



CANTHO UNIVERSITY

Cấu trúc dữ liệu

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

Bộ môn Công Nghệ Phần Mềm



NỘI DUNG SẼ HỌC

- 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 (Double – Lists)



HÀNG ĐỢI (QUEUE)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
 - DÙNG MẢNG DI CHUYỂN TỊNH TIẾN
 - DÙNG MẢNG VÒNG
 - DÙNG DSLK

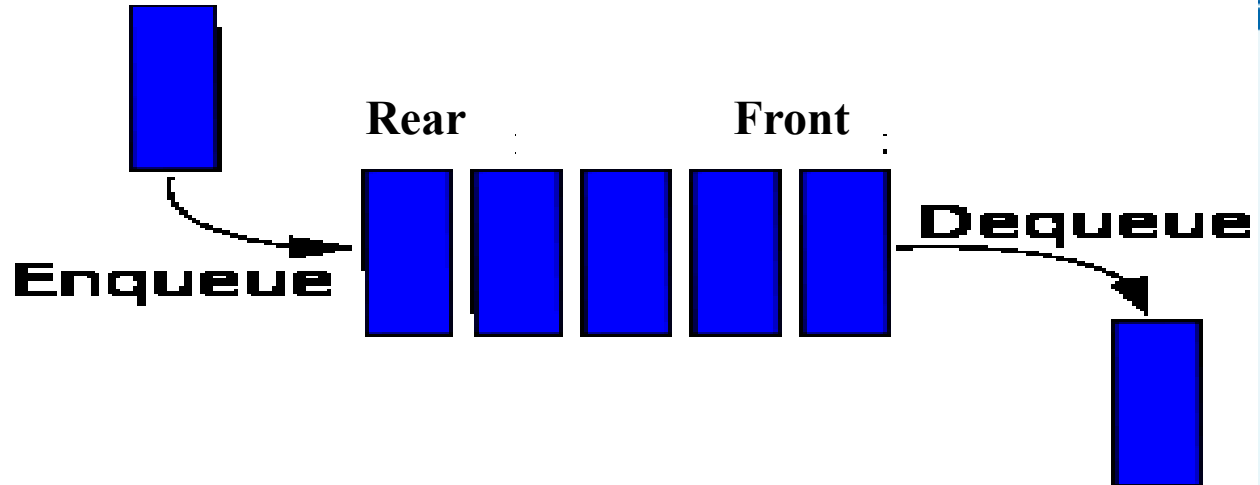


HÀNG ĐỢI (QUEUE)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
 - DÙNG MẢNG DI CHUYỂN TỊNH TIẾN
 - DÙNG MẢNG VÒNG
 - DÙNG DSLK



ĐỊNH NGHĨA HÀNG ĐỢI



Hàng đợi

- Là **một dạng danh sách đặc biệt**, mà phép thêm vào (enQueue) được thực hiện ở đầu cuối hàng (REAR), còn phép loại bỏ (deQueue) được thực hiện ở đầu kia của danh sách, gọi là đầu hàng (FRONT).
- Cách làm việc theo dạng **FIFO** (First In First Out).



HÀNG ĐỢI (QUEUE)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
 - DÙNG MẢNG DI CHUYỂN TÍNH TIẾN
 - DÙNG MẢNG VÒNG
 - DÙNG DSLK



CÁC PHÉP TOÁN

Phép toán	Diễn giải
<code>makenullQueue(Q)</code>	Tạo một hàng đợi rỗng (Q)
<code>emptyQueue(Q)</code>	Kiểm tra xem hàng đợi Q có rỗng không
<code>fullQueue(Q)</code>	Kiểm tra hàng đầy
<code>enqueue(x, Q)</code>	Thêm phần tử x vào cuối hàng đợi Q
<code>dequeue(Q)</code>	Xóa phần tử tại đầu hàng đợi Q
<code>front(Q)</code>	Trả về phần tử đầu tiên của hàng đợi Q



HÀNG ĐỢI (QUEUE)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
 - DÙNG MẢNG DI CHUYỂN TÌNH TIẾN
 - DÙNG MẢNG VÒNG
 - DÙNG DSLK

