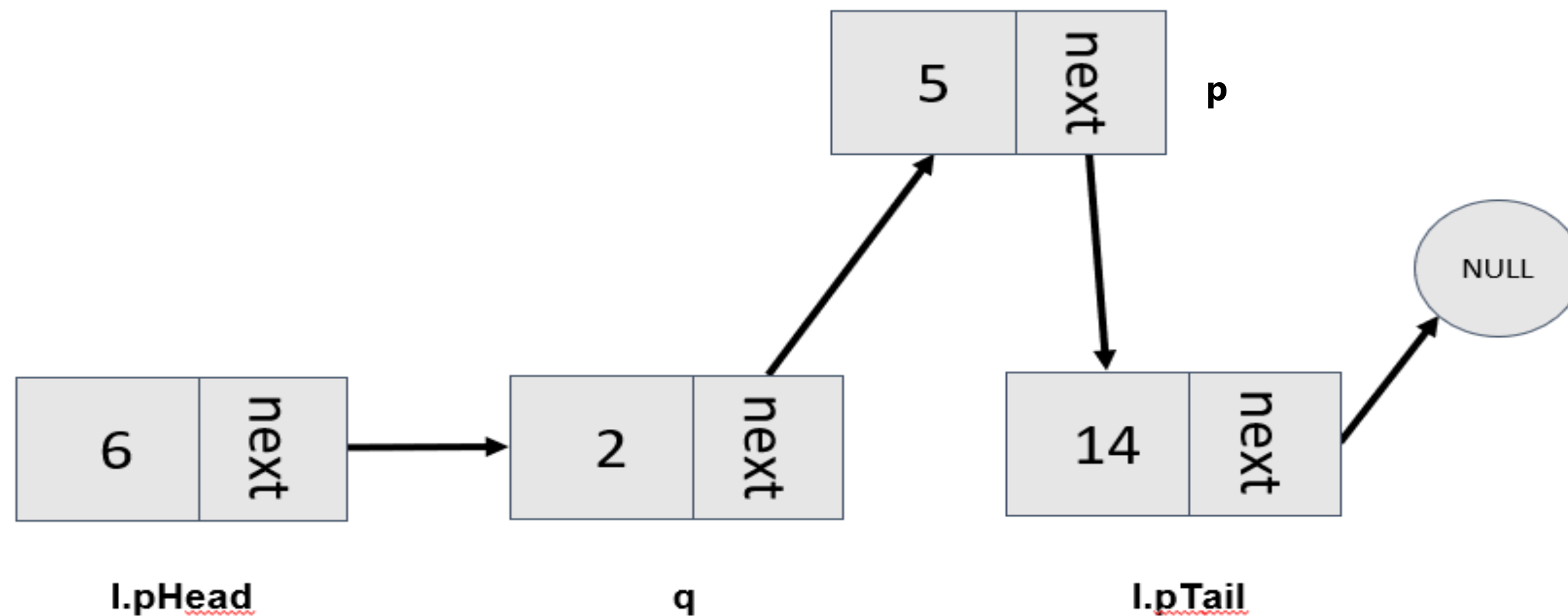


Sharing is learning

Thêm 1 node vào DSLKĐ

5c. Thêm node p vào sau node q trong DSLKĐ

- Nếu q là node cuối danh sách thì chỉ cần dùng hàm add tail để thêm node p vào danh sách
- Nếu không thì: Cho pnext của q trở tới p còn pnext của p trở tới con trỏ sau q.



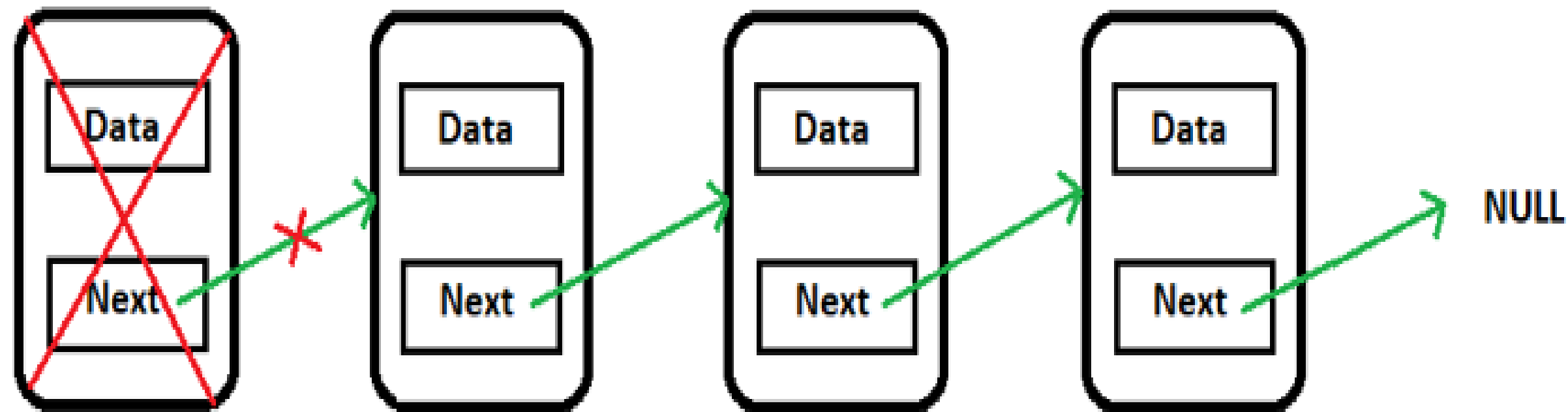
6. Xóa node trong danh sách

- Nguyên tắc: Phải cô lập node cần xóa trước khi xóa.
- Các vị trí cần xóa:
 - + Xóa node đứng đầu DSLK
 - + Xóa node đứng cuối DSLK
 - + Xóa node có khoá bằng x



Xóa 1 node trong DSLKĐ

6a. Xóa 1 node p đầu DSLK



codelearn.io



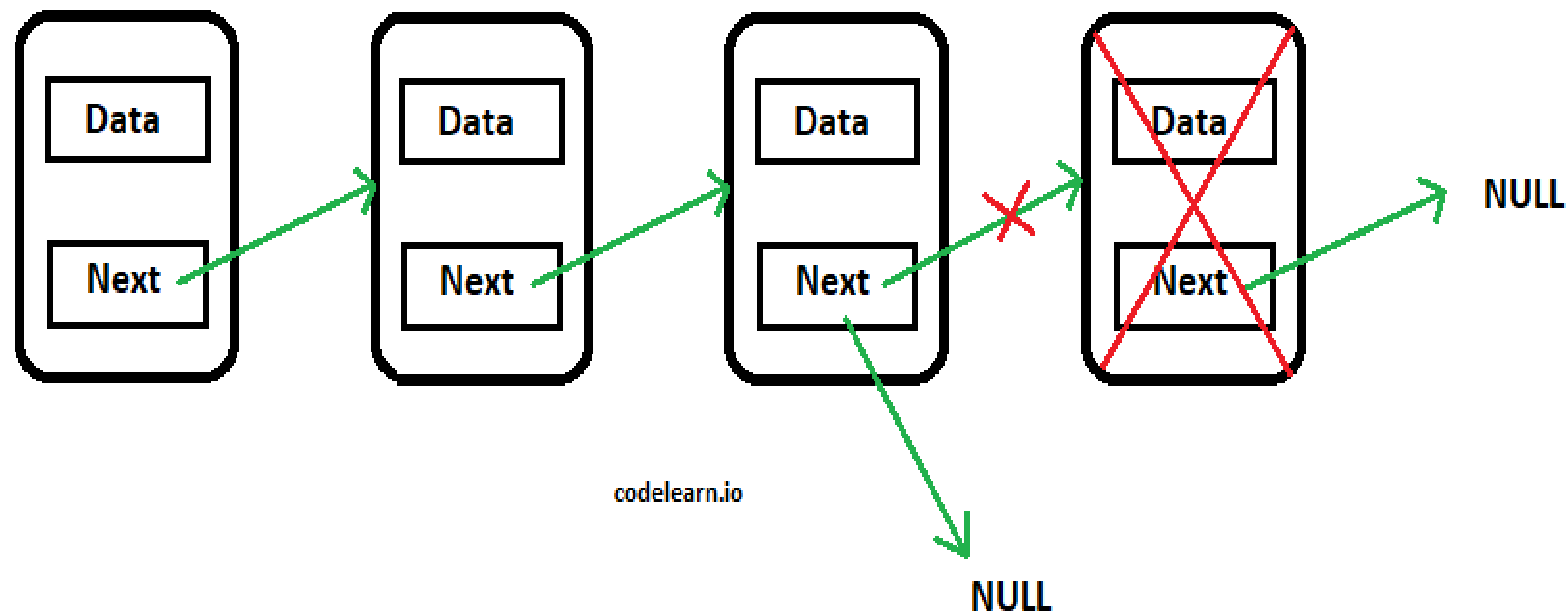
Sharing is learning



Sharing is learning

Xóa 1 node trong DSLKĐ

6b. Xóa 1 node p ở cuối DSLK



Sharing is learning



Sharing is learning

Một số code tham khảo



Sharing is learning

```
struct List {  
    Node *head;  
    Node *tail;  
};
```

```
void Createlist(List& l) {  
    l.head = NULL;  
    l.tail = NULL;  
}
```

```
void AddHead(List& l, Node* p) {  
    if (l.head == NULL)  
    {  
        l.head = p;  
        l.tail = l.head;  
    }  
    else  
    {  
        p->next = l.head;  
        l.head = p;  
    }  
}
```

```
void AddTail(List& l, Node* p)  
{  
    if (l.head == NULL)  
    {  
        l.head = p;  
        l.tail = p;  
    }  
    else  
    {  
        l.tail->next = p;  
        l.tail = p;  
    }  
}
```

```
void InsertAfterq(List& l, Node* p, Node* q)  
{  
    if (q != NULL)  
    {  
        p->Next = q->Next;  
        q->Next = p;  
        if (l.Tail == q)  
            l.Tail = p;  
    }  
    else // list rỗng  
        AddHead(l, p);  
}
```


