NỘI DUNG

DANH SÁCH LIÊN KẾT ĐƠN (LIST)

ThS.Nguyễn Thúy Loan

Bài toán

Giả sử cần phải quản lý sinh viên của một trường ĐH.

1.Tìm cấu trúc dữ liệu để lưu trữ thông tin trên.

2.Xuất thông tin trên ra màn hình

3.Thêm thông tin 1 SV vào DS

4.Xóa 1 thông tin khỏi DS

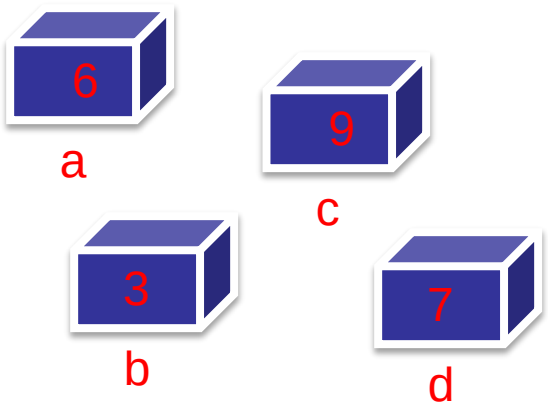
5.Tìm 1 phần tử có trong DS

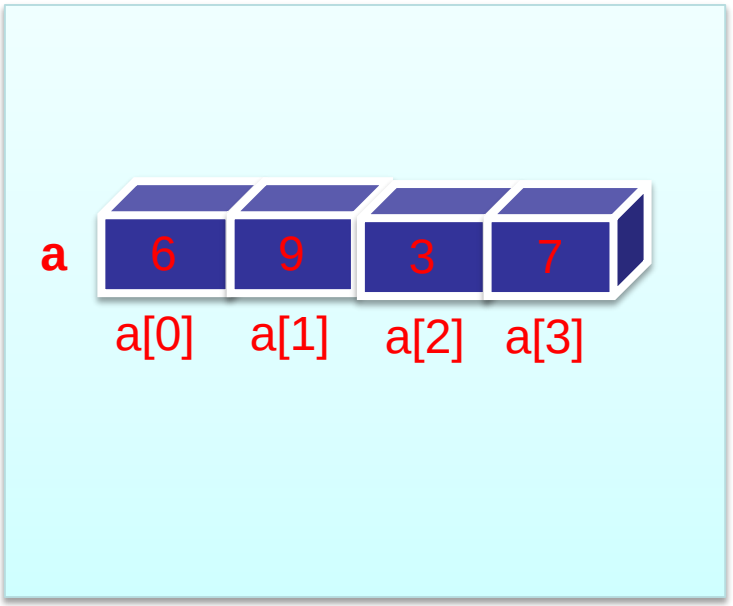
6.……

I. Cách 1

1. Lưu trên 1 mảng dữ liệu có cấu trúc:

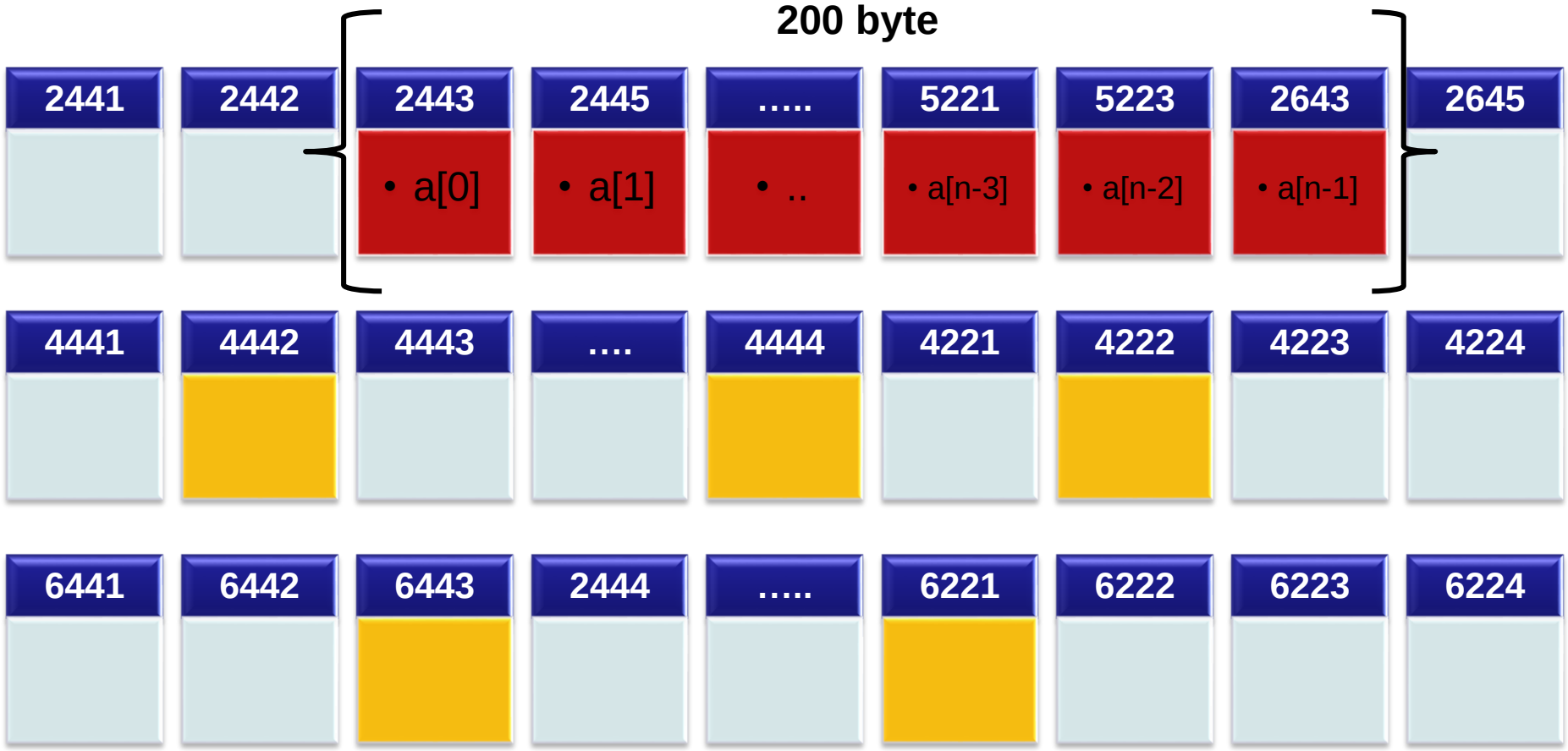
int a [100]



