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)*

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)*

DANH SÁCH

- 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

CÁC PHÉP TOÁN - BÀI TẬP

- Muốn thêm phần tử x vào đầu hay cuối danh sách ta gọi phép toán nào và gọi phép toán đó như thế nào?

CÁC PHÉP TOÁN - BÀI TẬP

- Muốn thêm phần tử x vào đầu hay cuối danh sách ta gọi phép toán nào và gọi phép toán đó như thế nào?

  - Thêm phần tử x vào đầu danh sách

    ```
    insertList(x, first(L), L)
    ```

  - Thêm phần tử x vào cuối danh sách

    ```
    insertList(x, endList(L), L)
    ```

CÁC PHÉP TOÁN - BÀI TẬP

- Cho hàm swap(p,q) hoán đổi nội dung của hai phần tử tại vị trí p và q trong danh sách. Hãy dùng các phép toán trừu tượng trên danh sách, viết chương trình con sort (có tham số đầu vào là 1 danh sách) sắp xếp danh sách theo thứ tự tăng dần?

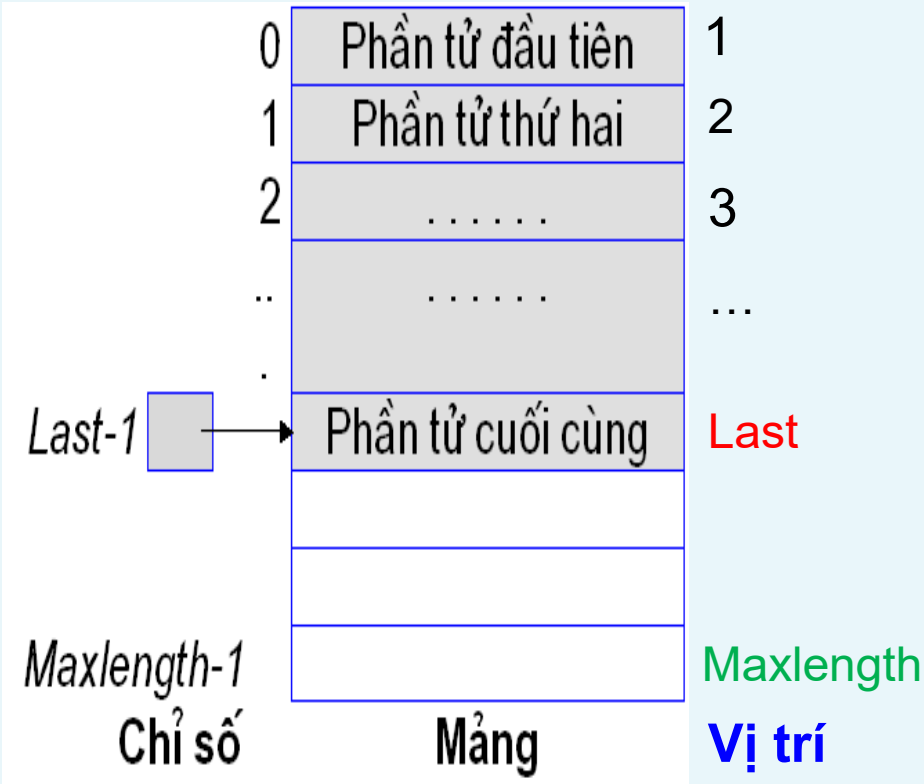
CÁC PHÉP TOÁN - BÀI TẬP

```
void sort(List L){
        Position p,q;       //kiểu vị trí của các phần tử trong danh sách
        p= first(L);        //vị trí phần tử đầu tiên trong danh sách
        while (p!=endList(L)){
                q=next(p,L);          //vị trí phần tử đứng ngay sau phần tử p
                while (q!=endList(L)){
                    if (retrieve(p,L) > retrieve(q,L))
                        swap(p,q);  //hoán đổi nội dung 2 phần tử
                    q=next(q,L);
                }
                p=next(p,L);
         }
}
```

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)

CÀI ĐẶT DANH SÁCH BẰNG MẢNG (DANH SÁCH ĐẶC)

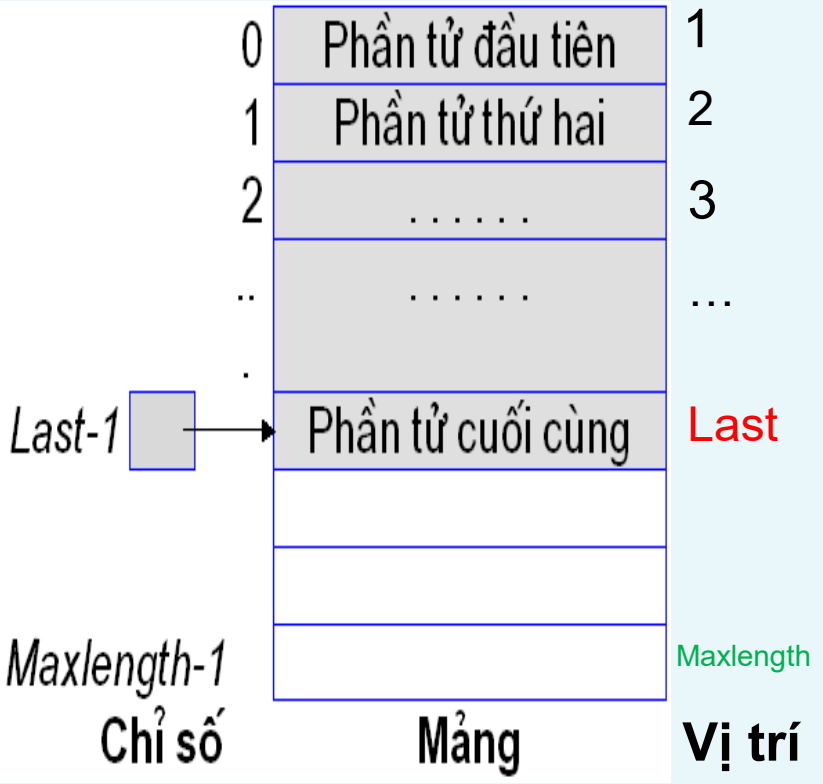


- Dùng 1 *mảng* (Elements) để lưu trữ liên tiếp các phần tử, bắt đầu từ vị trí đầu tiên.
- Phải ước lượng *số phần tử tối đa* của danh sách (Maxlength).
- Phải lưu trữ *độ dài hiện tại* của danh sách (Last).