Stack và Queue



Sharing is learning



STACK

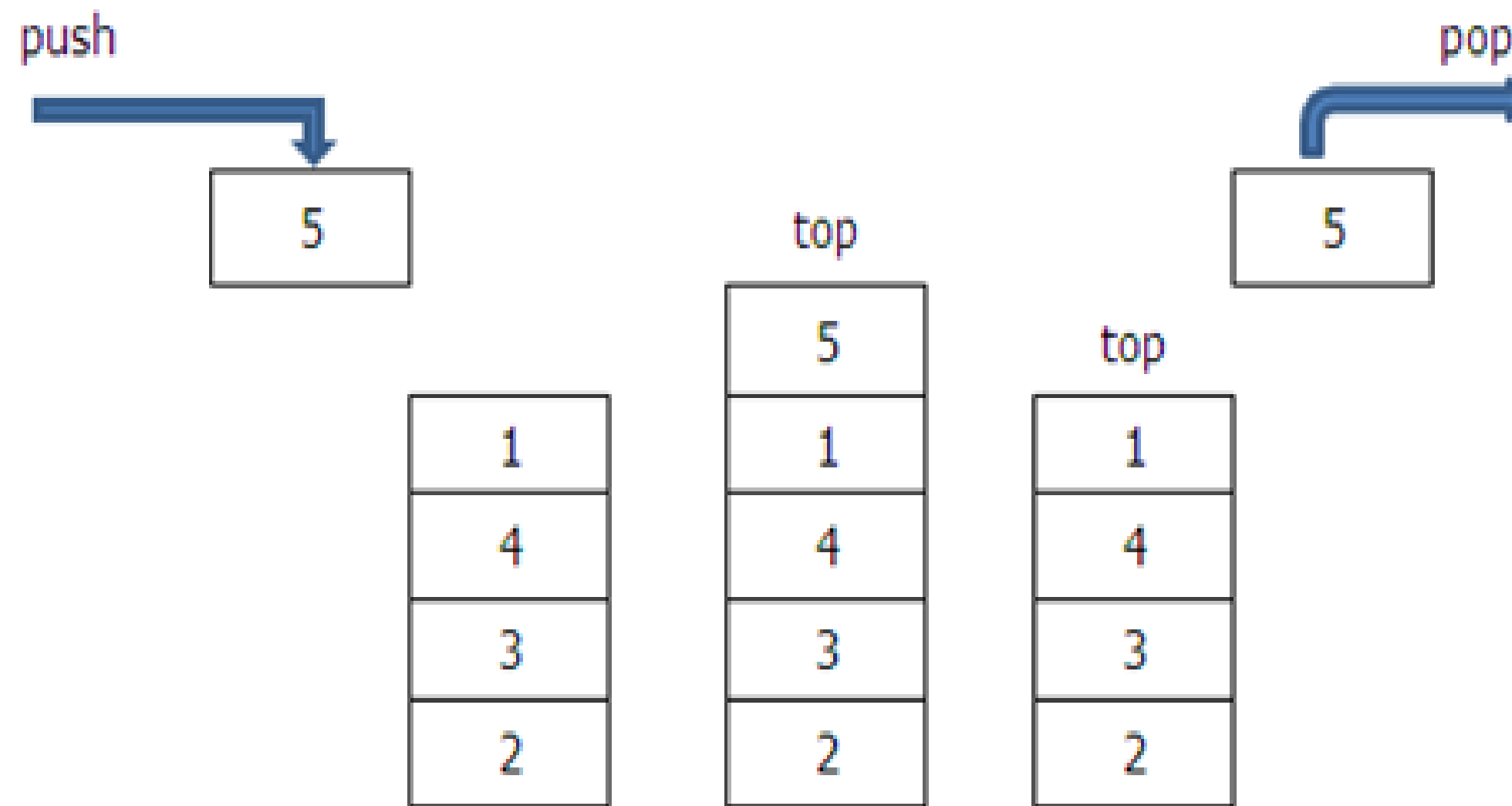


Sharing is learning

Stack

1. Cơ chế

LIFO (Last In First Out)



- **Chú ý:** Tất cả các thao tác với ngăn xếp chỉ tác động đến *phần tử ở đầu* ngăn xếp.



Sharing is learning



Sharing is learning

Stack



2. Các thao tác với stack

- **IsEmptyStack**: kiểm tra stack rỗng không.
- **IsFullStack** : kiểm tra stack đầy không.
- **Push**: thêm 1 phần tử vào **đỉnh** stack, có thể làm stack đầy.
- **Pop**: lấy ra 1 phần tử từ **đỉnh** stack, có thể làm stack rỗng.
- **Top**: kiểm tra phần tử đầu stack.

2. Cách cài đặt Stack

- Sử dụng mảng 1 chiều.
- Sử dụng DSLK đơn.

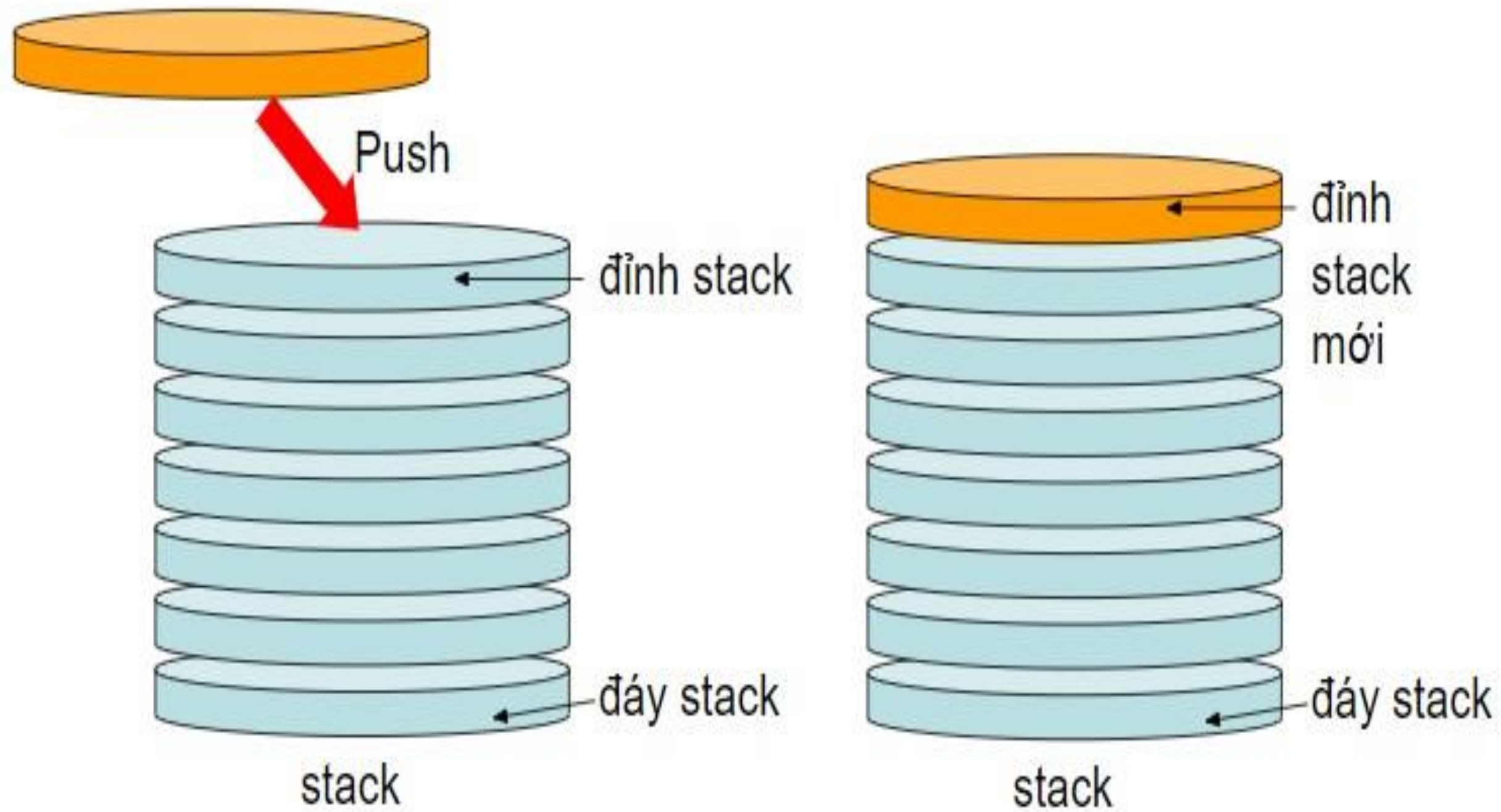


Sharing is learning

Push: Thêm 1 phần tử vào Stack



Sharing is learning

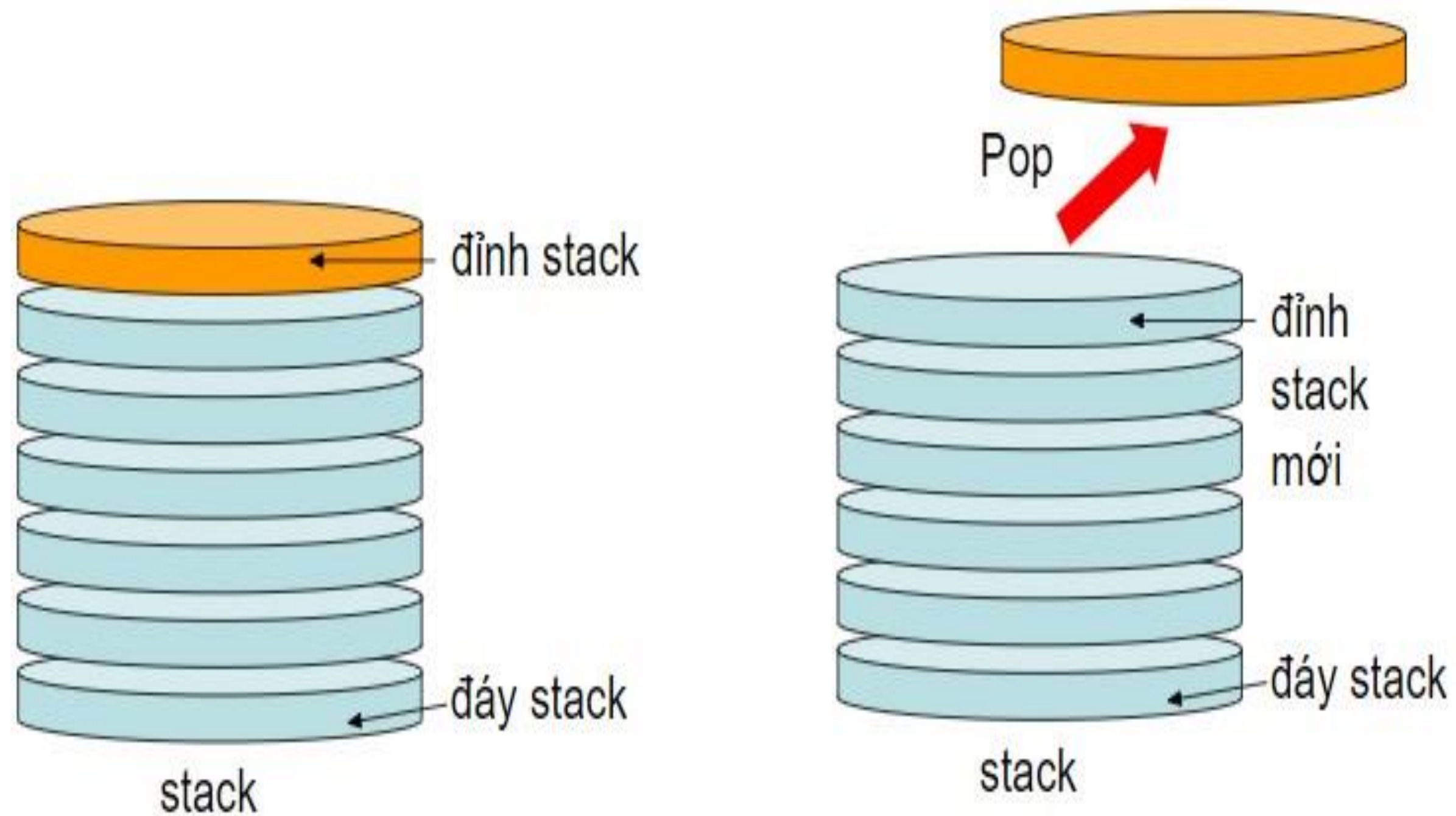


Sharing is learning

Pop: Lấy 1 phần tử từ Stack



Sharing is learning



Sharing is learning

Stack



Sharing is learning

Một số code tham khảo:

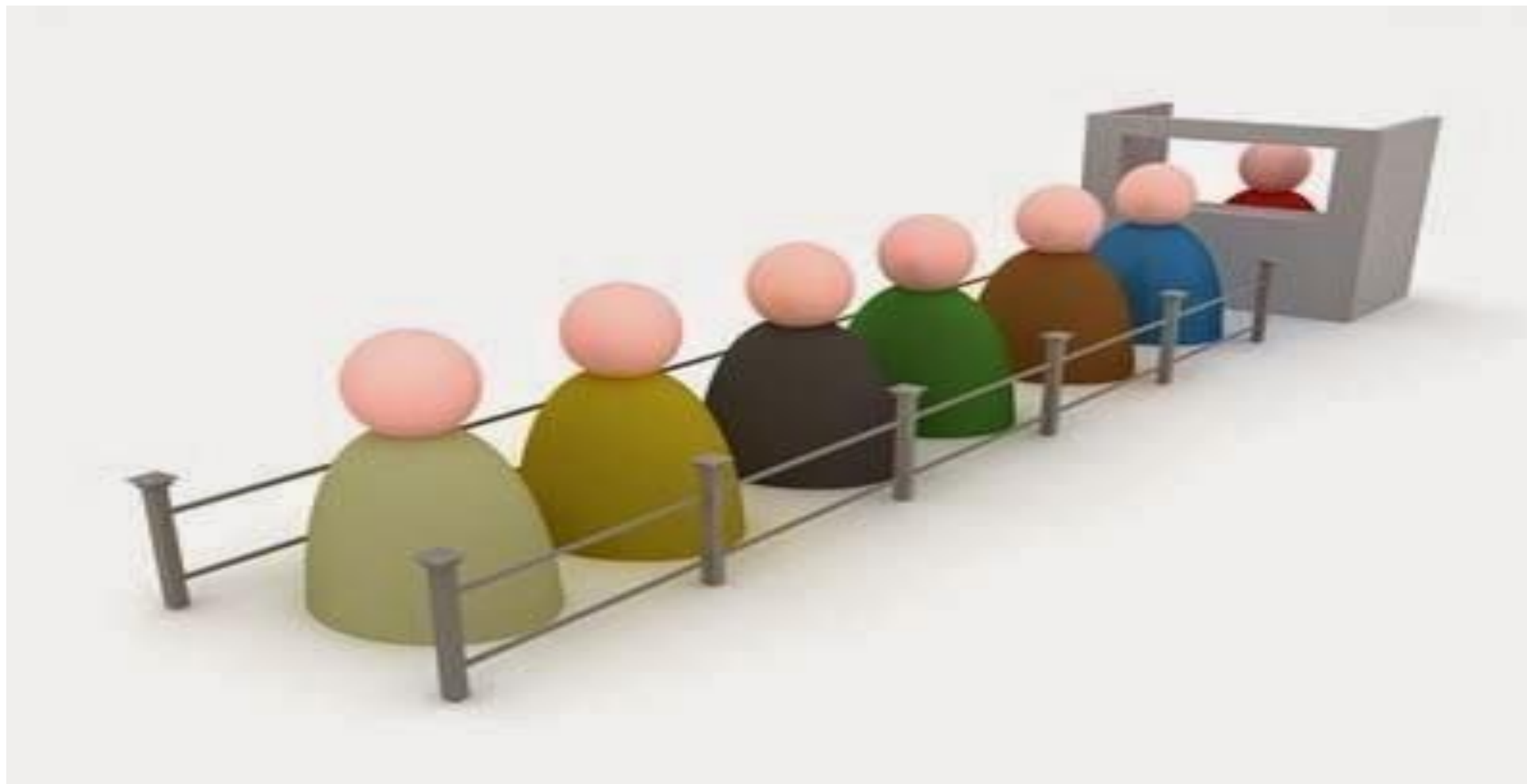
```
19 struct Node
20 {
21     int n;
22     Node* pNext;
23 };
24 struct Stack
25 {
26     Node* pHead;
27 };
29 void CreateStack(Stack& s)
30 {
31     s.pHead = NULL;
32 }
34 bool IsEmpty(Stack& s)
35 {
36     if (s.pHead == NULL)
37         return true;
38     else
39         return false;
```

```
42 bool pop(Stack& s, int& a)
43 {
44     if (IsEmpty(s))
45         return false;
46     Node* p = s.pHead;
47     a = p->n;
48     s.pHead = s.pHead->pNext;
49     delete p;
50     p = NULL;
51     return true;
52 }
53
54 void push(Stack& s, int a)
55 {
56     Node* p = new Node;
57     p->n = a;
58     p->pNext = NULL;
59     if (s.pHead == NULL)
60     {
61         s.pHead = p;
62     }
63     else
64     {
65         p->pNext = s.pHead;
66         s.pHead = p;
67     }
68 }
```


Queue

1. Cơ chế

FIFO (First In First Out)



Chú ý: Tất cả các thao tác với hàng đợi sẽ tác động đến *phần tử đầu và cuối* của hàng đợi.



Sharing is learning



Sharing is learning

Queue



Sharing is learning

2. Các thao tác trên Queue

- **EnQueue(O)**: Thêm đối tượng O vào cuối hàng đợi.
- **DeQueue()**: Lấy đối tượng ở đầu hàng đợi
- **isEmpty()**: Kiểm tra xem hàng đợi có rỗng hay không?
- **Front()**: Trả về giá trị của phần tử nằm đầu hàng đợi mà không hủy nó

3. Cách cài đặt Queue

- Sử dụng mảng 1 chiều.
- Sử dụng DSLK đơn.

