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.

CANTHO UNIVERSITY

THỰC HÀNH - DANH SÁCH ĐẶC

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

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: $a_1, a_2, a_3, ..., a_n$.
- 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).

CANTHO UNIVERSITY

CÁC PHÉP TOÁN TRÊN DANH SÁCH

Tên phép toán	Công dụng
makenullList(L)	Khởi tạo một danh sách L rỗng
emptyList(L)	Kiểm tra xem danh sách L có rỗng hay không
fullList(L)	Kiểm tra xem danh sách L có đầy hay không
first(L)	Trả về kết quả là vị trí của phần tử đầu danh sách, endList(L) nếu danh sách rỗng
endList(L)	Trả về vị trí *sau phần tử cuối* trong ds L
insertList(x,P,L)	Xen phần tử có nội dung x vào danh sách L tại vị trí P, phép toán không được xác định (thông báo lỗi) nếu vị trí P không tồn tại trong danh sách
deleteList(P,L)	Xóa phần tử tại vị trí P trong danh sách L, phép toán không được xác định (thông báo lỗi) nếu vị trí P không tồn tại trong danh sách

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, kết quả không xác định (có thể thông báo lỗi) nếu vị trí P không có trong danh sách
locate(x,L)	Trả về kết quả là vị trí của phần tử có nội dung x đầu tiên trong danh sách L, endList(L) nếu không tìm thấy
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, kết quả không xác định 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, kết quả không xác định nếu ví trí P là vị trí đầu tiên hoặc không có trong danh sách L
printList(L)	Hiển thị các phần tử trong danh sách L theo thứ tự xuất hiện

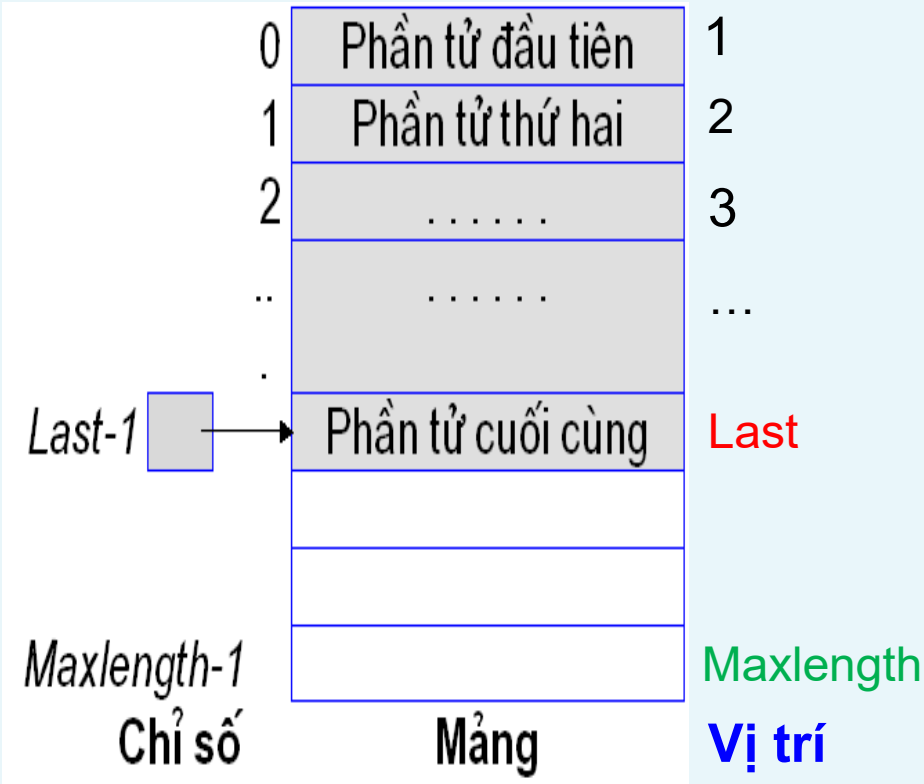
CÀI ĐẶT DANH SÁCH

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

Kiểu dữ liệu trừu tượng - Lưu ý

- Cài đặt kiểu dữ liệu trừu tượng:
  - Tổ chức lưu trữ: **cấu trúc dữ liệu** (*khai báo dữ liệu*).
  - Viết **chương trình con** thực hiện các phép toán (*khai báo phép toán*).

