

Cấu trúc dữ liệu

CÁC KIỂU DỮ LIỆU TRỪU TƯỢNG CƠ BẢN (BASIC ABSTRACT DATA TYPES)

Bộ môn Công Nghệ Phần Mềm



MỤC TIÊU

- Nắm vững các kiểu dữ liệu trừu tượng như: danh sách, ngăn xếp, hàng đợi.
- Cài đặt các kiểu dữ liệu trừu tượng bằng ngôn ngữ lập trình cụ thể.
- Ứng dụng được các kiểu dữ liệu trừu tượng trong bài toán thực tế.



NỘI DUNG

- Kiểu dữ liệu trừu tượng danh sách (LIST)
- Kiểu dữ liệu trừu tượng ngăn xếp (STACK)
- Kiểu dữ liệu trừu tượng hàng đợi (QUEUE)
- *Danh sách liên kết kép (Doubly-Linked Lists)*



KHÁI NIỆM VỀ DANH SÁCH

- Là tập hợp hữu hạn các phần tử có cùng kiểu.
- Kiểu chung được gọi là kiểu phần tử (element type).
- Ta thường biểu diễn dạng: $a_1, a_2, a_3, \dots, a_n$.
- Nếu
 - $n=0$: danh sách rỗng.
 - $n>0$: phần tử đầu tiên là a_1 , phần tử cuối cùng là a_n .
- Độ dài của danh sách: số phần tử của danh sách.
- Các phần tử trong danh sách có thứ tự tuyến tính theo vị trí xuất hiện. Ta nói a_i đứng trước a_{i+1} ($i=1..n-1$).



CÁC PHÉP TOÁN TRÊN DANH SÁCH

Tên phép toán	Công dụng
<code>makenullList(L)</code>	Khởi tạo một danh sách L rỗng
<code>emptyList(L)</code>	Kiểm tra xem danh sách L có rỗng hay không
<code>fullList(L)</code>	Kiểm tra xem danh sách L có đầy hay không
<code>insertList(x,P,L)</code>	Xen phần tử có nội dung x vào danh sách L tại vị trí P
<code>deleteList(P,L)</code>	Xóa phần tử tại vị trí P trong danh sách L
<code>endList(L)</code>	Trả về vị trí sau phần tử cuối trong ds L
<code>locate (X,L)</code>	Trả về kết quả là vị trí của phần tử có nội dung X đầu tiên trong danh sách L, <code>endList(L)</code> nếu không tìm thấy



CANTHO UNIVERSITY

CÁC PHÉP TOÁN TRÊN DANH SÁCH

Tên phép toán	Công dụng
retrieve(P,L)	Trả về nội dung phần tử tại vị trí P trong danh sách L, thông báo lỗi nếu vị trí P không có trong ds
next(P,L)	Trả về kết quả là vị trí của phần tử đi sau phần tử tại vị trí P trong danh sách L, endList(L) nếu phần tử tại vị trí P là phần tử cuối cùng, thông báo lỗi nếu vị trí P không có trong danh sách
previous(P,L)	Trả về kết quả là vị trí của phần tử đứng trước phần tử tại vị trí P trong danh sách L, thông báo lỗi nếu vị trí P là vị trí đầu tiên hoặc không có trong danh sách L
first(L)	Trả về kết quả là vị trí của phần tử đầu danh sách, endList(L) nếu danh sách rỗng
printList(L)	Hiển thị các phần tử trong danh sách L theo thứ tự xuất hiện