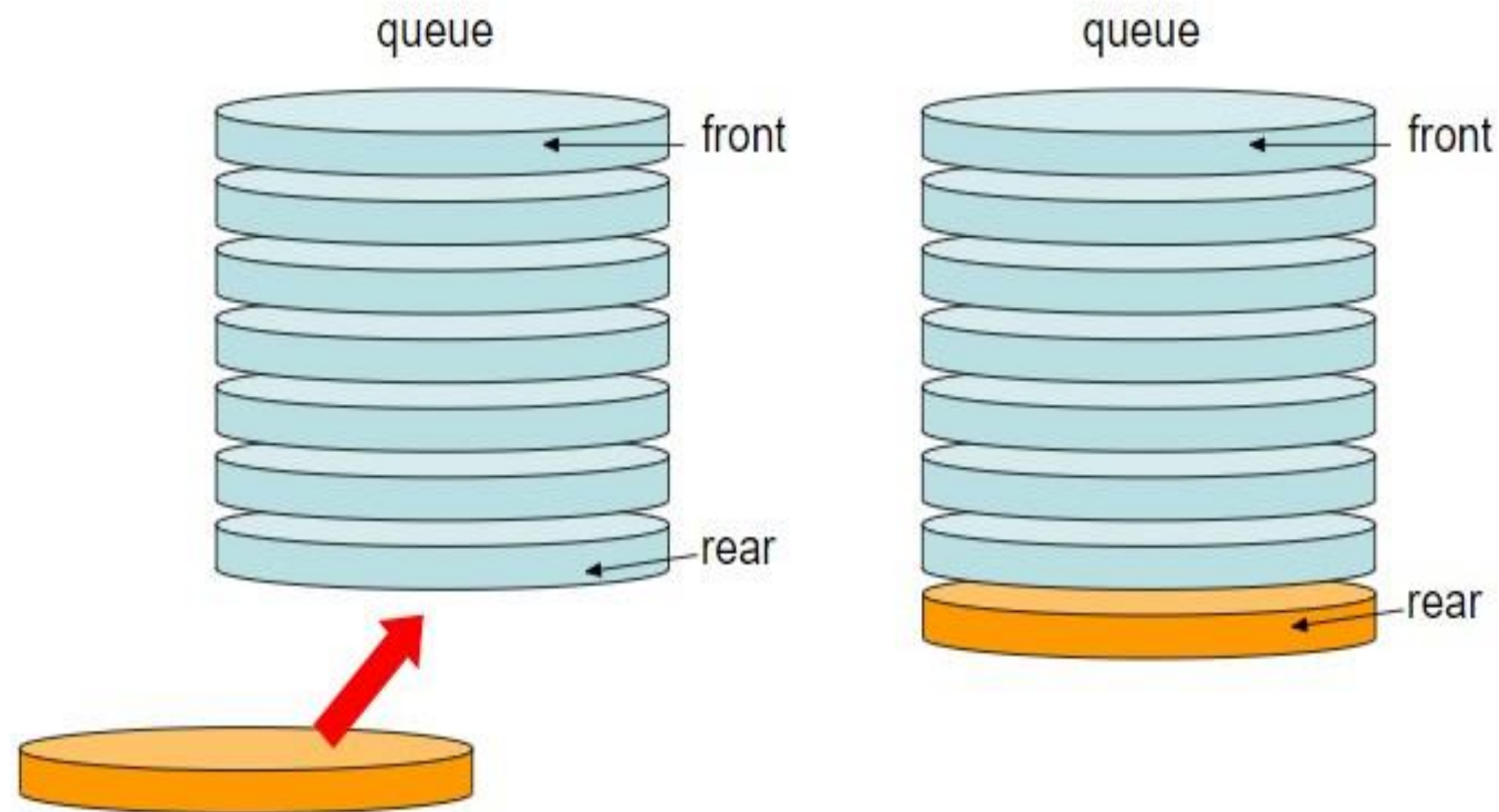


Sharing is learning

Enqueue: Thêm vào cuối Queue



Sharing is learning

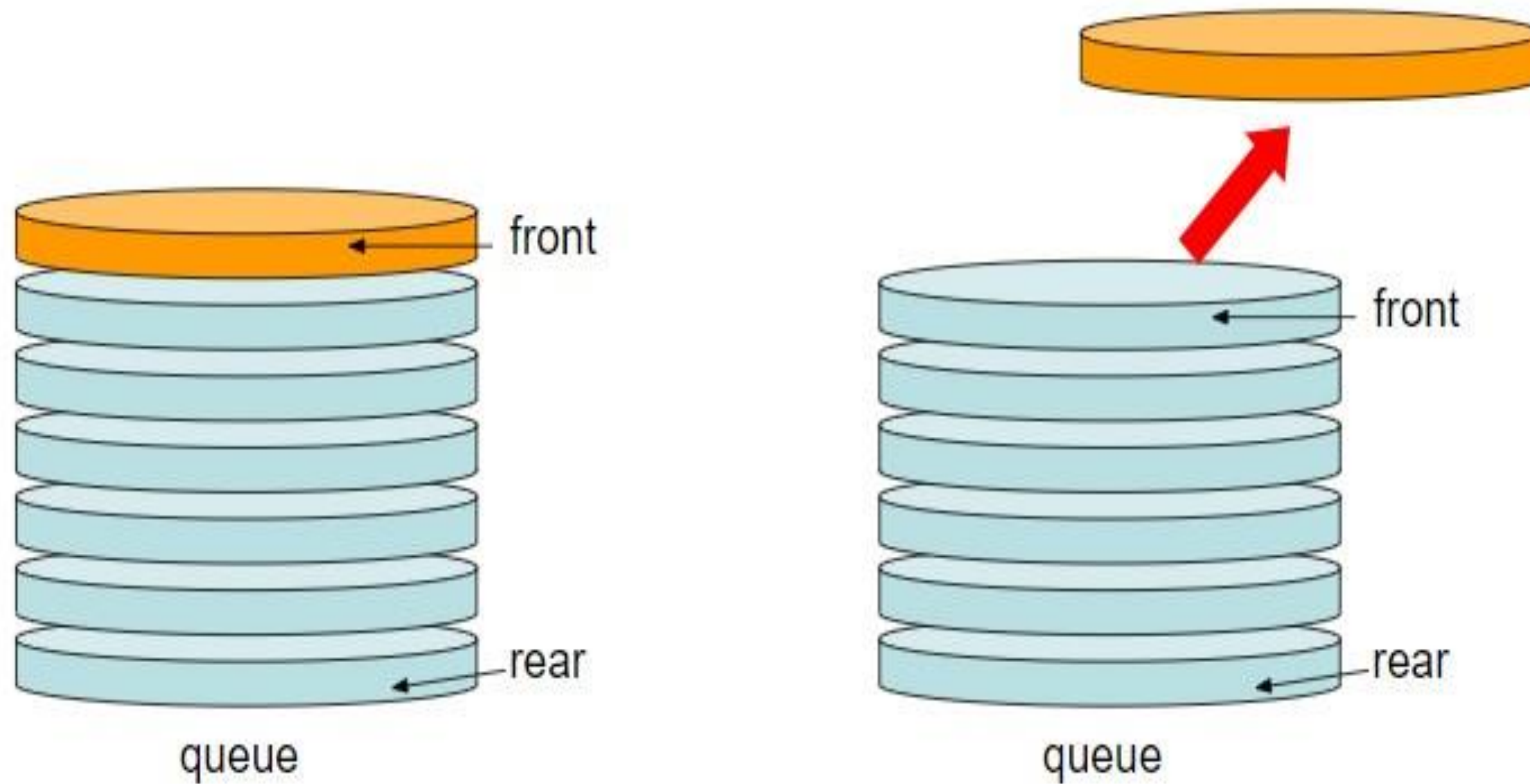


Sharing is learning

Deque: Lấy ở đầu Queue



Sharing is learning



Sharing is learning

Queue



Một số code tham khảo:

```
11 struct node {
12     int n;
13     node* pNext;
14 };
15 struct Queue {
16     node* pHead;
17     node* pTail;
18 };
19
56 void CreateQueue(Queue& l)
57 {
58     l.pHead = NULL;
59     l.pTail = NULL;
60 }
61
81 bool isEmpty(Queue l)
82 {
83     if (l.pHead == NULL)
84         return true;
85     else
86         return false;
87 }
```

```
88 bool dequeue(Queue& l, int &x)
89 {
90     if (isEmpty(l))
91         return false;
92     node* p = l.pHead;
93     x = p->n;
94     l.pHead = p->pNext;
95     delete p;
96     p = NULL;
97     return true;
98 }
99 void enqueue(Queue& l, int x)
100 {
101     node* tem = new node;
102     tem->n = x;
103     tem->pNext = NULL;
104     AddTail(l, tem);
105 }
```

Hàm AddTail của queue có cấu trúc tương tự với hàm AddTail của list

Nội dung Training



Sharing is learning

I. Danh sách liên kết đơn, ngăn xếp, hàng đợi

II. Giải thuật tìm kiếm.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

1. Tìm kiếm tuyến tính (Linear search):

- I. Ý tưởng và minh họa.
- II. Mã nguồn.

2. Tìm kiếm tuyến tính cải tiến:

- I. Ý tưởng và minh họa.
- II. Mã nguồn.

3. Tìm kiếm nhị phân (Binary search):

- I. Ý tưởng và minh họa.
- II. Mã nguồn.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

1. Tìm kiếm tuyến tính (Linear search):

- I. Ý tưởng và minh họa.*
- II. Mã nguồn.*



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

1. Tìm kiếm tuyến tính (Linear search):

- Ý tưởng: Duyệt từ phần tử đầu tiên cho đến khi **tìm được phần tử cần tìm** và **trả về vị trí xuất hiện phần tử đó**. Nếu **không tìm thấy** trả về **-1**.
- Áp dụng cho dãy có **hoặc không có thứ tự**.
- Độ phức tạp thuật toán là **$O(n)$** .



Sharing is learning

Giải thuật tìm kiếm



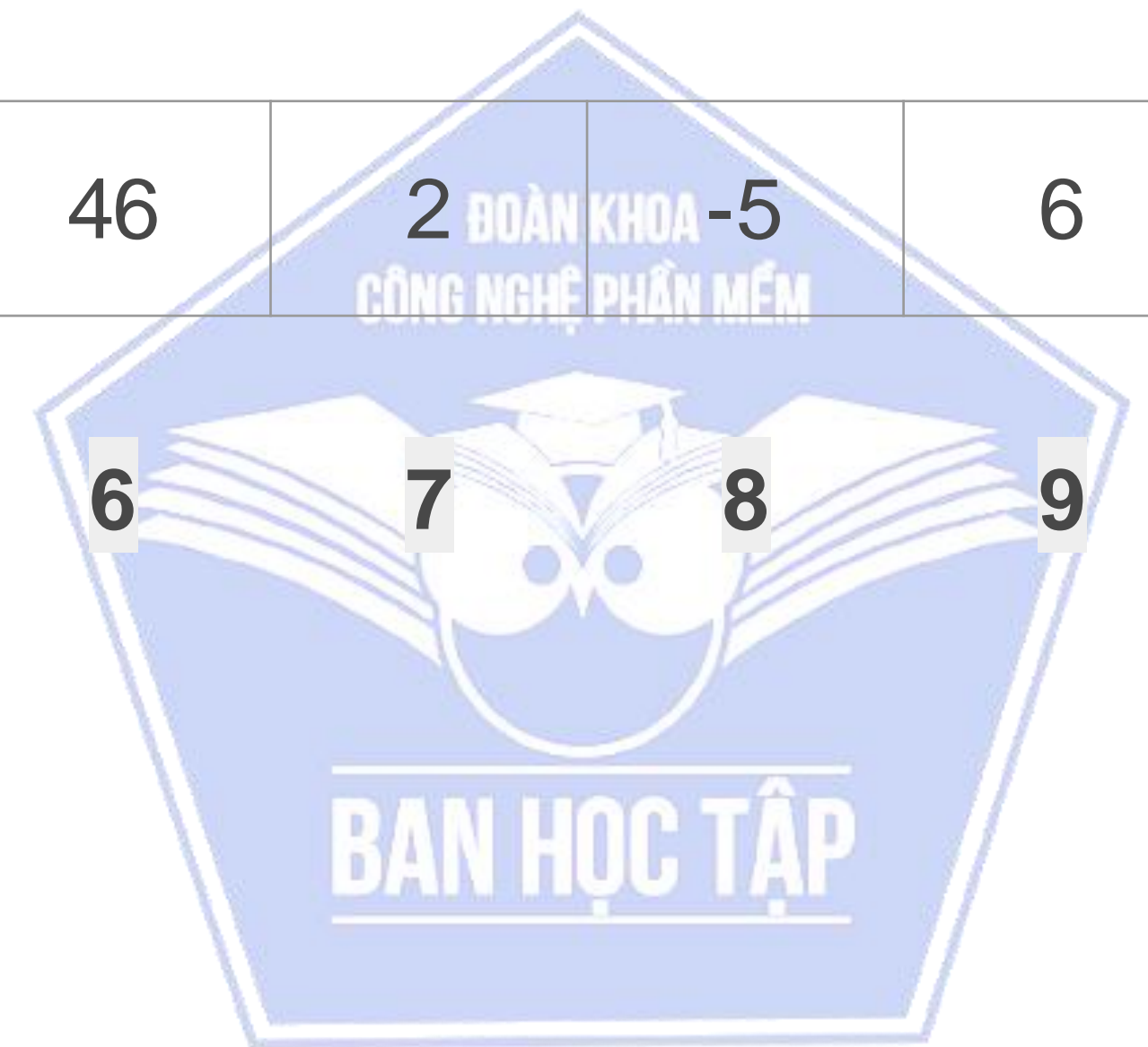
Sharing is learning

1. Tìm kiếm tuyến tính (Linear search):

Ví dụ minh họa:

A:	-2	3	1	5	2	0	46	2	-5	6
VT:	0	1	2	3	4	5	6	7	8	9

- Tìm $x = 5$ ---> Trả về vị trí 3.
- Tìm $x = 2$ ---> Trả về vị trí 4.
- Tìm $x = -1$ ---> Trả về -1.