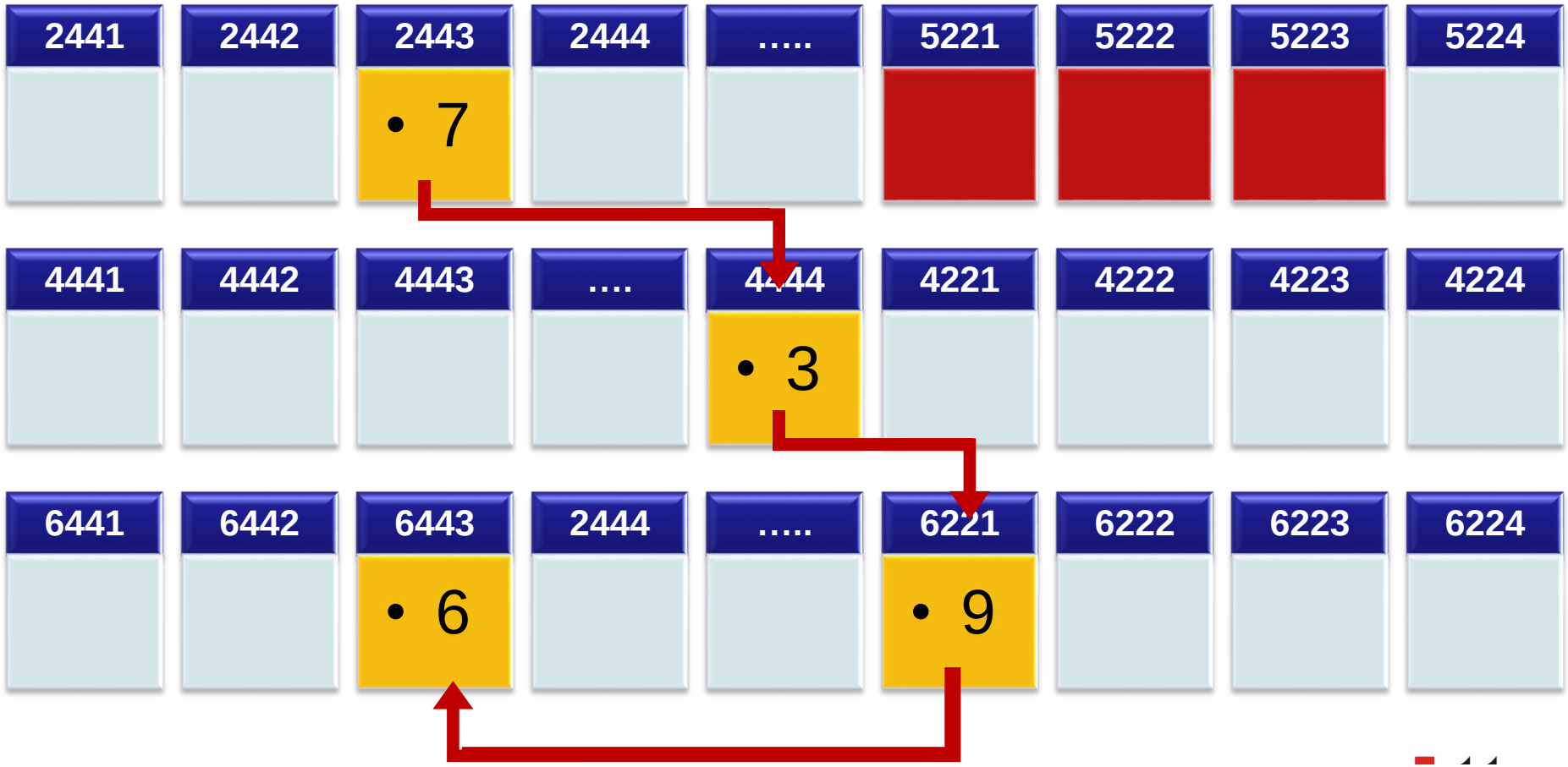


ThS.Nguyễn Thúy Loan

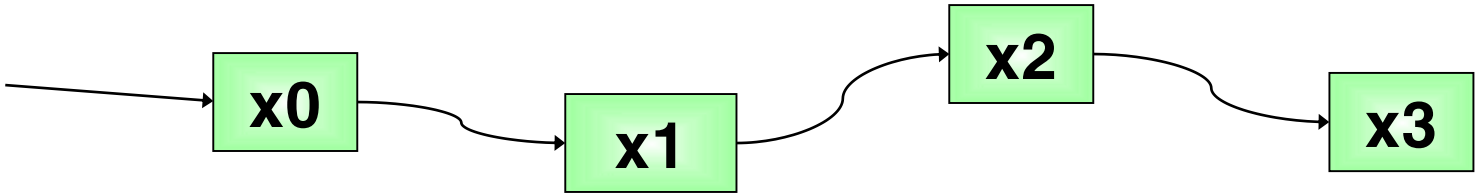
Lưu trữ trên mảng một chiều int a[100]



