Data Structure & Algorithm

Topic 4: Các kiểu dữ liệu trừu tượng cơ bản- Hàng đợi (Queue)

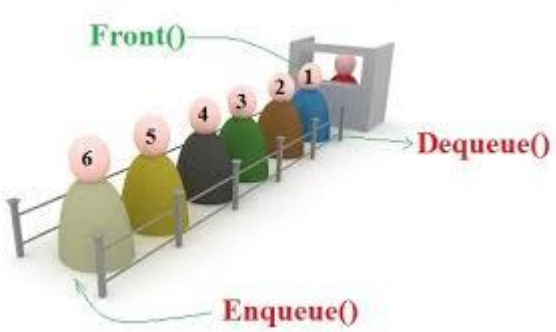
Hàng đợi – Queue

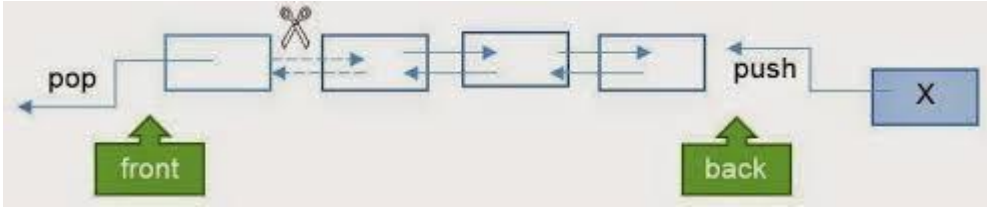
- Khái niệm Queue
- Các phép toán trên Queue
- Cài đặt Queue bằng Mảng
- Cài đặt Queue bằng DSLK

Khái niệm Hàng đợi – Queue

- **Queue**
  - Là một danh sách đặc biệt mà việc thêm và bớt phần tử được thực hiện ở 2 đầu khác nhau.
  - THÊM ở cuối danh sách
  - BỚT ở đầu danh sách

  - Nguyên tắc hoạt động: FIFO (*first in – first out*): vào trước ra trước





Khái niệm Hàng đợi – Queue

- Queue
  - Chứa các phần tử có cùng kiểu dữ liệu
  - Vị trí để loại bỏ phần tử (đầu danh sách): front
  - Vị trí để thêm phần tử (cuối danh sách): rear

- Có 2 cách cài đặt
  - Mảng
  - Danh sách liên kết

Khái niệm Hàng đợi – Queue

- **Các thao tác cơ bản**
  - Khởi tạo một hàng đợi rỗng: MakeNull_Queue (Q)
  - Kiểm tra hàng đợi có rỗng không: isEmpty_Queue (Q)
  - Lấy giá trị của phần tử đầu tiên của Queue: Front (Q)
  - Thêm phần tử X vào cuối hàng: Enqueue (Q,X)
  - Xóa phần tử tại đầu hàng đợi: Dequeue (Q)
  - Kiểm tra hàng đầy: isFull_Queue (Q)

Cài đặt Queue

- Cài đặt bằng Mảng
- Cài đặt bằng Danh sách

Cài đặt Queue bằng Mảng

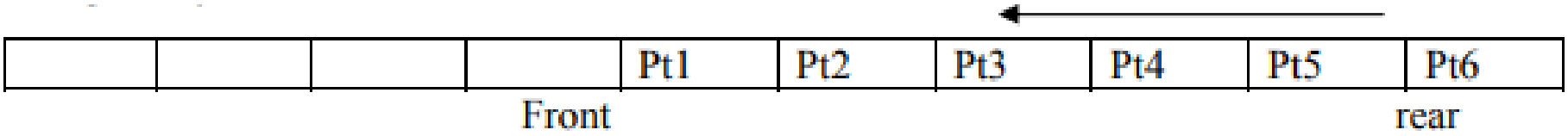
- Dùng 1 mảng để lưu trữ liên tiếp các phần tử của Queue.
- Các phần tử đưa vào Queue bắt đầu từ vị trí có chỉ số thấp nhất của mảng (phần tử chỉ số 0, chỉ số 1, chỉ số, 2,…)
- Giả sử Hàng đợi có n phần tử
  - **front = 0**
  - **rear = n-1**