CÀI ĐẶT DANH SÁCH BẰNG MẢNG (DANH SÁCH ĐẶC)

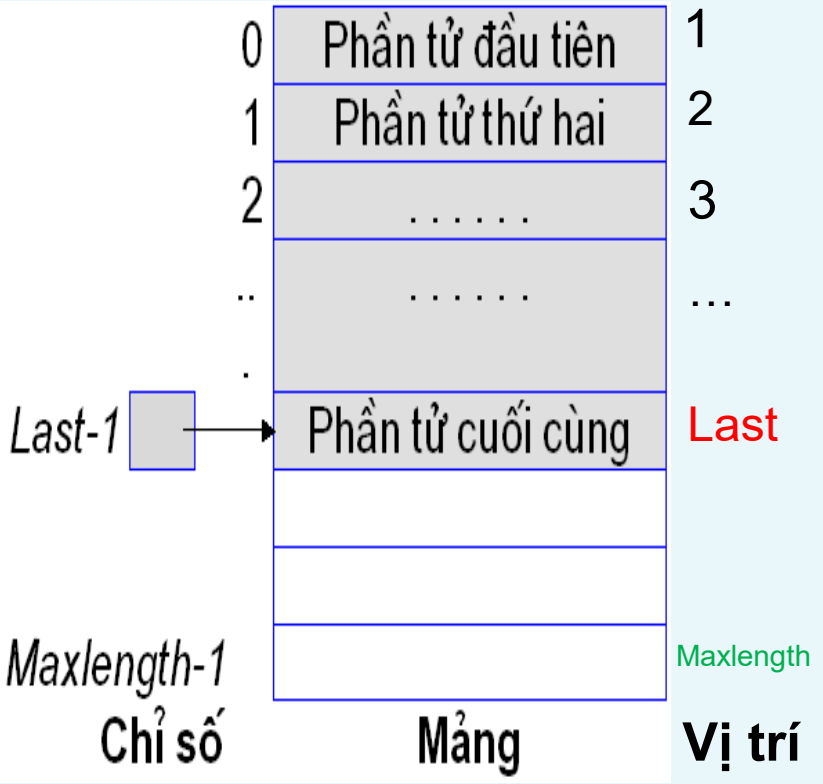


- Dùng 1 *mảng* (Elements) để lưu trữ liên tiếp các phần tử, bắt đầu từ vị trí đầu tiên.
- Phải ước lượng *số phần tử tối đa* của danh sách (Maxlength).
- Phải lưu trữ *độ dài hiện tại* của danh sách (Last).

Ta định nghĩa vị trí của một phần tử trong danh sách là "chỉ số của mảng tại vị trí lưu trữ phần tử đó + 1".

KHAI BÁO

```
//Độ dài tối đa của danh sách-ds
#define MaxLength <n>
//kiểu của phần tử trong ds
typedef <datatype> ElementType;
//kiểu vị trí cuả các phần tử
typedef int Position;
typedef struct {
   //mảng chứa các phần tử của ds
   ElementType Elements[MaxLength];
   //giữ độ dài danh sách
   Position Last;
 } List;
List L;
```



Kiểu dữ liệu trừu tượng - Lưu ý

- Cài đặt kiểu dữ liệu trừu tượng:
  - Tổ chức lưu trữ: **cấu trúc dữ liệu** (*khai báo dữ liệu*).
  - Viết **chương trình con** thực hiện các phép toán (khai báo phép toán).

Hàm

- Định nghĩa hàm

```
<kiểu kết quả>  Tên hàm (
                    [<kiểu t số> <tham số>]
                    [,<kiểu t số><tham số>][…])
{
      [Khai báo biến cục bộ]
      [Các câu lệnh thực hiện hàm]
      [return [<Biểu thức>];]
}
```

Tham số hình thức

- Sử dụng hàm/ Gọi hàm

Tham số thực tế

```
<Tên hàm>([Danh sách các tham số])
```

Hàm

- Định nghĩa hàm

```
<kiểu kết quả>  Tên hàm (
      void                [<kiểu t số> <tham số>]
                          [,<kiểu t số><tham số>][…])
{
      [Khai báo biến cục bộ]
      [Các câu lệnh thực hiện hàm]
      [return [<Biểu thức>];]
}
```

Tham số hình thức

- Sử dụng hàm/ Gọi hàm

Tham số thực tế

```
<Tên hàm>([Danh sách các tham số])
```

Hàm

Các phương pháp truyền tham số

- Truyền giá trị:
  - Là phương pháp truyền tham số mà sau đó *hàm được truyền* có được một phiên bản được lưu trữ riêng biệt giá trị của các tham số đó.
  - Khi truyền giá trị, thì giá trị gốc (được truyền) sẽ không bị thay đổi cho dù hàm được truyền có thay đổi các giá trị này đi nữa.
- Truyền bằng con trỏ:
  - Là phương pháp truyền tham số mà nó cho phép *hàm được truyền* tham khảo đến vùng nhớ lưu trữ giá trị gốc của tham số.
  - Nếu ta truyền bằng con trỏ thì giá trị gốc của tham số có thể được thay đổi bởi các mã lệnh bên trong hàm.