II. Cách 2 Lưu trữ trên DANH SÁCH LIÊN KẾT ĐƠN

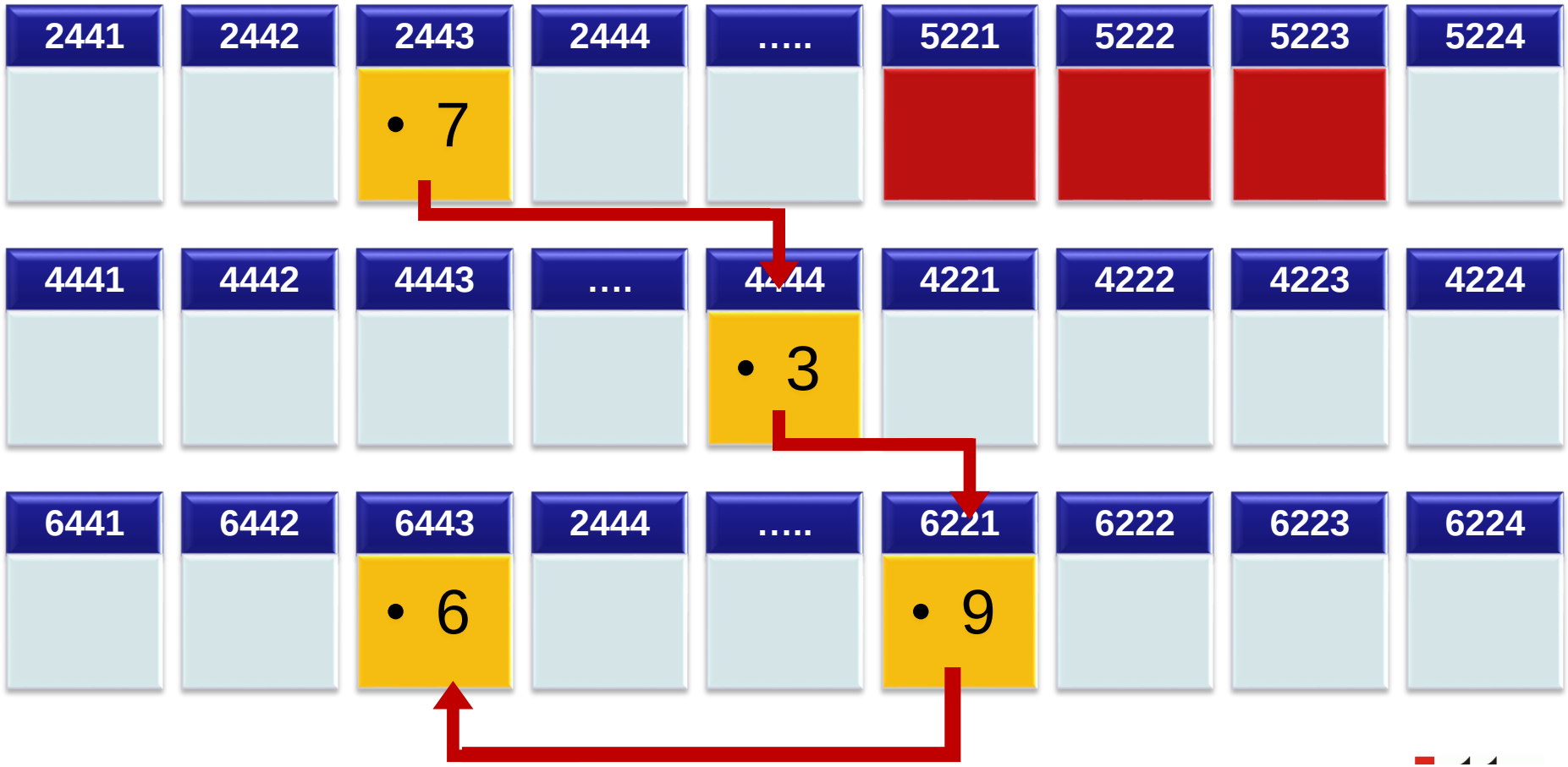


1. Tổ Chức Của DSLK Đơn



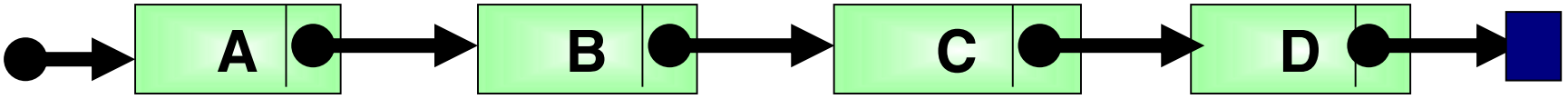
- Mỗi phần tử liên kết với phần tử đứng liền sau trong danh sách
- Mỗi phần tử trong danh sách liên kết đơn là một cấu trúc có hai thành phần
  - ***Thành phần dữ liệu***: Lưu trữ **thông tin** về bản thân phần tử
  - ***Thành phần liên kết***: Lưu **địa chỉ** phần tử đứng sau trong danh sách hoặc bằng NULL nếu là phần tử cuối danh sách.

II. Cách 2 Lưu trữ trên DANH SÁCH LIÊN KẾT ĐƠN

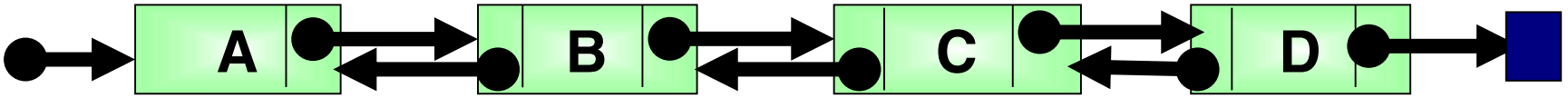


Các loại danh sách liên kết

- **Danh sách liên kết đơn**: Mỗi phần tử liên kết với phần tử đứng sau nó trong danh sách



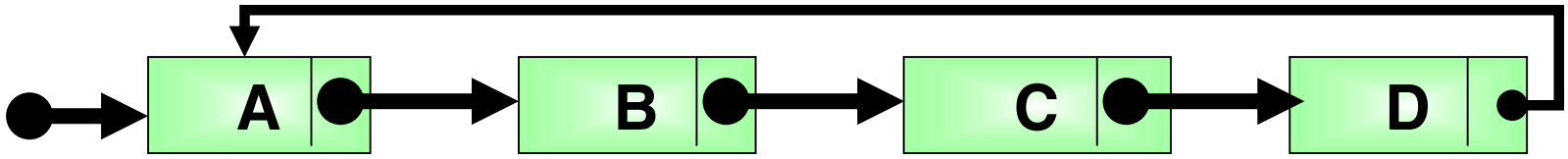
- **Danh sách liên kết kép**: Mỗi phần tử liên kết với phần tử đứng trước và sau nó trong danh sách



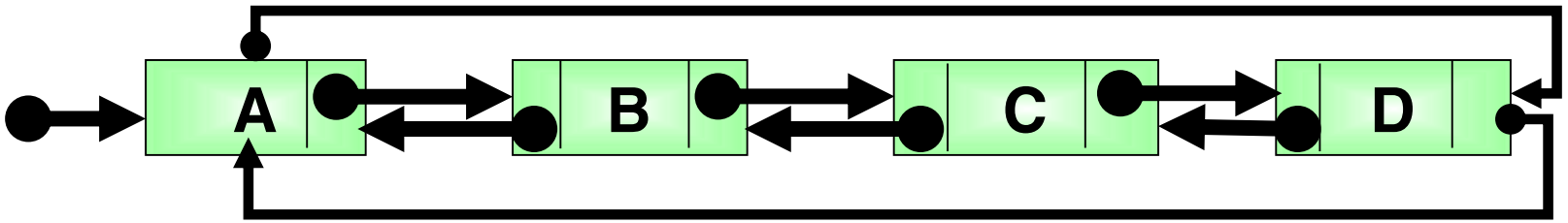
- **Danh sách liên Vòng**: Phần tử cuối danh sách liên kết với phần tử đầu danh sách

Các loại danh sách liên kết (tt)

- **Danh sách liên Vòng:** Phần tử cuối danh sách liên với phần tử đầu danh sách
  - Danh sách liên kết đơn vòng

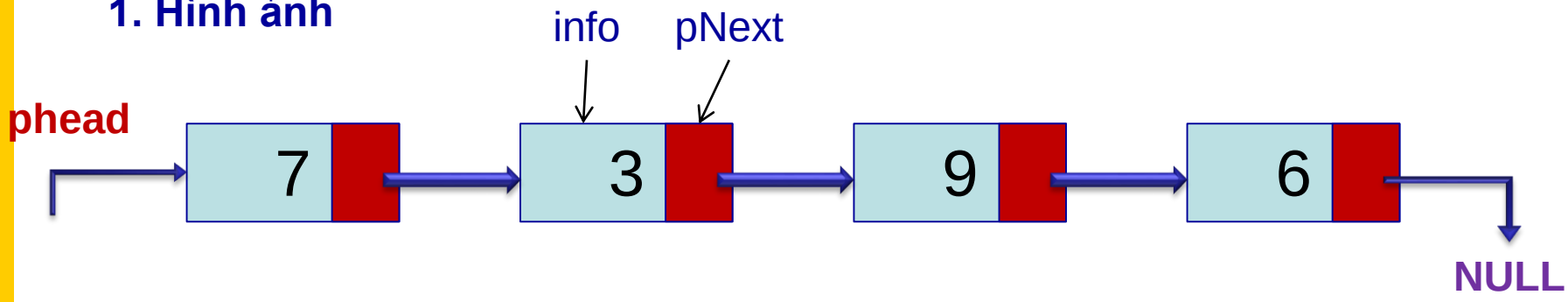


- Danh sách liên kết đôi vòng

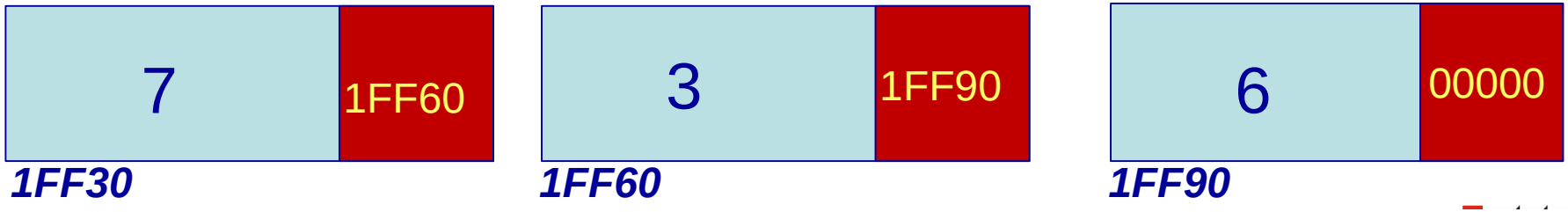


II. DANH SÁCH LIÊN KẾT ĐƠN

1. Hình ảnh



- Các phần tử trong danh sách liên kết kết nối với nhau theo dãy, trong đó:
  - ❖phead là con trỏ chỉ đến phần tử đầu tiên của danh sách liên kết.
  - ❖ Phần tử cuối của danh sách liên kết với vùng liên kết có nội dung NULL.
  - ❖ Mỗi nút của danh sách có trường info chứa nội dung của nút và trường pNext là con trỏ chỉ đến nút kế tiếp trong danh sách.



