
CANTHO UNIVERSITY

# 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 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)

CANTHO UNIVERSITY

# HÀNG ĐỢI (QUEUE)

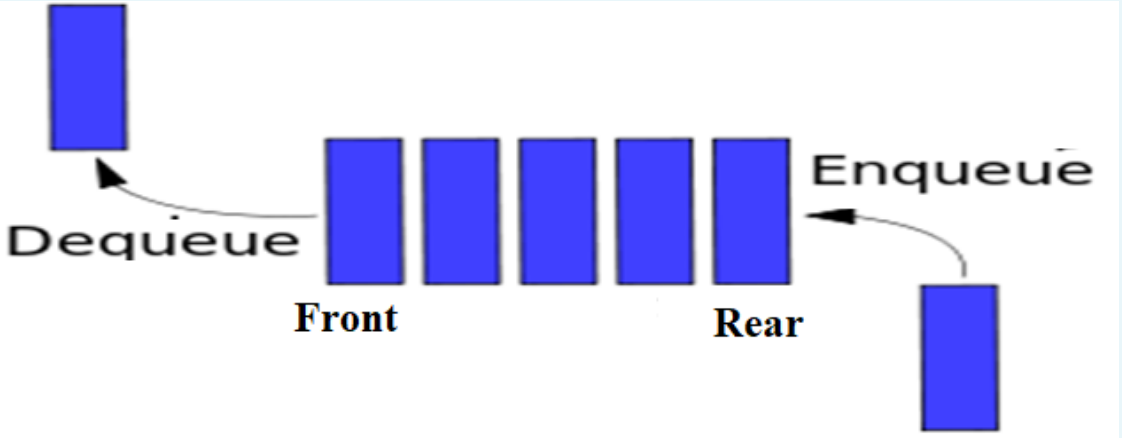
- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
  - DÙNG MẢNG DI CHUYỂN TỊNH TIẾN
  - DÙNG MẢNG VÒNG
  - DÙNG DANH SÁCH LIÊN KẾT

CANTHO UNIVERSITY

# HÀNG ĐỢI (QUEUE)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
  - DÙNG MẢNG DI CHUYỂN TỊNH TIẾN
  - DÙNG MẢNG VÒNG
  - DÙNG DANH SÁCH LIÊN KẾT

# ĐỊNH NGHĨA HÀNG ĐỢI




Hàng đợi (Queue)

- Là một dạng danh sách đặc biệt, mà phép *thêm vào* (enQueue) được thực hiện ở đầu cuối của danh sách, gọi là *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).

CANTHO UNIVERSITY

# HÀNG ĐỢI (QUEUE)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
  - DÙNG MẢNG DI CHUYỂN TỊNH TIẾN
  - DÙNG MẢNG VÒNG
  - DÙNG DANH SÁCH LIÊN KẾT

# CÁC PHÉP TOÁN

| Phép toán | Diễn giải |
|---|---|
| makenullQueue(*Q*) | Tạo một hàng đợi rỗng (Q) |
| emptyQueue(*Q*) | Kiểm tra xem hàng đợi Q có rỗng không |
| fullQueue(*Q*) | Kiểm tra hàng đợi Q đầy |
| enQueue(*x*,*Q*) | Thêm phần tử x vào cuối hàng đợi Q |
| deQueue(*Q*) | Xóa phần tử tại đầu hàng đợi Q |
| front(*Q*) | Trả về phần tử đầu tiên của hàng đợi Q |

CANTHO UNIVERSITY

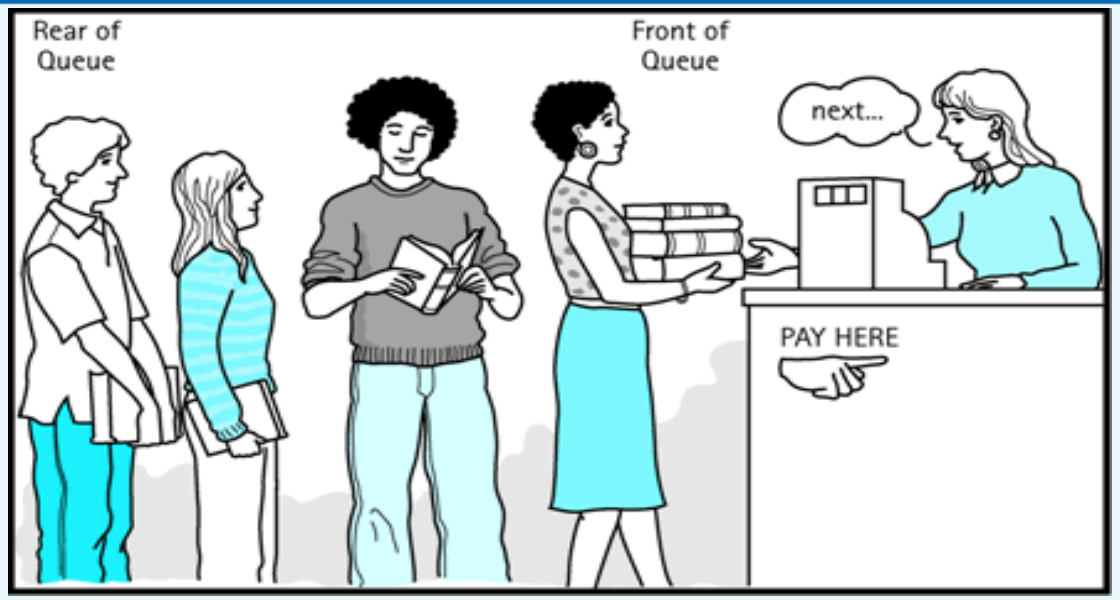
# HÀNG ĐỢI (QUEUE)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
  - DÙNG MẢNG DI CHUYỂN TỊNH TIẾN
  - DÙNG MẢNG VÒNG
  - DÙNG DANH SÁCH LIÊN KẾT

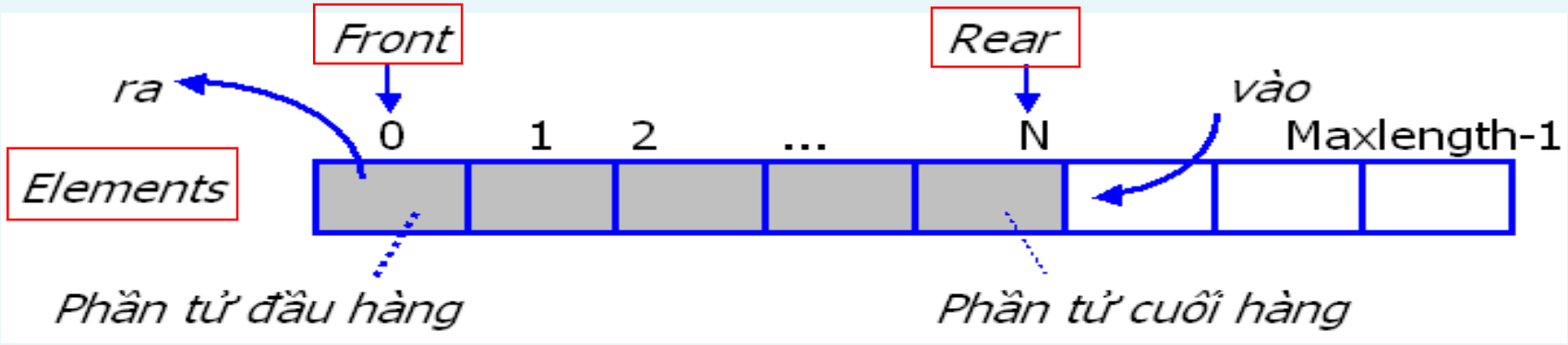
# Kiểu dữ liệu trừu tượng - Lưu ý

- 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*).