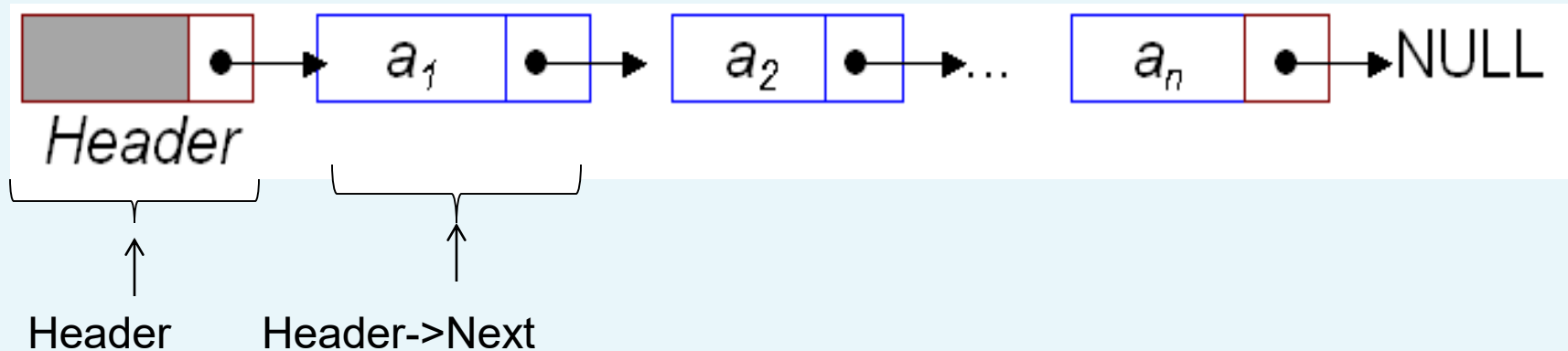
DANH SÁCH

- Khái niệm danh sách
- Các phép toán trên danh sách
- Cài đặt danh sách
 - Dùng mảng (DS ĐẶC)
 - Dùng con trỏ (DS LIÊN KẾT)



CÀI ĐẶT DANH SÁCH BẰNG CON TRỎ

- Mô hình

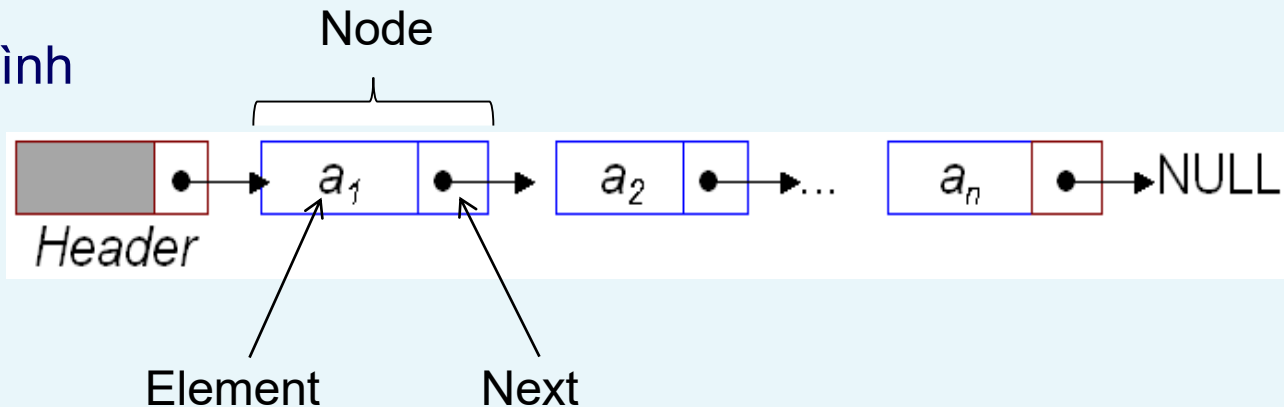


- Nối kết các phần tử liên tiếp nhau bằng con trỏ
 - Phần tử a_i trỏ tới phần tử a_{i+1} .
 - Phần tử a_n trỏ tới phần tử đặc biệt NULL.
 - Phần tử Header trỏ tới phần tử đầu tiên a_1 .