Con trỏ đến cấu trúc

Con trỏ trỏ đến biến kiểu cấu trúc

- Truy xuất các phần tử của cấu trúc: có 2 cách
  - Dùng toán tử * (dereference) kết hợp với toán tử . (dot)

```
(*p).diemTBTL = 8.5;
strcpy((*p).hoten, "TCAN");
strcpy((*p).ngaysinh, "23/12/78");
```

sv1.DiemTBTL=8.5;

  - Dùng toán tử -> (arrow operator)

```
p->diemTBTL = 8.5
strcpy(p->hoten, "TCAN");
strcpy(p->ngaysinh, "23/12/78");
```

sv1.DiemTBTL=8.5;

```
typedef struct {
   char hoten[30];
   char ngaysinh[11];
   float diemTBTL;
} Sinhvien;
```

```
Sinhvien sv1;
Sinhvien *p = &sv1;
```

```
Sinhvien sv1;
Sinhvien *p;
p = &sv1;
```

LƯU Ý (Định nghĩa hàm, gọi hàm, phương pháp truyền tham số)

```
void makenullList(List *pL){
      pL->Last=0;
}
```

```
(*pL).Last=0
```

- Giả sử ta có khai báo biến:

```
List   L1;
List  *L2;
```

Hãy viết lời gọi hàm makenullList() khởi tạo danh sách L1 và danh sách được trỏ bởi con trỏ *L2 rỗng?

```
makenullList(&L1);
makenullList(L2);
```

```
int emptyList(List L){
      return L.Last==0;
}
```

- Giả sử ta có khai báo biến:

```
List   L1;
List  *L2;
```

Hãy viết lời gọi hàm emptyList() kiểm tra xem danh sách L1 và danh sách được trỏ bởi con trỏ *L2 có rỗng hay không?

```
emptyList(L1);
emptyList(*L2);
```

Viết hàm tìm phần tử x trong danh sách

```
Position locate(ElementType x, List L){
   Position P;
   int Found = 0;
   P = first(L); //vị trí phần tử đầu tiên
   /*trong khi chưa tìm thấy và chưa kết thúc danh sách
   thì xét phần tử kế tiếp*/
   while ((P != endList(L)) && (Found == 0))
       if (retrieve(P,L) == x) Found = 1;
       else P = next(P,L);
   return P;
}
```

LƯU Ý (cài đặt, ứng dụng)

Ứng dụng ADT: tái sử dụng, tổng quát, không phụ thuộc vào chi tiết cài đặt bên trong

```
Position locate(ElementType x, List L){
   Position P;
   P = 1; //vị trí phần tử đầu tiên
   /*trong khi chưa tìm thấy và chưa kết thúc danh sách
   thì xét phần tử kế tiếp*/
   while (P != L.Last+1)
       if (L.Elements[P-1] == x) return P;
       else P = P+1;
   return P;
}
```

Gắn với cách cài đặt cụ thể (cài đặt danh sách bằng mảng), can thiệp chi tiết bên trong

CANTHO UNIVERSITY

KHỞI TẠO DANH SÁCH RỖNG

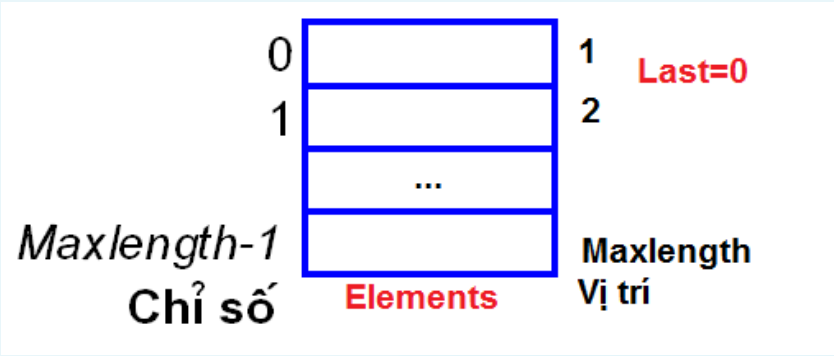
- Cho độ dài danh sách bằng 0.

```
void makenullList(List *pL){
        pL->Last=0;
}
```

(*pL).Last=0

- Hãy viết hàm makenullList không có tham số vào và trả về một danh sách rỗng?