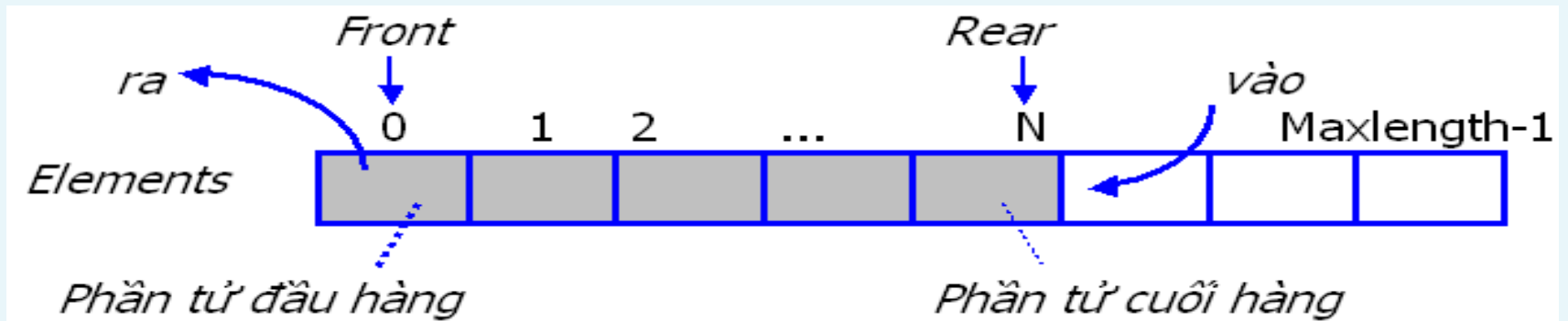


CANTHO UNIVERSITY

CÀI ĐẶT HÀNG BẰNG MẢNG DI CHUYỂN TÌNH TIẾN

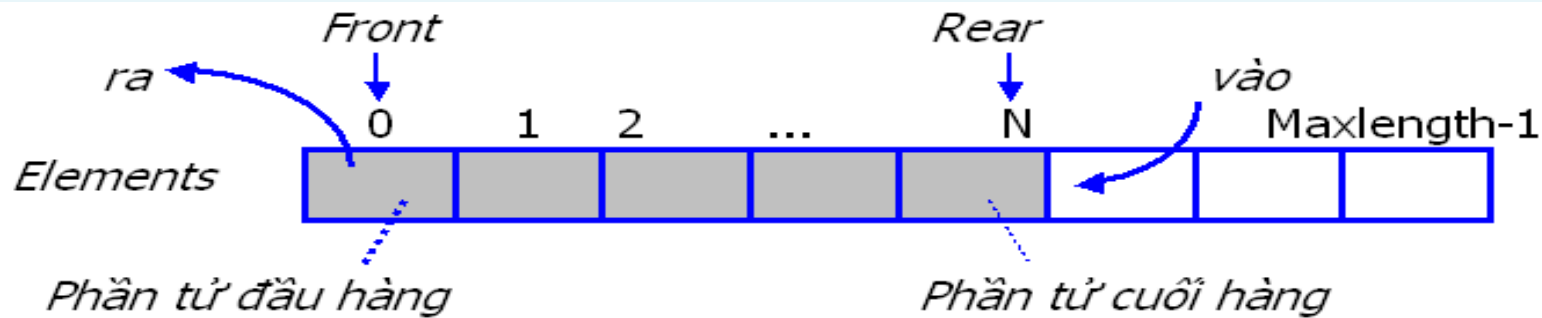


- Mô hình





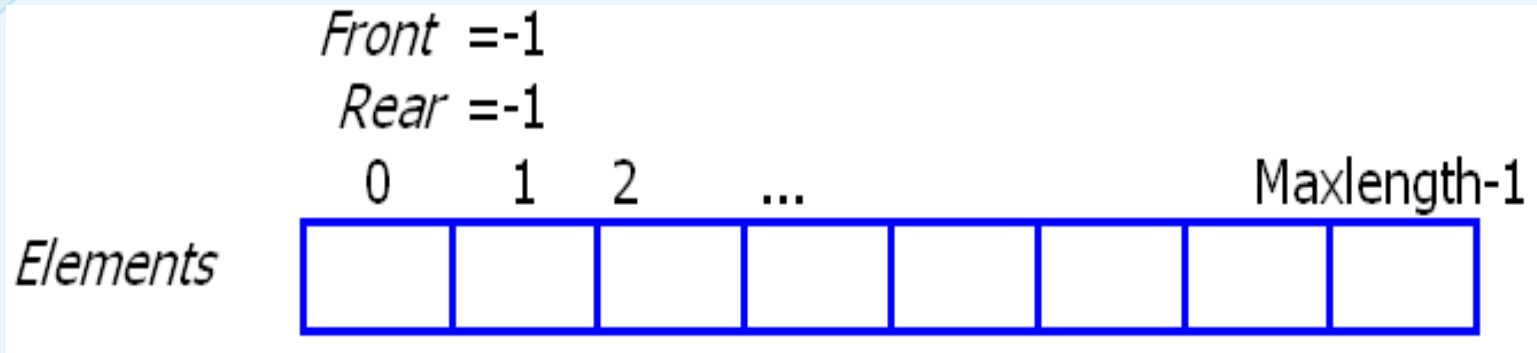
KHAI BÁO



```
//Chiều dài tối đa của mảng
#define MaxLength ...
//Kiểu dữ liệu của các phần tử trong hàng
typedef ... ElementType;
typedef struct {
    //Lưu trữ nội dung các phần tử
    ElementType Elements[MaxLength];
    //Chỉ số đầu và cuối hàng
    int Front, Rear;
} Queue;
Queue Q;
```



KHỞI TẠO HÀNG Q RỖNG

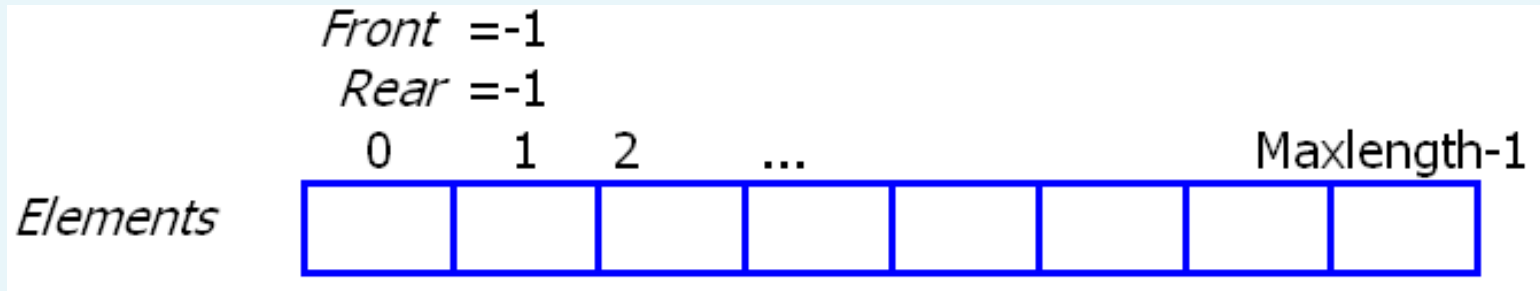


- Front và Rear không trở đến vị trí hợp lệ nào
- Ta cho $Front=Rear=-1$

```
void makenullQueue (Queue *pQ) {  
    pQ->Front=-1;  
    pQ->Rear=-1;  
}
```



KIỂM TRA HÀNG RỖNG

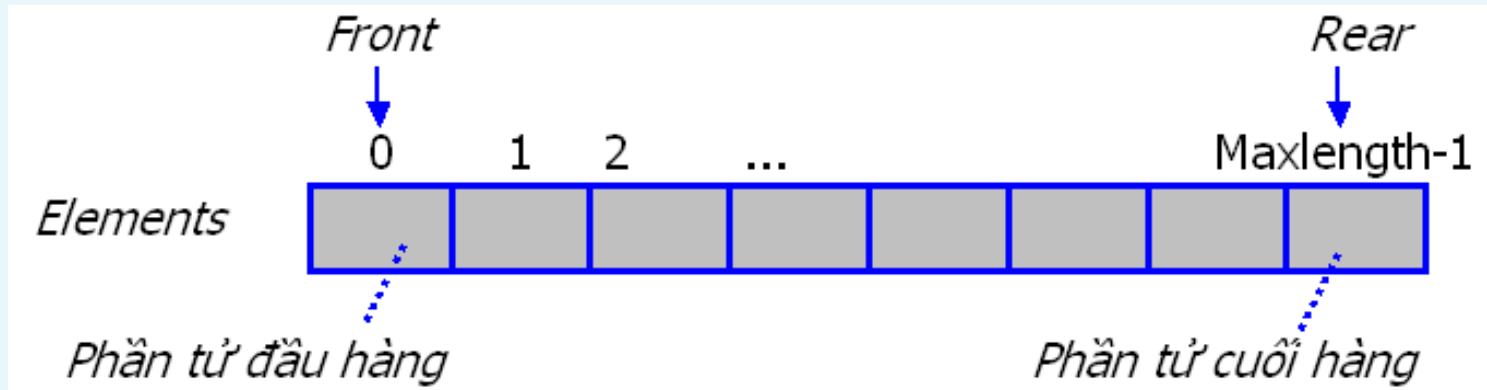


– Hàng rỗng khi $front = -1$

```
int emptyQueue (Queue Q) {  
    return (Q.Front == -1);  
}
```



KIỂM TRA HÀNG ĐẦY



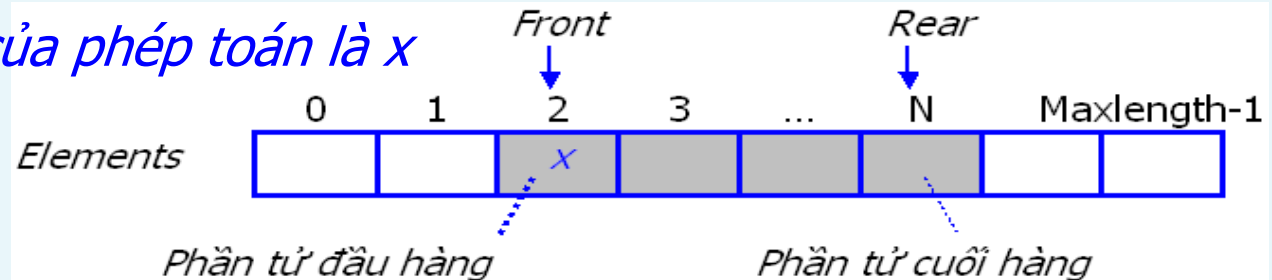
– Hàng đầy khi số phần tử hiện có trong hàng=Maxlength

```
int fullQueue (Queue Q) {  
    return ((Q.Rear-Q.Front+1)==MaxLength) ;  
}
```