CÀI ĐẶT DANH SÁCH BẰNG CON TRỎ

- Mô hình



- Khai báo

//kiểu của phần tử trong danh sách

```
typedef <DataType> ElementType;
```

```
struct Node{
```

```
    ElementType Element; //Chứa nội dung của phần tử
```

```
    struct Node *Next; //Con trỏ chỉ đến phần tử kế tiếp
```

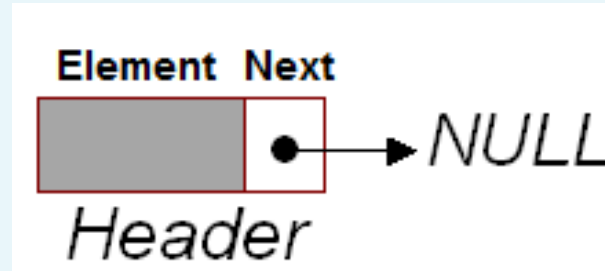
```
};
```

```
typedef struct Node *Position; //Kiểu vị trí
```

```
typedef Position List;
```



KHỞI TẠO DANH SÁCH RỖNG



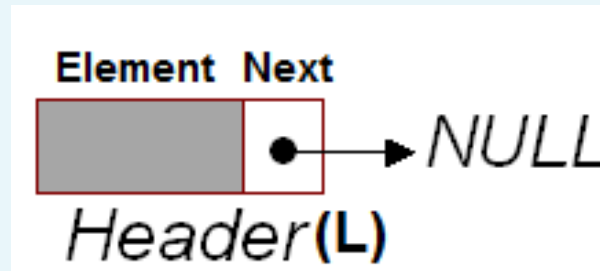
- Cấp phát vùng nhớ cho Header
- Cho trường Next của Header trở về NULL

```
void makenullList(List *pL)
{
    (*pL) = (struct Node*) malloc(sizeof(Node));
    (*pL) -> Next = NULL;
}
```



KIỂM TRA DANH SÁCH RỖNG

- Xem trường Next của ô Header có trỏ đến NULL hay không?

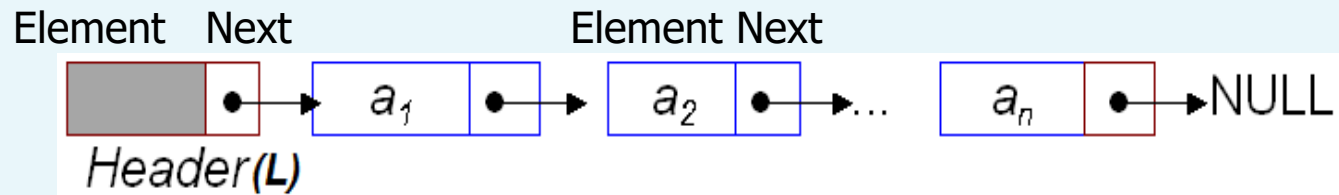


```
int emptyList(List L)
{
    return (L->Next==NULL) ;
}
```



XÁC ĐỊNH VỊ TRÍ PHẦN TỬ

- Vị trí phần tử đầu tiên



Position first(List L)

{

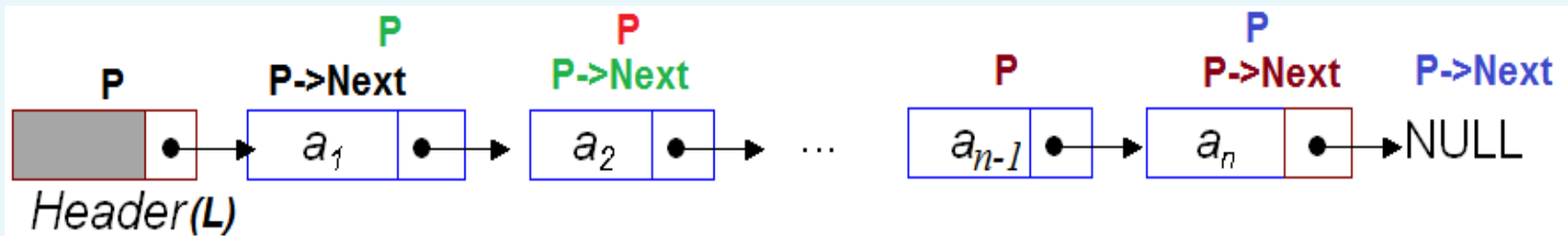
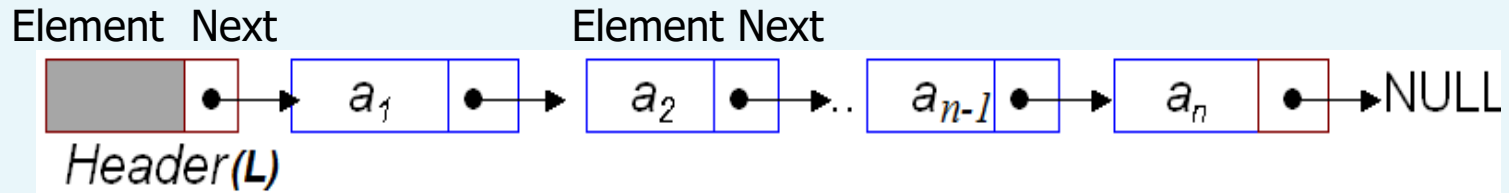
return L;