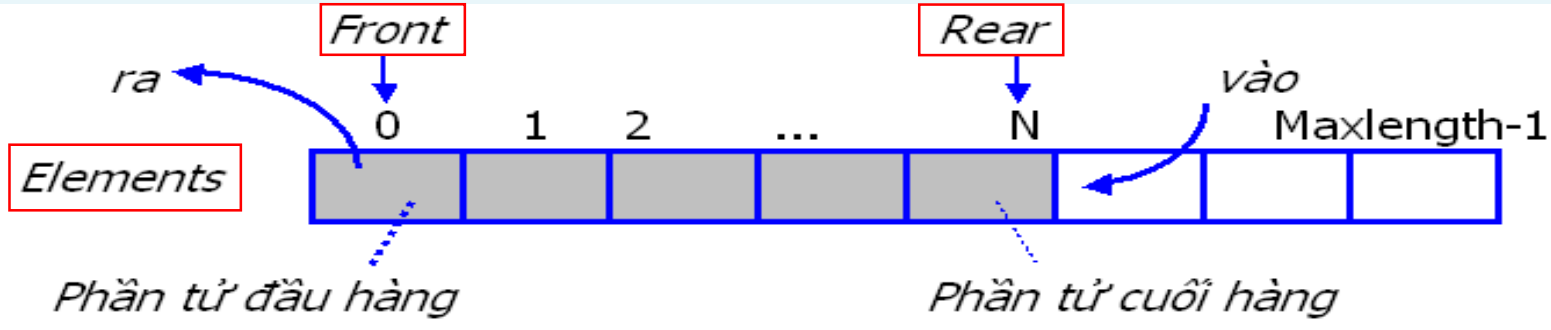
# CÀI ĐẶT HÀNG BẰNG MẢNG DI CHUYỂN TỊNH TIẾN




- Mô hình



# KHAI BÁO



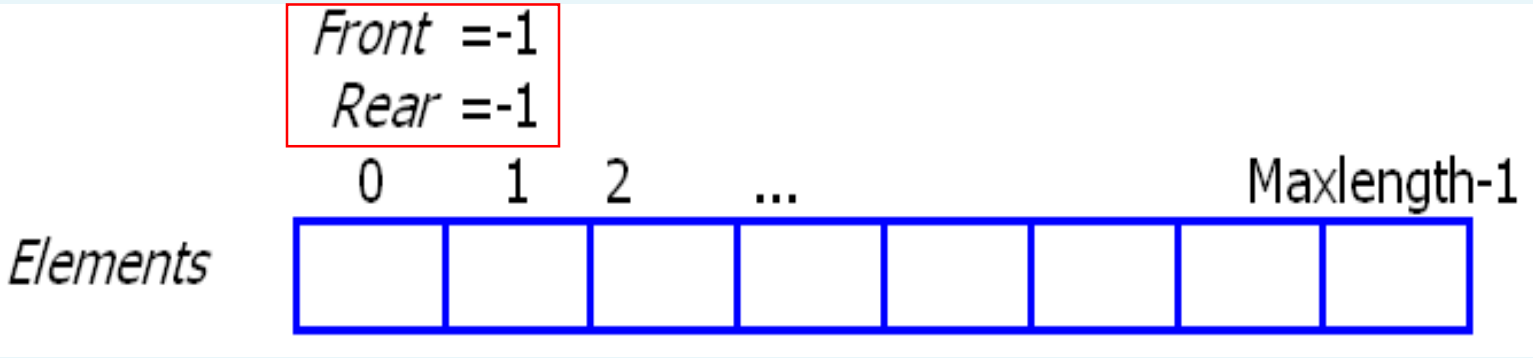

```
#define MaxLength <n> //Chiều dài tối đa của mảng
typedef <datatype> ElementType; //Kiểu dữ liệu của các phần tử trong hàng
typedef struct {
   ElementType Elements[MaxLength]; //Lưu trữ nội dung các phần tử
   int         Front, Rear; //Chỉ số đầu và cuối hàng
} Queue;
Queue Q;
```

# KHỞI TẠO HÀNG RỖNG

```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
   ElementType Elements[MaxLength];
   int Front, Rear;
} Queue;
```

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

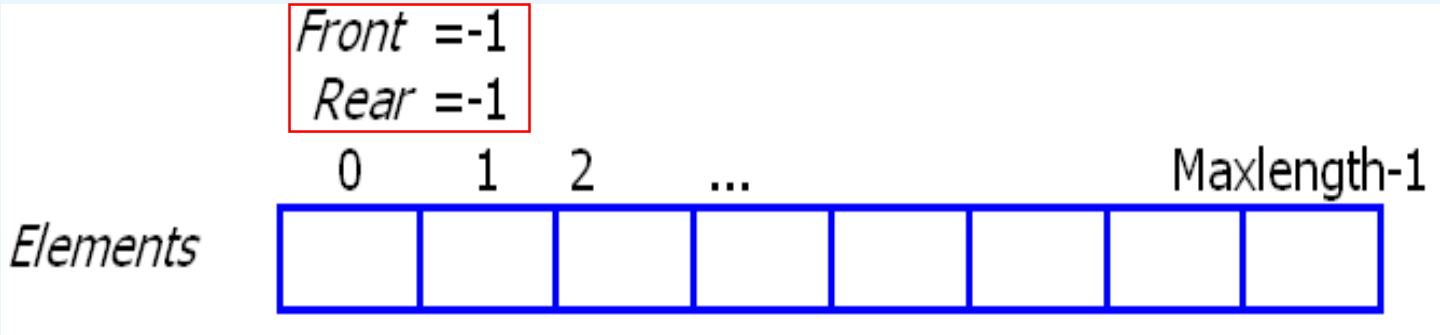
- Front và Rear không trỏ đến  vị trí hợp lệ nào
- Ta cho Front=Rear=-1



# KHỞI TẠO HÀNG RỖNG

```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
   ElementType Elements[MaxLength];
   int Front, Rear;
} Queue;
```

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

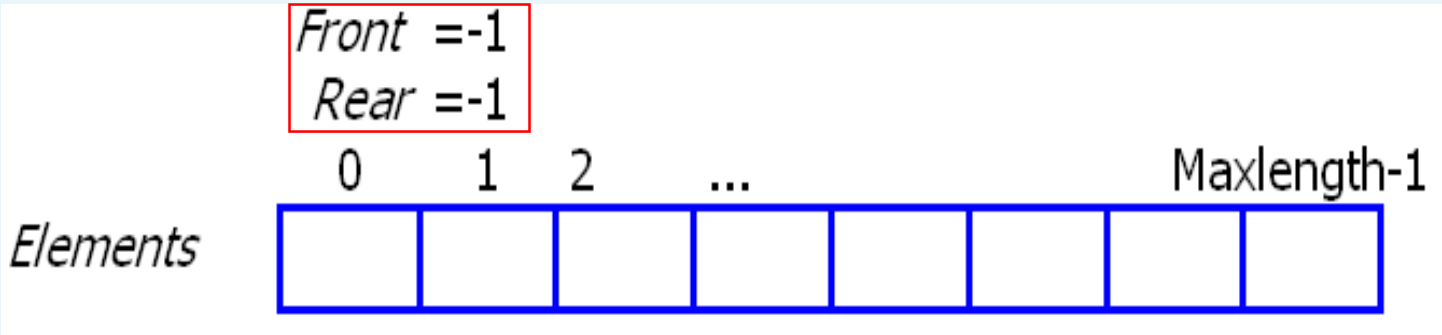



```
void makenullQueue(Queue *pQ){
      pQ->Front=-1;
      pQ->Rear=-1;
}
```

# KHỞI TẠO HÀNG RỖNG

```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
   ElementType Elements[MaxLength];
   int Front, Rear;
} Queue;
```

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



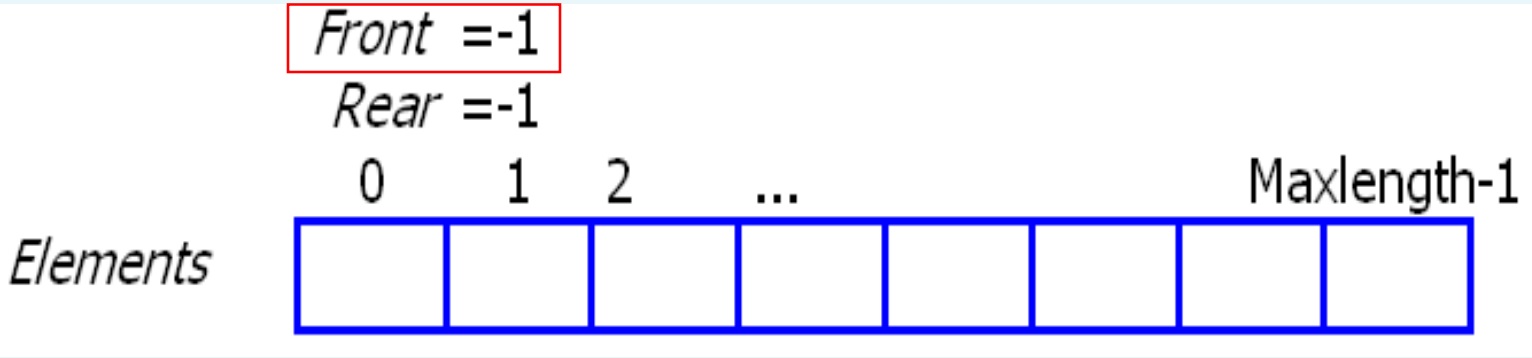

```
Queue makenullQueue(){
      Queue Q;    Q.Front=-1;
      Q.Rear=-1; return Q;
}
```

# KIỂM TRA HÀNG RỖNG

```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
   ElementType Elements[MaxLength];
   int Front, Rear;
} Queue;
```