TRẢ VỀ PHẦN TỬ ĐẦU HÀNG

Kết quả của phép toán là x



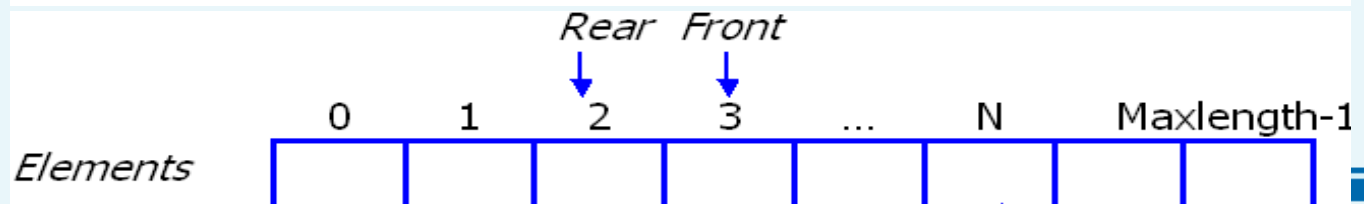
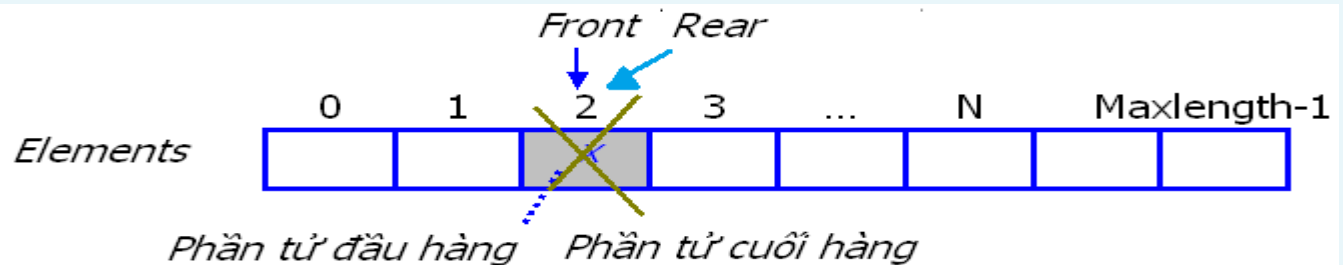
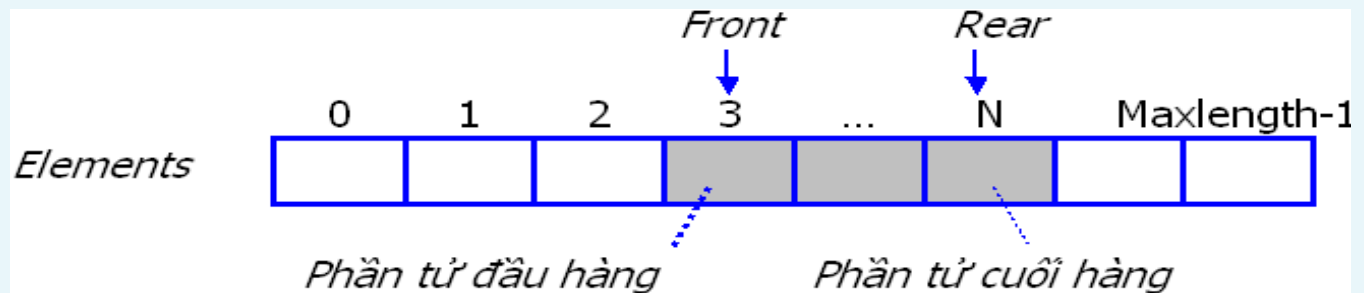
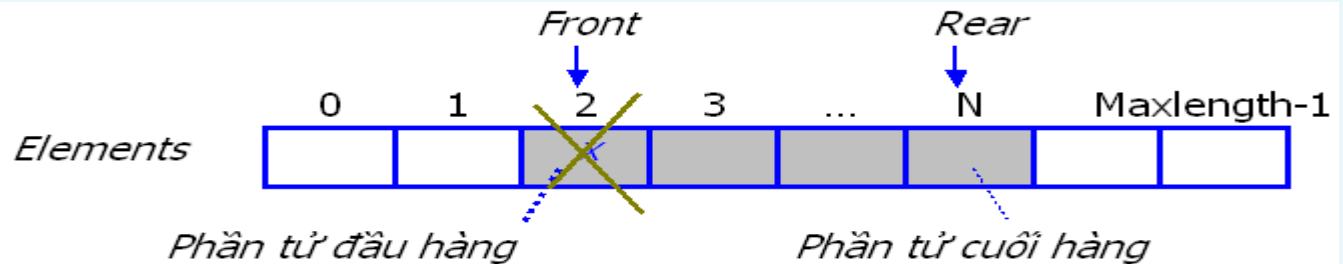
=> Giải thuật:

- Nếu hàng Q rỗng thì thông báo lỗi
- Ngược lại, trả về giá trị được lưu trữ tại ô có chỉ số là Front

```
ElementType front (Queue Q) {  
    if emptyQueue (Q)  
        printf ("Hang rong");  
    else  
        return Q.Elements [Q.Front];  
}
```



XÓA MỘT PHẦN TỬ KHỎI HÀNG





XÓA MỘT PHẦN TỬ KHỎI HÀNG

Giải thuật:

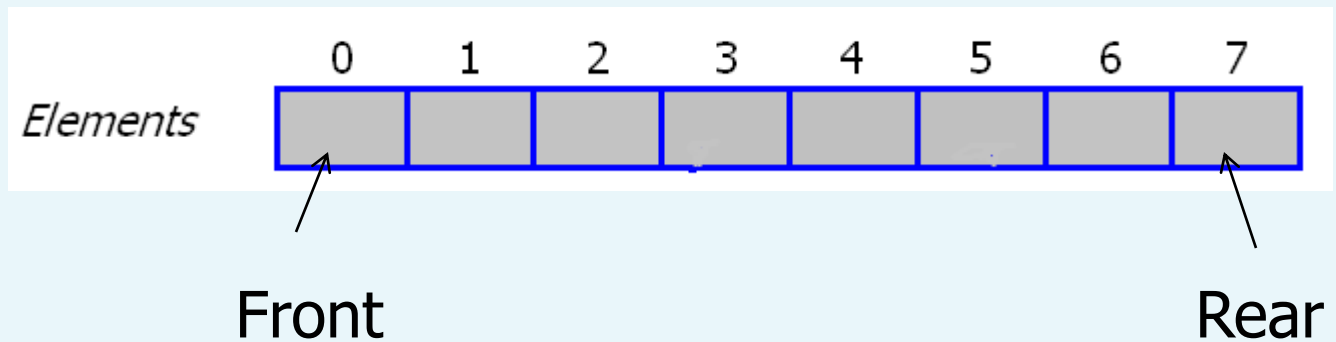
- Nếu hàng Q rỗng thì thông báo lỗi.
- Ngược lại:
 - Tăng Front lên 1 đơn vị.
 - Nếu $(\text{Front} > \text{Rear})$ tức hàng chỉ còn 1 phần tử thì khởi tạo lại hàng rỗng.

```
void deQueue (Queue *pQ) {  
    if (!emptyQueue (*pQ) ) {  
        pQ->Front=pQ->Front+1;  
        if (pQ->Front>pQ->Rear)  
            makenullQueue (Q); //Dat lai hang rong  
    }  
    else printf("Loi: Hang rong!");  
}
```




