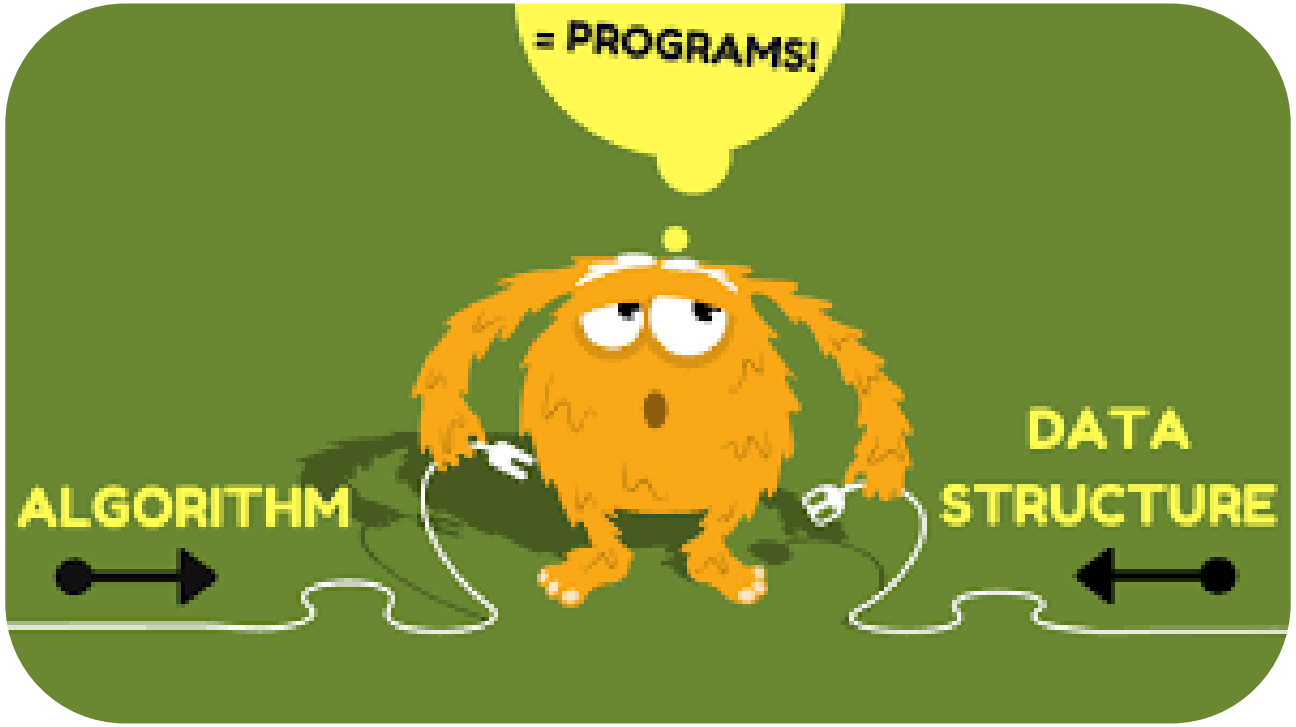
NGĂN XẾP VÀ HÀNG ĐỢI



MỤC TIÊU BÀI HỌC

- Tìm hiểu cấu trúc dữ liệu kiểu ngăn xếp và hàng đợi
- Cài đặt ngăn xếp và hàng đợi bằng mảng
- Tìm hiểu nhược điểm và cách khắc phục nhược điểm của Queue
- Tìm hiểu khả năng ứng dụng của ngăn xếp và hàng đợi