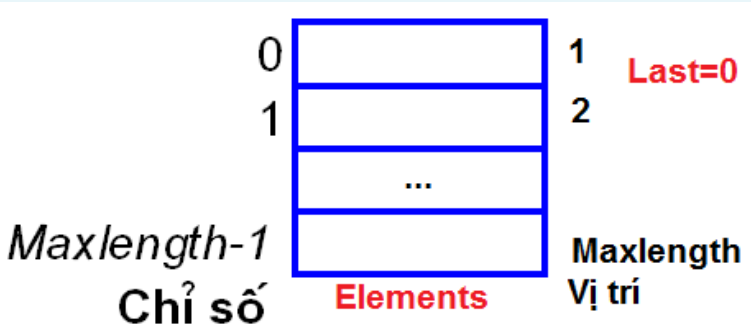
```
List makenullList(){
        List L; // L biến kiểu cấu trúc
        L.Last=0;
        return L;
}
```



KIỂM TRA DANH SÁCH RỖNG/ĐẦY

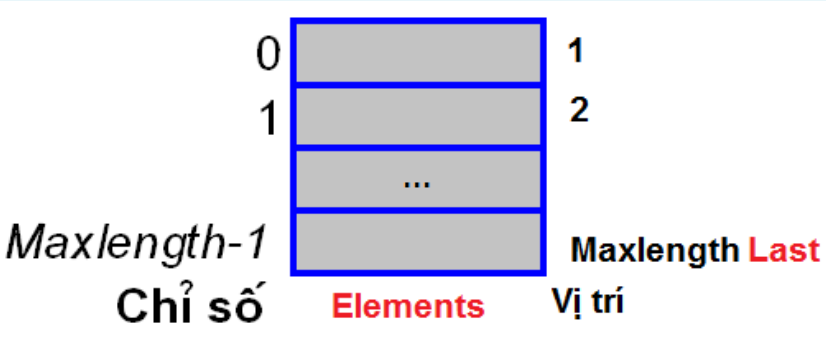
- Xem độ dài danh sách có bằng 0 không?

```
int emptyList(List L){
    return L.Last==0;
}
```



- Xem độ dài danh sách có bằng MaxLength không?

```
int fullList(List L){
    return L.Last==MaxLength;
}
```

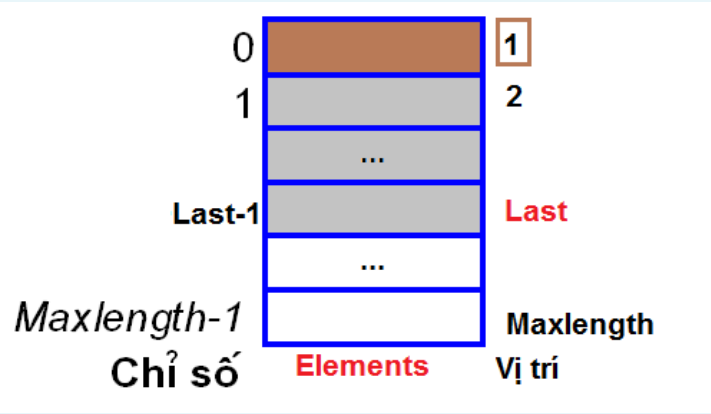


CANTHO UNIVERSITY

XÁC ĐỊNH VỊ TRÍ ĐẦU / SAU VỊ TRÍ CUỐI

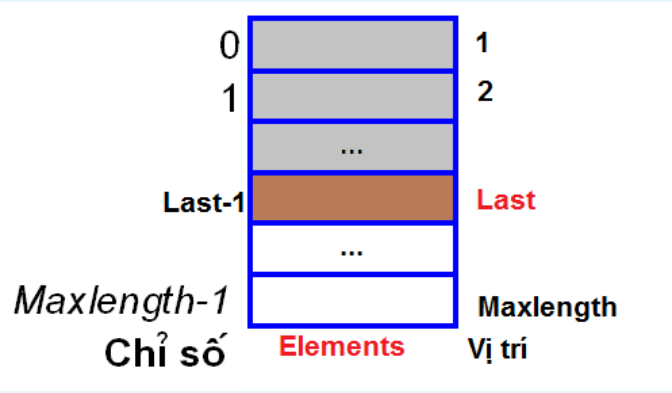
- Xác định vị trí đầu tiên trong danh sách.

```
Position first(List L){
   return 1;
}
```