}



XÁC ĐỊNH VỊ TRÍ PHẦN TỬ

- Vị trí sau phần tử cuối cùng



Position `endList(List L)` {

Position P ;

$P = \text{first}(L)$; // $P = L$;

while ($P \rightarrow \text{Next} \neq \text{NULL}$) $P = P \rightarrow \text{Next}$;

return P ; }

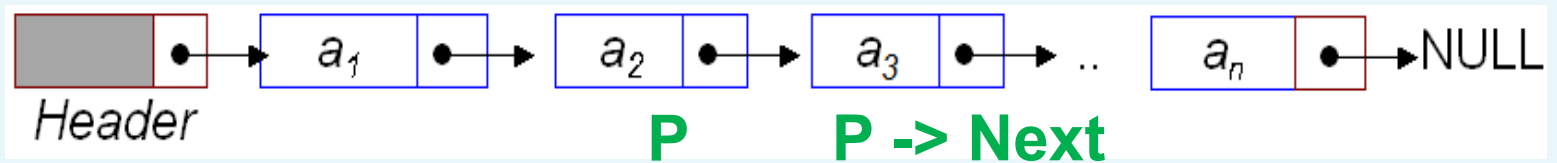


XÁC ĐỊNH VỊ TRÍ PHẦN TỬ

- Vị trí phần tử kế tiếp

Element Next

Element Next



Position next(Position P, List L)

{

return P->Next;