NỘI DUNG BÀI HỌC

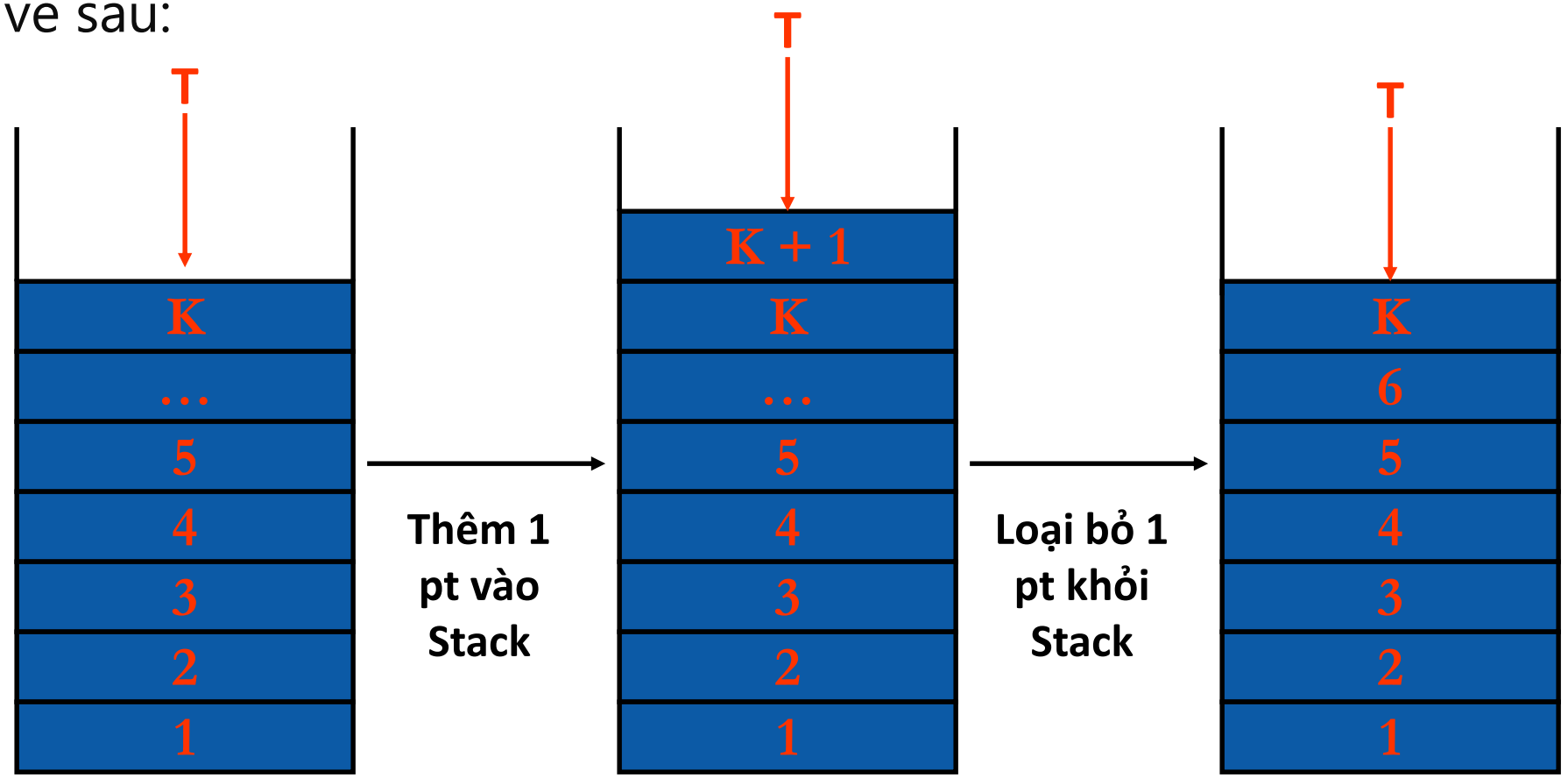
1. Stack
2. Lưu trữ Stack bằng mảng
3. Queue
4. Lưu trữ Queue bằng mảng
5. Nhược điểm cài đặt Queue bằng mảng
6. Ứng dụng của Stack và queue