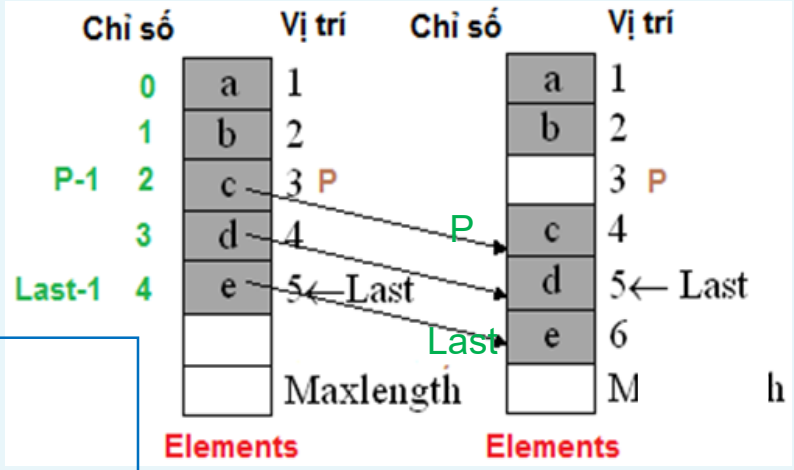
- Xác định vị trí sau phần tử cuối trong danh sách.

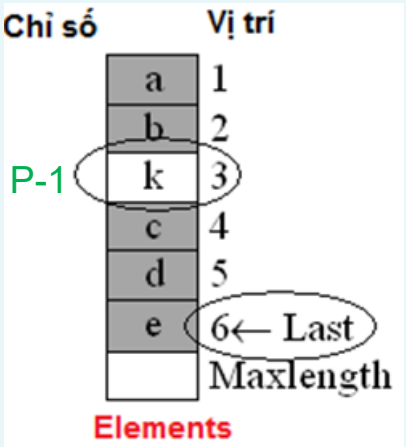
```
Position endList(List L){
  return L.Last+1;
}
```



XEN PHẦN TỬ x VÀO VỊ TRÍ P

```
void insertList(ElementType x,Position P, List *pL){
    if (pL->Last==MaxLength)
        printf("Danh sach day");
    else if ((P<1 || (P>(pL->Last+1)))
        printf("Vi tri khong hop le");
    else {
        Position Q;
       /*Dời các phần tử từ vị trí P
      đến cuối dsách ra sau 1 vị trí*/
        for(Q=pL->Last;Q>=P;Q--)
            pL->Elements[Q]=pL->Elements[Q-1];
        //Đưa x vào vị trí P
        pL->Elements[P-1]=x;
        //Tăng độ dài danh sách lên 1
        pL->Last++;     }
}
```