THÊM MỘT PHẦN TỬ VÀO HÀNG

- Trường hợp hàng đầy

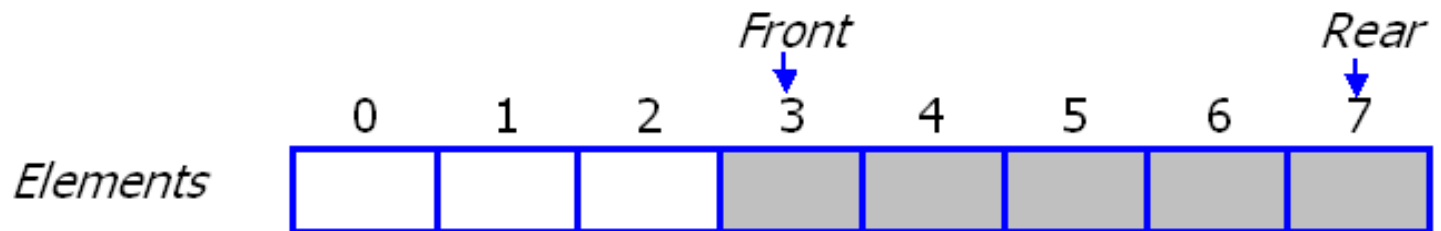


Thông báo lỗi “Lỗi: hàng đầy!”

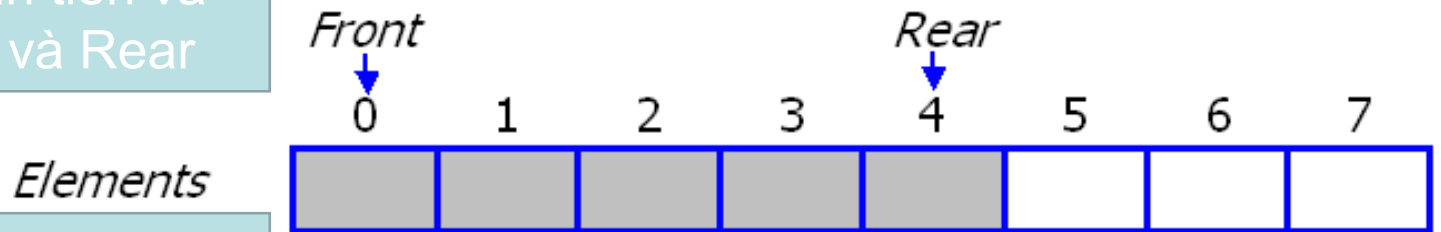


THÊM MỘT PHẦN TỬ VÀO HÀNG

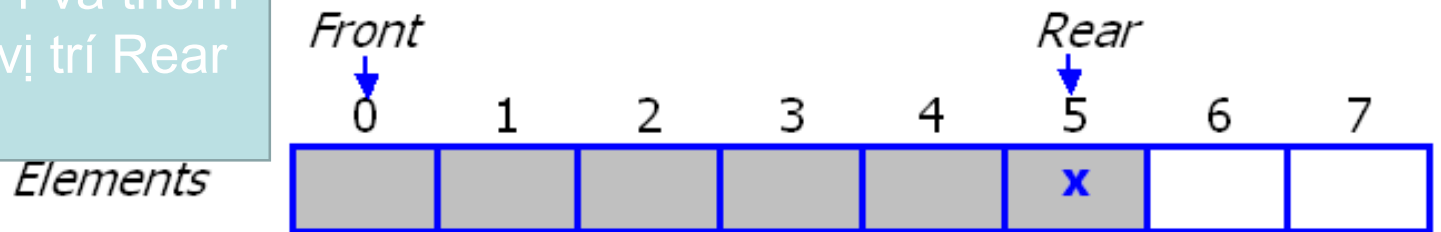
- Trường hợp hàng chưa đầy nhưng bị tràn



Di chuyển tịnh tiến và
gán lại Front và Rear



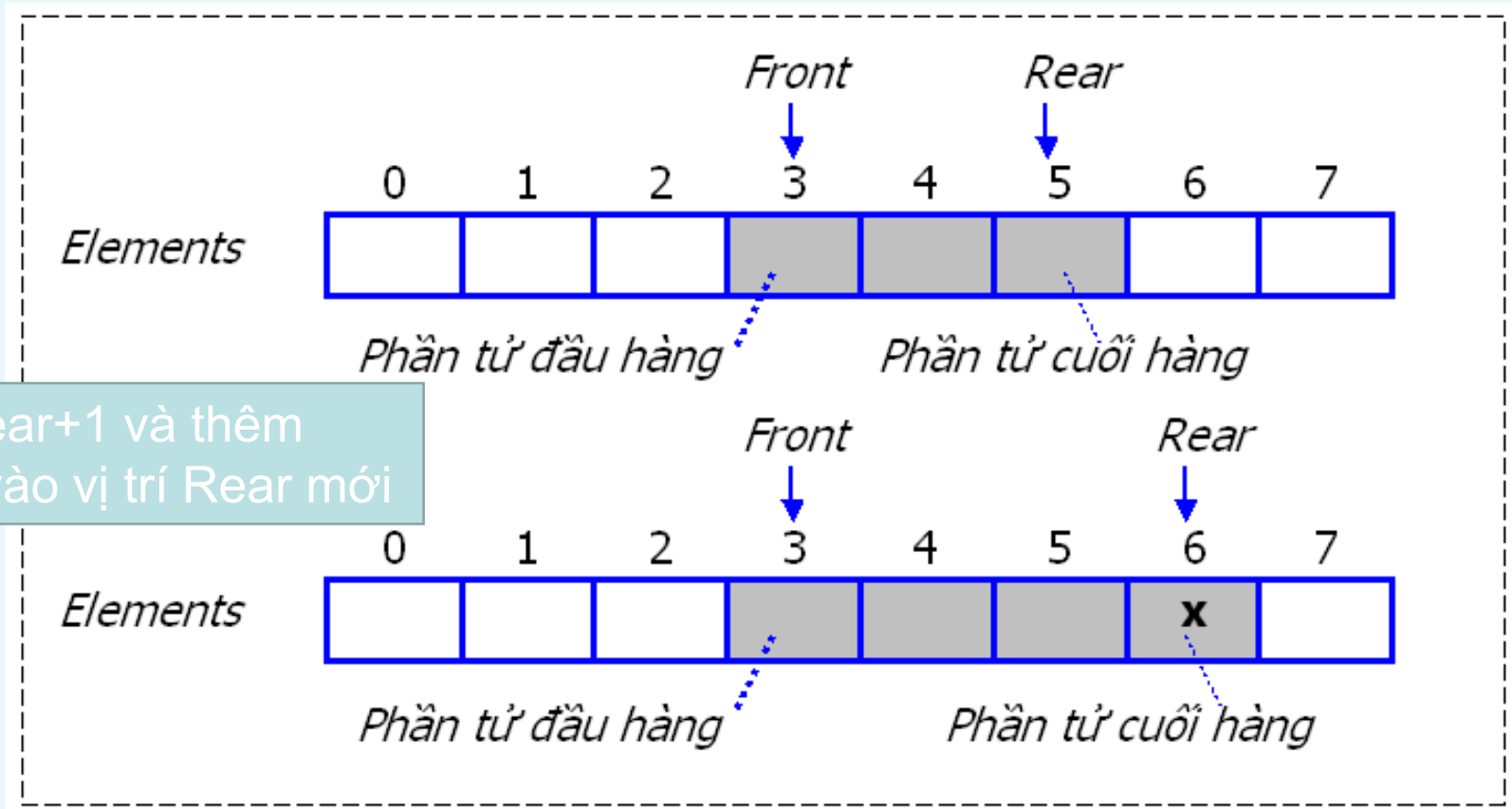
$Rear = Rear + 1$ và thêm
phần tử vào vị trí Rear
mới