Cài đặt Queue bằng Mảng

- Khi xóa một phần tử, front tăng lên 1
- Khi thêm 1 phần tử, rear tăng lên 1

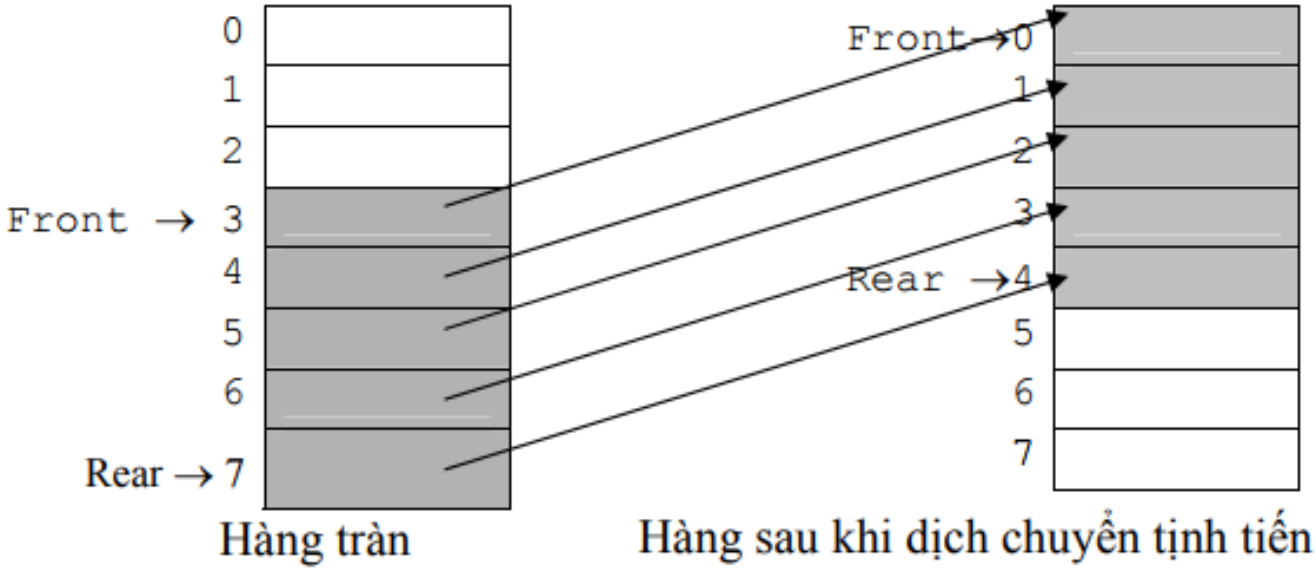
→ Hàng đợi có khuynh hướng đi xuống, đến một lúc không thể thêm vào hàng được (rear=maxlength-1) dù mảng còn nhiều chỗ trống (các vị trí trước front), trường hợp này ta gọi là hàng bị tràn

- Trường hợp toàn bộ mảng đã chứa các phần tử của hàng ta gọi là hàng bị đầy.