Sharing is learning

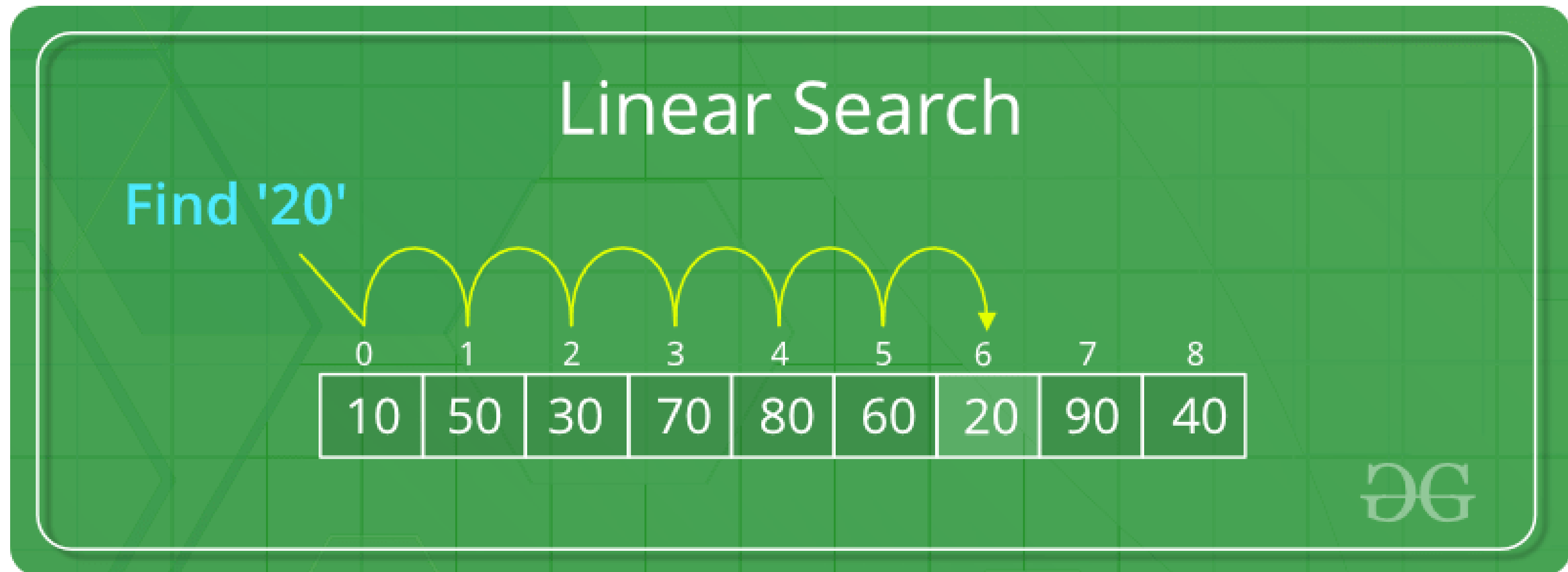
Giải thuật tìm kiếm



Sharing is learning

1. Tìm kiếm tuyến tính (Linear search):

Ví dụ minh họa:



Giải thuật tìm kiếm



Sharing is learning

1. Tìm kiếm tuyến tính (Linear search):

Các bước tiến hành:

- Bước 1: Khởi tạo và gán $i=0$
- Bước 2: So sánh $a[i]$ với giá trị x cần tìm, có 2 khả năng:
 - $A[i] = x$, tìm thấy x , dừng.
 - $A[i] \neq x$, chuyển sang bước 3.
- Bước 3: $i=i+1$, xét tiếp phần tử kế tiếp trong mảng. Nếu $i=N$ (hết mảng), dừng. Ngược lại: lặp lại bước 2.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

1. Tìm kiếm tuyến tính (Linear search):

❖ Mã nguồn:

```
1  int linearSearch(int A[], int n, int x)
2  {
3      for (int i = 0; i < n; i++)
4      {
5          if (A[i] == x)
6              return i;
7      }
8      return -1;
9  }
```



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

1. Tìm kiếm tuyến tính (Linear search):

- Best case: $O(1)$
- Worst case: $O(n)$
- Average case: $O(n)$



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm tuyến tính cải tiến:

- I. Ý tưởng và minh họa.*
- II. Mã nguồn.*



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm tuyến tính cải tiến:

- Số phép so sánh của thuật toán tìm kiếm tuyến tính trong trường hợp xấu nhất là $2n$.
- Để giảm thiểu số phép so sánh trong vòng lặp cho thuật toán, ta có thể thêm phần tử "lính canh" vào cuối dãy.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm tuyến tính cải tiến:

```
5  int linearSearchUpdated(int a[], int n, int x)
6  {
7      int i = 0;
8      a[n] = x; //a[n] là phần tử "lính canh"
9      while (a[i] != x)
10     {
11         i++;
12     }
13     if (i == n)
14     {
15         return 0; //không tìm thấy x
16     }
17     else
18     {
19         return 1; //tìm thấy x
20     }
21 }
22
```



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

3. Tìm kiếm nhị phân (Binary search):

- I. Ý tưởng và minh họa.*
- II. Mã nguồn.*



Sharing is learning

Giải thuật tìm kiếm



2. Tìm kiếm nhị phân (Binary search):

Binary search

steps: 0



Sequential search

steps: 0





2. Tìm kiếm nhị phân (Binary search):

- **Áp dụng cho dãy đã có thứ tự (Tăng dần/giảm dần).**
- Độ phức tạp thuật toán là **$O(\log(n))$.**

Qui ước: ta viết $\log_2(n)$ là $\log(n)$.



2. Tìm kiếm nhị phân (Binary search):

Giả sử ta có mảng tăng dần, ta có ý tưởng thuật toán như sau:

Xét đoạn mảng **arr[left...right]** cần tìm kiếm phần tử x . Ta so sánh x với phần tử ở vị trí giữa của mảng (**mid = (left + right)/2**):

- Nếu phần tử **arr[mid] = x**. Kết luận và thoát chương trình
- Nếu **arr[mid] < x**. Chỉ thực hiện tìm kiếm trên đoạn **arr[mid+1...right]**
- Nếu **arr[mid] > x**. Chỉ thực hiện tìm kiếm trên đoạn **arr[left...mid-1]**



2. Tìm kiếm nhị phân (Binary search):

Các bước tiến hành:

- Bước 1: $\text{left} = 1$; $\text{right} = N - 1$;
- Bước 2:
 - ✓ $\text{Mid} = (\text{left} + \text{right}) / 2$
 - ✓ So sánh $A[\text{mid}]$ với X , có 3 khả năng:
 - $A[\text{mid}] = x$: tìm thấy
 - $A[\text{mid}] > x$: $\text{right} = \text{mid} - 1$;
 - $A[\text{mid}] < x$: $\text{left} = \text{mid} + 1$;
- Bước 3: nếu $\text{left} \leq \text{right}$, tức là còn phần tử trong dãy hiện hành, tiếp tục lặp lại bước 2. Ngược lại: dừng, kết thúc giải thuật.



Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa:

A:	-95	-10	0	1	9	15	46	53	98	911
VT:	0	1	2	3	4	5	6	7	8	9

Tìm $x = -10$?

$x = 911$?

$x = 3$?



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

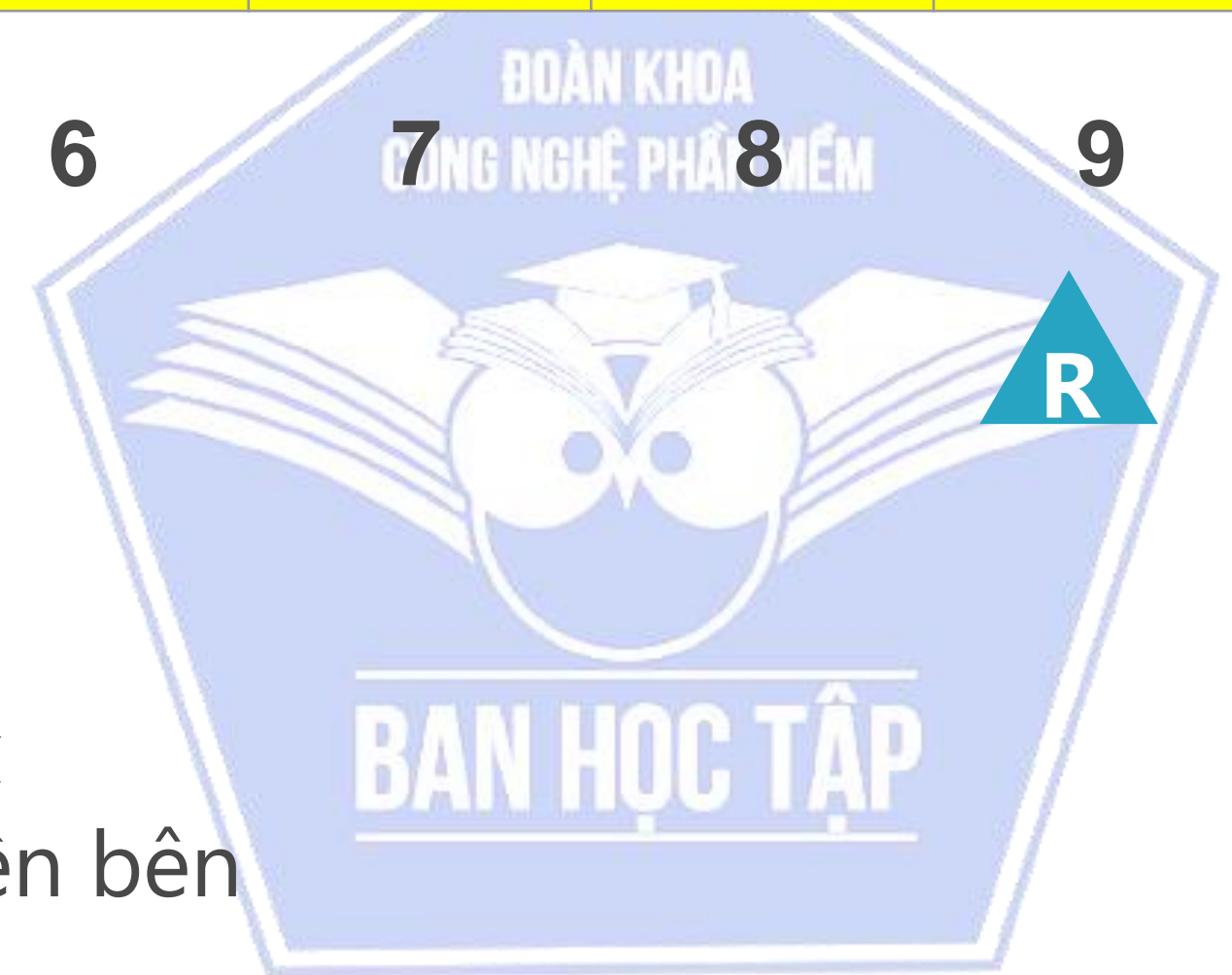
2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa: tìm $x = -10$

A:	-95	-10	0	1	9	15	46	53	98	911
VT:	0	1	2	3	4	5	6	7	8	9
	L				Mid					R

- $L = 0$
- $R = 9$
- $Mid = (L + R) / 2 = 4$

- $A[Mid] = 9 \rightarrow A[Mid] \neq x$
- $x < A[Mid] \rightarrow x$ thuộc miền bên trái mảng A.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa: tìm $x = -10$

A:	-95	-10	0	1	9	15	46	53	98	911
VT:	0	1	2	3	4	5	6	7	8	9
	L	Mid		R						

- $L = 0$
- $R = 3$
- $Mid = (L + R) / 2 = 1$

➤ $A[Mid] = -10 \rightarrow A[Mid] = x$

\Rightarrow Đã tìm thấy x ở vị trí thứ 1 trong mảng A.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