1
Stack

STACK

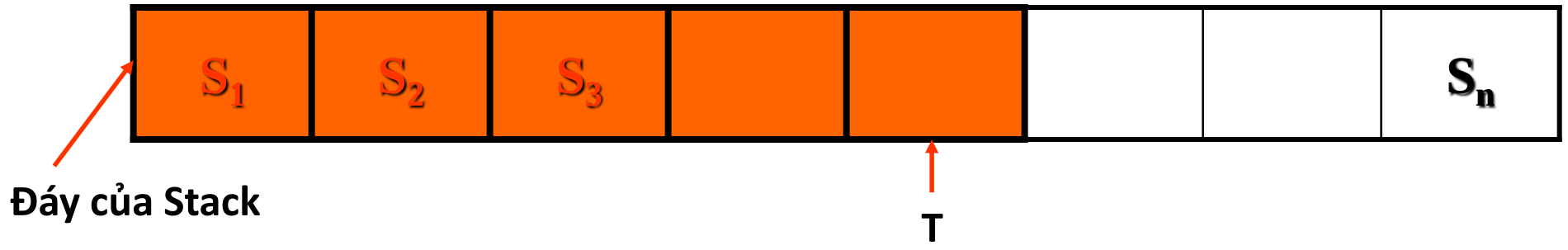
- Khái niệm: Stack (ngăn xếp) là 1 kiểu DSTT đặc biệt, mà phép bổ sung và phép loại bỏ luôn luôn thực hiện ở 1 đầu gọi là đỉnh (top).
- Có thể hình dung nó như 1 cơ cấu của 1 hộp chứa đạn súng tiểu liên. Lắp đạn hay lấy đạn ra cũng chỉ ở 1 đầu hộp. Viên đạn mới nạp vào sẽ nằm ở đỉnh (top), còn viên nạp vào đầu tiên nằm ở đáy (bottom). Viên đạn nạp vào sau cùng lại chính là viên đạn lên nòng súng trước tiên.
- Nguyên tắc "vào sau ra trước" của stack đã đưa tới 1 tên gọi khác: danh sách kiểu LIFO (Last In First Out).

- Stack có thể rỗng hoặc bao gồm 1 số phần tử. Minh hoạ stack như hình vẽ sau:



LƯU TRỮ STACK BẰNG MẢNG

- Có thể lưu trữ stack bởi 1 mảng lưu trữ S gồm n phần tử.
- Nếu T là địa chỉ của phần tử đỉnh của stack thì T sẽ có giá trị biến đổi khi stack hoạt động.
- Nếu quy ước dùng địa chỉ tương đối (như chỉ số) thì khi stack rỗng T = 0.
- Nếu mỗi phần tử của stack ứng với 1 ô nhớ thì khi 1 phần tử bị loại khỏi stack, T sẽ giảm đi 1. Có thể thấy cấu trúc lưu trữ của stack như hình vẽ sau:



LƯU TRỮ STACK BẰNG MẢNG

- Khi mô tả Stack bằng mảng:
  - ✓ Việc bổ sung 1 phần tử vào Stack tương đương với việc thêm 1 phần tử vào cuối mảng. Thủ tục Push
  - ✓ Việc loại bỏ một phần tử khỏi Stack tương đương với việc loại bỏ một phần tử ở cuối mảng. Hàm Pop
  - ✓ Stack bị tràn khi bổ sung vào mảng đã đầy. Last = Max
  - ✓ Stack là rỗng khi số phần tử thực sự đang chứa trong mảng = 0. Last = 0

CÀI ĐẶT STACK BẰNG MẢNG

- Khởi tạo mảng Data:

```
#define Max 100 //so phan tu toi da cua Stack
typedef int item; //kieu du lieu cua Stack
struct Stack
{
    int Top; //Dinh Top
    item Data[Max]; //Mang cac phan tu
};
```

CÀI ĐẶT STACK BẰNG MẢNG

- Khởi tạo danh sách rỗng:

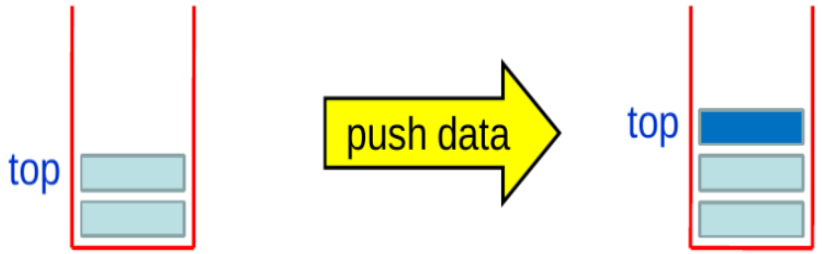
```
void Init (Stack &S) //khoi tao Stack rong
{
    S.Top = 0; //Stack rong khi Top la 0
}

int Isempty(Stack S) //kiem tra Stack rong
{
    return (S.Top == 0);
}

int Isfull(Stack S) //kiem tra Stack day
{
    return (S.Top == Max); //}
```