XEN PHẦN TỬ x VÀO VỊ TRÍ P

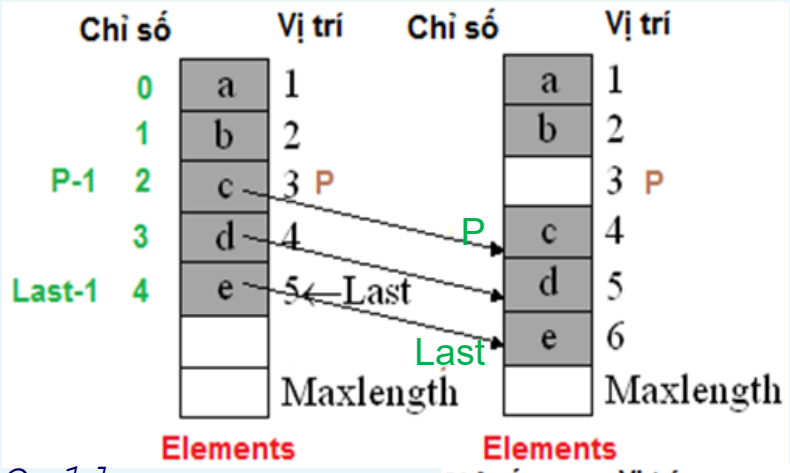
CANTHO UNIVERSITY

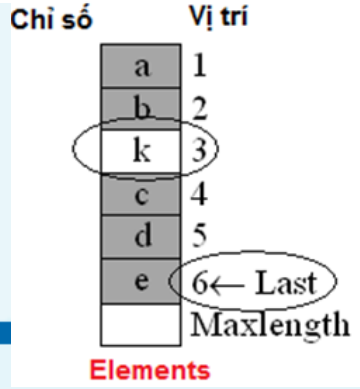
Cài đặt

```
void insertList(ElementType x,Position P, List *pL){
    if (pL->Last==MaxLength)
        printf("Danh sach day");
    else if ((P<1 || (P>(pL->Last+1)))
        printf("Vi tri khong hop le");
    else {
        Position Q;
       /*Dời các phần tử từ vị trí P
      đến cuối dsách ra sau 1 vị trí*/
        for(Q=pL->Last;Q>=P;Q--)
            pL->Elements[Q]=pL->Elements[Q-1];
        //Đưa x vào vị trí p
        pL->Elements[P-1]=x;
        //Tăng độ dài danh sách lên 1
        pL->Last++;
    }  }
```

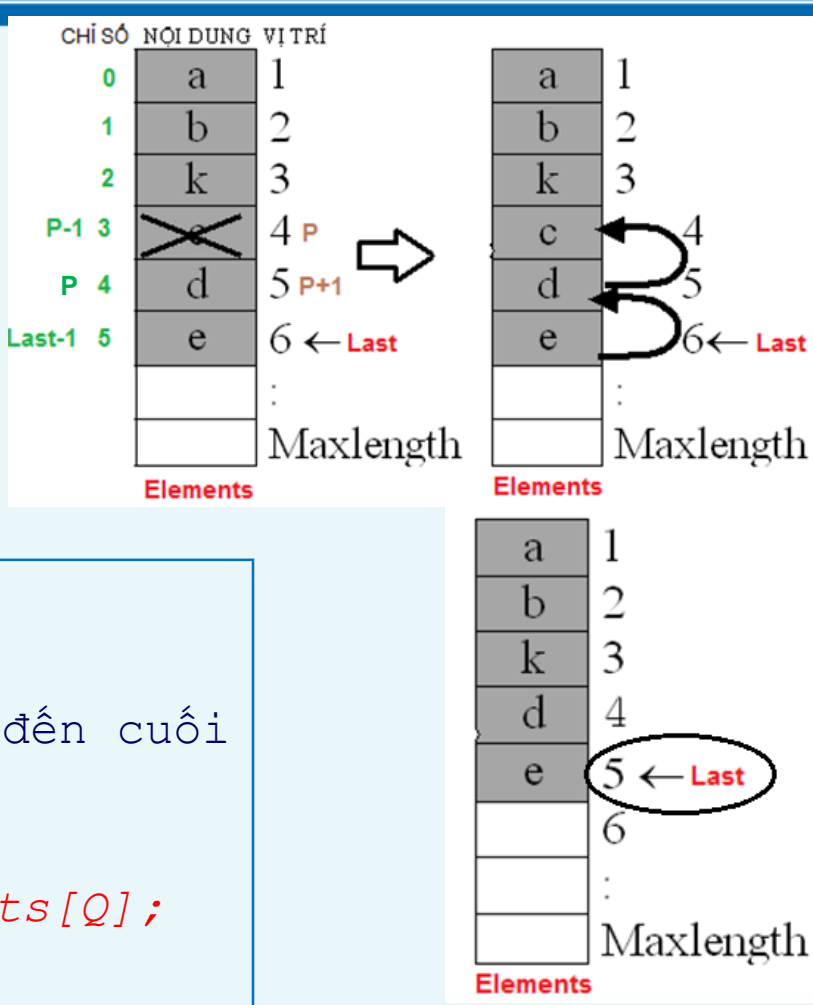
fullList(*pL)

P<first(*pL)) || (P>endList(*pL))