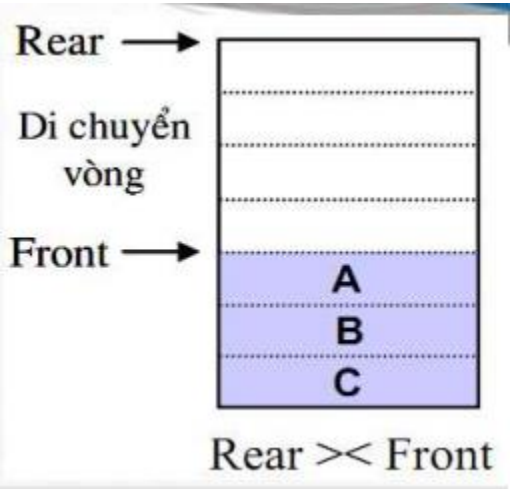
Cài đặt Queue bằng Mảng

- Cách khắc phục Hàng tràn
  - **Di chuyển tịnh tiến**
    - Dời toàn bộ hàng lên front -1 vị trí
    - Trong trường hợp này ta luôn có front <= rear.



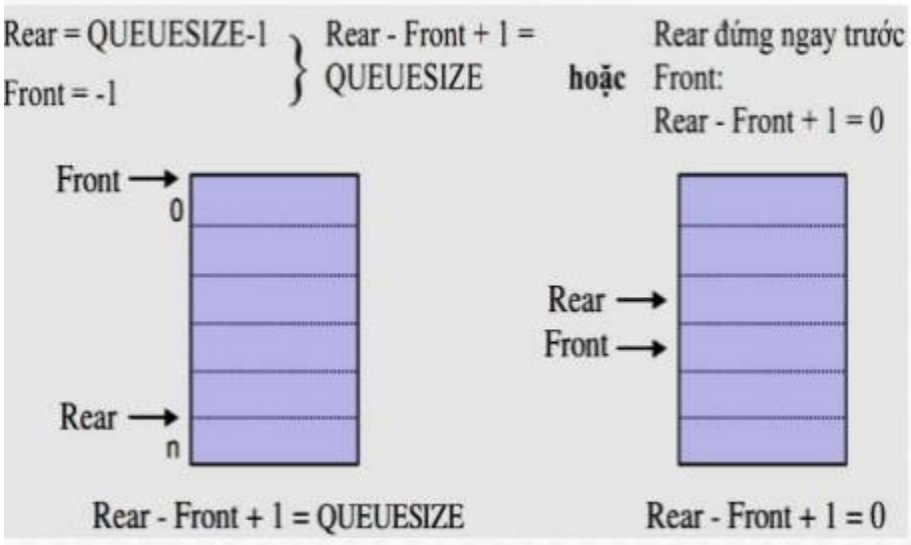
Cài đặt Queue bằng Mảng

- Cách khắc phục Hàng tràn
  - **Dùng mảng xoay vòng**
    - Xem mảng như là một vòng tròn nghĩa là khi hàng bị tràn nhưng chưa đầy ta thêm phần tử mới vào vị trí 1 của mảng, thêm một phần tử mới nữa thì thêm vào vị trí 2 (nếu có thể)
    - front có thể lớn hơn rear



Cài đặt Queue bằng Mảng

- Cách khắc phục Hàng tràn
  - **Dùng mảng xoay vòng**
    - Hàng đợi bị đầy:
      - hàng đợi không có phần tử nào trống:
      - front = 0 và rear = n hoặc rear đứng ngay trước front => nếu tiếp tục thêm vào sẽ bị mất dữ liệu.



Cài đặt Queue bằng Mảng

- **Khai báo**
  - Để quản lý một Hàng đợi, ta cần quản lý đầu hàng (front) và cuối hàng (rear)

```
#define MaxLength 100 // chiều dài tối đa của mảng
typedef int ElementType;
typedef struct Queue
{
    ElementType Data[MaxLength]; // Lưu trữ nội dung các
    phần tử
    int Front; // Chỉ số đầu
    int Rear; // Chỉ số đuôi
};
```

Cài đặt Queue bằng Mảng

- **Các thao tác cơ bản -** Khởi tạo một hàng đợi rỗng: MakeNull_Queue (Q)

```
void MakeNull_Queue(Queue &Q)
{
/*
Lúc này Front và Rear không trỏ đến vị trí hợp lệ nào trong
mảng vậy ta có thể cho front và rear đều bằng -1
*/
    Q.Front = -1;
    Q.Rear = -1;
}
```