CÀI ĐẶT STACK BẰNG MẢNG

- Thêm phần tử vào stack:




```
void Push(Stack &S, item x) //them phan tu vao Stack
{
    if (!Isfull(S))
    {
        S.Data[S.Top] = x; //Gan du lieu
        S.Top ++; //Tang Top len 1
    }
}
```

CÀI ĐẶT STACK BẰNG MẢNG

- Lấy dữ liệu từ top nhưng không xóa:

```
int Peak(Stack S) //Lay phan tu o dau Stack nhung khong xoa
{
    return S.Data[S.Top-1]; //Lay du lieu tai Top
}
```

- Xóa và lấy dữ liệu tại Top (Pop):

CÀI ĐẶT STACK BẰNG MẢNG

- Xóa và lấy dữ liệu tại Top (Pop) (tt):

```
int Pop(Stack &S) //Loai bo phan tu khoi Stack
{
    if (!Isempty(S))
    {
        S.Top --; //Giam Top
        return S.Data[S.Top]; //Lay du lieu tai Top
    }
}
```

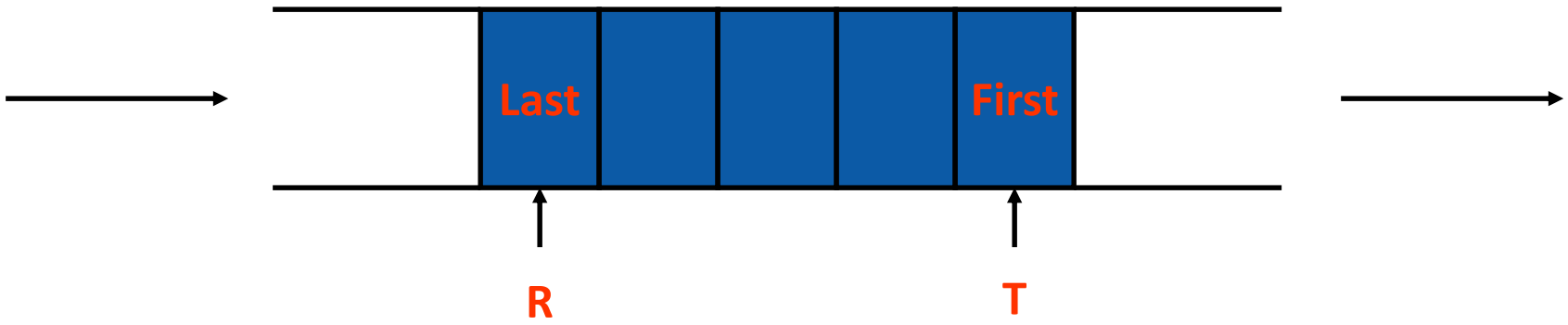
ỨNG DỤNG CỦA STACK

- Stack có rất nhiều ứng dụng trong tin học.
- Căn cứ vào nguyên tắc "vào sau ra trước" của Stack, người ta ứng dụng nó trong một số bài toán điển hình như sau:
  - ✓ Bài toán đổi cơ số
  - ✓ Bài toán định giá biểu thức số học theo ký pháp nghịch đảo Balan
  - ✓ Ngoài ra, Stack còn sử dụng trong các bài toán đệ quy: Bài toán tính N!, bài toán tháp Hà Nội, …