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

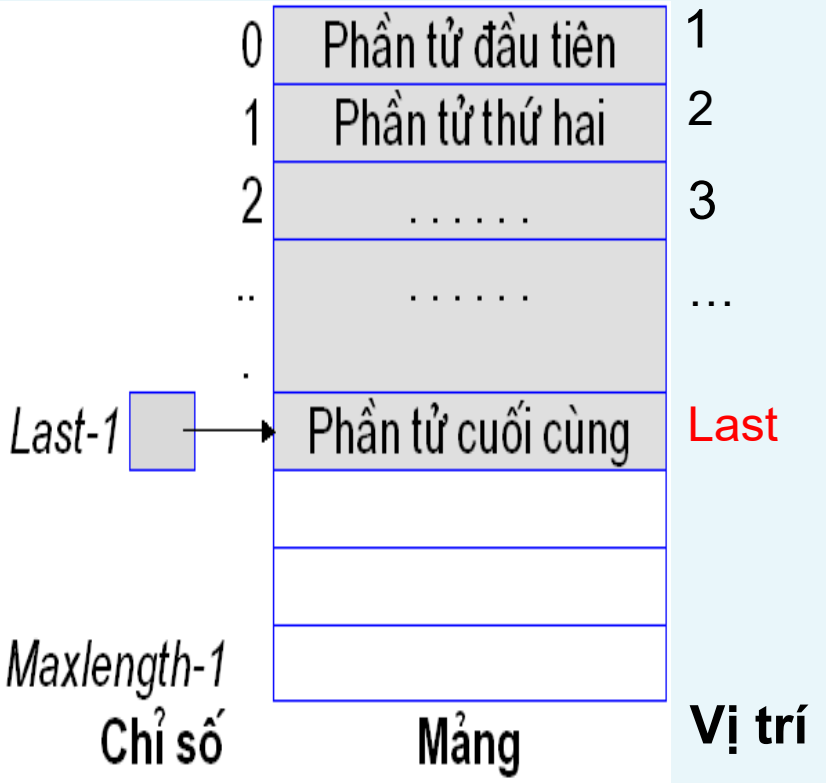
```
//Độ dài tối đa của danh sách-ds
#define MaxLength <n>
//kiểu của phần tử trong ds
typedef <datatype> ElementType;
//kiểu vị trí cuả các phần tử
typedef int Position;
typedef struct {
   //mảng chứa các phần tử của ds
   ElementType Elements[MaxLength];
   //giữ độ dài danh sách
   Position Last;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



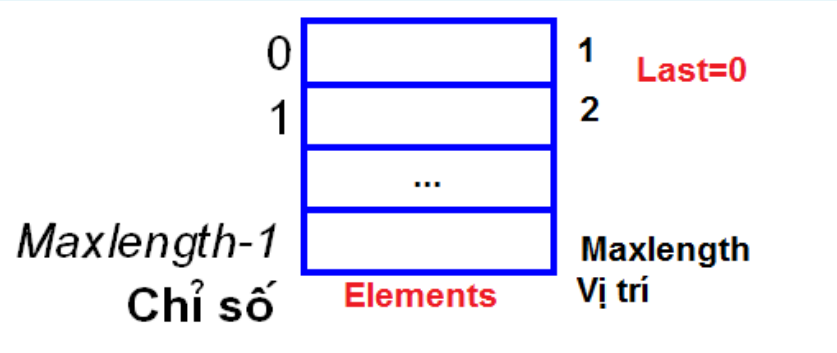
KHAI BÁO

Ví dụ: Khai báo một danh sách đặc có thể chứa tối đa 300 số nguyên?

```
//Độ dài tối đa của danh sách
#define MaxLength 300
//kiểu của phần tử trong danh sách
typedef int 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;
```



KHỞI TẠO DANH SÁCH RỖNG



- Cho hàm:

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

(*pL).Last=0

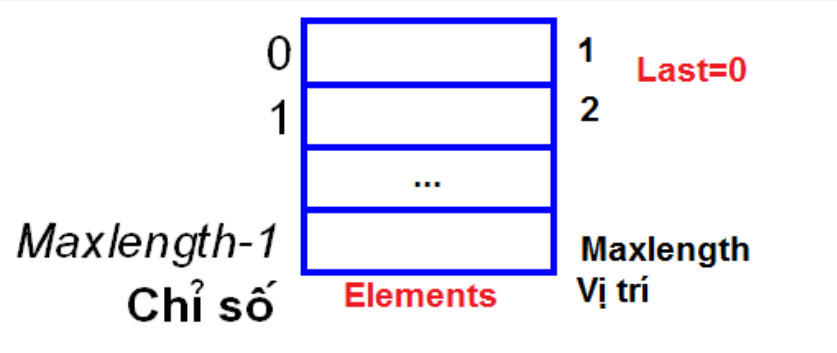
- Giả sử ta có khai báo biến:

```
List   L;
```

```
makenullList(&L);
```

Hãy viết lời gọi hàm makenullList()?

KHỞI TẠO DANH SÁCH RỖNG



- Cho hàm makenullList không có tham số vào và trả về một danh sách rỗng:

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

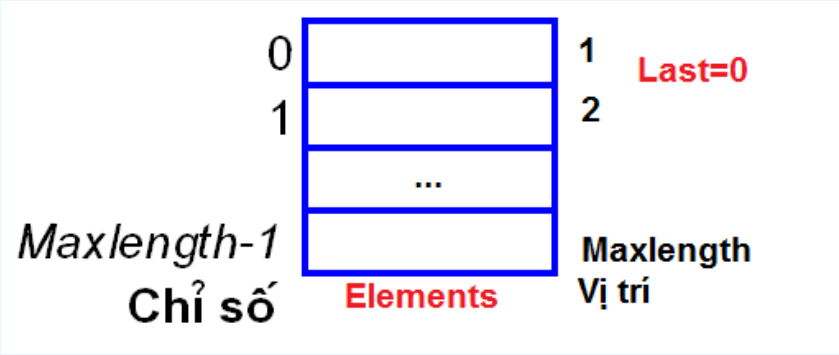
- Giả sử ta có khai báo biến:

```
List   L;
```

Hãy viết lời gọi hàm makenullList()?