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

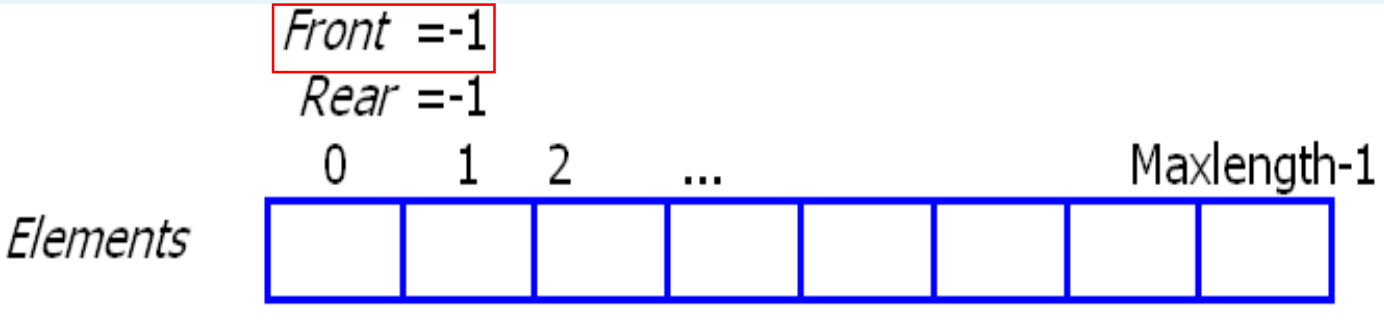
– Hàng rỗng khi Front=-1



# KIỂM TRA HÀNG RỖNG

```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
   ElementType Elements[MaxLength];
   int Front, Rear;
} Queue;
```

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



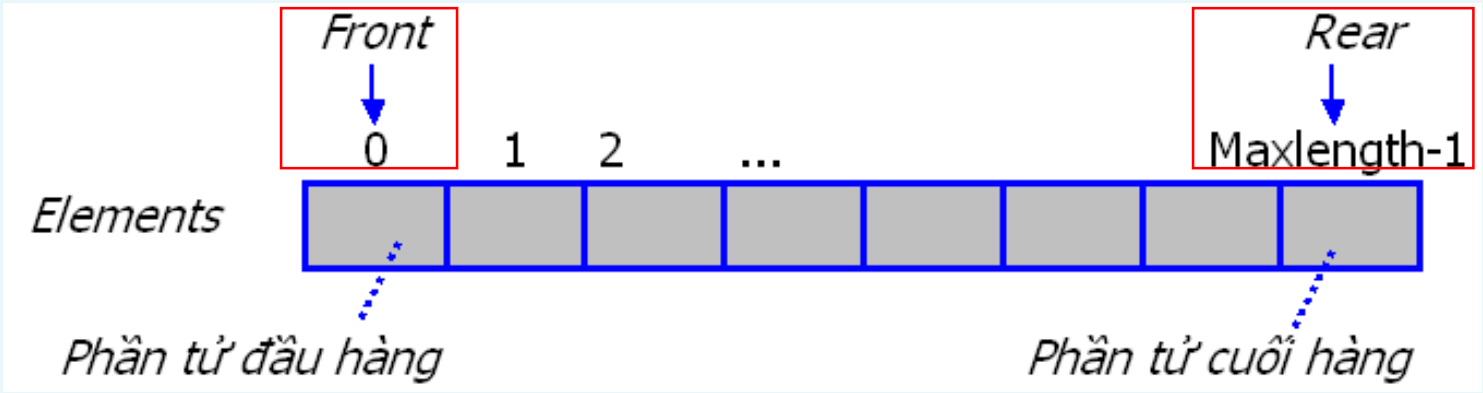

```
int emptyQueue(Queue Q){
    return (Q.Front==-1);
}
```

# KIỂM TRA HÀNG ĐẦY

```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
   ElementType Elements[MaxLength];
   int Front, Rear;
} Queue;
```

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

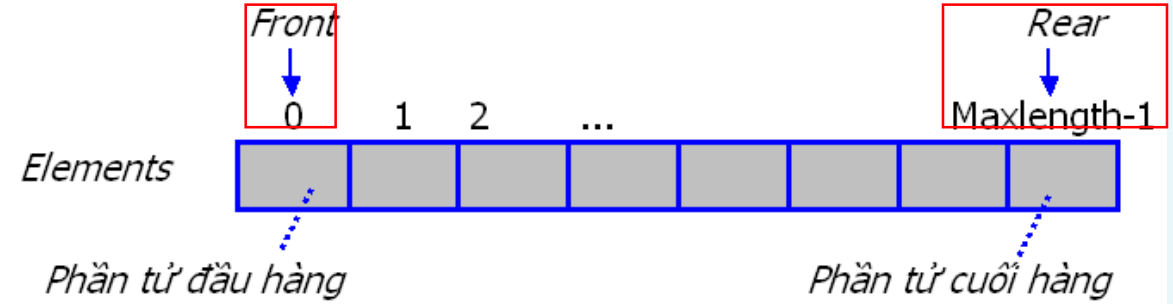
– Hàng đầy khi số phần tử hiện có trong hàng=MaxLength



# KIỂM TRA HÀNG ĐẦY

```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
   ElementType Elements[MaxLength];
   int Front, Rear;
} Queue;
```

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




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

Số p.tử của hàng Q

# TRẢ VỀ PHẦN TỬ ĐẦU HÀNG

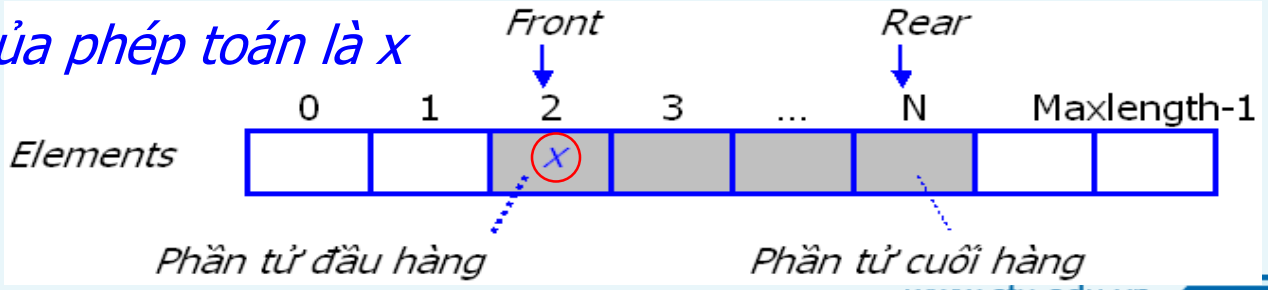
```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
   ElementType Elements[MaxLength];
   int Front, Rear;
} Queue;
```

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

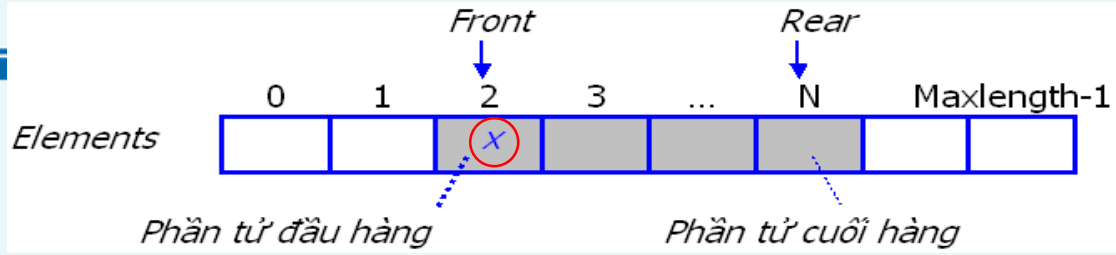
Giải thuật:

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

*Kết quả của phép toán là x*



# TRẢ VỀ PHẦN TỬ ĐẦU HÀNG



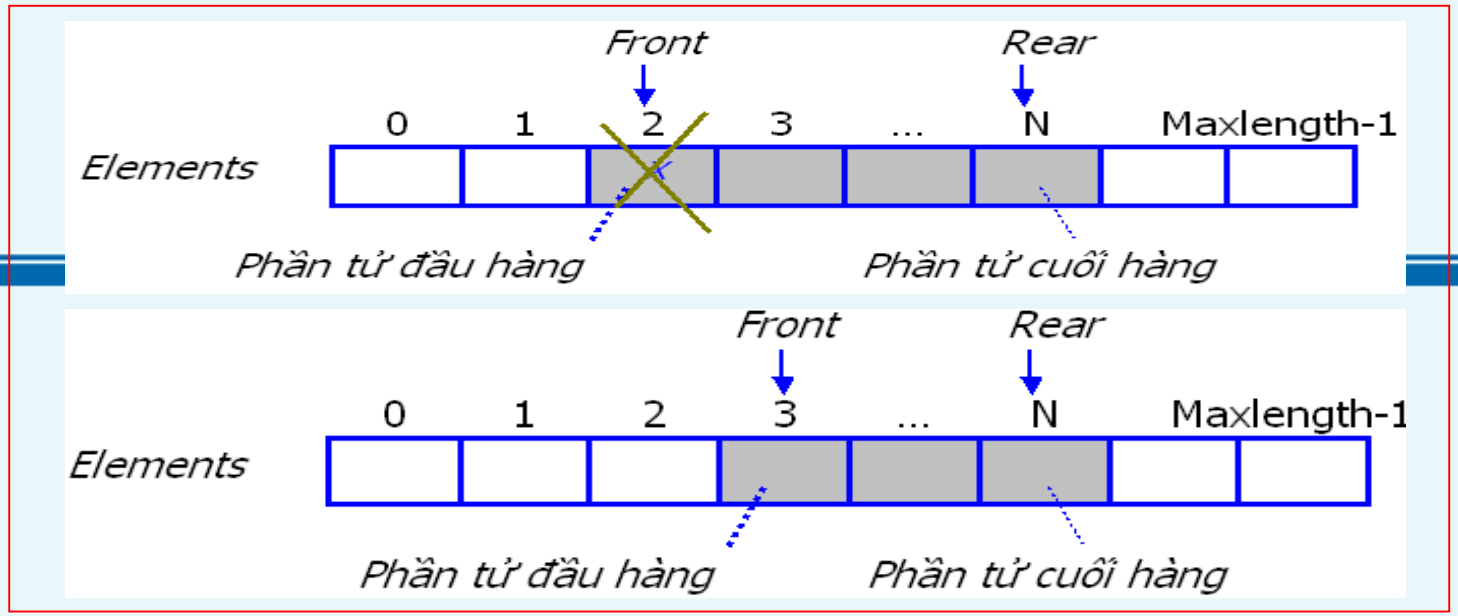

```
#define MaxLength <n>
typedef <datatype> ElementType;
typedef struct {
   ElementType Elements[MaxLength];
   int Front, Rear;
} Queue;
```

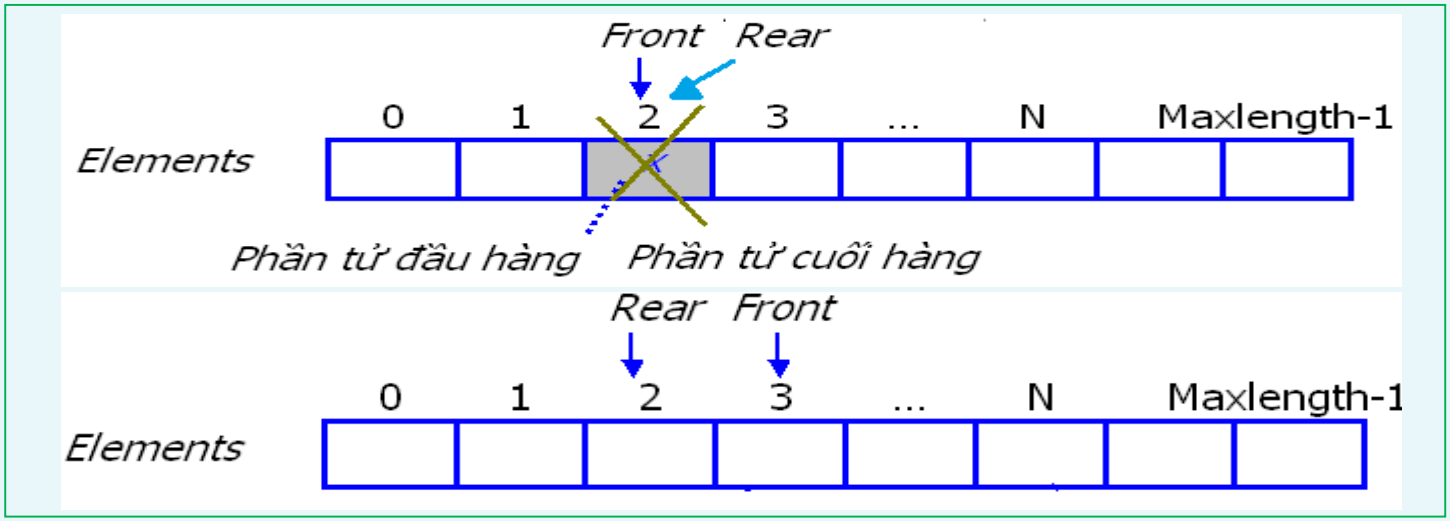
```
ElementType front(Queue Q){
    if emptyQueue(Q)
        printf ("Hang rong");
    else
        return Q.Elements[Q.Front];
}
```