QUEUE

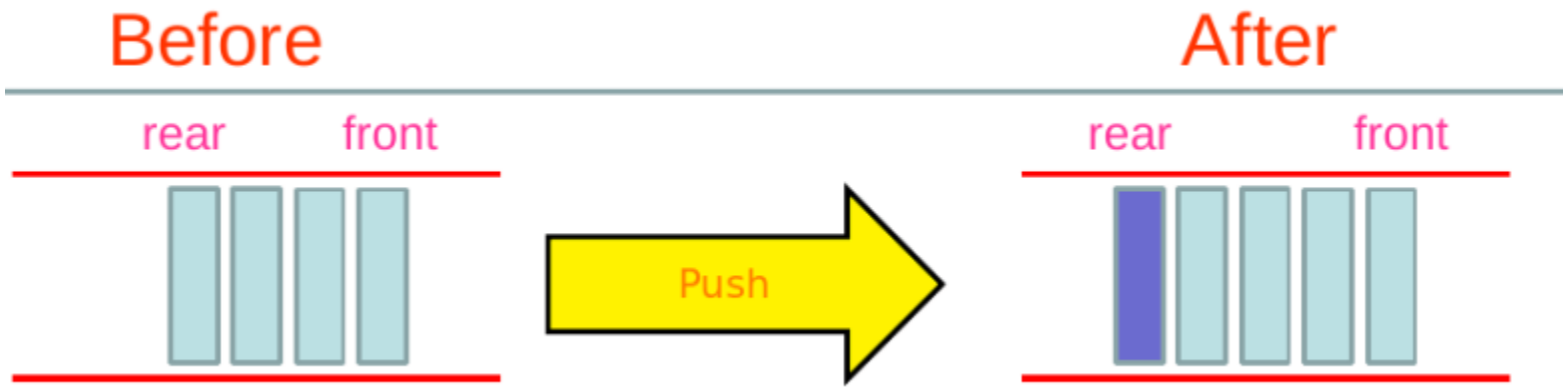
- Khái niệm: Queue (hàng đợi) là kiểu DSTT mà phép bổ sung được thực hiện ở 1 đầu, gọi là lối sau (rear), và phép loại bỏ thực hiện ở 1 đầu khác, gọi là lối trước (front).
- Cơ cấu của queue giống như 1 hàng đợi, chẳng hạn mua vé xe lửa, khách hàng nào xếp hàng trước sẽ được mua vé trước, vào ở 1 đầu và ra ở 1 đầu khác.

- Vì nguyên tắc "vào trước thì ra trước" như vậy, queue còn được gọi là danh sách kiểu FIFO (First In First Out).



CÀI ĐẶT QUEUE BẰNG MẢNG

- Khi mô tả Queue bằng mảng, ta có hai chỉ số First và Last, First lưu chỉ số phần tử đầu Queue còn Last lưu chỉ số cuối Queue.
- Khởi tạo Queue rỗng: First := 1 và Last := 0;
- Để thêm 1 phần tử vào Queue, ta tăng Last lên 1 và đưa giá trị đó vào phần tử thứ Last.



CÀI ĐẶT QUEUE BẰNG MẢNG

- Để loại một phần tử khỏi Queue, ta lấy giá trị ở vị trí First và tăng First lên 1.
- Khi Last tăng lên hết khoảng chỉ số của mảng thì mảng đã đầy, không thể đẩy thêm phần tử vào nữa.
- Khi First > Last thì tức là Queue đang rỗng.
- Như vậy chỉ 1 phần của mảng từ vị trí First tới Last được sử dụng làm Queue.

CÀI ĐẶT QUEUE BẰNG MẢNG

- Khởi tạo mảng:

```
typedef int item; //kieu du lieu

struct Queue
{
    int Front, Rear; //front: phan tu dau hang, rear: phan tu cuoi hang
    item Data[Max]; //Mang cac phan tu
    int count; //dem so phan tu cua Queue
};
```

CÀI ĐẶT QUEUE BẰNG MẢNG

➢Khởi tạo Queue rỗng:

```
void Init (Queue &Q) //khoi tao Queue rong
{
    Q.Front = 0; //phan tu dau
    Q.Rear = -1; // phan tu cuoi o -1 (khong co phan tu trong Q)
    Q.count = 0; //so phan tu bang 0
}
```