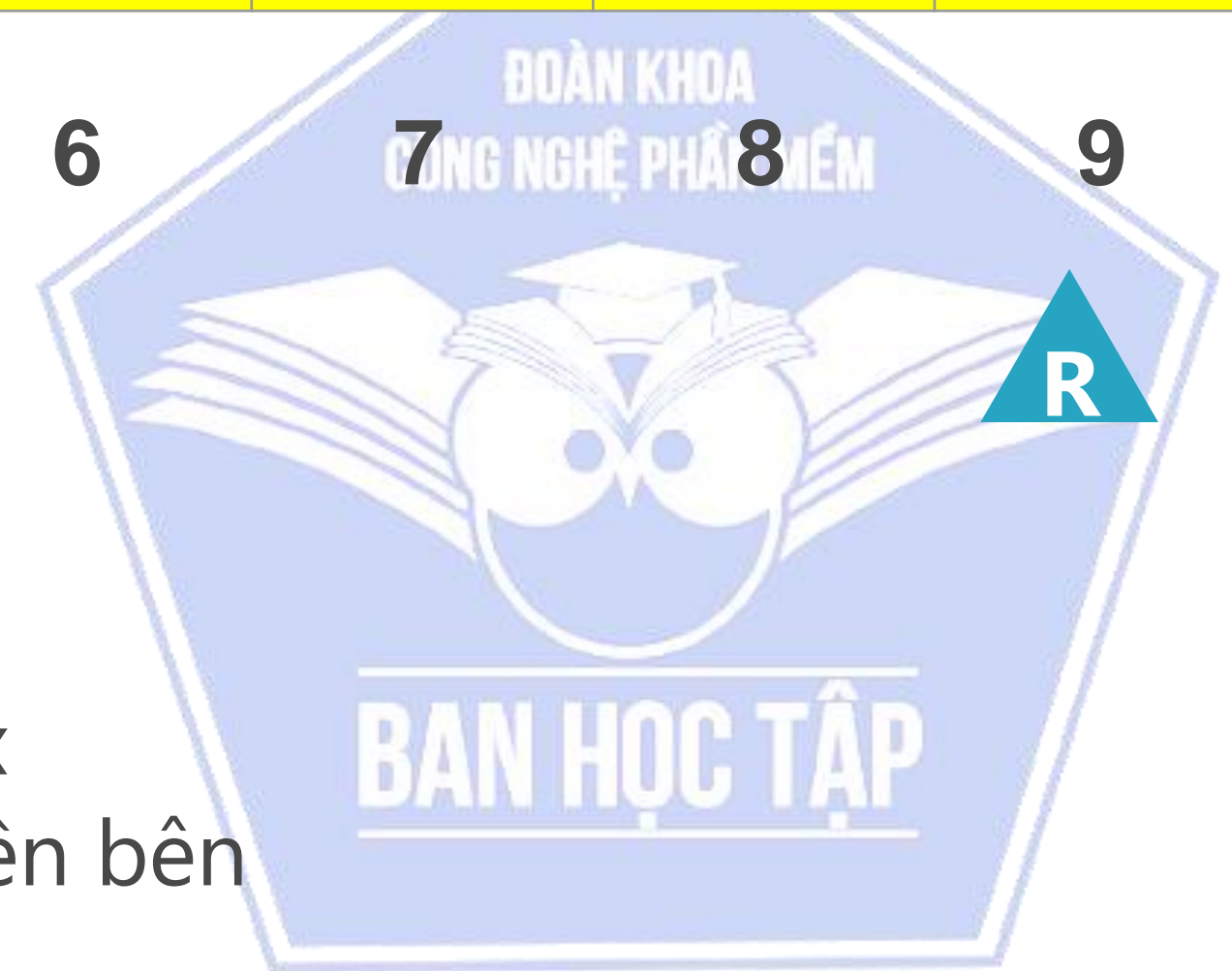
2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa: tìm $x = 911$

A:	-95	-10	0	1	9	15	46	53	98	911
VT:	0	1	2	3	4	5	6	7	8	9
	L				Mid					R

- $L = 0$
- $R = 9$
- $Mid = (L + R) / 2 = 4$

- $A[Mid] = 9 \rightarrow A[Mid] \neq x$
- $x > A[Mid] \rightarrow x$ thuộc miền bên phải mảng A.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa: tìm $x = 911$

A:	-95	-10	0	1	9	15	46	53	98	911
VT:	0	1	2	3	4	5	6	7	8	9
						L		Mid		R

- $L = 5$
- $R = 9$
- $Mid = (L + R) / 2 = 7$

- $A[Mid] = 53 \rightarrow A[Mid] \neq x$
- $x > A[Mid] \rightarrow x$ thuộc miền bên phải mảng A.

Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa: tìm $x = 911$

A:

-95

-10

0

1

9

15

46

53

98

911

VT:

0

1

2

3

4

5

6

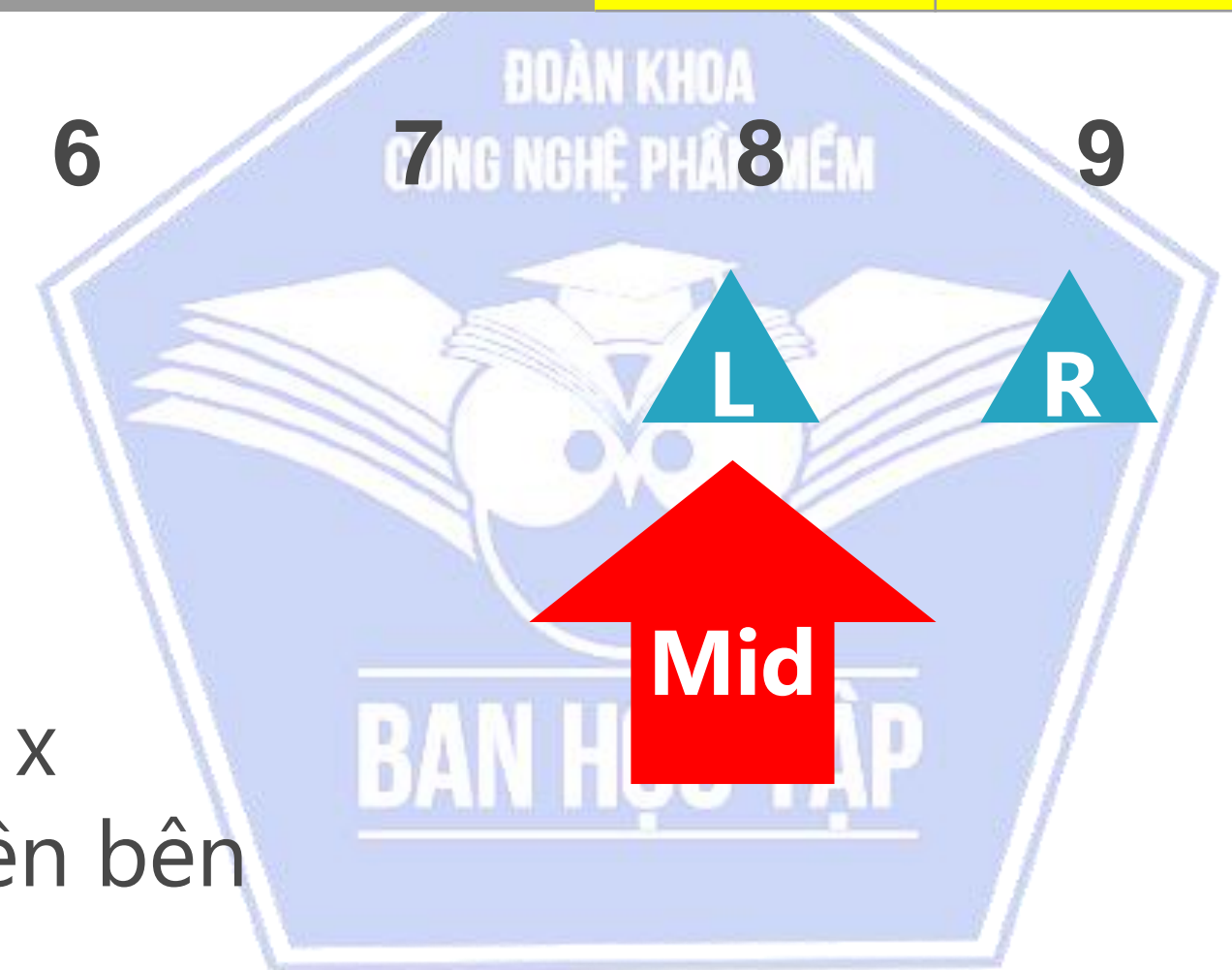
7

8

9

- $L = 8$
- $R = 9$
- $Mid = (L + R) / 2 = 8$

- $A[Mid] = 98 \rightarrow A[Mid] \neq x$
- $x > A[Mid] \rightarrow x$ thuộc miền bên phải mảng A.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa: tìm $x = 911$

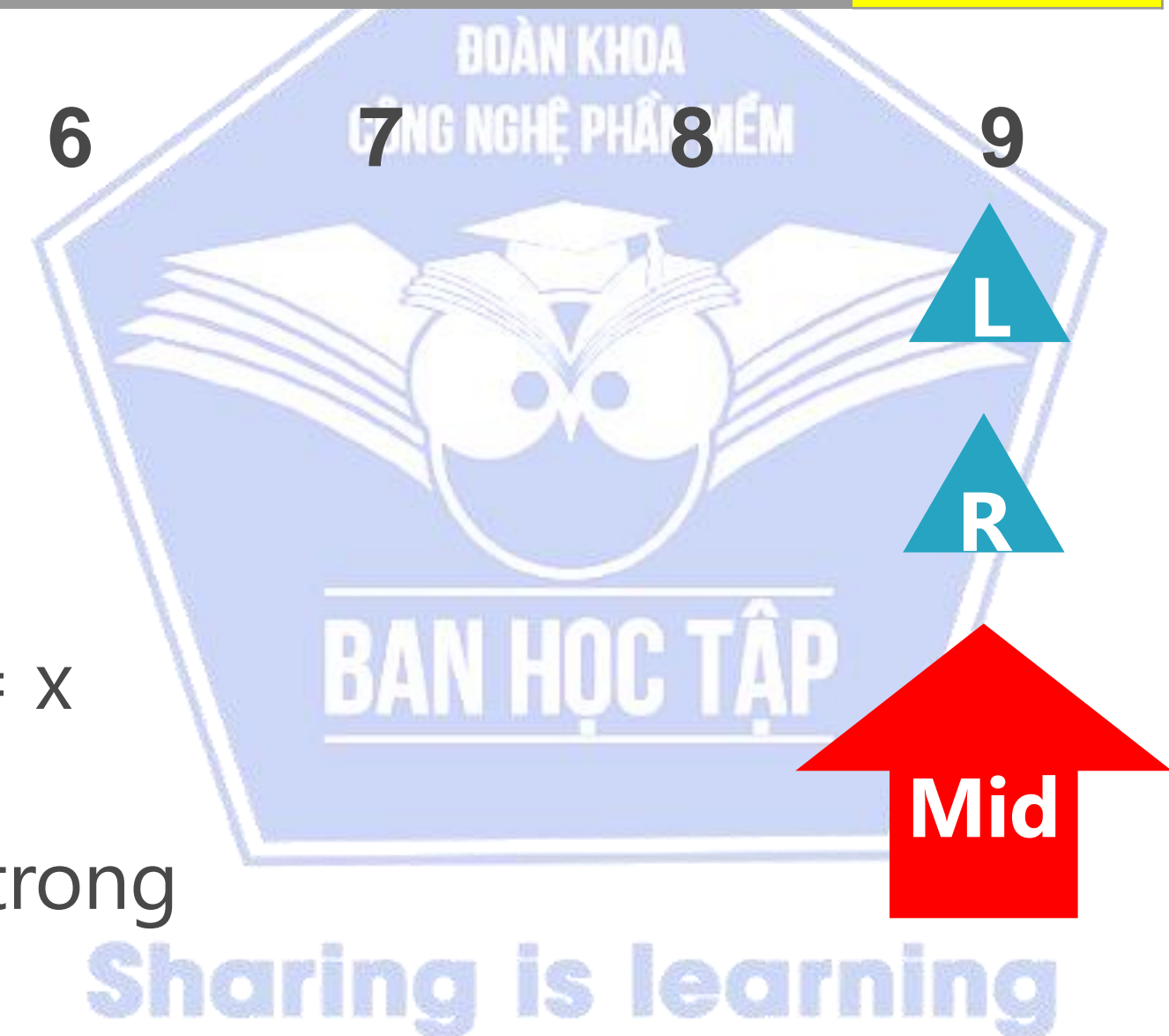
A: -95 -10 0 1 9 15 46 53 98 **911**

VT: 0 1 2 3 4 5 6 7 8 9

- $L = 9$
- $R = 9$
- $Mid = (L + R) / 2 = 9$

➤ $A[Mid] = 911 \rightarrow A[Mid] = x$

\Rightarrow Đã tìm thấy x ở vị trí 9 trong mảng A.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