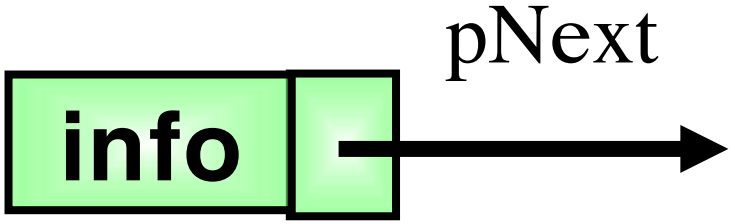
DANH SÁCH LIÊN KẾT ĐƠN

1. Nhận xét

- Cấu trúc danh sách liên kết là cấu trúc động
- Cấp phát nút cho danh sách liên kết bằng biến động.
- Việc cấp phát nút và giải phóng nút trên danh sách xảy ra khi chương trình đang chạy.
- Các nút có thể nằm rải rác ở nhiều nơi khác nhau trong bộ nhớ (phân bố không liên tục).

2. CTDL của DSLK đơn

- Cấu trúc dữ liệu của 1 nút trong List đơn
- **struct node**




```
1.{
2.      KDL info;
3.      struct node *pNext;
4.};
5.typedef struct node NODE;
```

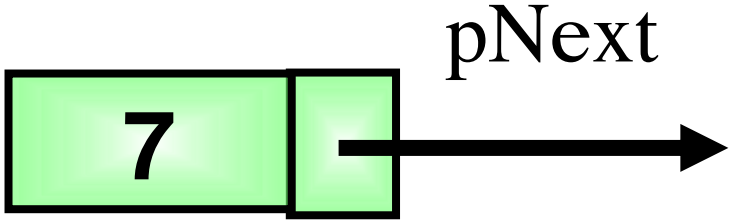
Ví dụ : khi khai báo NODE* a;

a - > info ;

a - > pNext ;

2. CTDL của DSLK đơn

- Cấu trúc dữ liệu của 1 nút trong List đơn là số nguyên
- **struct node**




```
1. {
2.     int info;
3.     struct node *pNext;
4. };
5. typedef struct node NODE;
```

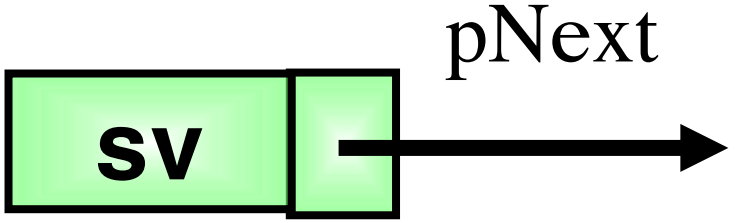
Ví dụ : khi khai báo NODE* a;

a - > info =7;

a - > pNext =NULL ;

2. CTDL của DSLK đơn

- Cấu trúc dữ liệu của 1 nút trong List đơn là SV
- **struct node**



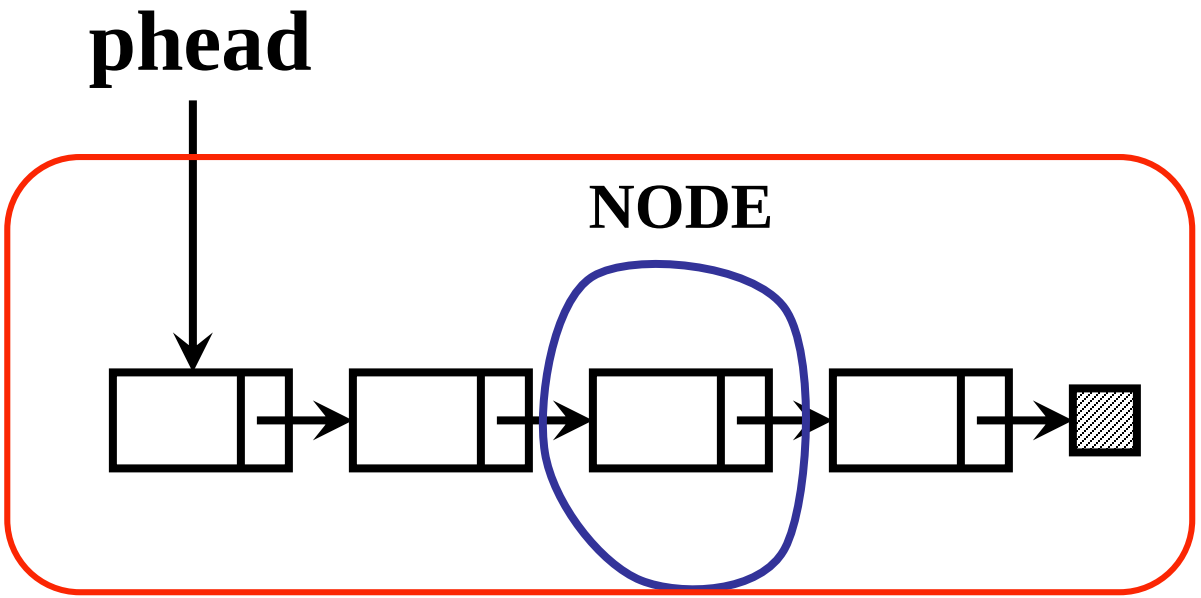

```
1.{
2.      sv info;
3.      struct node *pNext;
4.};
5.typedef struct node NODE;
```

Ví dụ : khi khai báo NODE* a;

a - > info = lưu thông tin 1 sv;

a - > pNext =NULL ;

Hình ảnh xâu đơn một trỏ

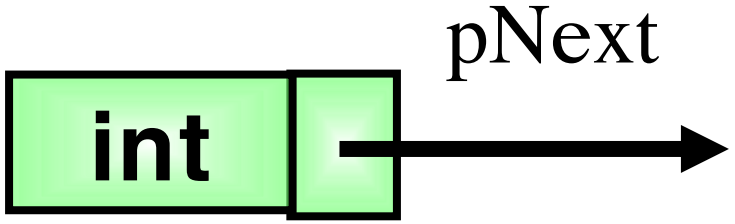


NODE *phead

VÍ DỤ 1 CTDL DANH SÁCH LIÊN KẾT ĐƠN

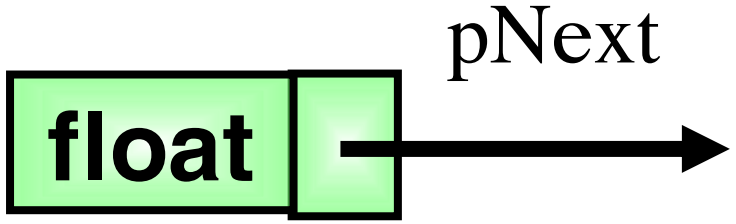
Ví dụ 1: Hãy khai báo CTDL cho dslk đơn các số NGUYÊN.

```
1.struct node
2.{
3.     int info;
4.     struct node*pNext;
5.};
6.typedef struct node NODE;
7.NODE *phead;
```