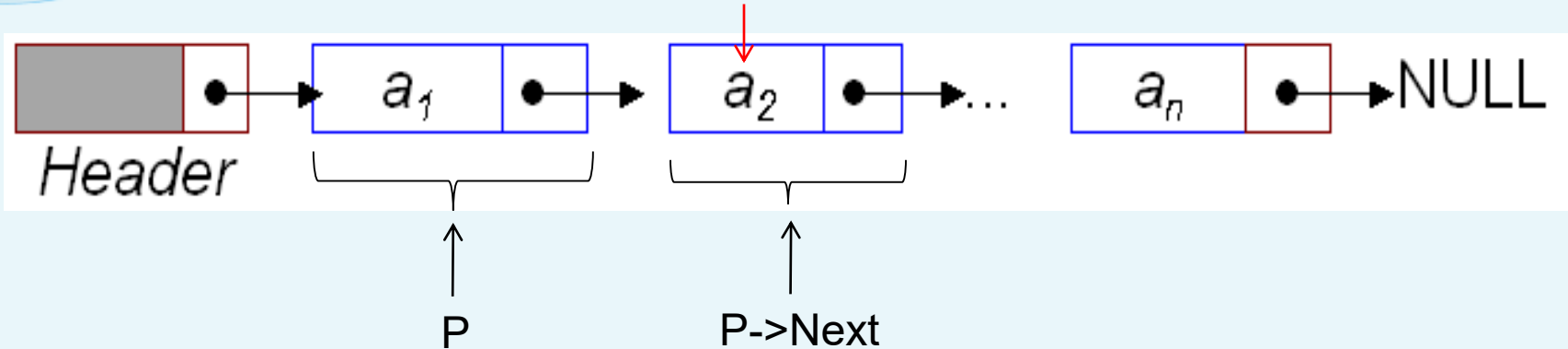
}

Cho nhận xét đánh giá độ phức tạp so với cách dùng mảng



XÁC ĐỊNH NỘI DUNG PHẦN TỬ

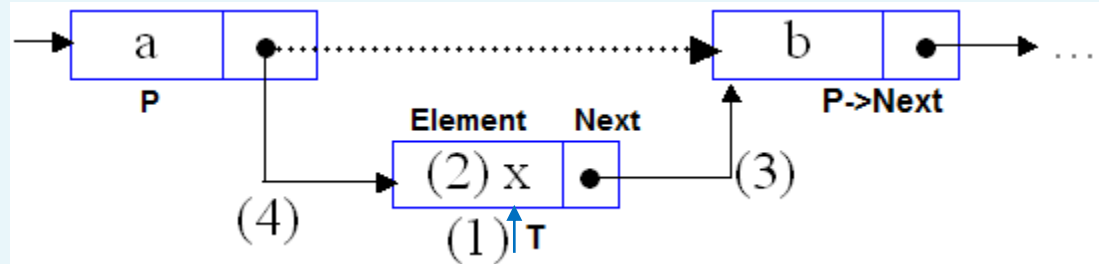
$P \rightarrow \text{Next} \rightarrow \text{Element}$ (nội dung của P)



```
ElementType retrieve(Position P, List L)
{
    if (P->Next != NULL)
        return P->Next->Element;
}
```



XEN MỘT PHẦN TỬ VÀO DANH SÁCH



- Để xen phần tử x vào vị trí P của L , ta làm như sau:
 - Cấp phát 1 ô mới để lưu trữ phần tử x .
 - Nối kết lại các con trỏ để đưa ô mới này vào vị trí P .

```
void insertList(ElementType x, Position P, List *pL)
{
```

```
    Position T;
```

```
    T = (struct Node*) malloc(sizeof(Node));
```

```
    T->Element = x;
```

```
    T->Next = P->Next;
```

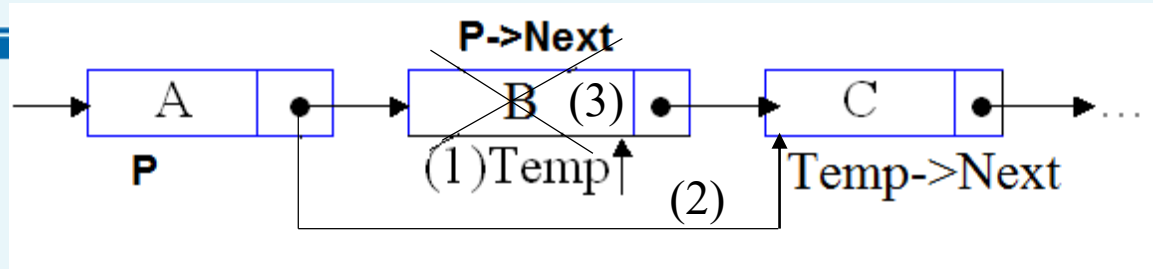
```
    P->Next = T;
```

```
}
```

Cho nhận xét đánh giá độ phức tạp
so với cách dùng mảng



XÓA MỘT PHẦN TỬ KHỎI DANH SÁCH



=> Muốn xóa phần tử ở vị trí P trong danh sách ta cần nối kết lại các con trỏ bằng cách cho P trở tới phần tử đứng sau phần tử thứ P.

```
void deleteList(Position P, List *pL) {
```

```
    Position Temp;
```

```
    if (P->Next != NULL) {
```

```
        //giữ ô chứa phần tử bị xóa để thu hồi vùng nhớ
```

```
        Temp = P->Next;
```

```
        //nối kết con trỏ trỏ tới phần tử kế tiếp
```

```
        P->Next = Temp->Next;
```

```
        //thu hồi vùng nhớ
```

```
        free(Temp);
```