THÊM MỘT PHẦN TỬ VÀO HÀNG

- Trường hợp hàng chưa đầy và không bị tràn





THÊM MỘT PHẦN TỬ VÀO HÀNG

=> Giải thuật:

- Nếu hàng đầy thì thông báo lỗi.
- Ngược lại, nếu hàng tràn thì phải tịnh tiến tất cả phần tử lên “Front-1” vị trí.
- Tăng Rear 1 đơn vị và đưa giá trị x vào ô có chỉ số Rear mới này.



THÊM MỘT PHẦN TỬ VÀO HÀNG

```
void enqueue(ElementType x, Queue *pQ) {
    if (!fullQueue(*pQ)) {
        if (emptyQueue(*pQ)) pQ->Front=0;
        if (pQ->Rear==MaxLength-1) {
            //Di chuyển tính tiền ra trước Front -1 vì tri
            for(int i=pQ->Front; i<=pQ->Rear; i++)
                pQ->Elements[i-pQ->Front]=pQ->Elements[i];
            //Xác định vị trí Rear mới
            pQ->Rear=MaxLength - pQ->Front-1;
            pQ->Front=0;
        }
        //Tăng Rear để lưu nội dung mới
        pQ->Rear=pQ->Rear+1;
        pQ->Elements[pQ->Rear]=x;
    }
    else printf("Lỗi: Hàng đầy!");
}
```



HÀNG ĐỢI (QUEUE)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
 - DÙNG MẢNG DI CHUYỂN TÌNH TIẾN
 - DÙNG MẢNG VÒNG
 - DÙNG DSLK

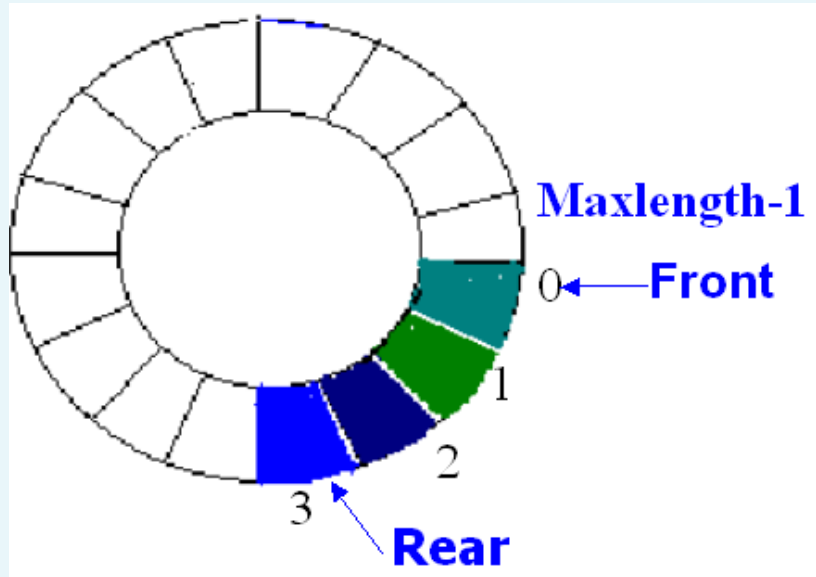


CÀI ĐẶT HÀNG BẰNG MẢNG VÒNG

- Mô hình

- Khai báo

```
#define MaxLength ...  
typedef ... ElementType;  
typedef struct {  
    //Lưu trữ nội dung các phần tử  
    ElementType Elements[MaxLength];  
    //Chỉ số đầu và đuôi hàng  
    int Front, Rear;  
} Queue;
```





KHỞI TẠO HÀNG RỖNG

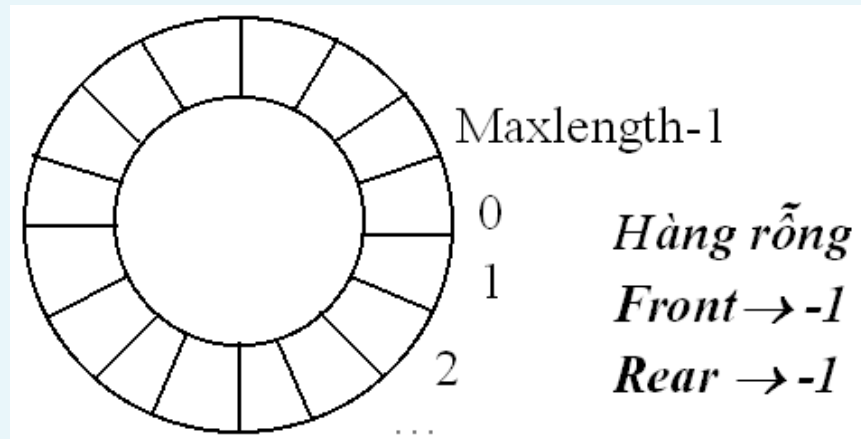


- Front và Rear không trở đến vị trí hợp lệ nào
- Ta cho $Front = Rear = -1$

```
void makenullQueue (Queue *pQ) {  
    pQ->Front=-1;  
    pQ->Rear=-1  
}
```