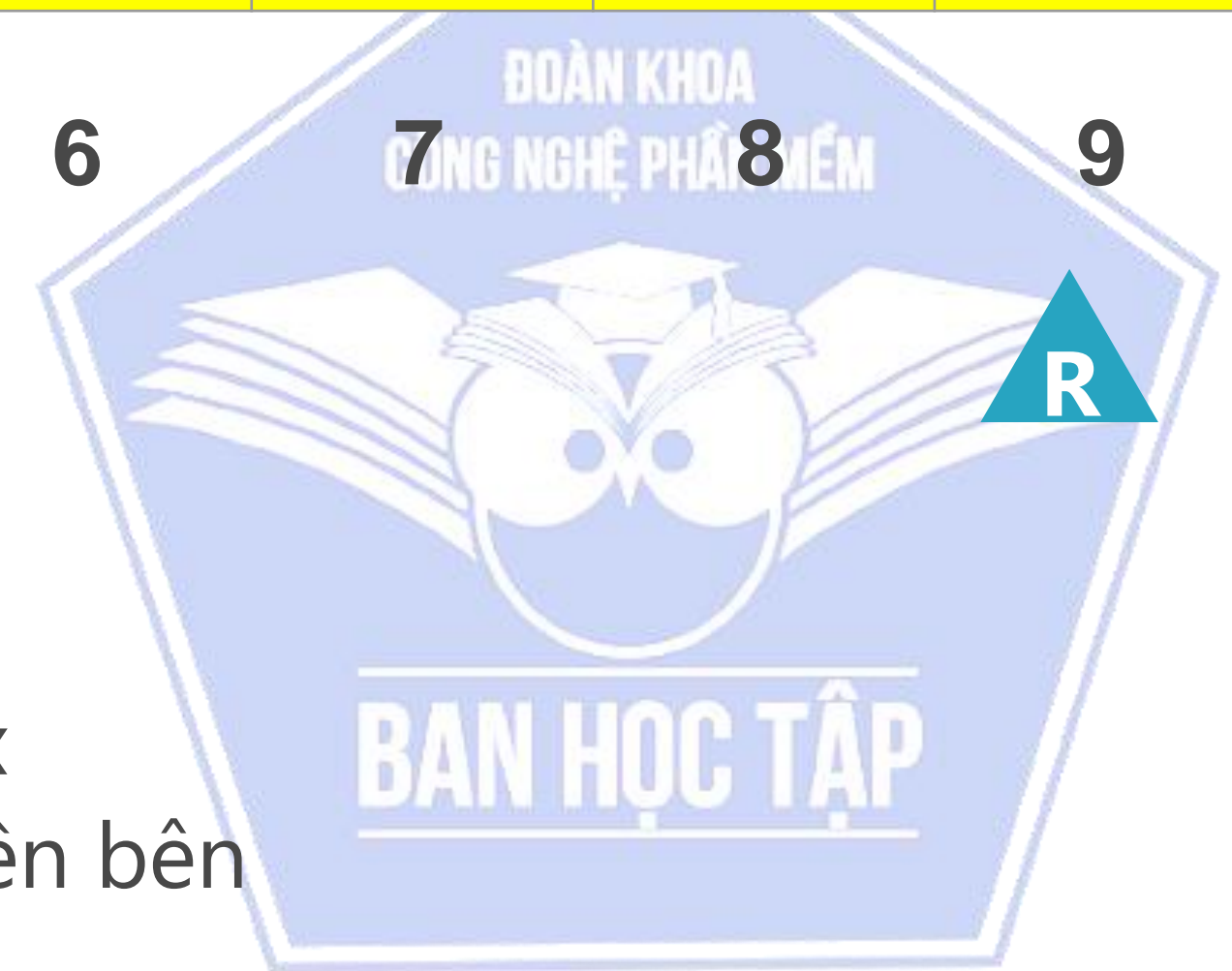
2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa: tìm $x = 3$

A:	-95	-10	0	1	9	15	46	53	98	911
VT:	0	1	2	3	4	5	6	7	8	9
	L				Mid					R

- $L = 0$
- $R = 9$
- $Mid = (L + R) / 2 = 4$

- $A[Mid] = 9 \rightarrow A[Mid] \neq x$
- $x < A[Mid] \rightarrow x$ thuộc miền bên trái mảng A.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa: tìm $x = 3$

A:	-95	-10	0	1	9	15	46	53	98	911
VT:	0	1	2	3	4	5	6	7	8	9
	L	Mid		R						

- $L = 0$
- $R = 3$
- $Mid = (L + R) / 2 = 1$

- $A[Mid] = -10 \rightarrow A[Mid] \neq x$
- $x > A[Mid] \rightarrow x$ thuộc miền bên phải mảng A.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa: tìm $x = 3$

A:	-95	-10	0	1	9	15	46	53	98	911
VT:	0	1	2	3	4	5	6	7	8	9
			L	R						

➤ $L = 2$

➤ $R = 3$

➤ $Mid = (L + R) / 2 = 2$

Mid

• $A[Mid] = 2 \rightarrow A[Mid] \neq x$

➤ $x > A[Mid] \rightarrow x$ thuộc miền bên phải mảng A.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa: tìm $x = 3$

A: -95 -10 0 1 9 15 46 53 98 911

VT: 0 1 2 3 4 5 6 7 8 9

L

R

Mid

- $L = 3$
- $R = 3$
- $Mid = (L + R) / 2 = 3$

- $A[Mid] = 1 \rightarrow A[Mid] \neq x$
- $x > A[Mid] \rightarrow x$ thuộc miền bên phải mảng A.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm nhị phân (Binary search):

Ví dụ minh họa: tìm $x = 3$

A: -95 -10 0 1 9 15 46 53 98 911

VT: 0 1 2 3 4 5 6 7 8 9



❖ $L > R \implies$ Không thu được miền nào
 \implies Kết luận không tìm thấy x trong mảng A.



Sharing is learning

Giải thuật tìm kiếm



Sharing is learning

2. Tìm kiếm nhị phân (Binary search):

❖ Mã nguồn:

```
1  int binarySearch(int A[], int n, int x)
2  {
3      int L = 0, R = n - 1;
4      while (L <= R)
5      {
6          int Mid = (L + R) / 2;
7          if (A[Mid] == x)
8              return Mid;
9
10         if (x > A[Mid])
11             L = Mid + 1;
12         else
13             R = Mid - 1;
14     }
15     return -1;
16 }
```



Sharing is learning

Giải thuật tìm kiếm

2. Tìm kiếm nhị phân (Binary search):

- Best case: **$O(1)$**
- Worst case: **$O(\log n)$**
- Average case: **$O(\log n)$**



Sharing is learning



Sharing is learning

Đề thi mẫu



Sharing is learning

Câu 1:

- Hãy trình bày ý tưởng của giải thuật tìm kiếm tuyến tính và cho biết độ phức tạp của giải thuật (1 điểm)
- Trình bày các bước (vẽ từng bước) giải thuật tìm kiếm tuyến tính thực hiện tìm giá trị $X=5$ trong mảng 6 số nguyên có giá trị: 81; 90; 62; 65; 12; 42 (1,5 điểm)

Câu 2: Người ta muốn lưu trữ danh sách hàng hóa tại công ty X với các thông tin chính yếu nhằm hỗ trợ nhanh trong tra cứu, với các thông tin: tên mặt hàng (chuỗi); giá mặt hàng (số nguyên); số lượng còn trong kho (số nguyên). Hãy thực hiện:

- Định nghĩa cấu trúc dữ liệu lưu danh sách các mặt hàng theo thông tin mô tả ở trên, sử dụng cấu trúc danh sách liên kết. (1 điểm)
- Viết hàm nhập vào danh sách 50 mặt hàng sử dụng cấu trúc dữ liệu ở câu 2a, biết rằng khi nhập lần lượt từng mặt hàng sẽ thêm vào cuối danh sách (1,5 điểm)
- Viết hàm nhập vào 2 số nguyên dương x, y ($x < y$), hiển thị lên màn hình danh sách mặt hàng có số lượng trong kho lớn hơn x và nhỏ hơn y . (1 điểm)

Đề thi mẫu



Sharing is learning

Câu 3: Hãy thực hiện chuyển đổi một số nguyên dương N ($N < 1000$) ở hệ thập phân sang biểu diễn ở hệ nhị phân (ví dụ: số 5 ở hệ thập phân sẽ là 101 ở hệ nhị phân), sử dụng cấu trúc ngăn xếp (stack), với các yêu cầu sau:

- Định nghĩa cấu trúc ngăn xếp để lưu trữ số nhị phân (1 điểm).
- Viết các hàm thao tác với cấu trúc ngăn xếp trong câu 3a: push; pop; kiểm tra stack rỗng; kiểm tra stack đầy (2 điểm).
- Viết hàm nhận đầu vào một số nguyên dương N ở hệ thập phân, chuyển đổi và hiển thị kết quả số N ở hệ nhị phân lên màn hình sử dụng cấu trúc, các hàm đã định nghĩa trong câu 3a, 3b (1 điểm)

Giải đề mẫu



Sharing is learning

Câu 1: (đề thi GK 2018-2019)

- a. Hãy trình bày ý tưởng của giải thuật tìm kiếm tuyến tính và cho biết độ phức tạp của giải thuật (1 điểm)

- **Ý tưởng giải thuật:** Xuất phát từ phần tử đầu tiên của mảng/danh sách A có N phần tử. So sánh phần tử X cần tìm với phần tử đang xét, nếu phần tử đang xét có giá trị bằng X thì thông báo có X trong mảng/danh sách và kết thúc giải thuật, ngược lại chuyển qua xem xét với phần tử tiếp theo. Khi xét đến phần tử cuối cùng nếu giá trị của phần tử này vẫn không bằng X thì thông báo không có X trong mảng/danh sách A và kết thúc giải thuật.