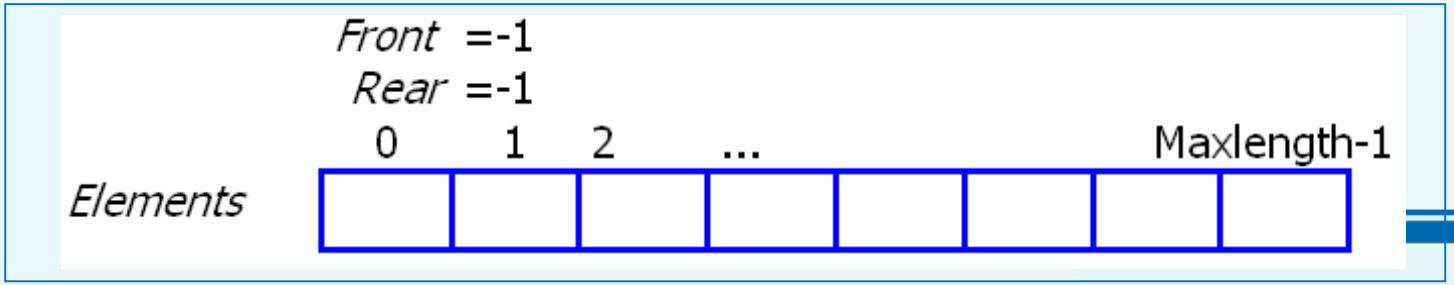
CANTHO UNIVERSITY

# XÓA MỘT PHẦN TỬ KHỎI HÀNG

- Trường hợp hàng không rỗng
- Trường hợp hàng rỗng



# 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 (Front >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(pQ); //Dat lai hang rong
  }
  else printf("Loi: Hang rong!");
}
```

# XÓA MỘT PHẦN TỬ KHỎI HÀNG

Giải thuật:

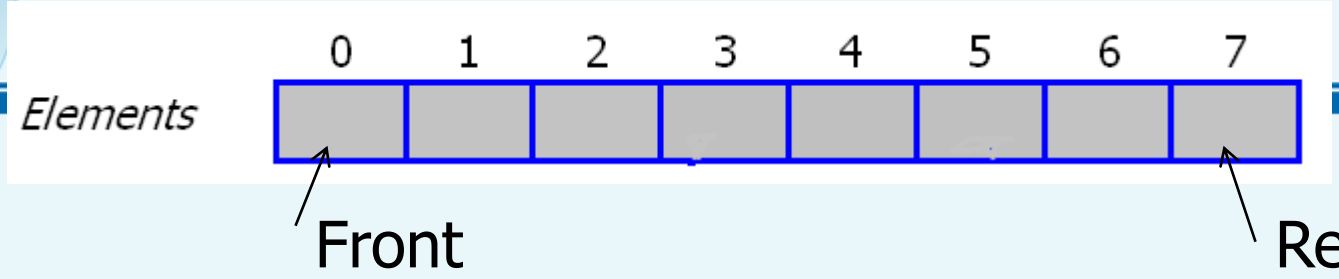
- Nếu hàng Q rỗng thì thông báo lỗi.
- Ngược lại:
  - Nếu (Front =Rear) tức hàng chỉ còn 1 phần tử thì khởi tạo lại hàng rỗng.
  - Ngược lại, tăng Front lên 1 đơn vị.

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

CANTHO UNIVERSITY

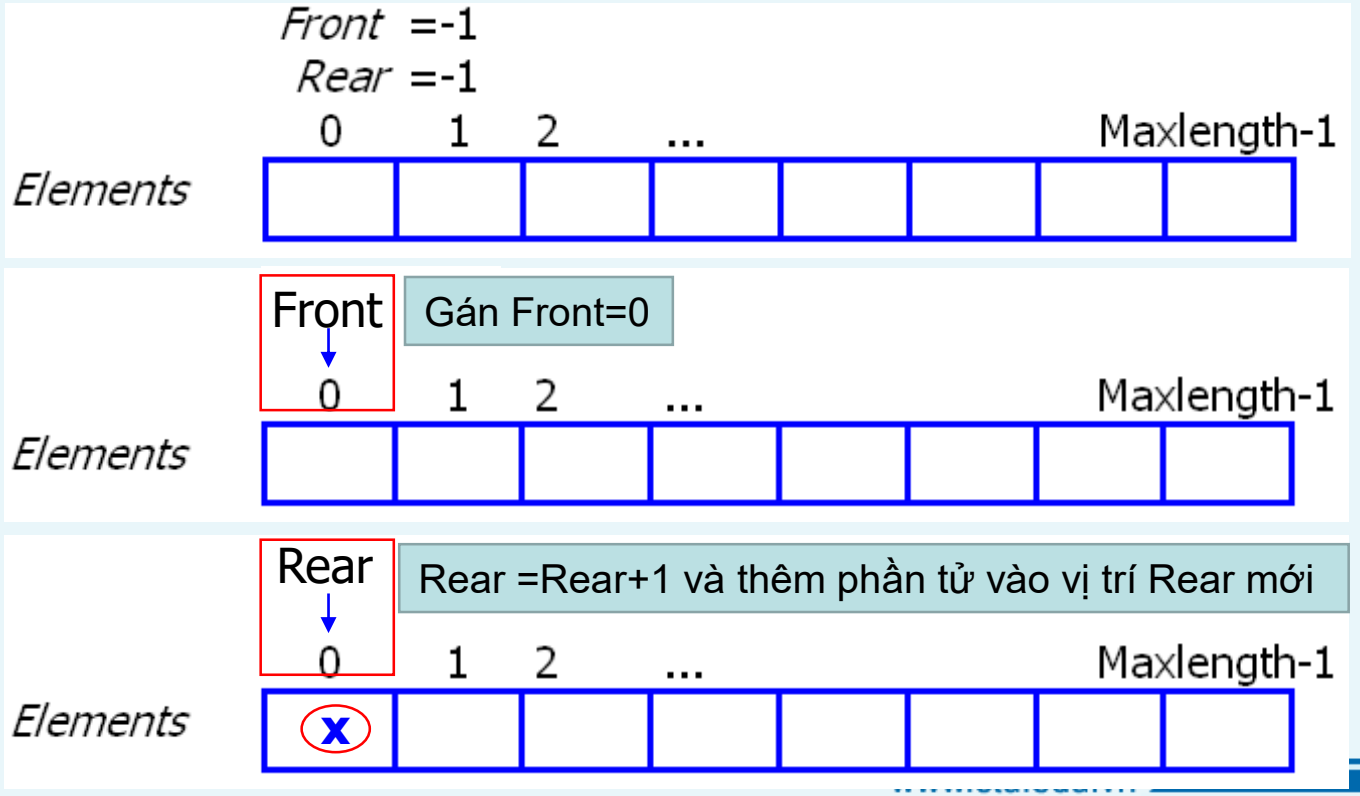
- Trường hợp hàng đầy

Thông báo lỗi "Lỗi: hàng đầy!"




- Trường hợp hàng rỗng

**THÊM MỘT PHẦN TỬ VÀO HÀNG**

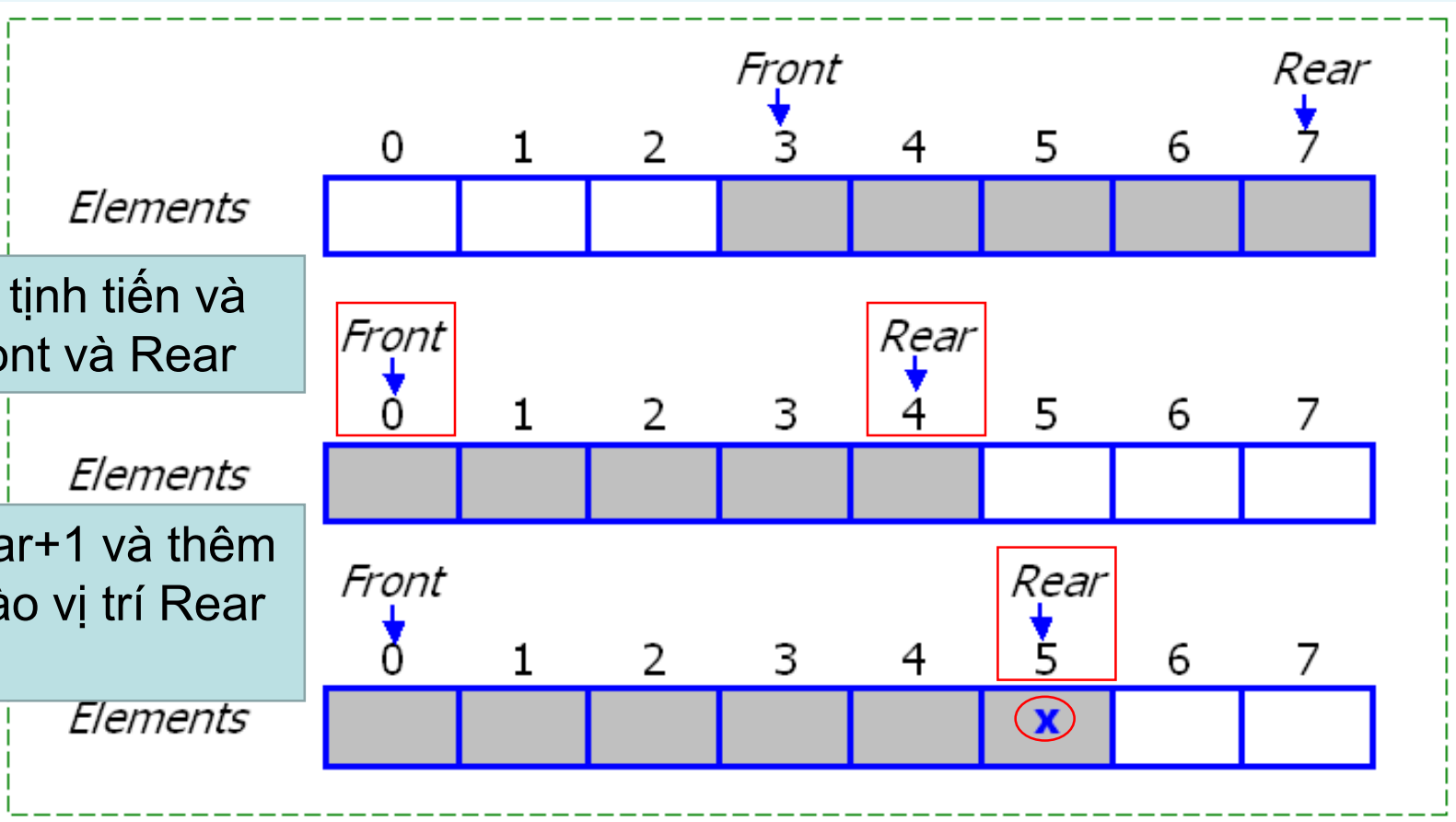


# THÊM MỘT PHẦN TỬ VÀO HÀNG

- Trường hợp hàng chưa đầy nhưng *bị tràn*

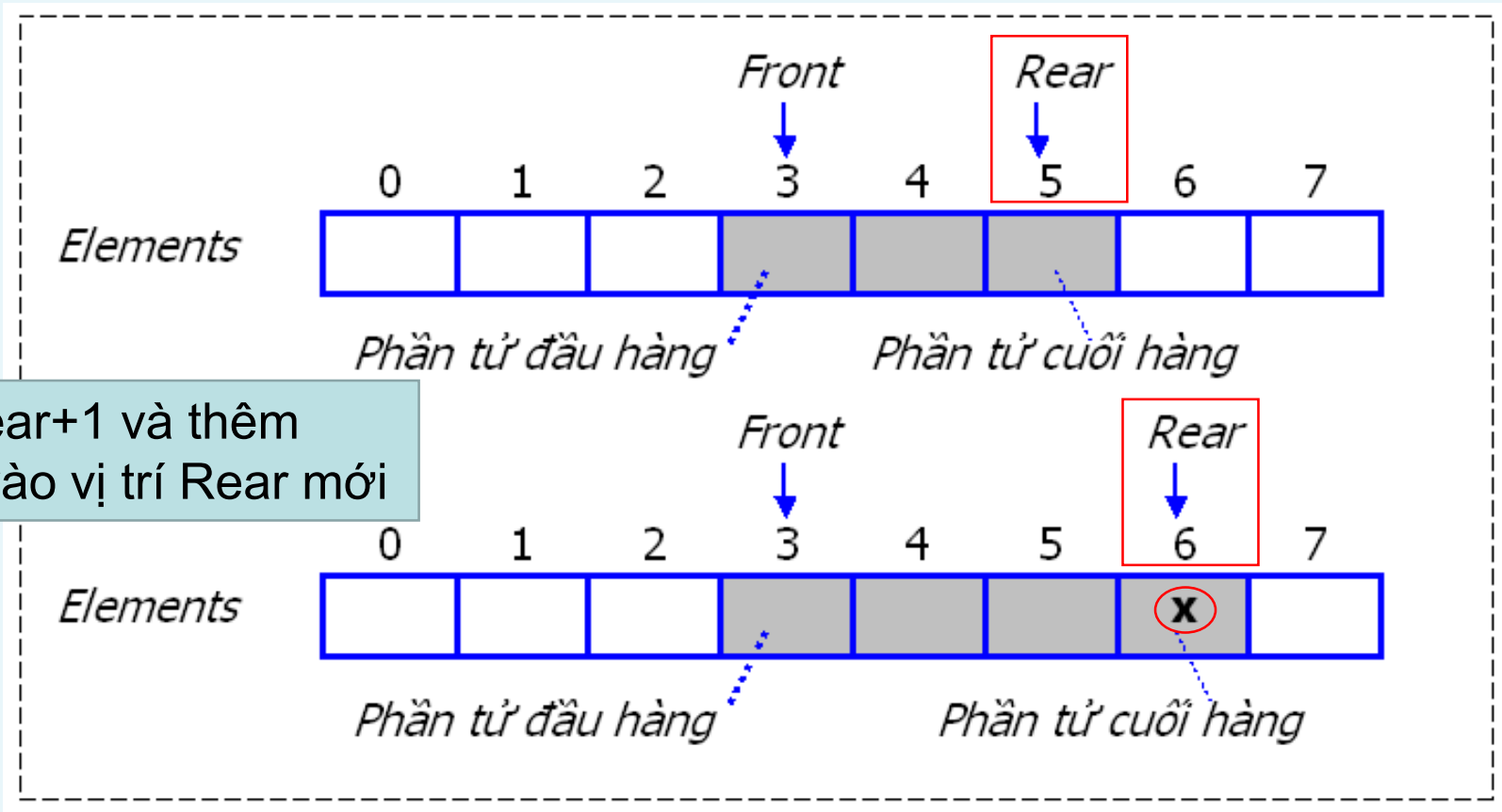
Di chuyển tịnh tiến và gán lại Front và Rear

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*




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

# 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 chuyen tinh tien ra truoc Front -1 vi tri
          for(int i=pQ->Front;i<=pQ->Rear;i++)
               pQ->Elements[i-pQ->Front]=pQ->Elements[i];
        //Xac dinh vi tri Rear moi
          pQ->Rear=MaxLength - 1 - pQ->Front;
          pQ->Front=0;
     }
     pQ->Rear=pQ->Rear+1; //Tang Rear de luu noi dung moi
     pQ->Elements[pQ->Rear]=x; // Dua x vao cuoi hang
   }
   else printf("Loi: Hang day!");
}
```