VÍ DỤ 2 CTDL DANH SÁCH LIÊN KẾT ĐƠN

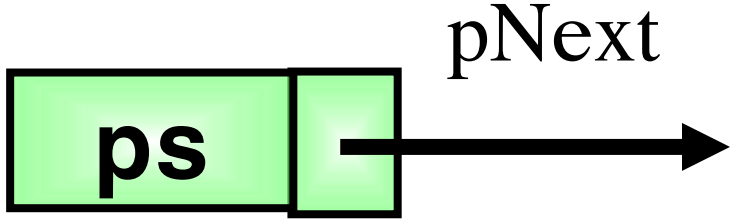
Ví dụ 2: Hãy khai báo CTDL cho dslk đơn các số thực.




```
1.struct node
2.{
3.     float info;
4.     struct node*pNext;
5.};
6.typedef struct node NODE;
7. NODE *phead;
```

Ví dụ 3: Hãy khai báo ctdl cho dslk đơn các phân số.

```
1.struct phanso
2.{     int tu;
3.      int mau;
4.};
5.typedef struct phanso PHANSO;
6.struct node
7.{     PHANSO info;
8.      struct node*pNext;
9.};
10.typedef struct node NODE;
11.NODE * phead;
```



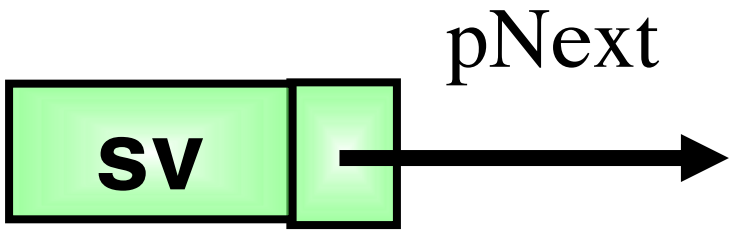
Kiểu dữ liệu nhiều thành phần

```
struct   ngay
{     int   ng  ;
      int   th  ;
      int   nm ;
};
typedef  struct ngay  date;
```

Kiểu dữ liệu nhiều thành phần

```
struct sinhvien
{       char ho[20];
        char ten[10];
        char mssv [11];
        char lop [9];
        char gioitinh;
        date ntns;
        float  toan, ly, tin , DTB ;
};
 typedef struct sinhvien   sv;
```

Ví dụ 3: Hãy khai báo ctdl cho dslk đơn SINH VIÊN.




```
1.struct node
2.{    sv info;
3.     struct node*pNext;
4.};
5.typedef struct node NODE;
6.NODE * phead;
```

3. Các thao tác trên DSLK đơn

1. Khởi tạo DSLK
2. Nhập dữ liệu
3. Tạo nút
4. Thêm lần lượt từng nút vào DSLK
5. Xuất Danh sách ra màn hình

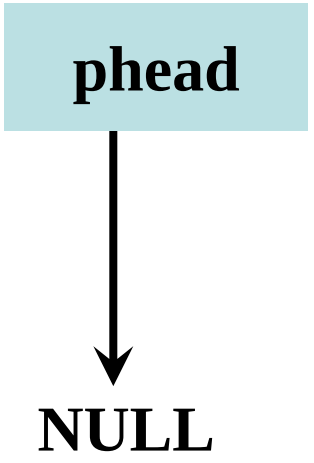
Các bước 2,3 và 4 lặp lại nhiều lần

1.KHỞI TẠO DANH SÁCH LIÊN KẾT ĐƠN

- Khái niệm: Khởi tạo danh sách liên kết đơn là tạo ra danh sách rỗng không chứa node nào hết.

- Định nghĩa hàm

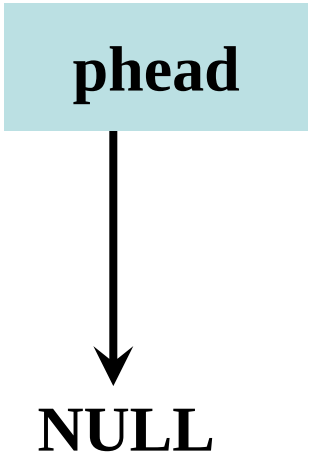
```
1.void Init(NODE *&phead)
2.{
3.     phead = NULL;
4.}
```



KIỂM TRA DANH SÁCH LIÊN KẾT ĐƠN RỖNG

- Khái niệm: Kiểm tra danh sách liên kết đơn rỗng là hàm trả về giá trị 1 khi danh sách rỗng. Trong tình huống danh sách không rỗng thì hàm sẽ trả về giá trị 0.
- Định nghĩa hàm

```
1.int IsEmpty(NODE *phead)
2.{    if (phead==NULL)
3.         return 1;
4.     return 0;
5.}
```



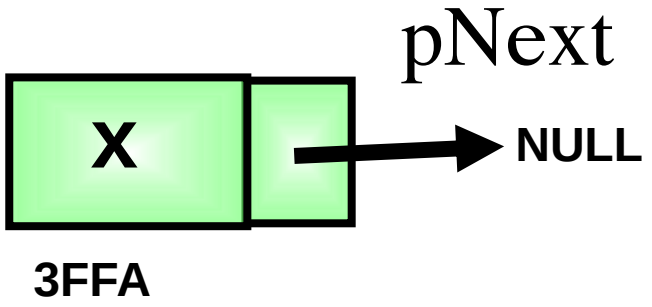
2. TẠO NODE CHO DANH SÁCH LIÊN KẾT ĐƠN

Khái niệm:

- ❖ **Tạo node cho danh sách liên kết đơn là :**
- ➢ **xin cấp phát bộ nhớ**
  - **Có kích thước bằng với kích thước của kiểu dữ liệu NODE**
  - **Để chứa thông tin đã được biết trước.**

2. TẠO NODE CHO DANH SÁCH LIÊN KẾT ĐƠN

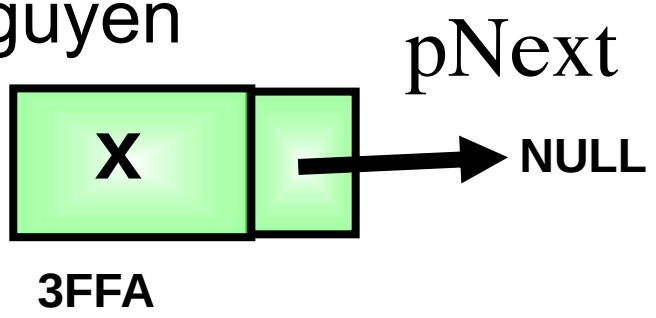
- Định nghĩa hàm trừu tượng




```
1.NODE* GetNode(KDL x)
2.{    NODE *p=new NODE;
3.     if (p==NULL)    return NULL;
4.     p->info = x;
5.     p->pNext = NULL;
6.     return p;
7.}
```

2. TẠO NODE CHO DANH SÁCH LIÊN KẾT ĐƠN

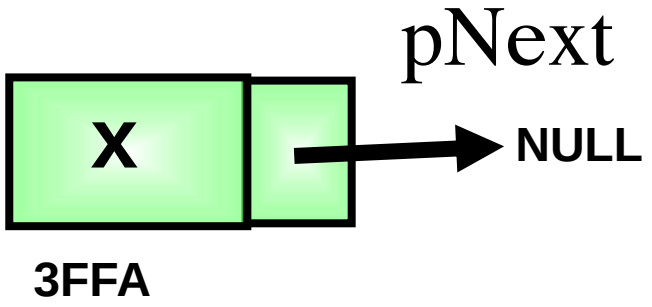
- Định nghĩa hàm tạo nút cho số nguyên




```
1.NODE* GetNode( int x)
2.{    NODE *p=new NODE;
3.     if (p==NULL)    return NULL;
4.     p->info = x;
5.     p->pNext = NULL;
6.     return p;
7.}
```