CÀI ĐẶT QUEUE BẰNG MẢNG

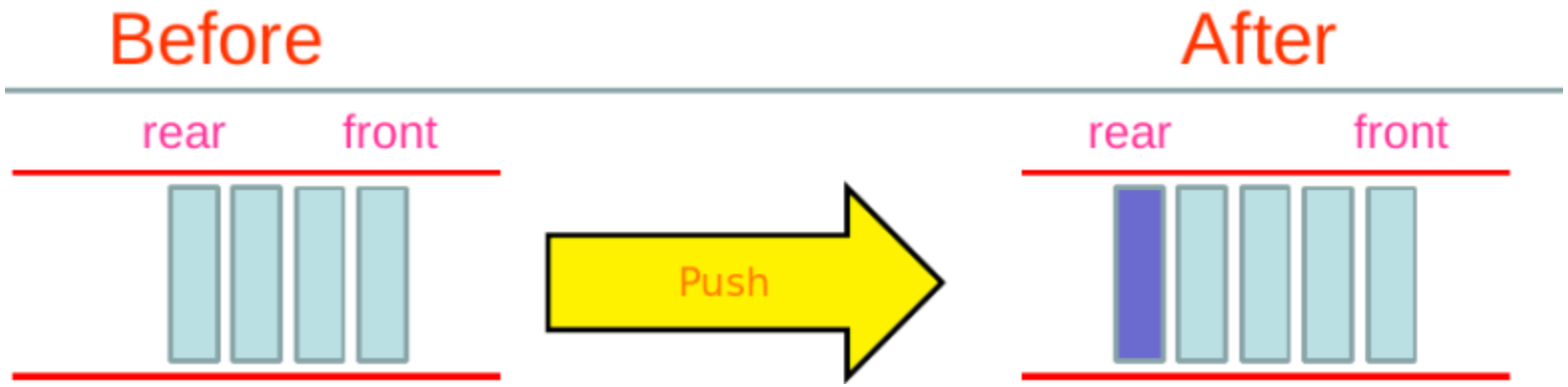
- Kiểm tra Queue rỗng, đầy:

```
int Isempty (Queue Q) //kiem tra Queue rong
{
    if (Q.count == 0) //so phan tu = 0 => rong
        return 1;
    return 0;
}

int Isfull (Queue Q) //kiem tra Queue day
{
    if (Q.count == Max) //so phan tu = Max => day
        return 1;
    return 0;
}
```

CÀI ĐẶT QUEUE BẰNG MẢNG

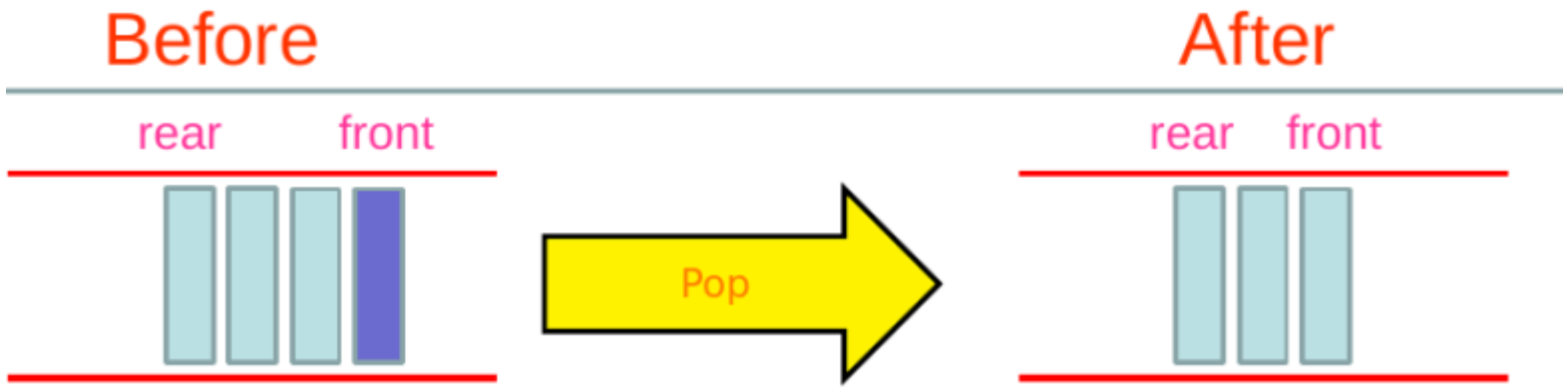
- Thêm phần tử vào Queue:




```
void Push(Queue &Q, item x) //them phan tu vao cuoi Queue
{
    if (Isfull(Q)) printf("Hang doi day !");
    else
    {
        Q.Data[++Q.Rear] = x; //tang Rear len va gan phan tu vao
        Q.count++; //tang so phan tu len }}
```

CÀI ĐẶT QUEUE BẰNG MẢNG

- Xóa phần tử đầu Queue




```
int Pop(Queue &Q) //Loai bo phan tu khoi dau hang doi
{ if (Isempty(Q)) printf("Hang doi rong !");
    else {  item x = Q.Data[Q.Front];
        for (int i=Q.Front; i<Q.Rear; i++) //di chuyen cac phan tu ve dau hang
            Q.Data[i] = Q.Data[i+1];
            Q.Rear--; // giam vi tri phan tu cuoi xuong
            Q.count--;//giam so phan tu xuong
        return x; //tra ve phan tu lay ra }}
```