XÓA MỘT PHẦN TỬ TẠI VỊ TRÍ P TRONG DS



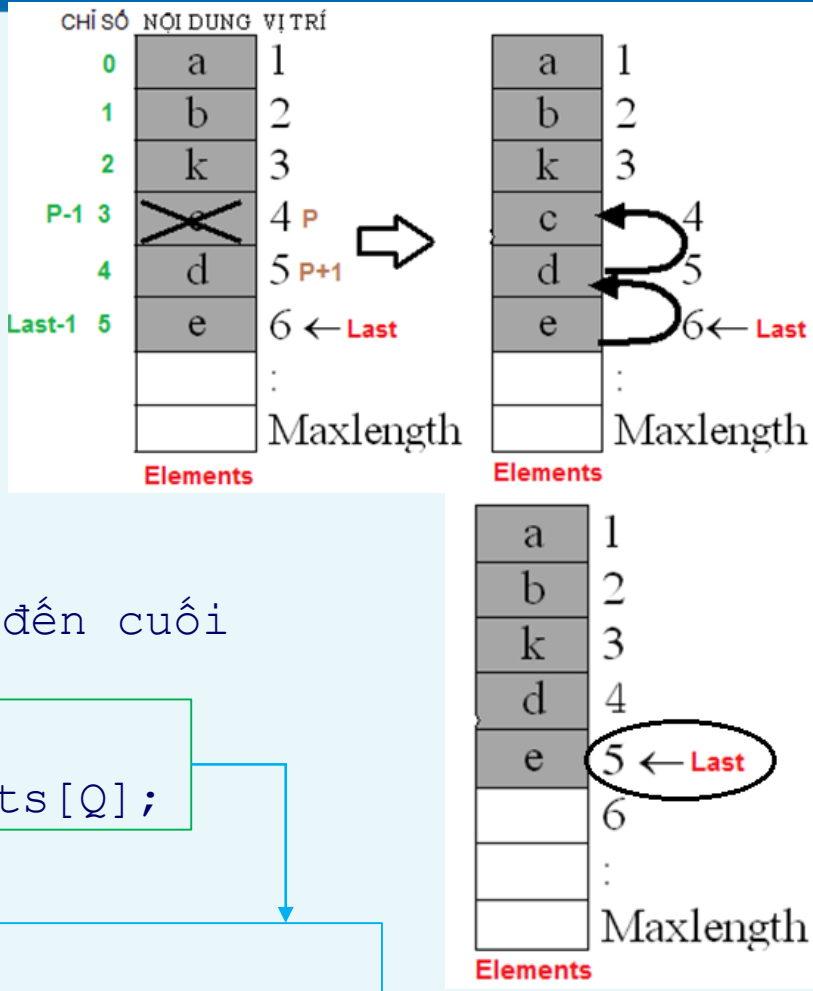

```
void deleteList(Position P,List *pL){
   if ((P<1) || (P>pL->Last))
        printf("Vi tri khong hop le");
   else if (emptyList(*pL))
        printf("Danh sach rong!");
   else{
        Position Q;
        /*Dời các phần tử từ vị trí P+1 đến cuối
          danh sách ra trước 1 vị trí*/
        for(Q=P;Q<pL->Last;Q++)
            pL->Elements[Q-1]=pL->Elements[Q];
        pL->Last--;
   }
}
```

XÓA MỘT PHẦN TỬ TẠI VỊ TRÍ P TRONG DS

```
void deleteList(Position P,List *pL){
   if ((P<1) || (P>pL->Last))
        printf("Vi tri khong hop le");
   else if (emptyList(*pL))
        printf("Danh sach rong!");
   else{
        Position Q;
        /*Dời các phần tử từ vị trí P+1 đến cuối
          danh sách ra trước 1 vị trí*/
        for(Q=P;Q<pL->Last;Q++)
            pL->Elements[Q-1]=pL->Elements[Q];
        pL->Last--;
   }
}
```

pL->Last==0

```
for(Q=P-1;Q<pL->Last-1;Q++)
     pL->Elements[Q]=pL->Elements[Q+1];
```



Độ phức tạp của deleteList()?

XÁC ĐỊNH VÍ TRÍ KẾ TIẾP/TRƯỚC

- Xác định vị trí kế tiếp trong danh sách.

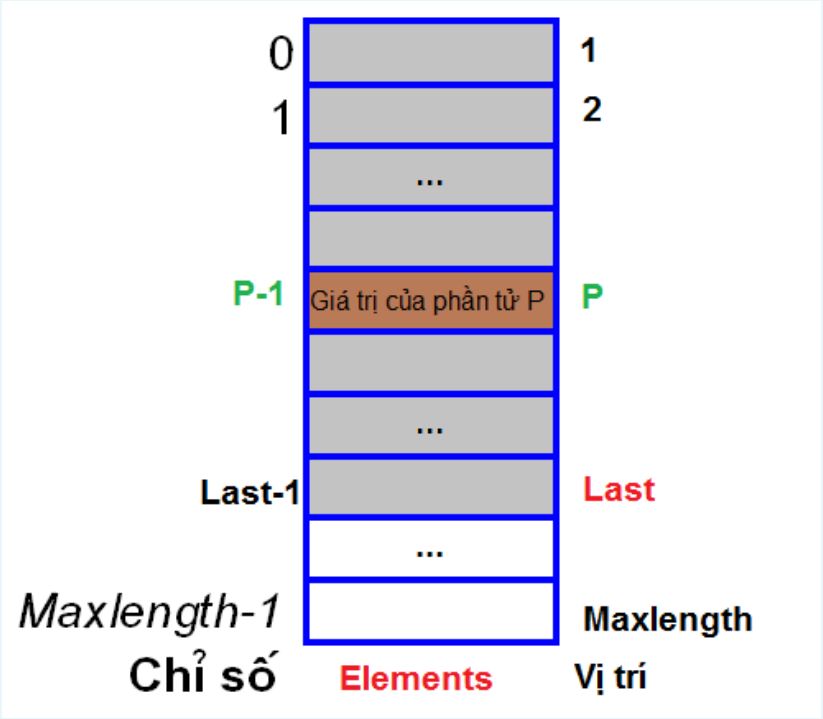
```
Position next(Position P, List L){
   return P+1;
}
```