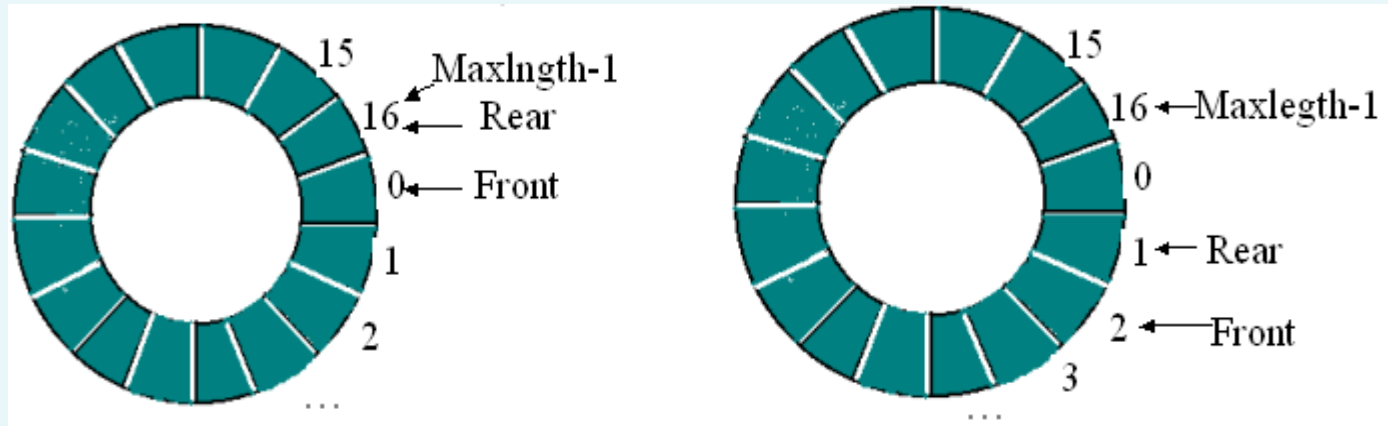

KIỂM TRA HÀNG RỖNG



```
int emptyQueue (Queue Q) {  
    return Q.Front == -1;  
}
```



KIỂM TRA HÀNG ĐẦY



Trường hợp $Q.Rear = Maxlength - 1$
và $Q.Front = 0$

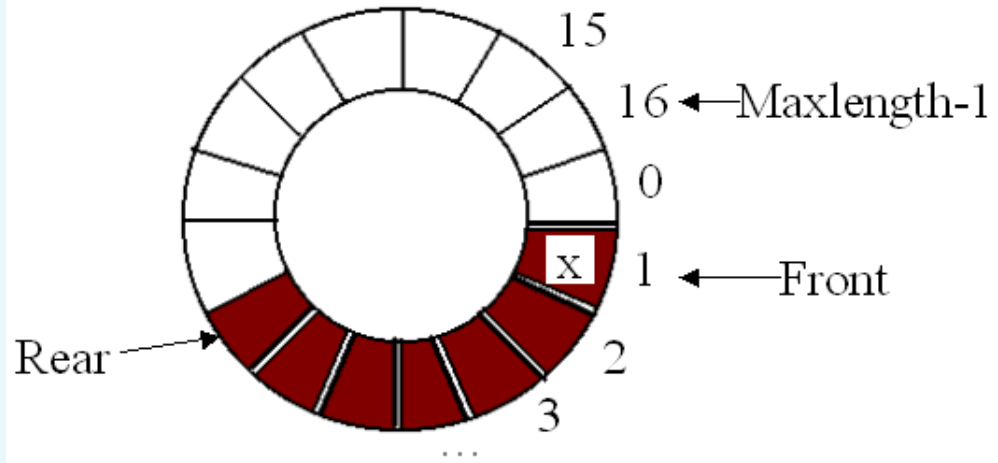
Trường hợp $Q.Front = Q.Rear + 1$

- Hàng đầy khi số phần tử hiện có trong hàng bằng $Maxlength$

```
int fullQueue (Queue Q) {  
    return (Q.Rear - Q.Front + 1) % MaxLength == 0;  
}
```



LẤY GIÁ TRỊ PHẦN TỬ ĐẦU HÀNG



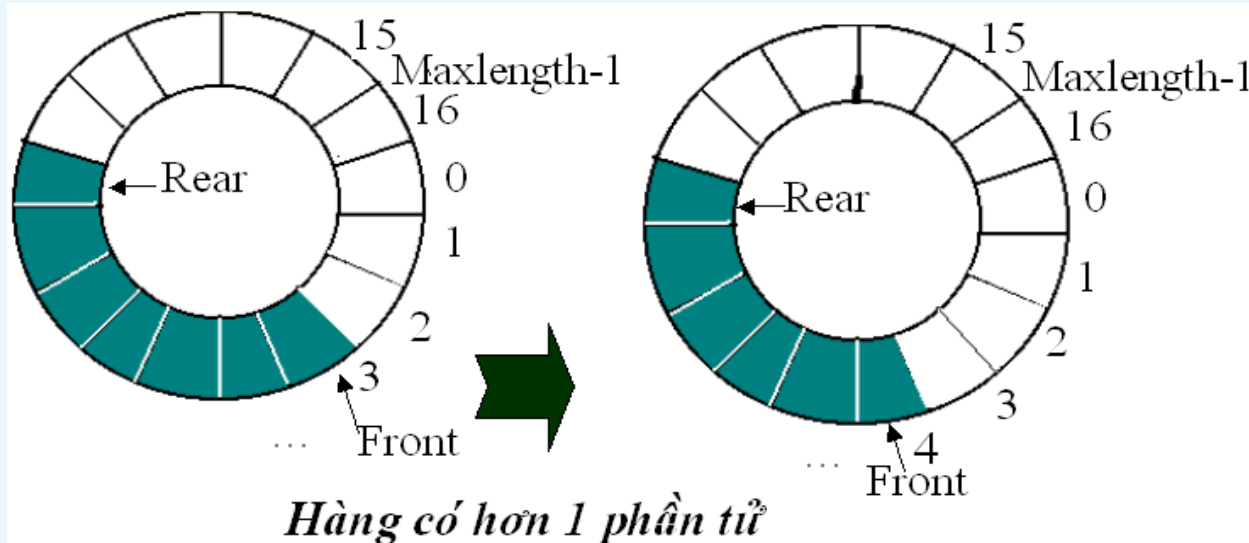
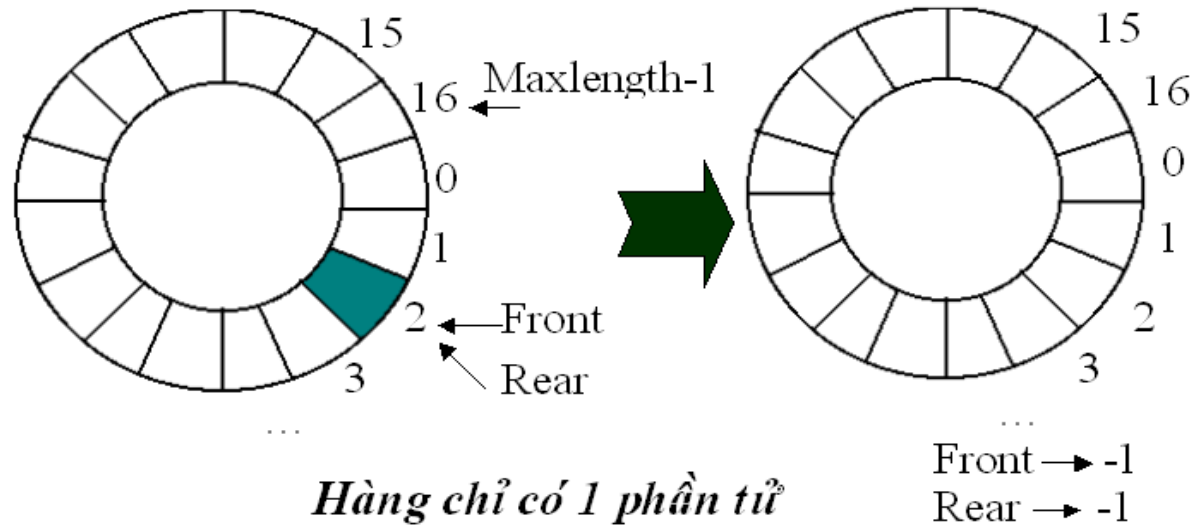
=>Giải thuật