```
L=makenullList();
```

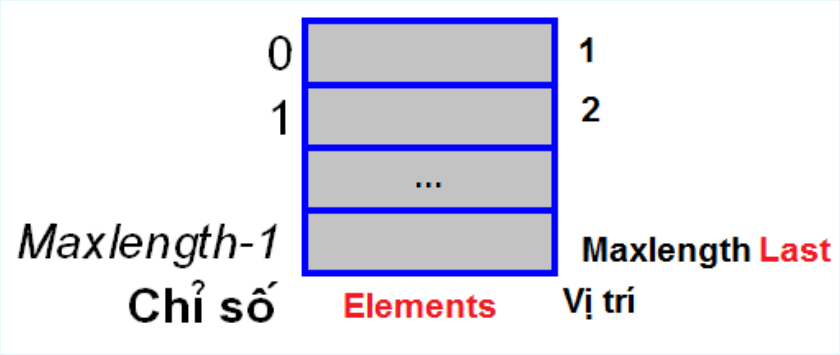
KIỂM TRA DANH SÁCH RỖNG



- Xem độ dài danh sách có bằng 0 không?

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

KIỂM TRA DANH SÁCH ĐẦY



- Xem độ dài danh sách có bằng MaxLength không?

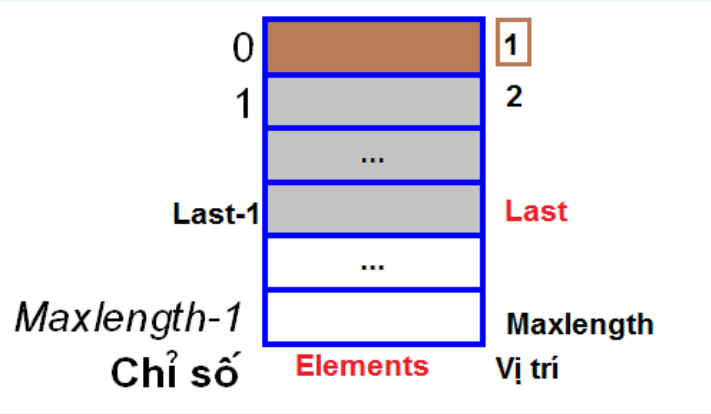
```
int fullList(List L) {
        return L.Last==MaxLength;
}
```

CANTHO UNIVERSITY

XÁC ĐỊNH VỊ TRÍ ĐẦU / SAU VỊ TRÍ CUỐI

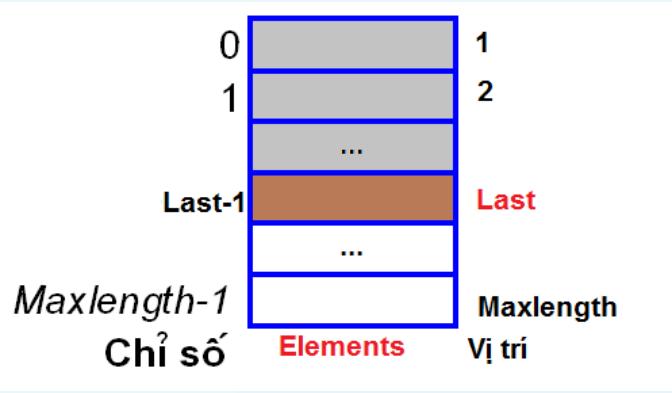
- Xác định vị trí đầu tiên trong danh sách.

```
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

- Xen phần tử x='k' vào vị trí P=3 trong danh sách L (chỉ số 2 trong mảng).