- **Độ phức tạp của giải thuật:** $O(N)$

Giải đề mẫu



Sharing is learning

Câu 1:

b. Trình bày các bước (vẽ từng bước) giải thuật tìm kiếm tuyến tính thực hiện tìm giá trị $X=5$ trong mảng 6 số nguyên có giá trị: 81; 90; 62; 65; 12; 42 (1,5 điểm)

A:	81	90	62	65	12	42
i:	0	1	2	3	4	5

Mảng A có $N=6$ phần tử, cần tìm $X=5$. Các bước giải thuật tìm kiếm như sau:

- *Bước 1:* $i=0$, so sánh $A[i] = A[0] = 81 \neq X = 5$, chuyển qua xem xét phần tử tiếp theo, $i=i+1$;
- *Bước 2:* $i=1$, so sánh $A[i] = A[1] = 90 \neq X = 5$, chuyển qua xem xét phần tử tiếp theo, $i=i+1$;

...

Tương tự với bước 3, 4 và 5.

- *Bước 6:* $i=5$, so sánh $A[i] = A[5] = 42 \neq X = 5$, đã xét xong phần tử cuối mảng, thông báo **không tìm thấy** X trong mảng, kết thúc giải thuật.

Đề thi mẫu



Bài tập tương tự: (trích câu 1 (5 điểm), đề thi GK 2019-2020)

a. Hãy hoàn thiện hàm tìm kiếm bên dưới: hàm này trả về vị trí của x trong mảng A.

- Nếu x xuất hiện ở nhiều vị trí trong mảng A, hàm **isAt** sẽ trả về vị trí sau cùng của x.

- Nếu không tìm thấy x trong mảng A, hàm **isAt** sẽ trả về giá trị -1.

Biết rằng mảng A[] là mảng các số nguyên dương, kết thúc bằng một phần tử có giá trị là -1, mảng có kích thước tối đa là 10000 phần tử.

```
int isAt(int A[], int x)
{
    ... /*insert code here*/
}
```

Ví dụ:

- cho mảng A[] = {1,7,6,5,6,8,-1} và x = 6, hàm isAt sẽ trả về 4.

- cho mảng A[] = {1,7,6,5,6,8,-1} và x = 9, hàm isAt sẽ trả về -1.

b. Hãy cho biết tên giải thuật đã sử dụng ở câu 1.a và ý tưởng của giải thuật.

Đề thi mẫu



Sharing is learning

Bài tập tương tự: (trích câu 1, đề thi GK 2019-2020)

a.

```
3  int isAt(int A[], int x)
4  {
5      int vitri = -1;
6      for (int i = 0; A[i] != -1; i++)
7      {
8          if (A[i] == x)
9              vitri = i;
10     }
11     return vitri;
```


Giải đề mẫu



Sharing is learning

Câu 2: Người ta muốn lưu trữ danh sách hàng hóa tại công ty X với các thông tin chính yếu nhằm hỗ trợ nhanh trong tra cứu, với các thông tin:

tên mặt hàng (chuỗi); giá mặt hàng (số nguyên); số lượng còn trong kho (số nguyên). Hãy thực hiện:

a. Định nghĩa cấu trúc dữ liệu lưu danh sách các mặt hàng theo thông tin mô tả ở trên, sử dụng cấu trúc danh sách liên kết. (1 điểm)

```
5 struct Node
6 {
7     char ten_mat_hang[100];
8     int gia_mat_hang;
9     int so_luong_con_trong_kho;
10    Node* pNext;
11 };
12
```

```
13 struct List
14 {
15     Node* pHead;
16     Node* pTail;
17 };
18
```

Giải đề mẫu



b. **Viết hàm nhập** vào danh sách 50 mặt hàng sử dụng cấu trúc dữ liệu động ở câu 2a, biết rằng khi nhập lần lượt từng mặt hàng sẽ **thêm vào cuối danh sách** (1,5 điểm)

```
28
29 void addTail(List& l, Node* p)
30 {
31     if (l.pHead == NULL)
32     {
33         l.pHead = p;
34         l.pTail = p;
35     }
36     else
37     {
38         l.pTail->pNext = p;
39         l.pTail = p;
40     }
41 }
```

Giải đề mẫu



b. **Viết hàm nhập** vào danh sách 50 mặt hàng sử dụng cấu trúc dữ liệu động ở câu 2a, biết rằng khi nhập lần lượt từng mặt hàng sẽ **thêm vào cuối danh sách** (1,5 điểm)

```
43 void enterList(List& l)
44 {
45     char s[100];
46     int gia, so_luong;
47     for (int i = 0; i < 50; i++)
48     {
49         cout << "Nhap ten mat hang: ";
50         cin.getline(s, 100);
51         cout << "\nNhap gia cua mat hang: ";
52         cin >> gia;
53         cout << "\nNhap so luong hang con trong kho: ";
54         cin >> so_luong;
55         Node *p = createNode(s, gia, so_luong);
56         addTail(l, p);
57     }
58 }
```


Giải đề mẫu



Sharing is learning

c. Viết hàm nhập vào 2 số nguyên dương x, y ($x < y$) và hiển thị lên màn hình danh sách mặt hàng có số lượng trong kho lớn hơn x và nhỏ hơn y . (1 điểm)

```
10 void Print(List l)
11 {
12     int x, y;
13     cout << "\nNhap gia tri x: ";
14     cin >> x;
15     while (x < 1)
16     {
17         cout << "\nGia tri x khong hop le!!! Vui long nhap lai: ";
18         cin >> x;
19     }
```

Giải đề mẫu



Sharing is learning

c. Viết hàm nhập vào 2 số nguyên dương x, y ($x < y$) và hiển thị lên màn hình danh sách mặt hàng có số lượng trong kho lớn hơn x và nhỏ hơn y . (1 điểm)

```
76      cout << "Nhap gia tri y: ";
77      cin >> y;
78      while (y <= x)
79      {
80
81          cout << "\nGia tri y khong hop le!!! Vui long nhap lai: ";
82          cin >> y;
83      }
```

Giải đề mẫu



Sharing is learning

c. Viết hàm nhập vào 2 số nguyên dương x, y ($x < y$) và hiển thị lên màn hình danh sách mặt hàng có số lượng trong kho lớn hơn x và nhỏ hơn y . (1 điểm)

```
85      Node* p;  
86      p = l.pHead;  
87      while (p)  
88      {  
89          if (p->so_luong_con_trong_kho > x && p->so_luong_con_trong_kho < y)  
90          {  
91              cout << p->ten_mat_hang << "\n";  
92          }  
93          p = p->pNext;  
94      }  
95  }
```


Giải đề mẫu



Sharing is learning

Câu 3: Hãy thực hiện chuyển đổi một số nguyên dương N ($N < 1000$) ở hệ thập phân sang biểu diễn ở hệ nhị phân (ví dụ: số 5 ở hệ thập phân sẽ là 101 ở hệ nhị phân), sử dụng cấu trúc ngăn xếp (stack), với các yêu cầu sau:

a. Định nghĩa cấu trúc ngăn xếp để lưu trữ số nhị phân (1 điểm).

```
5      #define MAX 100
6      ...
7      struct Stack
8      {
9          int n;
10         int a[MAX];
11     };
12
```

```
11     struct Node
12     {
13         int info;
14         Node* pNext;
15     };

```

Giải đề mẫu



b. Viết các hàm thao tác trúc ngăn xếp trong câu 3a: **push, pop, kiểm tra stack rỗng và kiểm tra stack đầy** (2 điểm)

```
17
18      /*hàm khởi tạo stack*/
19      - void createStack(Stack& s)
20      {
21          s.n = -1;
22      }
23
```

Giải đề mẫu



Sharing is learning

b. Viết các hàm thao tác trúc ngăn xếp trong câu 3a: push, pop, kiểm tra stack rỗng và kiểm tra stack đầy (2 điểm)

```
27  int isEmpty(Stack s)
28  {
29      if (s.n == -1)
30      {
31          return 1;
32      }
33      else return 0;
34  }
35
```


Giải đề mẫu



Sharing is learning

b. Viết các hàm thao tác trúc ngăn xếp trong câu 3a: push, pop, kiểm tra stack rỗng và kiểm tra stack đầy (2 điểm)

```
--
37  int isFull(Stack s)
38  {
39      if (s.n == MAX-1)
40      {
41          return 1;
42      }
43      else return 0;
44  }
```

Giải đề mẫu



Sharing is learning

b. Viết các hàm thao tác trúc ngăn xếp trong câu 3a: **push**, pop, kiểm tra stack rỗng và kiểm tra stack đầy (2 điểm)

```
46 void push(Stack& s, int x)
47 {
48     if (isFull(s) == 0)
49     {
50         s.n++;
51         s.a[s.n] = x;
52     }
53     else
54     {
55         cout << "\nNgan xep day!";
56     }
57 }
58
```

Giải đề mẫu



b. Viết các hàm thao tác trúc ngăn xếp trong câu 3a: push, pop, kiểm tra stack rỗng và kiểm tra stack đầy (2 điểm)

```
58
59  int pop(Stack& s)
60  {
61      if (isEmpty(s) == 0)
62      {
63          return s.a[s.n--];
64      }
65      else
66      {
67          cout << "\nNgan xep rong!";
68          return -1;
69      }
70  }
71
```


Giải đề mẫu



Sharing is learning

c. Viết hàm nhận đầu vào là 1 số nguyên dương N ở hệ thập phân, chuyển đổi và hiển thị kết quả số N ở hệ nhị phân lên màn hình (sử dụng cấu trúc, các hàm đã định nghĩa trong câu 3a, 3b) (1 điểm)

```
73 void chuyenSangHeNhiPhan(int n)
74 {
75     Stack s;
76     createStack(s);
77     int k;
78     while (n != 0)
79     {
80         k = n % 2;
81         push(s, k);
82         n /= 2;
83     }
84     cout << "\nSo sau khi duoc chuyen qua he Nhi phan la: ";
85     while (isEmpty(s) == 0)
86     {
87         cout << pop(s);
88     }
89 }
90
```

BAN HỌC TẬP KHOA CÔNG NGHỆ PHẦN MỀM

CHUỖI TRAINING GIỮA HỌC KÌ 2 NĂM HỌC 2021 - 2022



Sharing is learning

HẾT

**CẢM ƠN CÁC BẠN ĐÃ THEO DÕI.
CHÚC CÁC BẠN CÓ KẾT QUẢ THI THẬT TỐT!**



Ban học tập

Khoa Công Nghệ Phần Mềm
Trường ĐH Công Nghệ Thông Tin
ĐHQG Hồ Chí Minh



Email / Group

bht.cnpm.uit@gmail.com
fb.com/groups/bht.cnpm.uit