- Nếu hàng Q rỗng thì thông báo lỗi
- Ngược lại, trả về giá trị được lưu trữ tại ô có chỉ số là Front

```
ElementType front (Queue Q) {  
    if (emptyQueue (Q))  
        printf ("Hang rong");  
    else return Q.Elements[Q.Front];  
}
```

XÓA PHẦN TỬ ĐẦU HÀNG

Các trường hợp
có thể:





XÓA PHẦN TỬ ĐẦU HÀNG

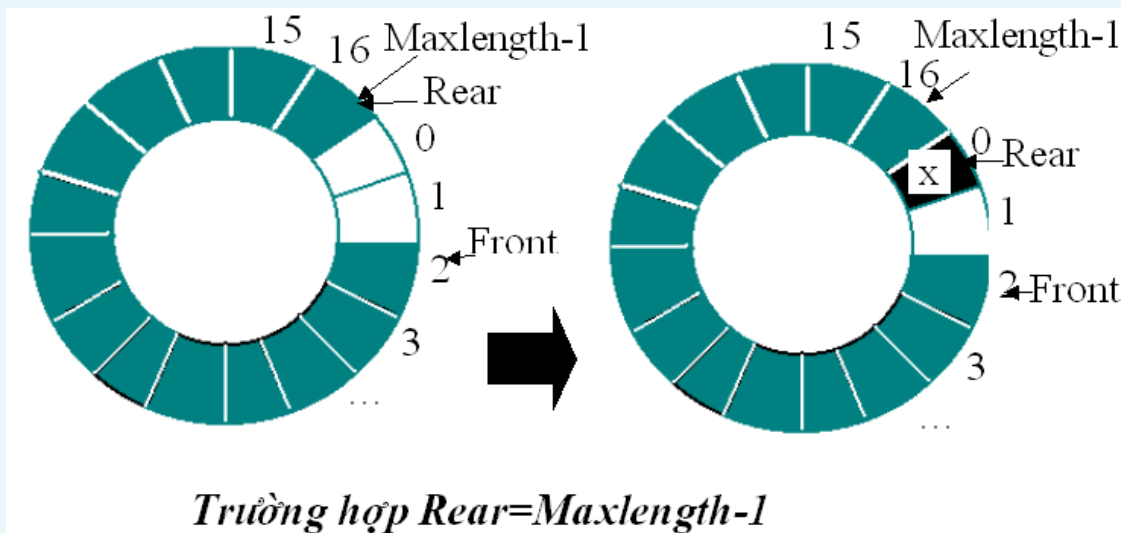
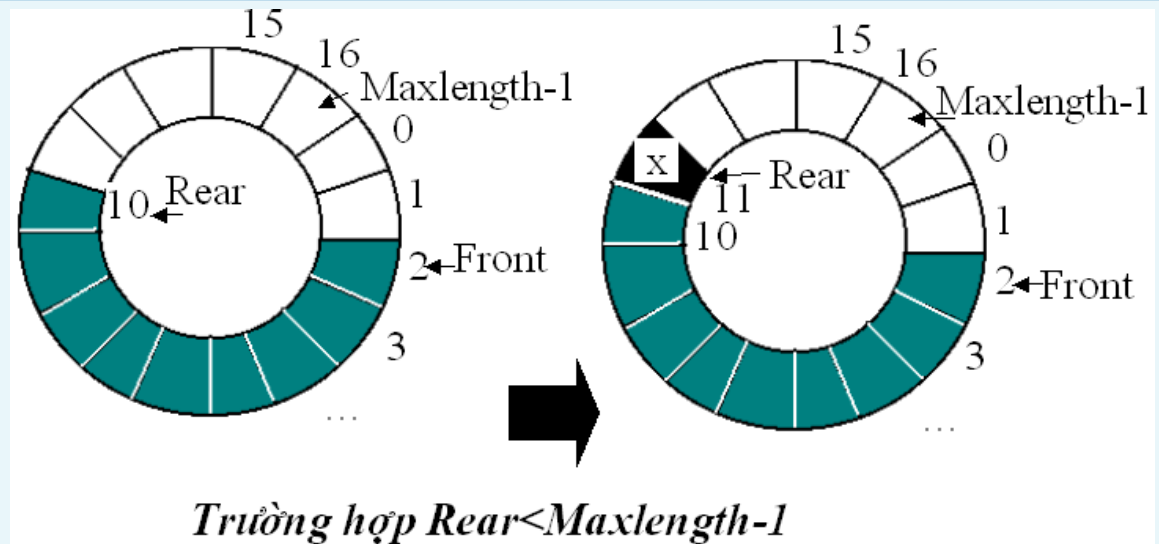
- Giải thuật :
 - Nếu hàng Q rỗng thì thông báo lỗi
 - Ngược lại:
 - Nếu $\text{Front} = \text{Rear}$ tức hàng chỉ còn 1 phần tử thì khởi tạo lại hàng rỗng
 - Ngược lại, thay đổi giá trị cho Front

```
void deQueue (Queue *pQ) {  
    if (!emptyQueue (*pQ)) {  
        if (pQ->Front==pQ->Rear)  
            makenullQueue (pQ) ;  
        else  
            pQ->Front= (pQ->Front+1) %MaxLength;  
    }  
    else printf("Loi: Hang rong!");  
}
```



THÊM PHẦN TỬ X VÀO HÀNG Q

Các trường hợp
có thể:





THÊM PHẦN TỬ X VÀO HÀNG Q

- Giải thuật :
 - Nếu hàng đầy thì thông báo lỗi.
 - Ngược lại, thay đổi giá trị Rear và đưa giá trị x vào ô có chỉ số Rear mới này.

```
void enqueue(ElementType X, Queue *pQ) {  
    if (!fullQueue(*pQ)) {  
        if (emptyQueue(*pQ))  
            pQ->Front=0;  
        pQ->Rear=(pQ->Rear+1) %MaxLength;  
        pQ->Elements[pQ->Rear]=X;  
    }  
    else printf("Loi: Hang day!");  
}
```

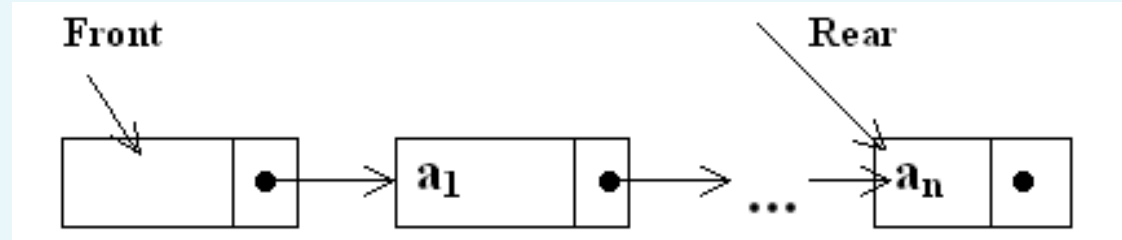


HÀNG ĐỢI (QUEUE)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
 - DÙNG MẢNG DI CHUYỂN TÌNH TIẾN
 - DÙNG MẢNG VÒNG
 - DÙNG DSLK