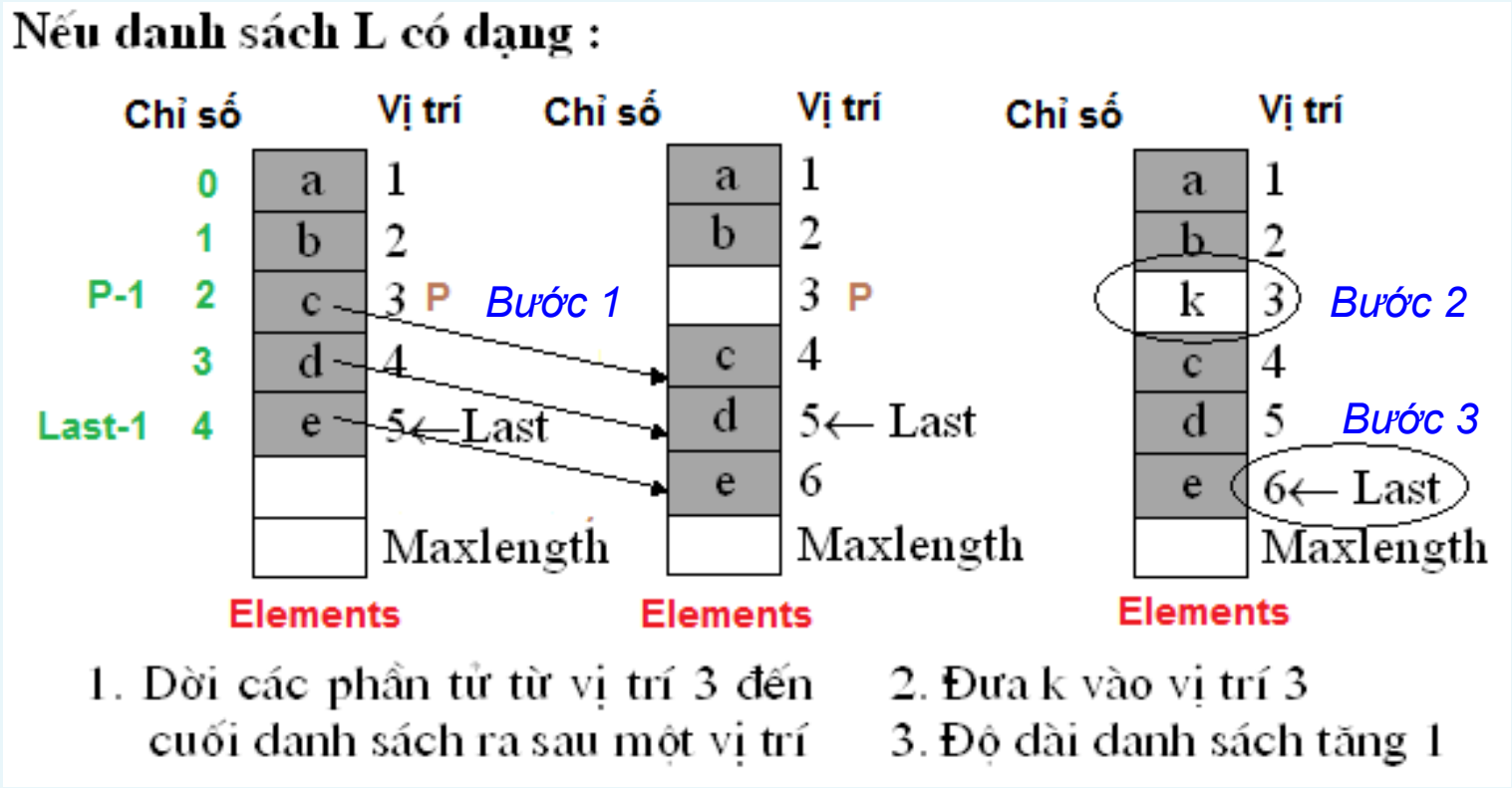
- **Nếu L có dạng :**

⇨ Trường hợp này mảng đầy => báo lỗi

Chỉ số mảng	Nội dung	Vị trí trong dsách
0	a	1
1	b	2
2	c	3
3	d	4
4	g	:
5	h	Maxlength ←Last

XEN PHẦN TỬ x VÀO VỊ TRÍ P

- Xen phần tử x=’k’ vào vị trí P=3 trong danh sách L (chỉ số 2 trong mảng).



XEN PHẦN TỬ x VÀO VỊ TRÍ P

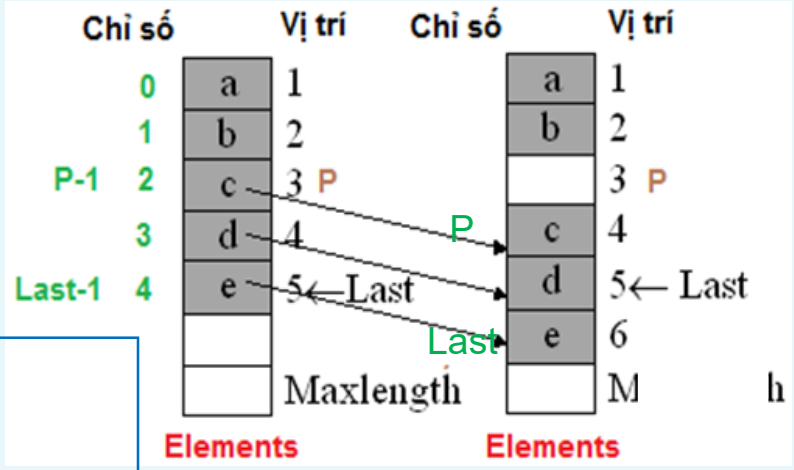
- Thuật toán

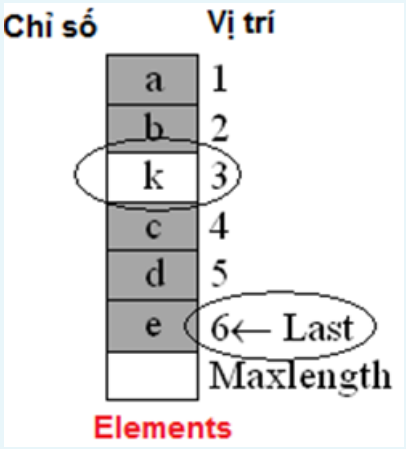
Để chèn x vào vị trí P của L, ta làm như sau:

- Nếu mảng đầy thì thông báo lỗi.
- Ngược lại, nếu vị trí P không hợp lệ thì báo lỗi.
- Ngược lại:
  - Dời các phần tử từ vị trí P đến cuối danh sách ra sau một vị trí.
  - Đưa phần tử mới x vào tại vị trí P.
  - Tăng độ dài danh sách lên 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