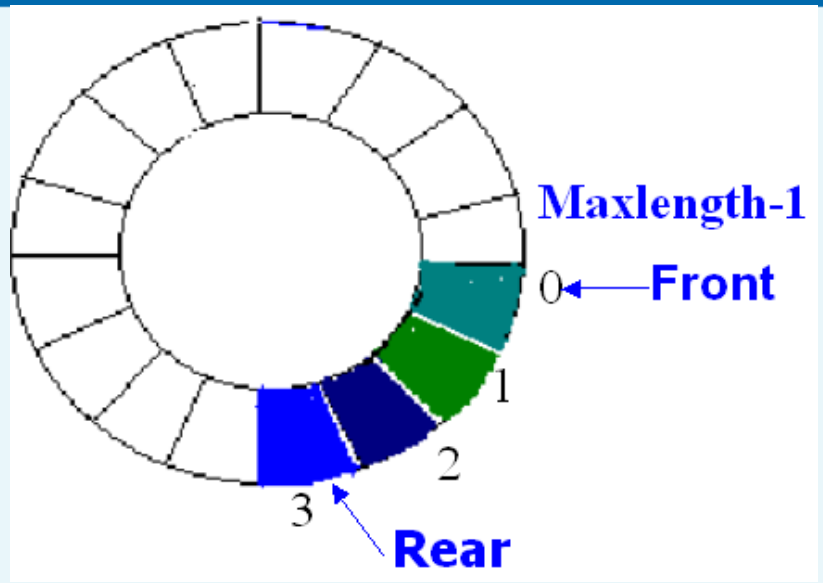
CANTHO UNIVERSITY

# HÀNG ĐỢI (QUEUE)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
  - DÙNG MẢNG DI CHUYỂN TỊNH TIẾN
  - DÙNG MẢNG VÒNG
  - DÙNG DANH SÁCH LIÊN KẾT

# CÀI ĐẶT HÀNG BẰNG MẢNG VÒNG

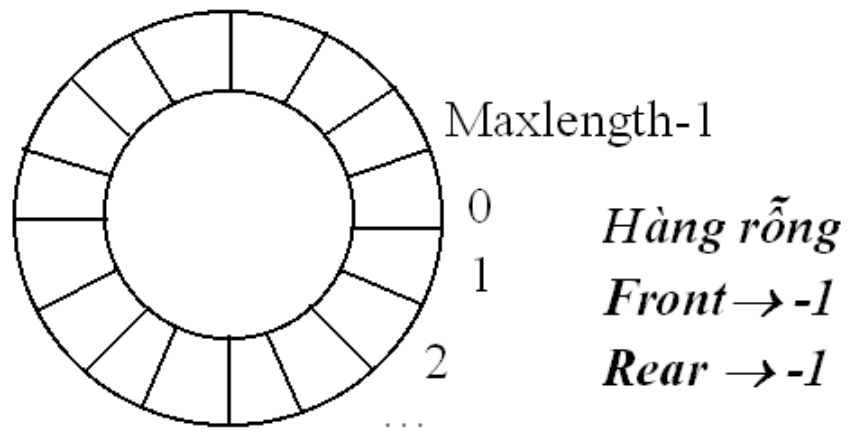
- Mô hình




- Khai báo

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

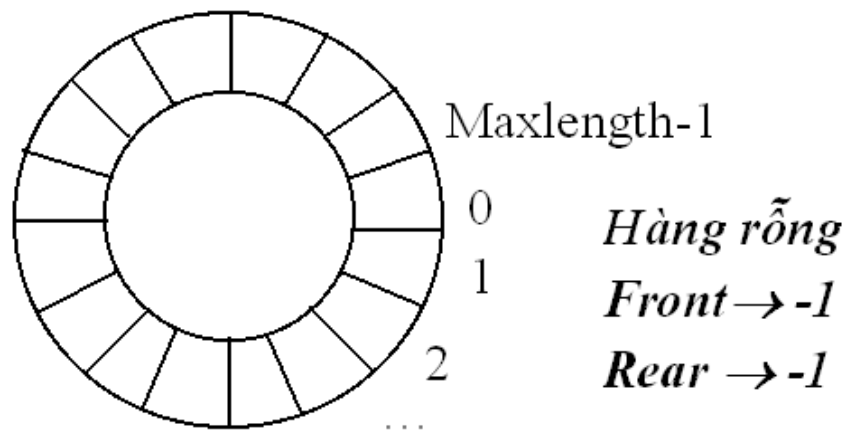
# KHỞI TẠO HÀNG RỖNG




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

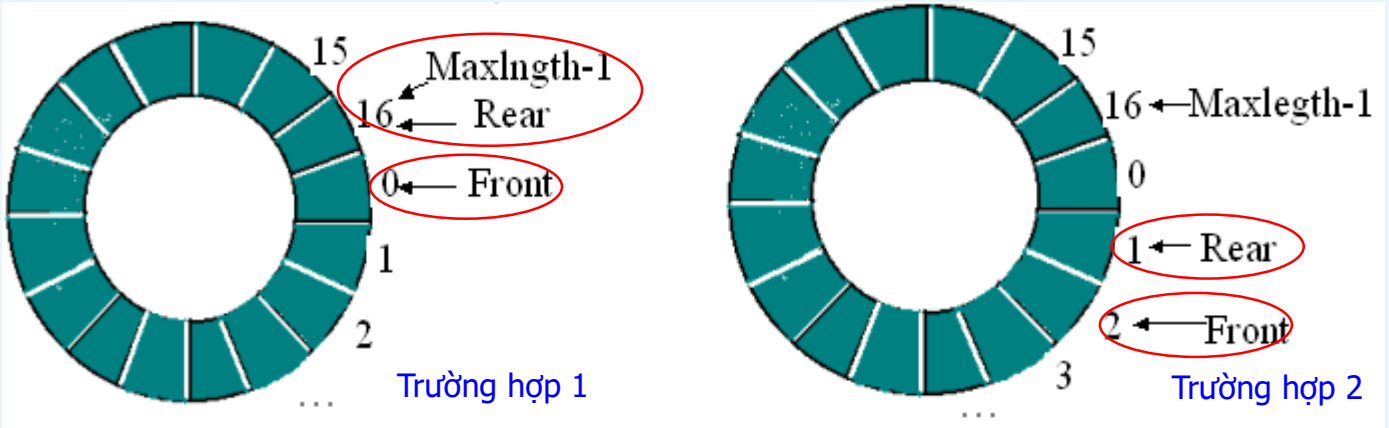
```
void makenullQueue(Queue *pQ){
        pQ->Front=-1;
        pQ->Rear=-1
}
```