Cài đặt Queue bằng Mảng

- **Các thao tác cơ bản -** Kiểm tra hàng đợi có rỗng không: isEmpty_Queue (Q)

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

Cài đặt Queue bằng Mảng

- **Các thao tác cơ bản -** Kiểm tra hàng đầy: isFull_Queue (Q)

```
int isFull_Queue(Queue Q)
{
    /*Hàng đầy nếu số phần tử hiện có trong hàng bằng số
    phần tử trong mảng*/
    return ((Q.Rear - Q.Front +1) == MaxLength);
}
```

Cài đặt Queue bằng Mảng

- **Các thao tác cơ bản -** Lấy giá trị của phần tử đầu tiên của Queue: Front (Q)

```
ElementType Front(Queue Q)
{
    if(isEmpty_Queue(Q))
        printf("Hang rong. Khong co ptu dau hang\n");
    else
        return Q.Data[Q.Front];
}
```

Cài đặt Queue bằng Mảng

- **Các thao tác cơ bản -** Xóa phần tử tại đầu hàng đợi: Dequeue (Q)

```
void Dequeue (Queue &Q) // Xóa phần tử đầu hàng
{
    // Nếu hàng rỗng thì không xóa được
    if(isEmpty_Queue(Q))
        printf("Hang rong. Khong xoa duoc\n");
    else
    {
        Q.Front ++;
        // Nếu như Hàng đã rỗng thì khởi tạo lại
        if (Q.Front > Q.Rear) MakeNull_Queue(Q);
    }
}
```

Cài đặt Queue bằng Mảng

- **Các thao tác cơ bản -** Thêm phần tử X vào cuối hàng: Enqueue (Q,X)
  - Các trường hợp có thể xảy ra
    - Hàng đầy: không thêm vào
    - Hàng rỗng: đặt lại chỉ số cho Front
    - Hàng tràn
      - Di chuyển tịnh tiến hàng lên Front vị trí
      - Đặt lại chỉ số cho Front và Rear

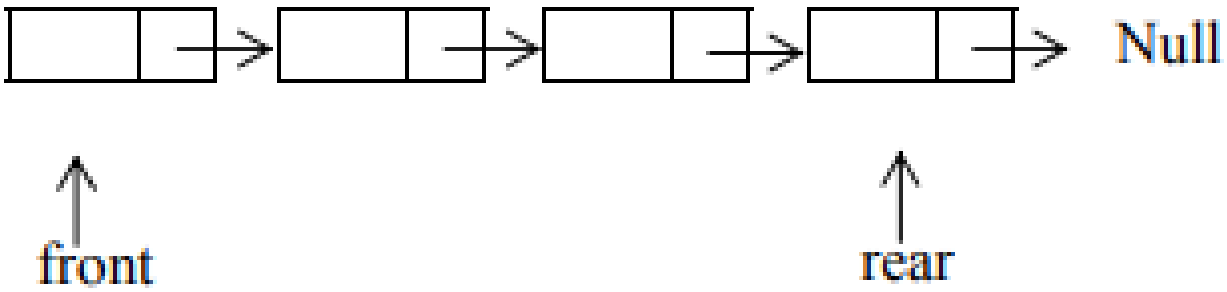
```
void Enqueue (Queue &Q, ElementType X)
{   // Nếu hàng đầy thì không thêm nữa
    if (isFull_Queue(Q)) printf ("Hang day. Khong them duoc nua!");
    else
    {   // Neu hang rong thi cho Front = 0
        if (isEmpty_Queue(Q)) Q.Front = 0;

        // Nếu hàng tràn thì di chuyển tịnh tiến rồi mới thêm vào đuôi
        phần tử
        if(Q.Rear == (MaxLength -1))
        {
            printf("Xu ly hang tran truoc khi them\n");
            // Di chuyển ra trước Front vị trí
            for (int i = Q.Front; i<= Q.Rear; i++)
                Q.Data[i-Q.Front] = Q.Data[i];
            // Đặt lại Front và Rear cho Hàng
            Q.Rear = MaxLength - 1 - Q.Front;
            Q.Front = 0;
        }

        // Them phan tu vao cuoi hang
        Q.Rear = Q.Rear +1;
        Q.Data[Q.Rear] = X;
    }
}
```