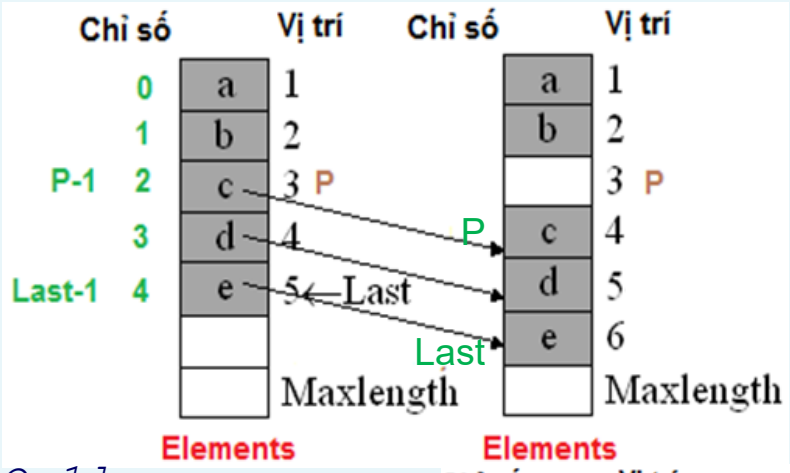
CANTHO UNIVERSITY

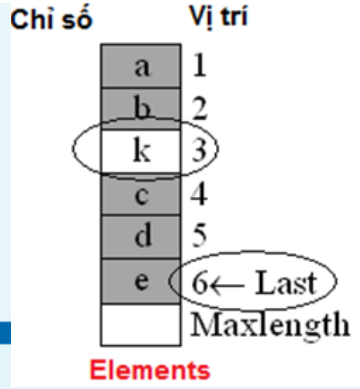
Cài đặt

```
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))

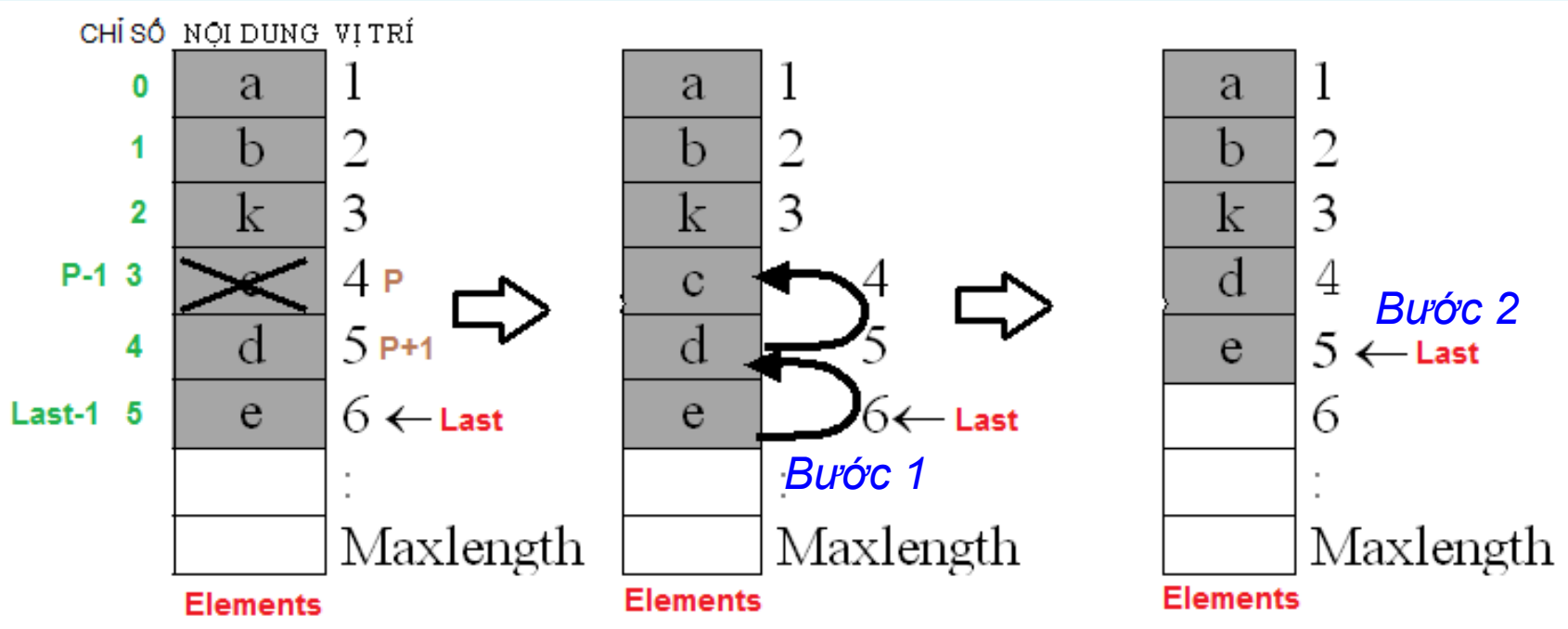




Độ phức tạp của insertList()?

XÓA MỘT PHẦN TỬ TẠI VỊ TRÍ P TRONG DS

- Ví dụ: Xóa phần tử vị trí P=4 của L.

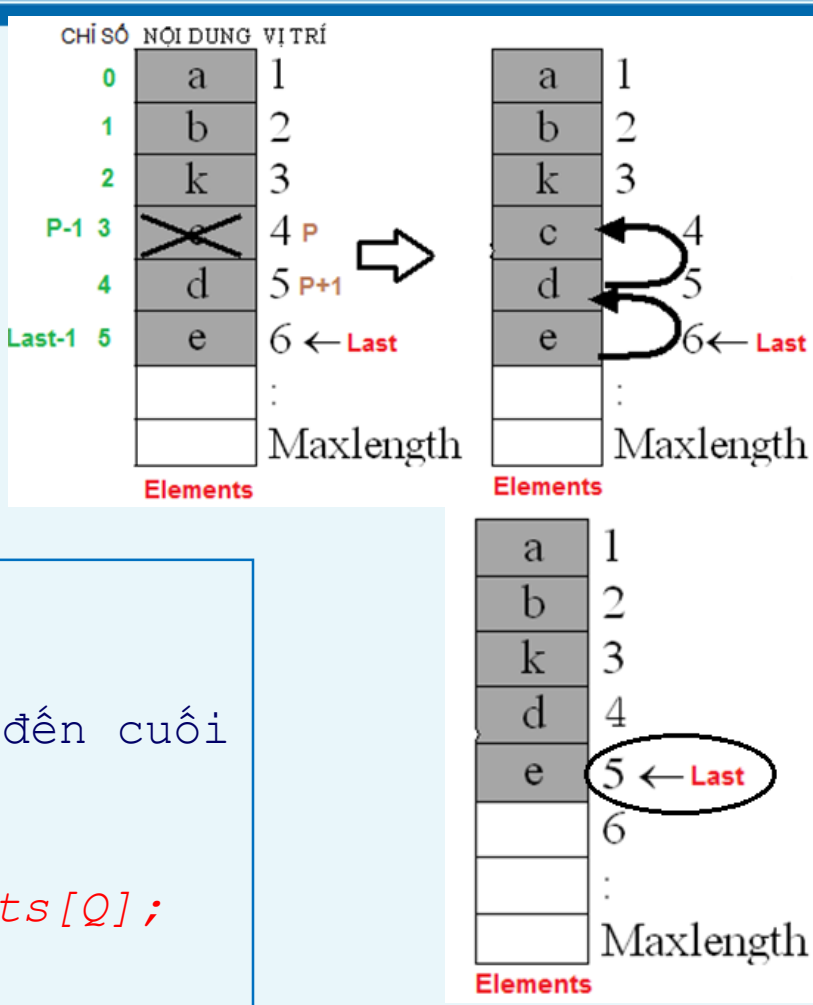