# KIỂM TRA HÀNG RỖNG




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

# KIỂM TRA HÀNG ĐẦY



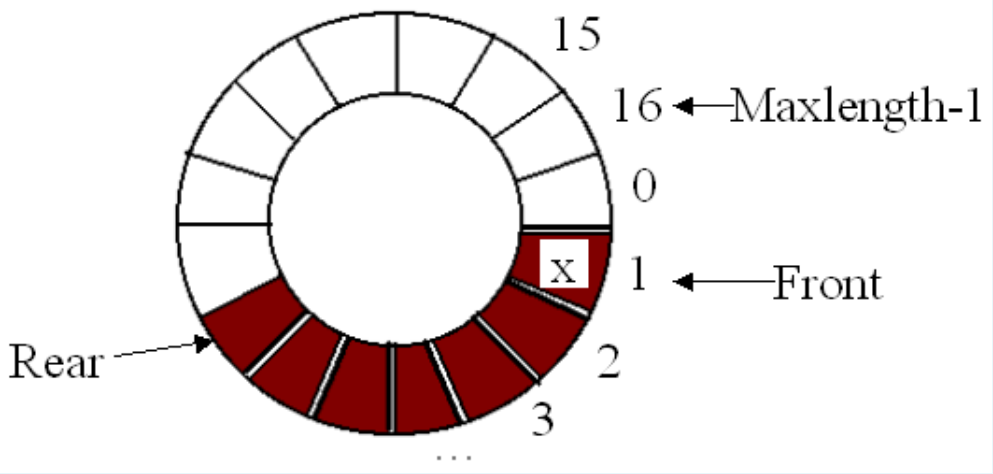

Trường hợp Q.Rear=Maxlength-1 và Q.Front =0

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;
}
```

# TRẢ VỀ PHẦN TỬ ĐẦU HÀNG

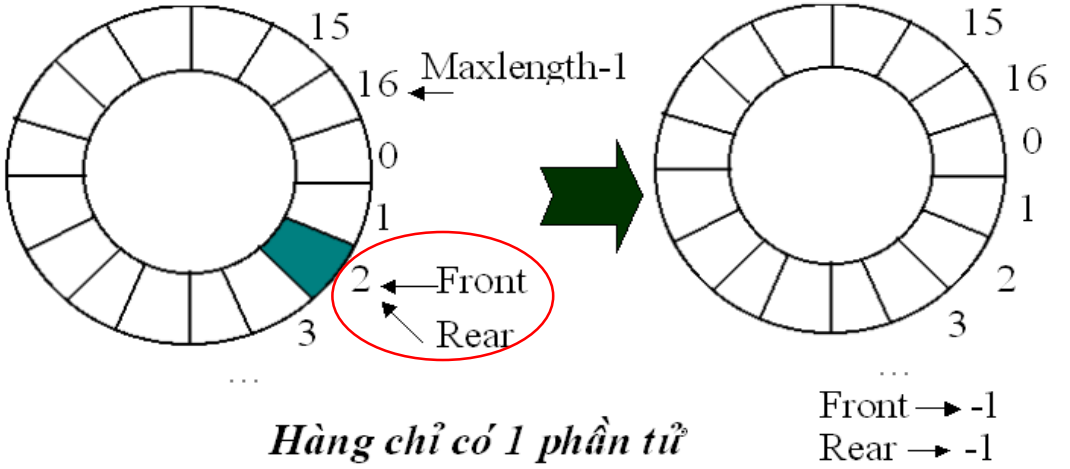



Giải thuật

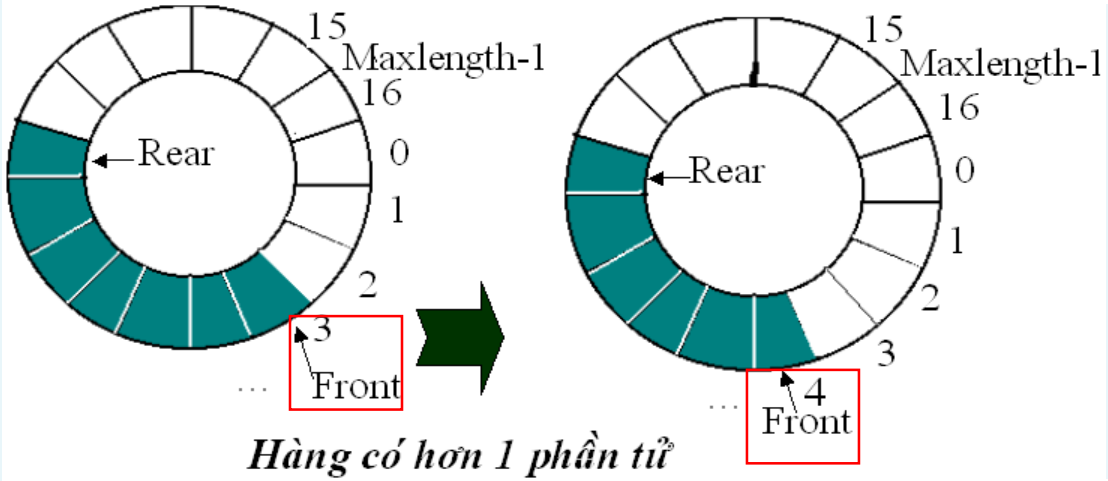
- Nếu hàng Q rỗng thì thông báo lỗi
- Ngược lại, trả về giá trị được lưu trữ tại ô có chỉ số là Front

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

CANTHO UNIVERSITY



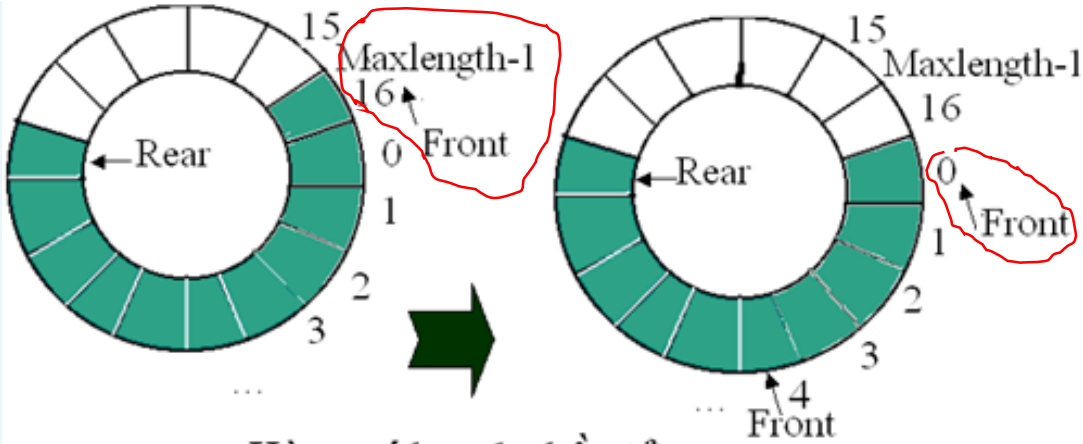

*Hàng chỉ có 1 phần tử*




*Hàng có hơn 1 phần tử*

Các trường hợp có thể:

# XÓA PHẦN TỬ ĐẦU HÀNG




*Hàng có hơn 1 phần tử*

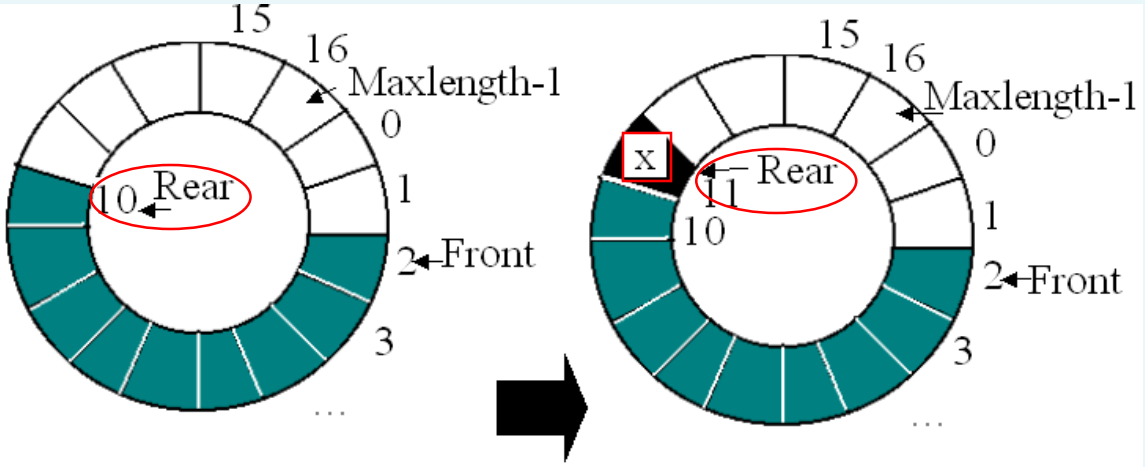
# XÓA PHẦN TỬ ĐẦU HÀNG

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

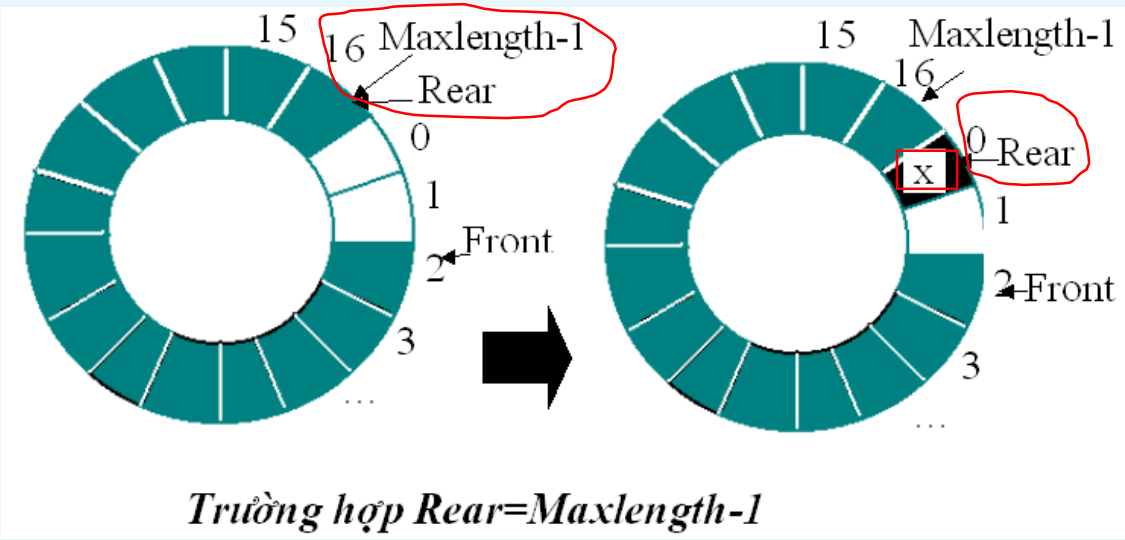
```
void deQueue(Queue *pQ){
    if (!emptyQueue(*pQ)){
        if (pQ->Front==pQ->Rear)
            makenullQueue(pQ);
        else
            pQ->Front=(pQ->Front+1)%MaxLength;
    }
    else printf("Loi: Hang rong!");
}
```

# THÊM PHẦN TỬ x VÀO HÀNG Q

Các trường hợp có thể:




*Trường hợp Rear<Maxlength-1*




*Trường hợp Rear=Maxlength-1*

# 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!");
}
```