CÀI ĐẶT QUEUE BẰNG MẢNG

- Xem thông tin phần tử đầu Queue:

```
item Qfront (Queue Q) //xem thong tin phan tu dau hang
{
    if (Isempty(Q)) printf("Hang doi rong !");
    else return Q.Data[Q.Front];
}
```

NHƯỢC ĐIỂM KHI LƯU TRỮ QUEUE BẰNG MẢNG

- Với 1 mảng kích thước tối đa 10000 phần tử đã cài đặt, thấy rằng nếu như làm 6000 lần Push, rồi 6000 lần Pop, rồi lại 6000 lần Push thì vẫn không có vấn đề gì xảy ra.
- Lý do vì chỉ số Last lưu đỉnh của Stack sẽ được tăng lên 6000 rồi lại giảm 0 rồi lại tăng trở lại lên 6000.

NHƯỢC ĐIỂM KHI LƯU TRỮ Q BẰNG MẢNG

- Nhưng đối với cách cài đặt Queue như trên thì sẽ gặp thông báo lỗi tràn mảng, bởi mỗi lần Push, chỉ số cuối hàng đợi Last cũng tăng lên và không bao giờ bị giảm đi cả.
- Đó chính là nhược điểm mà ta nói tới khi cài đặt: Chỉ có các phần tử từ vị trí First tới Last là thuộc Queue, các phần tử từ vị trí 1 tới First - 1 là vô nghĩa.

KHẮC PHỤC NHƯỢC ĐIỂM

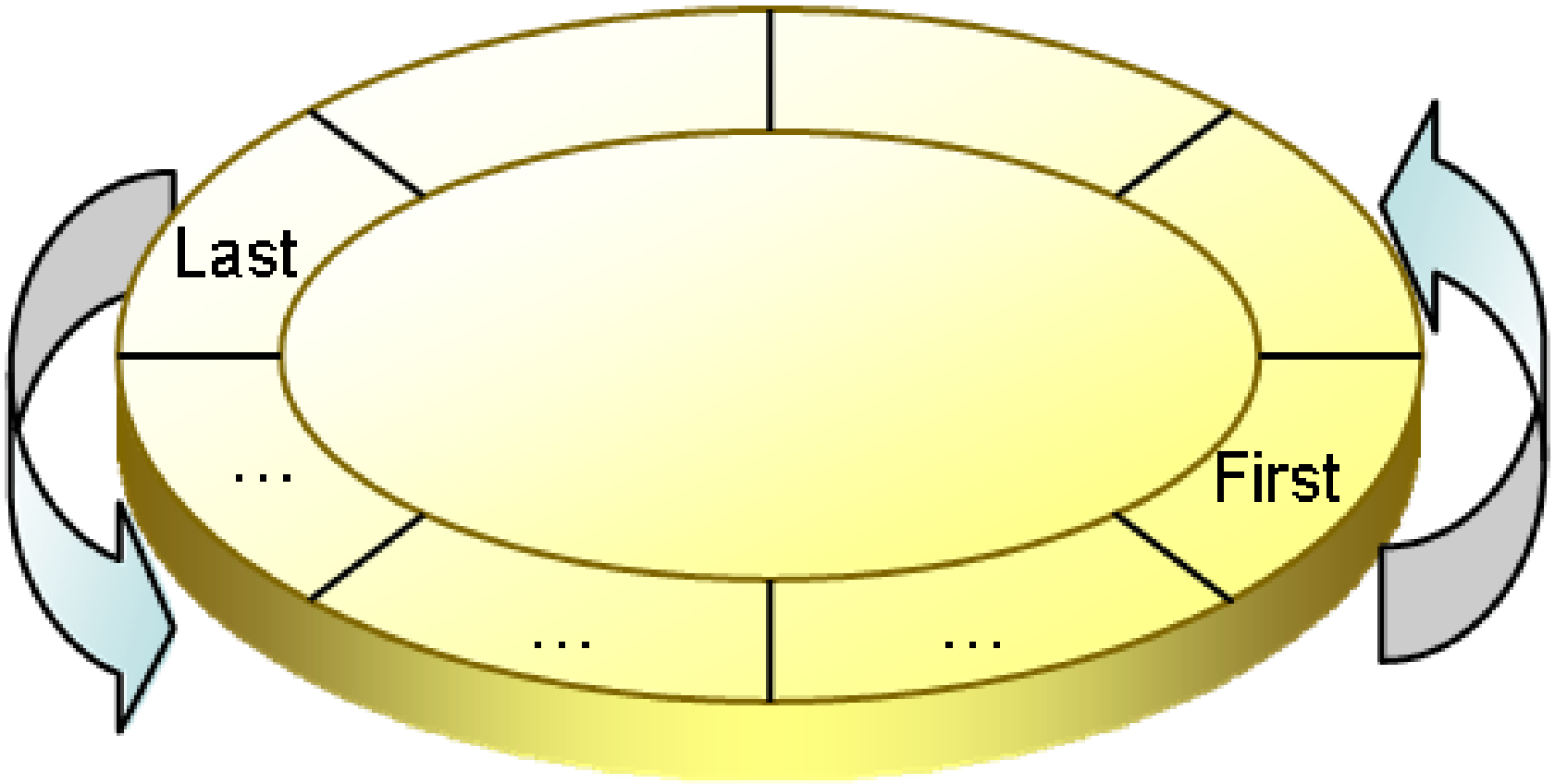
- Để khắc phục, mô tả Queue bằng 1 danh sách vòng (biểu diễn bằng mảng hoặc cấu trúc liên kết), coi như các phần tử của Queue được xếp quanh vòng theo 1 hướng nào đó.
  - ✓ Các phần tử nằm trên phần cung tròn từ vị trí First tới Last là các phần tử của Queue.
  - ✓ Có thêm 1 biến n để lưu số phần tử của Queue.