XÓA MỘT PHẦN TỬ TẠI VỊ TRÍ P TRONG DS

Thuật toán

- Nếu P là một vị trí không hợp lệ thì thông báo lỗi.
- Ngược lại, nếu danh sách rỗng thì thông báo lỗi.
- Ngược lại:
  - Di dời các phần tử từ vị trí P+1 đến cuối danh sách ra trước một vị trí.
  - Độ dài của danh sách giảm 1.

XÓA MỘT PHẦN TỬ TẠI VỊ TRÍ P TRONG DS



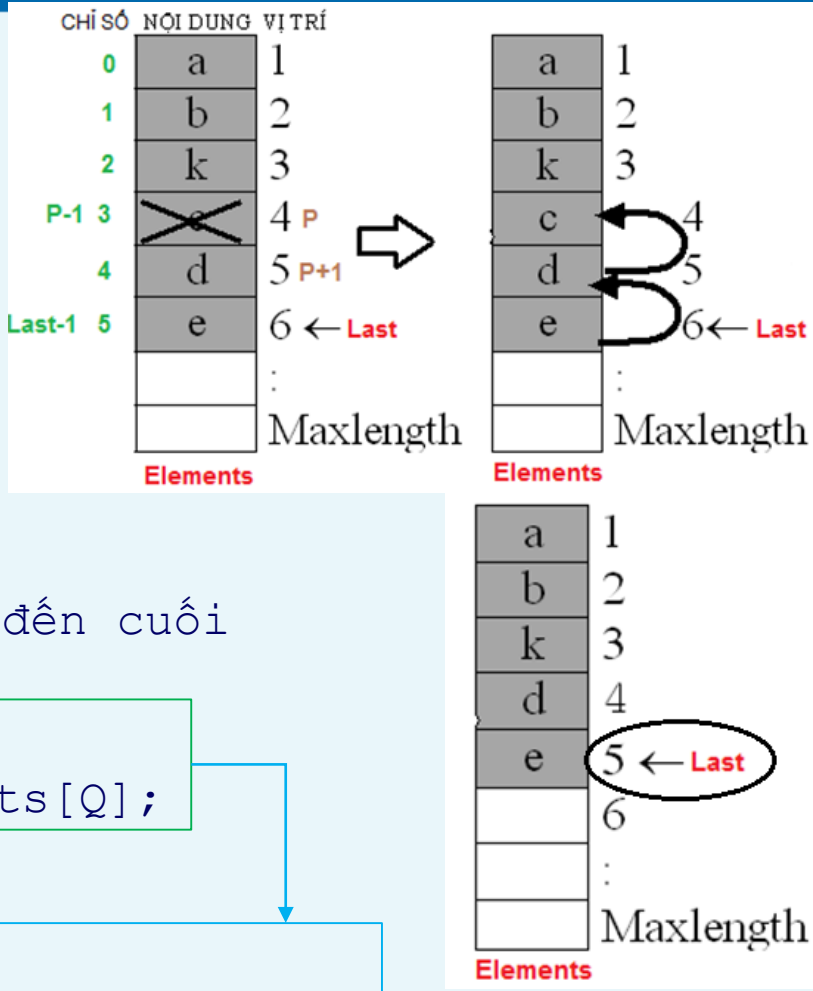