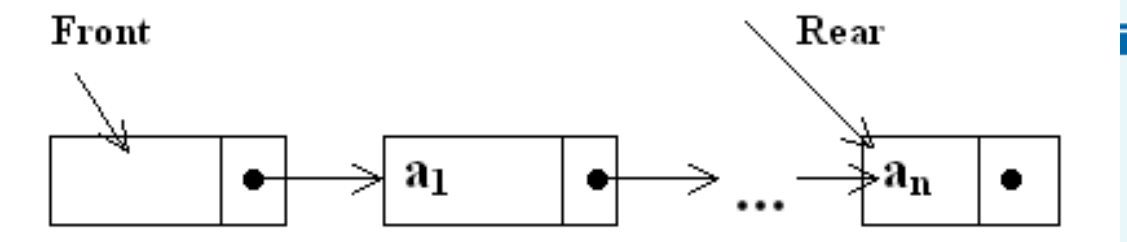
*Nhận xét về độ phức tạp của enQueue trong các trường hợp cài đặt bằng mảng tịnh tiến và mảng vòng?*

CANTHO UNIVERSITY

# HÀNG ĐỢI (QUEUE)

- ĐỊNH NGHĨA
- CÁC PHÉP TOÁN
- CÀI ĐẶT HÀNG ĐỢI
  - DÙNG MẢNG DI CHUYỂN TỊNH TIẾN
  - DÙNG MẢNG VÒNG
  - DÙNG DANH SÁCH LIÊN KẾT

# CÀI ĐẶT HÀNG BẰNG DSLK

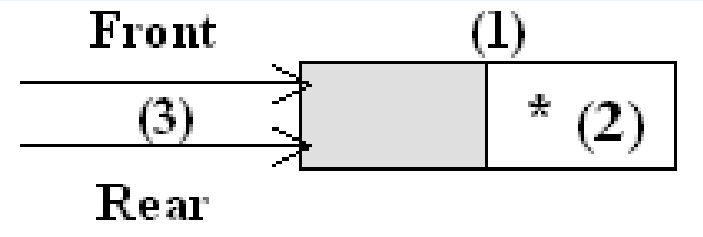



- Dùng 2 con trỏ Front và Rear để chỉ tới phần tử đầu hàng và cuối hàng

- Khai báo

```
typedef <datatype> 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;
```

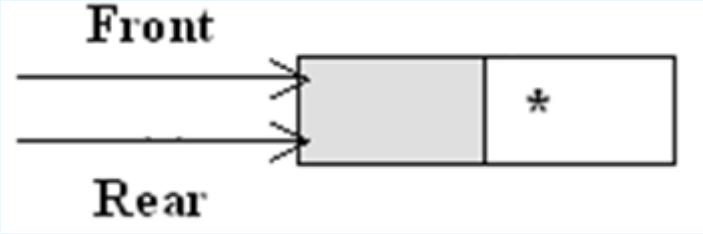
# KHỞI TẠO HÀNG Q RỖNG




– Cho Front và Rear cùng trỏ đến Header của hàng

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

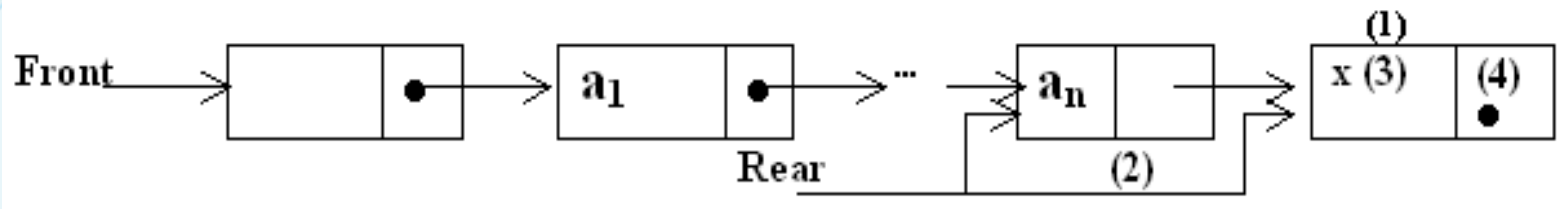
# KIỂM TRA HÀNG Q RỖNG




- 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

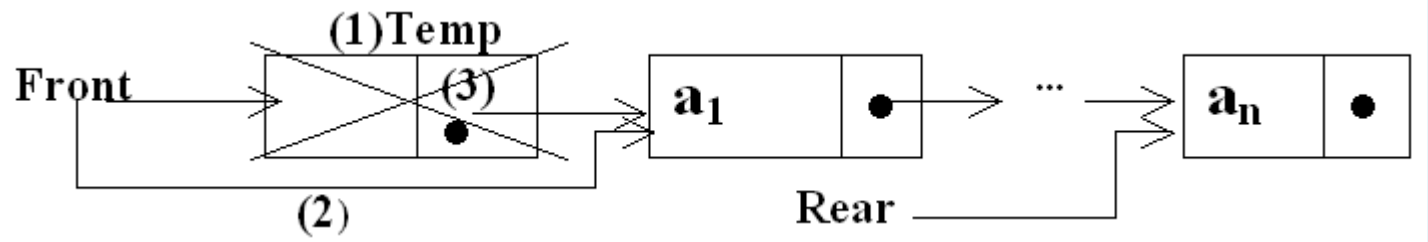



Giải thuật:

- Thêm 1 phần tử vào hàng ta thêm vào sau Rear (Rear->Next) 1 ô mới
- Cho Rear trỏ đến phần tử mới này và đặt giá trị thêm vào cho Rear
- Cho trường next của ô mới này trỏ 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;  //Trỏ 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 trỏ 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 nhập và 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.

# Q&A?