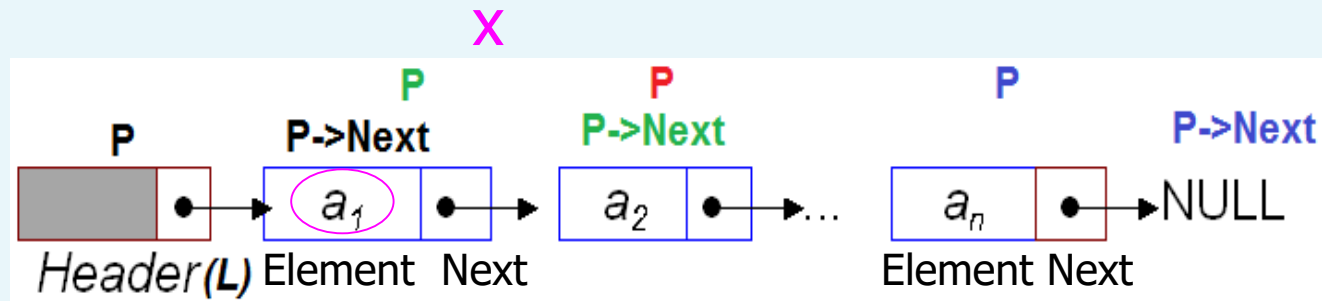
```
    }
```

```
}
```

Cho nhận xét đánh giá độ phức tạp so với cách dùng mảng



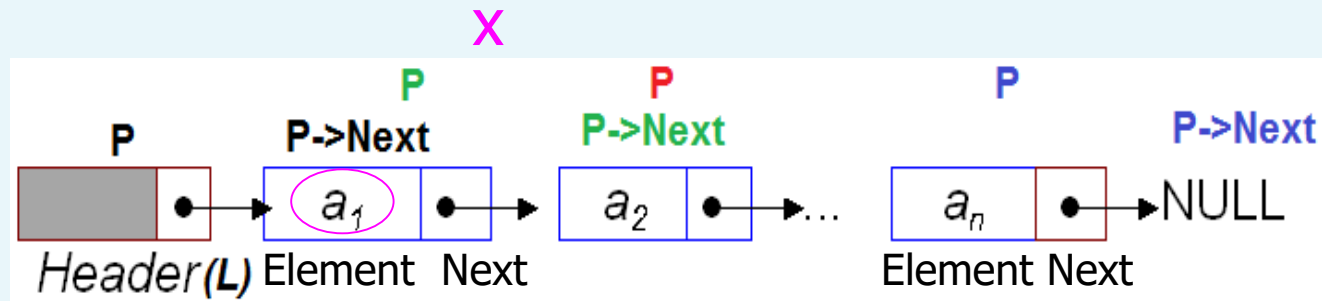
TÌM KIẾM MỘT PHẦN TỬ TRONG DANH SÁCH



- Bắt đầu từ phần tử đầu tiên trong danh sách, ta tiến hành tìm từ đầu danh sách cho đến khi tìm thấy hoặc cuối danh sách
 - Nếu giá trị tại vị trí P bằng x
 $\text{retrieve}(P, L) == x$ hay $P \rightarrow \text{Next} \rightarrow \text{Element} == x$
thì dừng tìm kiếm
 - Ngược lại (giá trị tại vị trí P khác x) thì đến vị trí kế tiếp
 $P = \text{next}(P, L)$