```
void deleteList(Position P,List *pL){
    if ((P<1) || (P>pL->Last))
        printf("Vi tri khong hop le");
    else if (emptyList(*pL))
        printf("Danh sach rong!");
    else{
        Position Q;
        /*Dời các phần tử từ vị trí P+1 đến cuối
          danh sách ra trước 1 vị trí*/
     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

CÁC PHÉP TOÁN KHÁC

- Xác định vị trí kế tiếp trong danh sách.

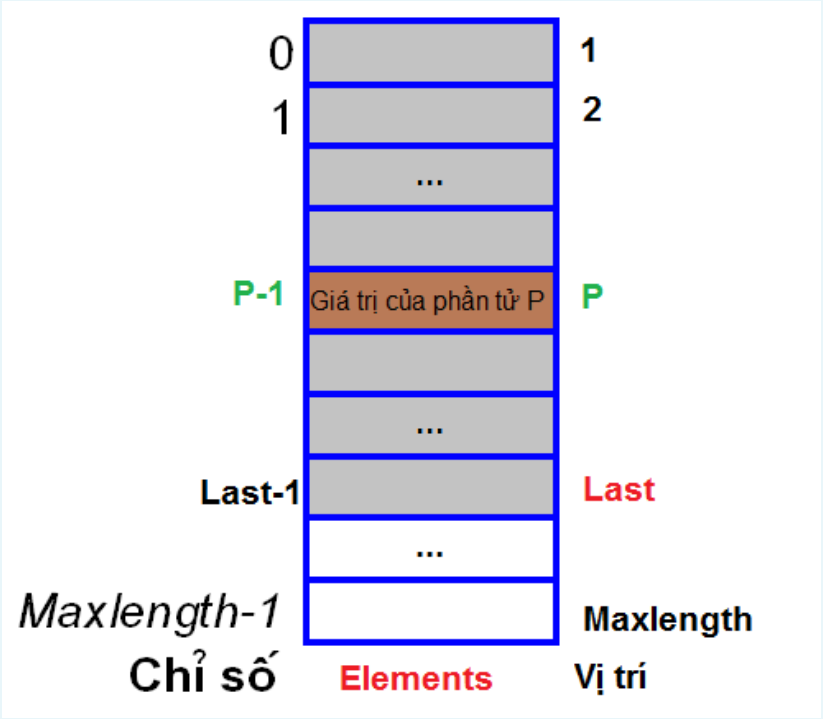
```
Position next(Position P, List L){
   return P+1;
}
```

- Xác định vị trí trước trong danh sách.

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

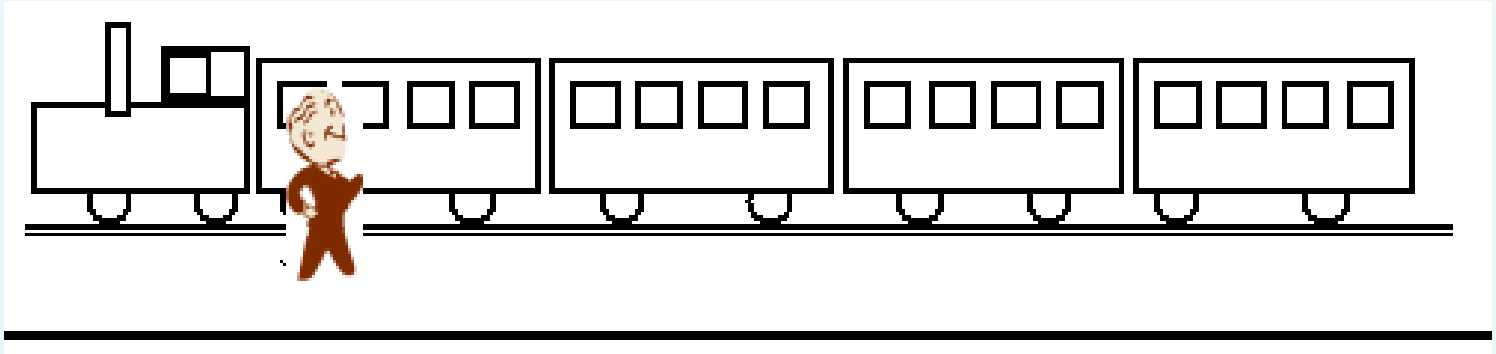
XÁC ĐỊNH NỘI DUNG PHẦN TỬ TẠI VỊ TRÍ P

- 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



Thuật toán

- 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

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

    thì dừng tìm kiếm.
  - Ngược lại (giá trị tại vị trí P khác x) thì đến vị trí kế tiếp

    ```
    P = next(P,L)
    ```
- Trả về vị trí phần tử được tìm thấy hoặc vị trí Last+1(*endList*) nếu không tìm thấy.

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()?

ĐÁNH GIÁ GIẢI THUẬT TÌM KIẾM

- Thời gian tìm kiếm
  - Nhanh nhất (tốt nhất) là khi nào, x ở đâu?
  - Xấu nhất khi nào?
- Độ phức tạp của giải thuật thường được xác định là trong trường hợp xấu nhất O(n).

CÁC PHÉP TOÁN KHÁC

- 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()?

CÁC PHÉP TOÁN KHÁC

- 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");
}
```

BÀI TẬP

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

- Nhập vào một danh sách các số nguyên và hiển thị danh sách vừa nhập ra màn hình.
- Thêm phần tử có nội dung x vào danh sách tại ví 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.
- *Sử dụng các phép toán trừu tượng trên danh sách, hãy viết hàm delete_duplicate(LIST &L) loại bỏ những giá trị trùng lặp trong danh sách.*

Q&A?

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

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

Tham số thực tế

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.

Hàm – Ví dụ

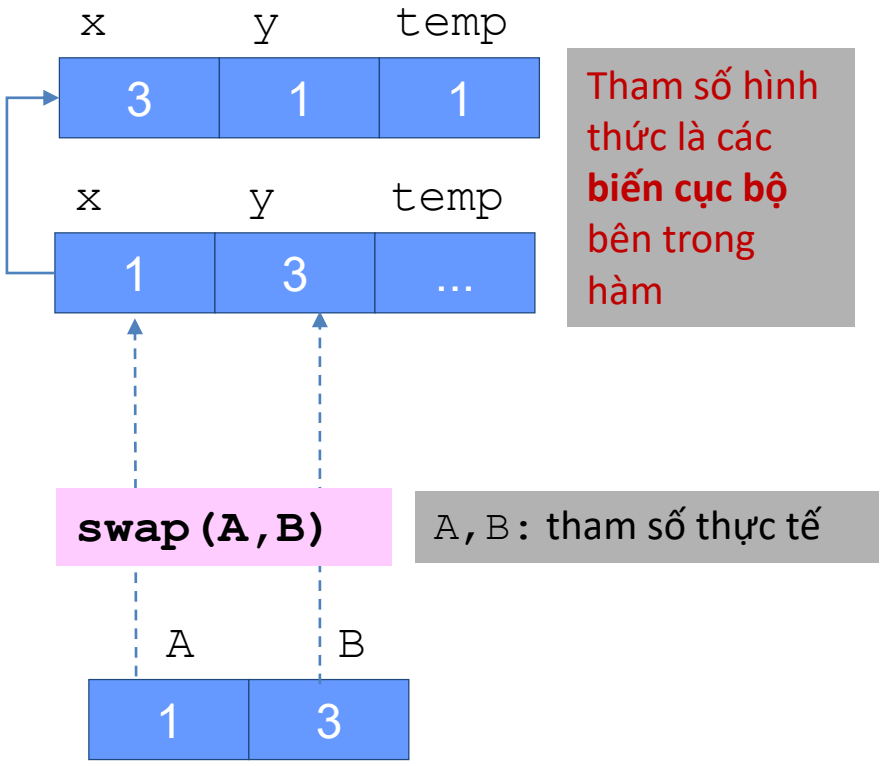
- Xét hàm swap dùng để đổi nội dung 2 biến x, y:

```
#include <stdio.h>
void swap(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}

int main(){
    int A, B;
    scanf("%d%d", &A, &B);
    printf("A=%d, B=%d\n",A,B);
    swap(A, B);
    printf("A=%d, B=%d\n",A,B);
    return 0;
}
```

Định nghĩa hàm

Gọi hàm

Truyền giá trị



Tham số hình thức là các **biến cục bộ** bên trong hàm

Kết quả chạy:

Nhập A, B: 1 3

Trước khi swap: A=1, B=3

Sau khi swap: A=1, B=3

Hàm – Ví dụ

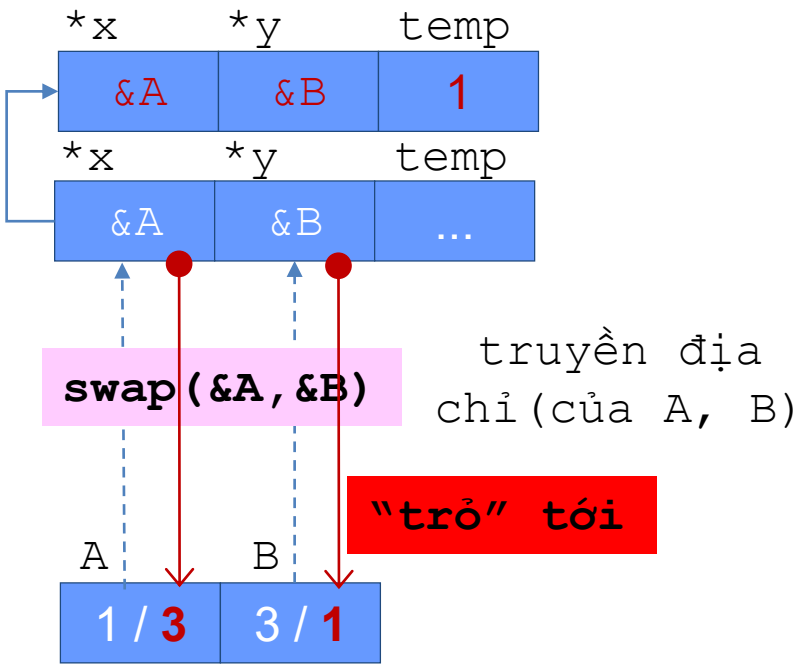
- Xét hàm swap dùng để đổi nội dung 2 biến x, y được truyền bằng con trỏ với dấu * được đặt trước 2 biến x, y:

```
#include <stdio.h>

void swap(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}

int main()
{
    int A, B;
    scanf("%d%d", &A, &B);
    printf("A=%d, B=%d\n",A,B);
    swap(&A, &B);
    printf("A=%d, B=%d\n",A,B);
    return 0;
}
```

Định nghĩa hàm

Gọi hàm

Truyền bằng con trỏ



Kết quả chạy:

Nhập A, B:1 3
Trước khi swap: A=1, B=3
Sau khi swap: A=3, B=1

Truy xuất từng trường của biến cấu trúc

- Cú pháp:

```
<tên biến kiểu cấu trúc>.<tên thành phần i>;
```

- Ví dụ:

```
struct Diem2D {
   int X;
   int Y;
} ;
struct Diem2D diem2D1;
```

C1

```
typedef struct {
   int X;
   int Y;
} Diem2D;
Diem2D diem2D1;
```

C2

```
struct Diem2D {
   int X;
   int Y;
} diem2D1;
```

C3

```
scanf("%d",&diem2D1.X);
scanf("%d",&diem2D1.Y);
```

Nhập

```
printf("%d",diem2D1.X);
printf("%d",diem2D1.Y);
```

Xuất

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");
```

  - 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;
```

BÀI TẬP

- Vận dụng các phép toán trên danh sách đặc để viết chương trình nhập vào một danh sách các số nguyên và hiển thị danh sách vừa nhập ra màn hình.
- Thêm phần tử có nội dung x vào danh sách tại ví 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.

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);
     }
}
```

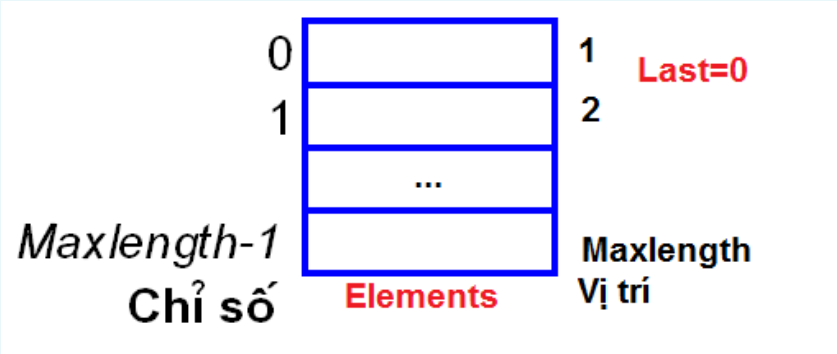
HIỂN THỊ DANH SÁCH RA MÀN HÌNH

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

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;                    }
```

KHỞI TẠO DANH SÁCH RỖNG



- Cho độ dài danh sách bằng 0.

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

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;                    }
```