2. TẠO NODE CHO DANH SÁCH LIÊN KẾT ĐƠN

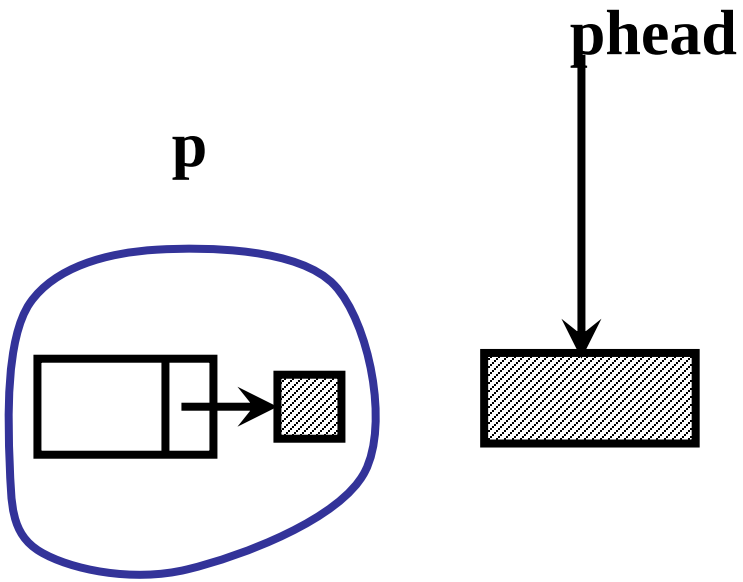
- Định nghĩa hàm tạo nút cho SV




```
1.NODE* GetNode( SV x)
2.{    NODE *p=new NODE;
3.     if (p==NULL)    return NULL;
4.     p->info = x;
5.     p->pNext = NULL;
6.     return p;
7.}
```

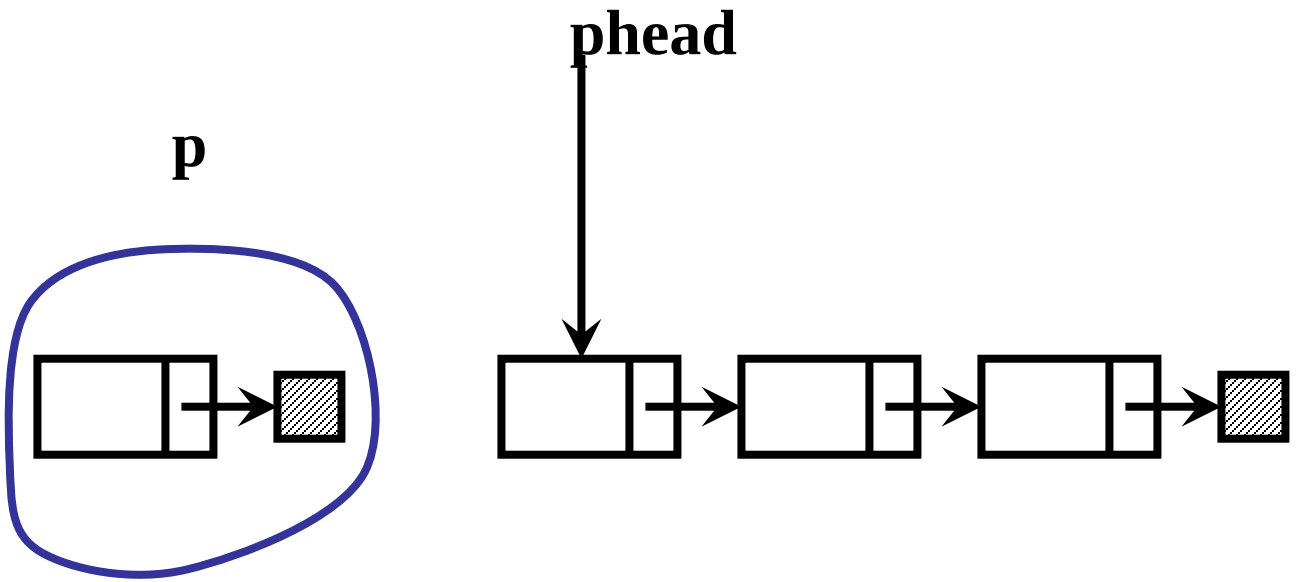
6. THÊM MỘT NODE VÀO ĐẦU DS LIÊN KẾT ĐƠN

- Khái niệm: Thêm một node vào đầu danh sách liên kết đơn là gắn node đó vào đầu danh sách.

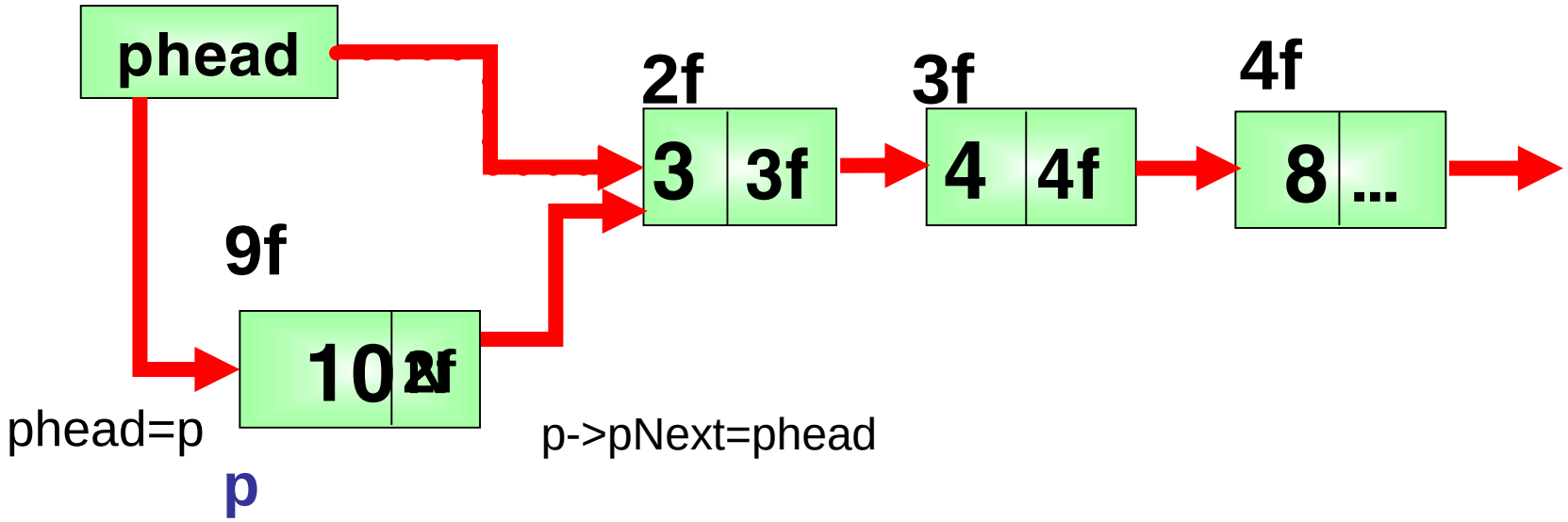


6. THÊM MỘT NODE VÀO ĐẦU DS LIÊN KẾT ĐƠN

- Khái niệm: Thêm một node vào đầu danh sách liên kết đơn là gắn node đó vào đầu danh sách.