KHẮC PHỤC NHƯỢC ĐIỂM

- Việc thêm 1 phần tử vào Queue tương đương với việc ta dịch chỉ số Last theo vòng 1 vị trí rồi đặt giá trị mới vào đó.
- Việc loại bỏ 1 phần tử trong Queue tương đương với việc lấy ra phần tử tại vị trí First rồi dịch chỉ số First theo vòng.

❖ Lưu ý là trong thao tác Push và Pop phải kiểm tra Queue tràn hay Queue cạn nên phải cập nhật lại biến n.

MÔ TẢ QUEUE BẰNG DANH SÁCH VÒNG



CÀI ĐẶT QUEUE DẠNG VÒNG

```
void Push_Circular(Queue &Q, item x) //them phan tu vao cuoi hang doi vong
{
    if (Isfull(Q)) printf("Hang doi day !");
    else
    {
        Q.Data[(++Q.Rear) % Max] = x;
        //tang Rear len va gan phan tu vao, Neu Rear dang o vi tri Max-1 thi tang ve vi
        Q.count++; //tang so phan tu len
    }
}
```

CÀI ĐẶT QUEUE DẠNG VÒNG

```
int Pop_Circular(Queue &Q) //Loai bo phan tu khoi dau hang doi vong
{
    if (Isempty(Q)) printf("Hang doi rong !");
    item x = Q.Data[Q.Front];
    Q.Front = (Q.Front++) % Max; // tang vi tri phan dau tien len,
                                 //neu dang o Max-1 thi ve 0
    Q.count--;//giam so phan tu xuong
    return x; //tra ve phan tu lay ra
}
```

ỨNG DỤNG CỦA STACK VÀ QUEUE

- Cần danh sách những sản phẩm vừa được mua gần nhất hoặc những khách hàng vừa mới truy cập. Sử dụng Stack để lưu trữ các thông tin trên, chỉ cần dùng method pop để lấy ra là chúng ta có được kết quả mong muốn.
- Queue thì mình sử dụng trong ghi log, dữ liệu log sẽ được thêm vào Queue và ghi lần lượt xuống đĩa, đảm bảo dữ liệu log theo thứ tự thời gian...

TỔNG KẾT

- Ngăn xếp là danh sách các phần tử mà việc thêm vào hay lấy ra các phần tử chỉ thực hiện ở Đỉnh ngăn xếp.
- Hàng đợi là danh sách các phần tử mà việc thêm phần tử được thực hiện ở cuối hàng, việc lấy ra phần tử thực hiện ở đầu hàng.


感謝你
Danke
Gracias
Grazie
Merci
Thanks!
THANK
YOU!


VAIO