TÌM KIẾM MỘT PHẦN TỬ TRONG DANH SÁCH

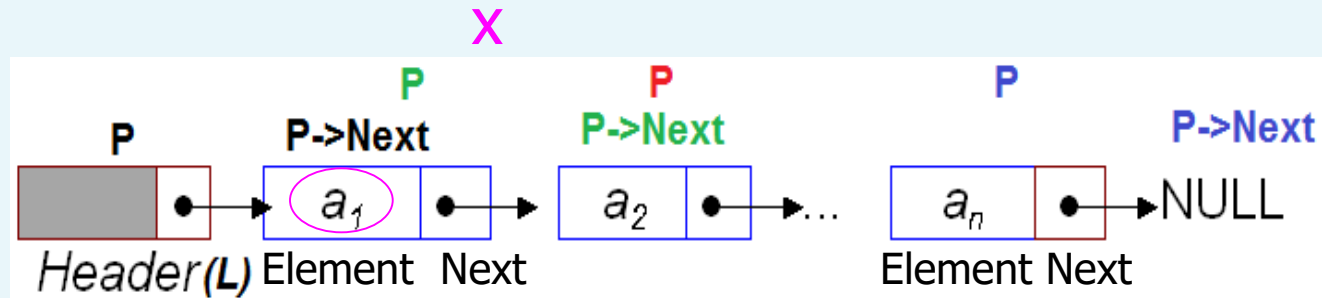


```
Position locate(ElementType x, List L){
    Position P;
    int Found = 0;
    P = first(L); next(P, L)
    while ((P->Next != NULL) && (Found == 0))
        if (P->Next->Element == X) Found = 1;
        else P = P->Next; retrieve(P, L)
    return P;    next(P, L)
}
```

Cài đặt lại hàm Locate bằng cách loại bỏ biến Found.



TÌM KIẾM MỘT PHẦN TỬ TRONG DANH SÁCH



Position locate(ElementType x, List L)

{

Position P;

P = first(L);

while (next(P,L) != **NULL**)

if (retrieve(P,L) == x)

return P;

else P = next(P,L);

return P; // endList(L)

}



Câu hỏi tìm vị trí

Cài đặt hàm

```
Position myLocate (ElementType X, int i,  
                  List L)
```

Trả về vị trí của lần xuất hiện thứ i của x trong L.



Trả lời - Câu hỏi tìm vị trí

```
Position myLocate(ElementType x, int i,
                    List L)
{
    Position P= first(L);
    int count =0;
    while (next(P,L) != NULL && count < i) {
        if (retrieve(P,L) == x)
            count++;
        if (count<i)
            P=next(P,L);
    }
    return P;
}
```



IN DANH SÁCH RA MÀN HÌNH

```
void printList(List L)
{
    Position P;
    P = first(L);
    while (P != endList(L))
    {
        printf("%d ", retrieve(P,L));
        P = next(P, L);
    }
    printf("\n");
}
```



Câu hỏi – Phép toán Previous

Hãy cài đặt phép toán

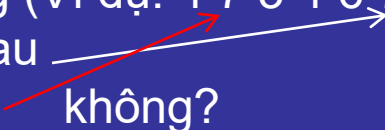
`Position previous (Position P, List L)`
bằng cách sử dụng các phép toán trừu tượng cơ bản
trên danh sách?



CANTHO UNIVERSITY

Câu hỏi – Phép toán Previous

```
Position previous(Position P, List L){  
    Position Q;  
    Q= first(L);  
    while (next(Q,L) != NULL)    //  
        if (next(Q,L) == P)  
            return Q;  
    else  
        Q = next(Q,L);  
    return NULL;  
}
```

Nếu danh sách có nhiều giá trị trùng (Ví dụ: 4 7 3 4 6 7 ...) thì khi tìm các vị trí của lần xuất hiện sau  thì kết quả phép toán trên có đúng không?
=> Viết đoạn chương trình để kiểm tra
Có thể cải thiện tốc độ của đoạn chương trình trên?