Minh họa thuật toán thêm vào đầu



6. THÊM MỘT NODE VÀO ĐẦU DS LIÊN KẾT ĐƠN

➢ Định nghĩa hàm:

```
1.void AddHead(NODE *&phead, NODE*p)
2.{    if(phead==NULL) phead=p;
     1.else
     2.{    p->pNext = phead;
3.          phead = p;
4.       }
5.}
```

7. NHẬP DỮ LiỆU VÀO DS LIÊN KẾT ĐƠN

Các thao tác phải thực hiện :

1. **Khởi tạo danh sách**
2. **Nhập dữ liệu**
3. **Tạo Node cho dữ liệu**
4. **Gắn Node vào danh sách**
5. **Lặp lại bước 2 ,3 và 4 cho đến khi hết dữ liệu cần nhập**

7. NHẬP DỮ LIỆU VÀO DS LIÊN KẾT ĐƠN

➢ Định nghĩa hàm trừu tượng

```
1.void Input(NODE *&phead)
2.{    int n;
3.     printf("Nhap n: "); scanf("%d",&n);
4.     Init(phead);
5.     for(int i=1;i<=n;i++)
6.     {    KDL x;
7.          nhap(x)
8.          NODE *p=GetNode(x);
9.          if (p!=NULL)
10.              AddHead(phead,p);
11.    }
12.}
```

Ví dụ: Nhập danh sách liên kết đơn các số nguyên.

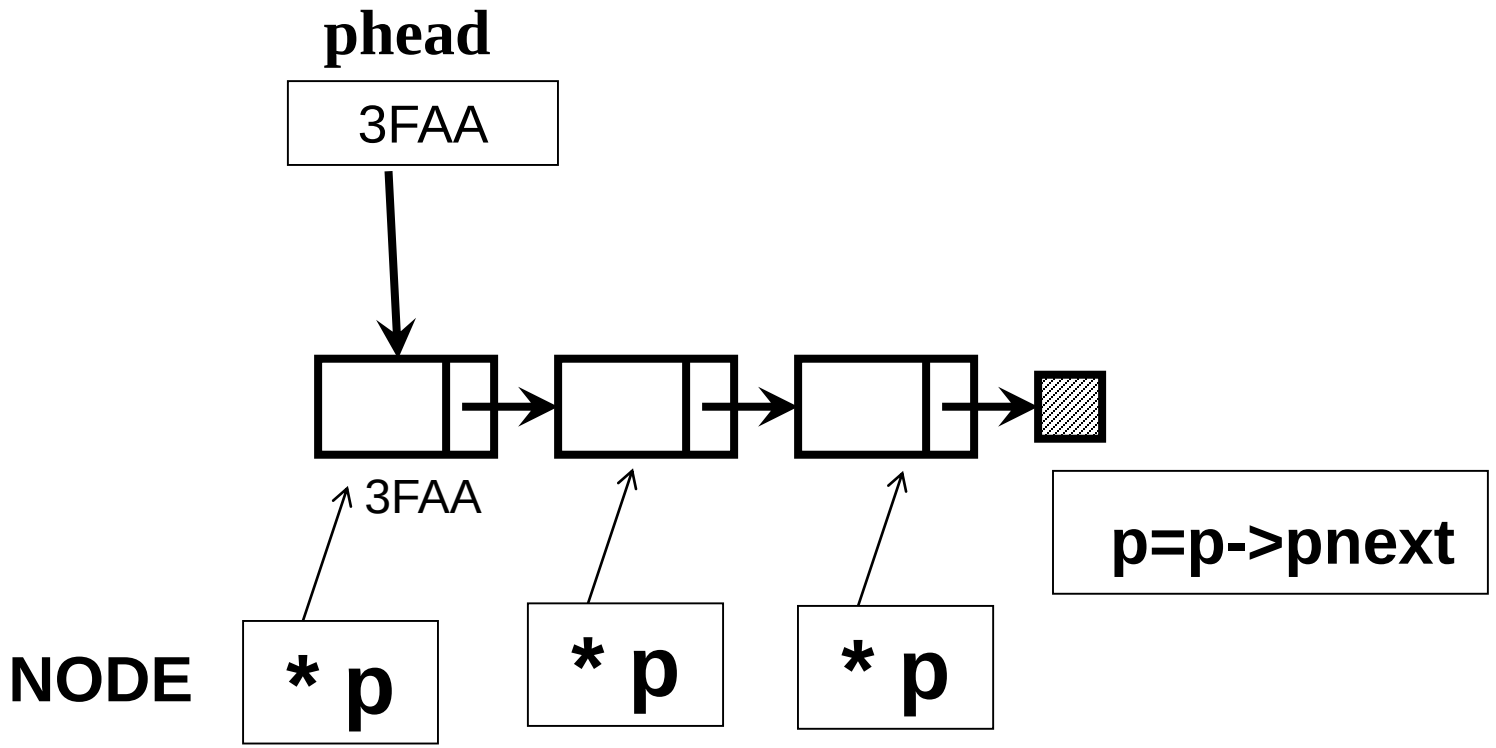
```
1.void Input(NODE *&phead)
2.{    int n;
3.     printf("Nhap n: ");scanf("%d",&n);
4.     Init(phead);
5.     for(int i=1;i<=n;i++)
6.     {
7.          int x;
8.          printf("Nhap so: ");
9.          scanf("%d",&x);
10.              NODE *p=GetNode(x);
11.               if (p!=NULL)
12.                   AddHead(phead,p);
13.     }
14.}
```

Ví dụ: Nhập danh sách liên kết đơn các số thực.

```
1.void Input(NODE *&phead)
2.{    int n;
3.     printf("Nhap n: ");scanf("%d",&n);
4.     Init(ptr);
5.     for(int i=1;i<=n;i++)
6.     {
7.          float x;
8.          printf("Nhap so: ");
9.          scanf("%f",&x);
10.              NODE *p=GetNode(x);
11.               if (p!=NULL)
12.                   AddHead(phead,p);
13.     }
14.}
```

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

- **Khái niệm: duyệt danh sách liên kết đơn là thăm qua tất cả các node mỗi node một lần.**



8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

- **Khái niệm: duyệt danh sách liên kết đơn là thăm qua tất cả các node mỗi node một lần.**
- Định nghĩa hàm trừu tượng

```
1.KDL <Tên Hàm>(NODE* phead)
2.{    ...
3.   for(NODE*p=phead;p!=NULL;p=p->pnext )
4.      {   ...
5.        //
6.      }
7.      ...
8.}
```

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

- **Khái niệm: duyệt danh sách liên kết đơn là thăm qua tất cả các node mỗi node một lần.**
- Định nghĩa hàm trừu tượng

```
1.KDL <Tên Hàm>(NODE* phead)
2.{    ...
3.     NODE *p=phead;
4.     while(p!=NULL)
5.     {   ...
6.        p=p->pNext;
7.     }
8.     ...
9.}
```

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

```
1.Void  xuat( NODE *phead )
2.{
3.   for(NODE*p=phead;p!=NULL;p=p->pnext )
4.   {
5.          printf("%d", p->info );
6.   }
7.
8.}
```

8. DUYỆT TUẦN TỰ DANH SÁCH LIÊN KẾT ĐƠN

```
1.void xuat (NODE *phead)
2.{
3.     NODE*p=phead;
4.     while(p!=NULL)
5.     {
          printf("%d", p->info );
6.         p=p->pNext;
7.     }
8.
9.}
```

VD:Tính tổng các số dslk đơn các số nguyên.

```
1.int Tong(NODE *phead)
2.{
3.     int s=0;
4.     NODE * p = phead;
5.     while(p!=NULL)
6.     {
7.        s=s+p->info;
8.        p=p->pNext;
9.     }
10.     return s
11.}
```