- Xác định vị trí trước trong danh sách.

```
Position previous(Position P, List L){
   return P-1;
}
```

XÁC ĐỊNH NỘI DUNG PHẦN TỬ TẠI VỊ TRÍ P

- Xác định nội dung phần tử tại vị trí P trong DS.




```
ElementType retrieve(Position P, List L){
   return L.Elements[P-1];
}
```

TÌM KIẾM PHẦN TỬ x TRONG DS

Cài đặt

```
Position locate(ElementType x, List L){
   Position P;
   int Found = 0;
   P = first(L); //vị trí phần tử đầu tiên
   /*trong khi chưa tìm thấy và chưa kết
   thúc danh sách thì xét phần tử kế tiếp*/
   while ((P != endList(L)) && (Found == 0))
      if (retrieve(P,L) == x) Found = 1;
      else P = next(P, L);
   return P;
}
```

TÌM KIẾM PHẦN TỬ x TRONG DS

Cài đặt

```
Position locate(ElementType x, List L){
   Position P;
   P = 1; //vị trí phần tử đầu tiên
   /*trong khi chưa tìm thấy và chưa kết
   thúc danh sách thì xét phần tử kế tiếp*/
   while (P != L.Last+1)
      if (L.Elements[P-1] == x) return P;
      else P = P+1;
   return P;
}
```

Độ phức tạp của locate()?

IN DANH SÁCH

- In danh sách.

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

Độ phức tạp của printList()?

IN DANH SÁCH

- In danh sách.

```
void printList(List L){
   Position P= 1;
   while (P != L.Last+1){
         printf("%d ", L.Elements[P-1]);
         P=P+1;
   }
   printf("\n");
}
```

NHẬP DANH SÁCH TỪ BÀN PHÍM

```
void readList(List *pL){
   int i,n;
   ElementType x;
   makenullList(pL);
   scanf("%d",&n);
   for(i=1;i<=n;i++) {
       scanf("%d",&x);
       insertList(x,endList(*pL),pL);
     }
}
```

CHƯƠNG TRÌNH CHÍNH

```
int main(){
    List L;
    ElementType x;
    Position P;
    readList(&L); // Nhap danh sach
    printList(L); //In danh sach len man hinh
    // Nhap noi dung phan tu can them
    scanf("%d",&x);
    // Nhap vi tri can them: ");
    scanf("%d",&P);
    insertList(x,P,&L);
    // In danh sach sau khi them phan tu
    printList(L);
    // Nhap noi dung phan tu can xoa
    scanf("%d",&x);
    P=locate(x,L);
    deleteList(P,&L);
    // In danh sach sau khi xoa
    printList(L);
    return 0;                    }
```

NHẬP DANH SÁCH TỪ BÀN PHÍM

```
List readList(){
   int i,n;
   ElementType x;
   List L;
   L=makenullList();
   scanf("%d",&n);
   for(i=1;i<=n;i++) {
       scanf("%d",&x);
       insertList(x,endList(L),&L);
     }
   return L;
}
```

CHƯƠNG TRÌNH CHÍNH

```
int main(){
    List L;
    ElementType x;
    Position P;
    L=readList(); // Nhap danh sach
    printList(L); //In danh sach len man hinh

    // Nhap noi dung phan tu can them
    scanf("%d",&x);
    // Nhap vi tri can them: ");
    scanf("%d",&P);
    insertList(x,P,&L);
    // In danh sach sau khi them phan tu
    printList(L);

    // Nhap noi dung phan tu can xoa
    scanf("%d",&x);
    P=locate(x,L);
    deleteList(P,&L);
    // In danh sach sau khi xoa
    printList(L);
    return 0;                    }
```