Cài đặt Queue

- Cài đặt bằng Mảng
- Cài đặt bằng Danh sách

Cài đặt Queue bằng DSLK

- Queue
  - Cài đặt như một DSLK, mỗi phần tử là 1 Node
  - Dùng 2 CON TRỎ Front và Rear trỏ tới phần tử đầu hàng và phần tử cuối hàng để quản lý hàng đợi



Cài đặt Queue bằng DSLK

- Khai báo

```
typedef int ElementType;
typedef struct Node
{
    ElementType Data;
    Node *Next;
};

typedef struct Queue
{
    Node* Front;//trỏ đầu hàng
    Node* Rear; //trỏ cuối hàng
};
```

Cài đặt Queue bằng DSLK

- **Các thao tác cơ bản -** Khởi tạo một hàng đợi rỗng: MakeNull_Queue (Q)

```
void MakeNull_Queue(Queue &Q)
{
    Q.Front = NULL;
    Q.Rear = NULL;
}
```

Cài đặt Queue bằng DSLK

- **Các thao tác cơ bản -** Kiểm tra hàng đợi có rỗng không: isEmpty_Queue (Q)

```
int isEmpty_Queue(Queue Q)
{
        return (Q.Front == NULL);
}
```

Cài đặt Queue bằng DSLK

- **Các thao tác cơ bản -** Xóa phần tử tại đầu hàng đợi: Dequeue (Q)

```
void Dequeue (Queue &Q)
{
    //Thực chất là xoá phần tử nằm ở vị trí đầu hàng
    //do đó ta chỉ cần cho front trỏ tới vị trí kế tiếp của
    //nó trong hàng
    if (isEmpty_Queue(Q))
        printf("Hang rong. Khong xoa duoc!\n");
    else
    {
        Node* temp = Q.Front;
        Q.Front = Q.Front->Next;
        free (temp);
    }
}
```

Cài đặt Queue bằng DSLK

Các thao tác cơ bản - **Thêm phần tử X vào cuối hàng**: Enqueue (Q,X)

```
void Enqueue (Queue &Q,ElementType X)
{// Chèn 1 phần tử có giá trị X vào cuối danh sách

   //Tạo 1 node mới mang giá trị X
    Node *temp = (Node*) malloc (sizeof(Node));
    temp->Data = X;
    temp->Next= NULL;
    // Nếu như hàng đợi đang rỗng thì thiết lập con trỏ cho
    Front
    if (isEmpty_Queue(Q)) Q.Front = temp;

    // Thêm vào cuối hàng Rear
    if (Q.Rear !=NULL) Q.Rear->Next = temp;

    // Thay đổi vị trí trỏ của Rear
    Q.Rear = temp;
}
```

Cài đặt Queue bằng DSLK

Các thao tác cơ bản – **Lấy giá trị của phần tử đầu tiên**: Front (Q,X)

```
ElementType Front(Queue Q)
{
    if (isEmpty_Queue(Q))
        printf("Hang doi rong, khong lay duoc.\n");
    else return (Q.Front->Data);
}
```

Ứng dụng của Queue

- Danh sách xếp hàng chờ mua vé tàu xe, chờ gửi xe, cho mượn sách ở thư viện
- Chia sẻ các tài nguyên (máy in, CPU, bộ nhớ,…)
- Tổ chức thực hiện đa chương trình
- …