VD:Tính tổng các số lẻ trong dslk đơn các số nguyên.

```
1.int TongLe(NODE *phead)
2.{
3.     int s=0;
4.     NODE * p = phead;
5.     while(p!=NULL)
6.     {
7.        if(p->info%2!=0)
8.           s=s+p->info;
9.        p=p->pNext;
10.     }
11.     return s
12.}
```

VD:Tim max trong dslk đơn các số nguyên.

```
1.int Timmax(NODE *phead)
2.{
3.     int max = phead->info ;
4.     NODE * p = phead;
5.     while(p!=NULL)
6.     {
7.        if( p->info > max )
8.              max= p->info;
9.        p=p->pNext;
10.     }
11.     return s
12.}
```

9. CHƯƠNG TRÌNH ĐẦU TIÊN DSLK ĐƠN

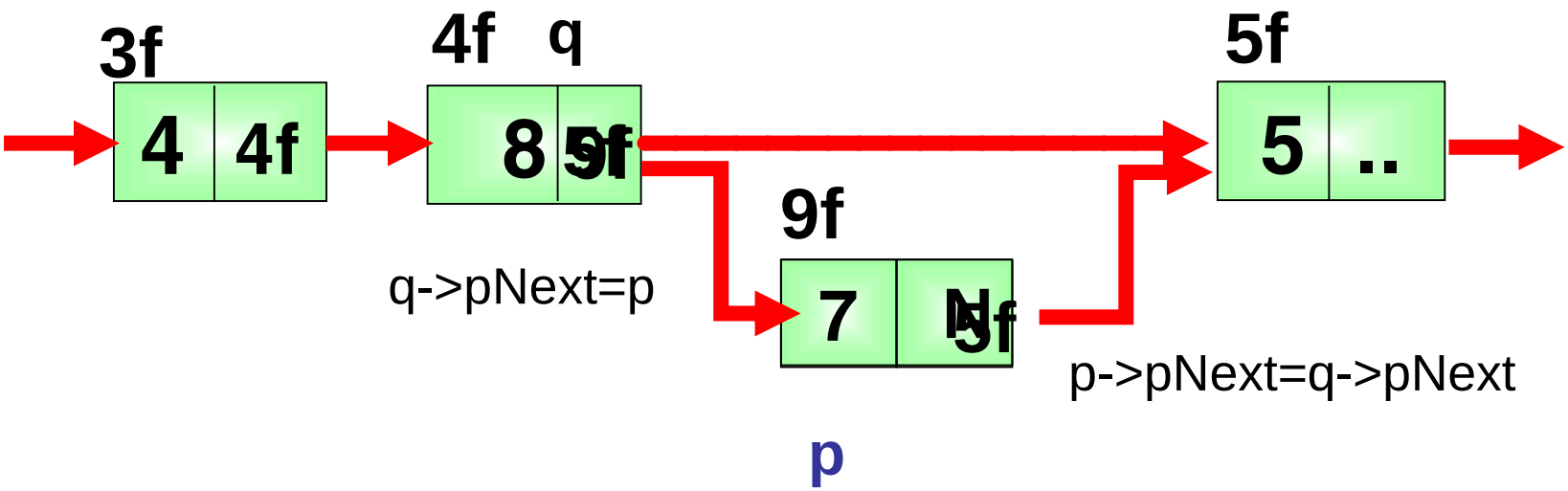
- Bài toán: Viết ctrình thực hiện các yêu cầu sau:
  - Nhập dslk đơn các số nguyên.
  - Tính tổng các giá trị trong dslk đơn.
  - Xuất dslk đơn.

```
1.#include "stdio.h"
2.#include "conio.h"
3.#include "math.h"
4.#include "string.h"
5.struct node
6.{
7.     int info;
8.     struct node *pNext;
9.};
10.typedef struct node NODE;
11.typedef NODE* phead;
```

9. CHƯƠNG TRÌNH ĐẦU TIÊN DSLK ĐƠN

```
// Khai báo hàm
void Init(NODEPTR&);
NODE* GetNode(int);
void AddHead(NODEPTR&, NODE*);
void Input(NODEPTR&);
void Output(NODEPTR);
int Tong(NODEPTR);
// Hàm main
void main()
{    NODE * lst;
     Input(lst);
     Output(lst);
     int kq = Tong(lst);
     printf("Tong la: %d",kq);
}
```

Thêm phần tử p vào sau phần tử q



Thêm phần tử p vào sau phần tử q

Ta cần thêm nút p vào sau nút q trong list đơn

Bắt đầu:

Nếu (q!=NULL) thì

B1: p->pNext = q->pNext

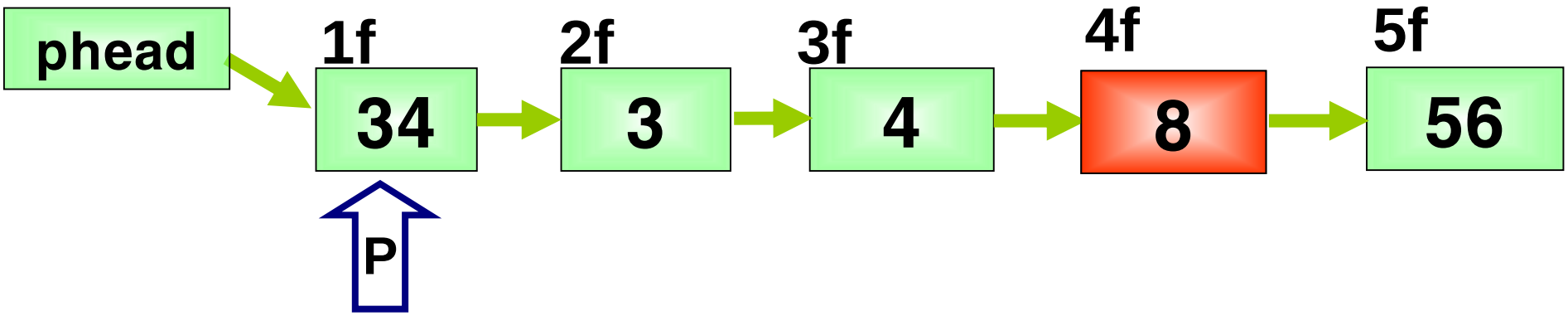
B2: q->pNext = p

Ngược lại: Thêm vào đầu danh sách

Cài đặt thuật toán

```
void addAfterq(NODE *&phead, NODE *p, NODE *q)
  {
      if(q!=NULL)
      {
        p->pNext=q->pNext;
        q->pNext=p;
      }
     else
        AddHead(phead,p);// thêm pvào đầu list
}
```

Minh họa thuật toán tìm phần tử trong DSLK



X = 8

Tìm thấy, hàm trả về địa chỉ của nút tìm thấy là 4f

Tìm 1 phần tử trong DSLK đơn

- Tìm tuần tự (hàm trả về), các bước của thuật toán tìm nút có info bằng x trong list đơn
- Bước 1: p=phead ;// địa chỉ của phần tử đầu trong ds đơn
- Bước 2:
- Trong khi p!=NULL và p->info!=x
- p=p->pNext;// xét phần tử kế
- Bước 3:
- + Nếu p!=NULL thì p lưu địa chỉ của nút có
- info = x
- + Ngược lại: Không có phần tử cần tìm

Tìm 1 phần tử trong DSLK đơn

Hàm tìm phần tử có info = x, hàm trả về địa chỉ của nút có info = x, ngược lại hàm trả về NULL

```
NODE * search( NODE *phead, int x)
{
    NODE *p = phead;
    while(p!=NULL && p->info!=x)
        p=p->pnext;
    return p;

}
```