CÀI ĐẶT HÀNG BẰNG DSLK



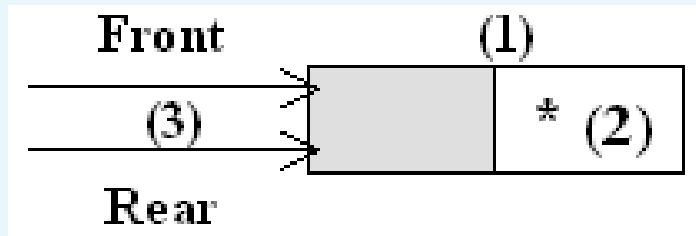
- Dùng 2 con trỏ Front và Rear để chỉ tới phần tử đầu hàng và cuối hàng

- Khai báo

```
typedef ... ElementType; //kiểu phần tử của hàng
struct Node{
    ElementType Element;
    struct Node* Next;      //Con trỏ chỉ ô kế tiếp
};
typedef struct Node* Position;
typedef struct{
    Position Front, Rear; //2 con trỏ
} Queue;
```



KHỞI TẠO HÀNG Q RỖNG



- Cho Front và Rear cùng trỏ đến Header của hàng

```
void makenullQueue (Queue *pQ) {  
    Position Header;  
    Header=(struct Node*)malloc(sizeof(struct Node));  
    //Cấp phát Header  
    Header->Next=NULL;  
    Q->Front=Header;  
    Q->Rear=Header;  
}
```



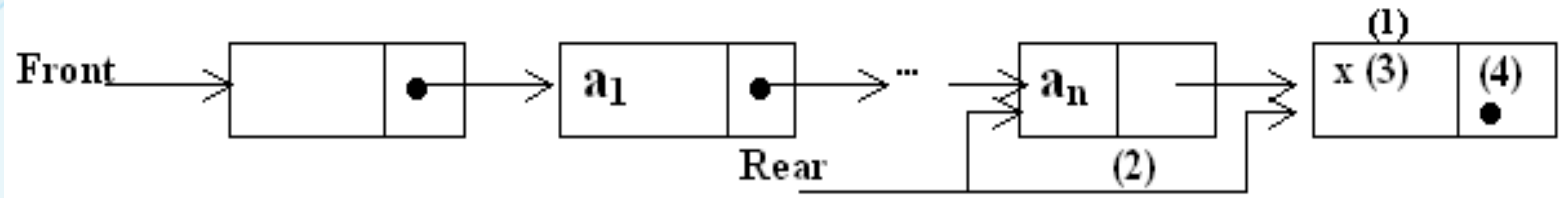
KIỂM TRA HÀNG Q RỖNG

- Kiểm tra xem Front và Rear có cùng chỉ đến 1 ô (Header) không?

```
int emptyQueue (Queue Q) {  
    return (Q.Front==Q.Rear) ;  
}
```



THÊM MỘT PHẦN TỬ x VÀO HÀNG Q



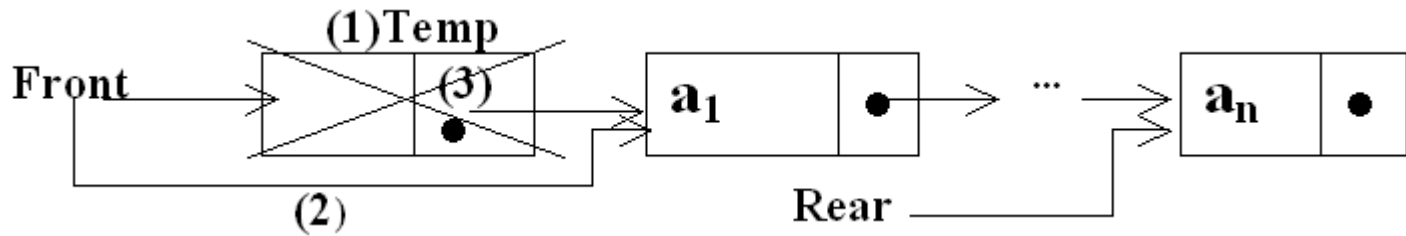
=>Giải thuật:

- Thêm 1 phần tử vào hàng ta thêm vào sau Rear (Rear->Next) 1 ô mới
- Cho Rear trở đến phần tử mới này và đặt giá trị thêm vào cho Rear
- Cho trường next của ô mới này trở tới NULL

```
void enqueue(ElementType X, Queue *pQ) {  
    //Thêm 1 Phần tử vào sau Rear  
    pQ->Rear->Next=(struct Node*)malloc(sizeof(struct Node));  
    pQ->Rear=pQ->Rear->Next; //Trở Rear đến phần tử mới  
    pQ->Rear->Element=X; //Đặt giá trị thêm vào cho Rear  
    pQ->Rear->Next=NULL; //Gán Next của Rear (ô mới) tới Null  
}
```



XÓA MỘT PHẦN TỬ KHỎI HÀNG Q



- Để xóa 1 phần tử khỏi hàng ta chỉ cần cho Front trở tới vị trí kế tiếp của nó trong danh sách

```
void deQueue (Queue *pQ) {  
    if (!emptyQueue(pQ)) {  
        Position Tempt;  
        Tempt=pQ->Front;  
        pQ->Front=pQ->Front->Next;  
        free(Tempt);  
    }  
    else printf("Loi : Hang rong");  
}
```



CÁC ỨNG DỤNG CỦA NGĂN XẾP VÀ HÀNG ĐỢI

- Bạn hãy liệt kê một số ứng dụng có sử dụng
 - Ngăn xếp?
 - Hàng đợi?



CÁC ỨNG DỤNG CỦA NGĂN XẾP VÀ HÀNG ĐỢI

- Ngăn xếp
 - Khử hàm đệ qui (Recursive Function)
 - Đánh giá biểu thức và phân tích cú pháp (Expression evaluation and syntax parsing)
 - Gọi hàm (Calling Function)
 - Thuật toán quay lui (Backtracking) vd: Puzzle, Sudoku...
- Hàng đợi
 - Quản lý in trên mạng.
 - Quản lý truyền thông điệp giữa 02 tiến trình, chương trình hay hệ thống.
 - Lập lịch biểu (VD: lịch cất cánh hay đáp máy bay trên 1 đường băng)



BÀI TẬP

Bài 1:

- Viết hàm để in các phần tử trong ngăn xếp.
- Viết hàm để in các phần tử trong hàng đợi.

Bài 2:

- Viết chương trình nhập vào một ngăn xếp chứa các số nguyên.
- Sau đó sử dụng một hàng đợi để đảo ngược thứ tự của các phần tử trong ngăn xếp đó.



CANTHO UNIVERSITY

Q&A?