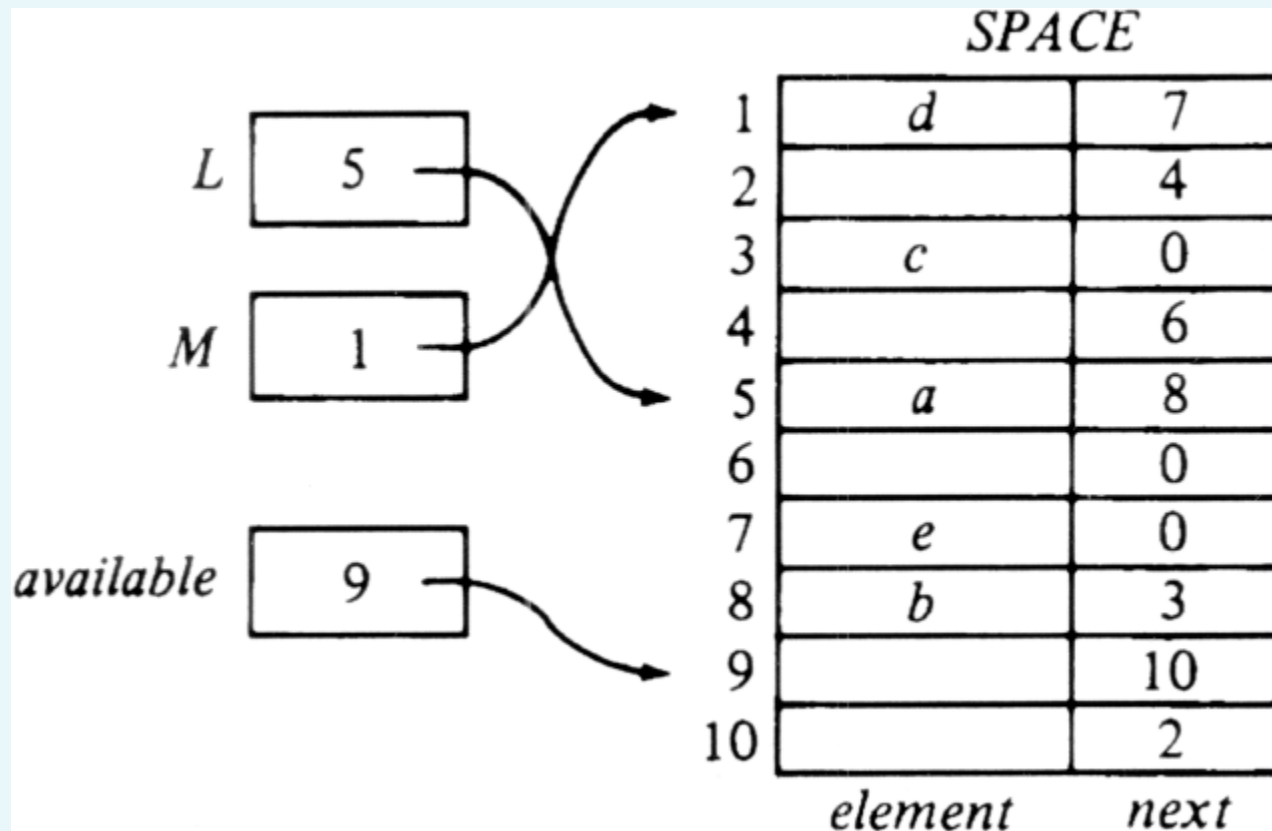
SO SÁNH 2 PHƯƠNG PHÁP CÀI ĐẶT DS

- Bạn hãy phân tích ưu và khuyết điểm của
 - Danh sách đặc
 - Danh sách liên kết
- Bạn nên chọn phương pháp cài đặt nào cho ứng dụng của mình?



Đọc thêm

Cài đặt danh sách bằng con nháy





BÀI TẬP

Vận dụng các phép toán trên danh sách liên kết để viết chương trình:

- Nhập vào một danh sách các số nguyên
- Hiển thị danh sách vừa nhập ra màn hình
- Thêm phần tử có nội dung x vào danh sách tại vị trí P (trong đó x và P được nhập từ bàn phím)
- Xóa phần tử đầu tiên có nội dung x (nhập từ bàn phím) ra khỏi danh sách
- Viết hàm

```
Position myLocate(ElementType x, int i, List L)
```



Trả lời bài tập

```
Position myLocate(ElementType x, int i, List L){
    Position P= first(L);
    int count =0;
    while (next(P,L) != NULL && count < i){
        if (retrieve(P,L) == x)
            count++;
        if (count<i)
            P=next(P,L);
    }
    return P;
}
```

Q&A?