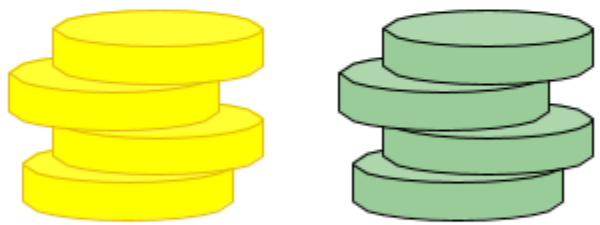
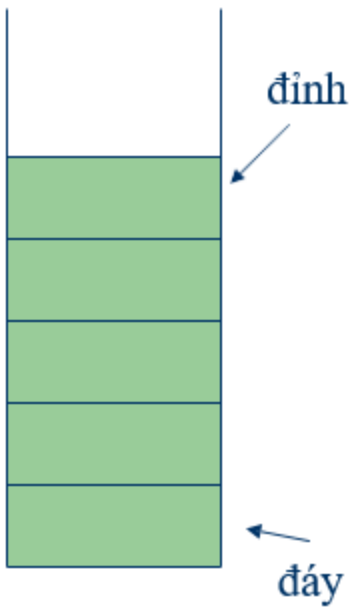
PHÂN TÍCH THIẾT KẾ GIẢI THUẬT

Chương III:
Stack và Queue



Danh sách kiểu ngăn xếp - Stack

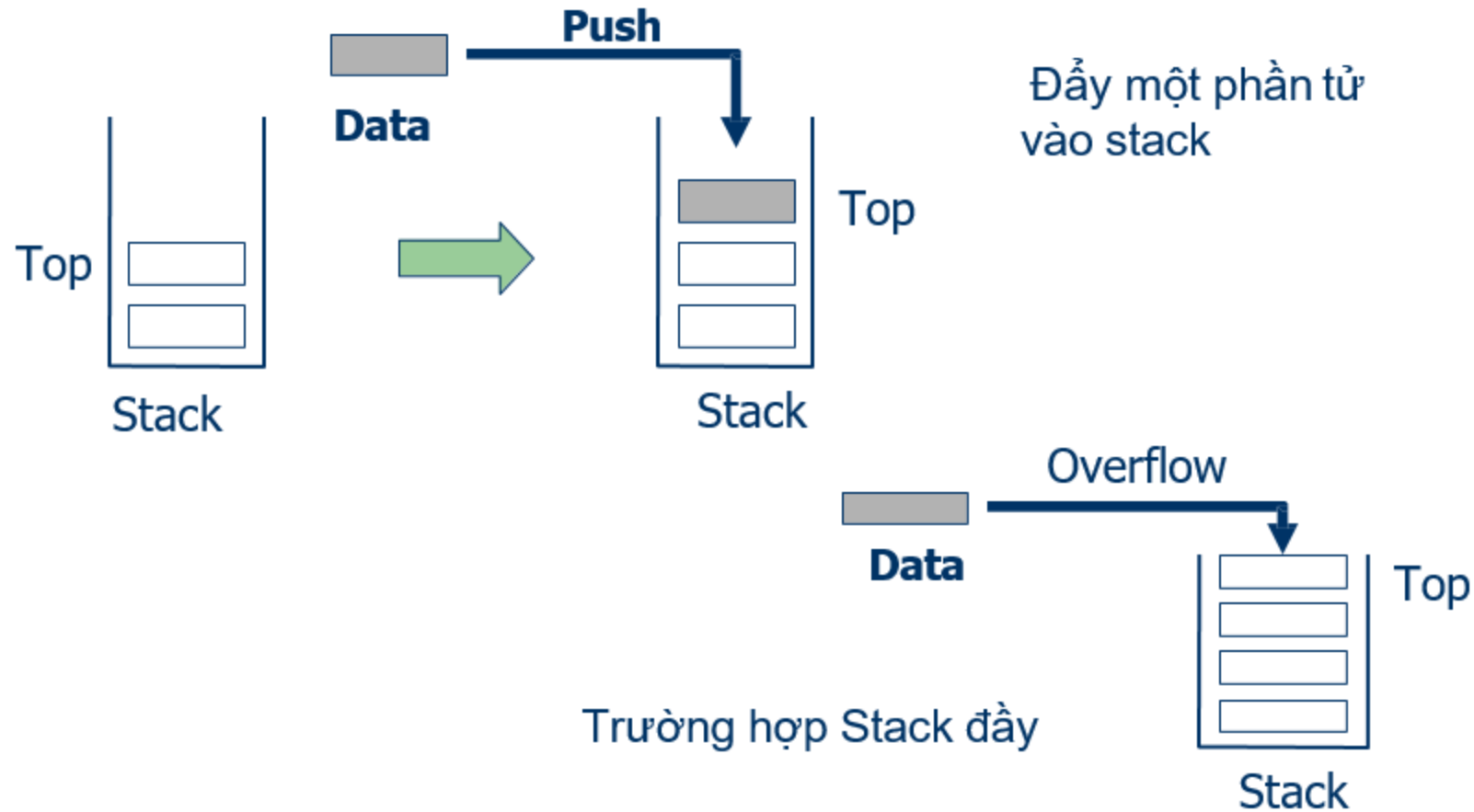
- Stack
  - Một kiểu danh sách tuyến tính đặc biệt
  - Phép bổ sung và phép loại bỏ tuân thủ theo cơ chế “vào sau ra trước” (last in first out) , được thực hiện ở đầu đỉnh



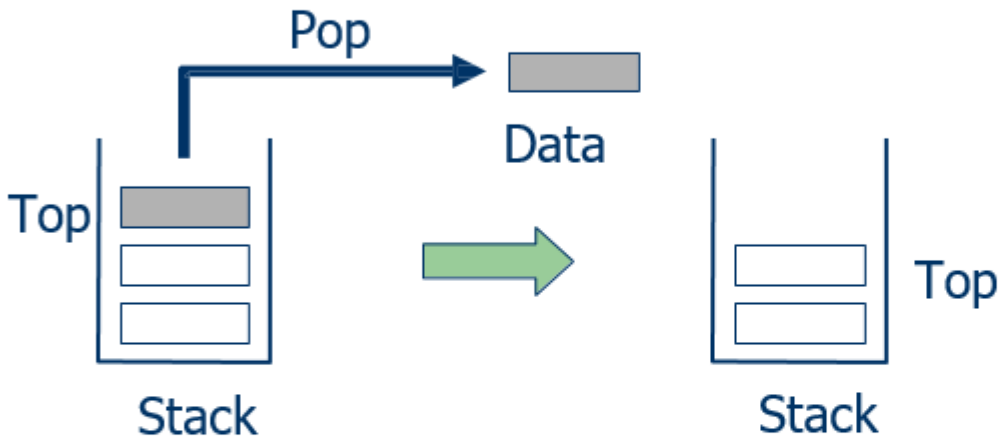
Danh sách kiểu ngăn xếp - Stack

- Hai thao tác cơ bản đối với danh sách kiểu ngăn xếp
  - push(Element e) : bổ sung phần tử vào Stack
  - Element pop(): Loại bỏ và trả ra giá trị của phần tử ở đỉnh Stack
- Các thao tác khác
  - Int size(): Trả ra số các phần tử trong Stack
  - Boolean isEmpty(): Kiểm tra xem Stack có rỗng không
  - Element top(): Trả ra giá trị của phần tử ở đỉnh Stack

Các thao tác cơ bản của Stack

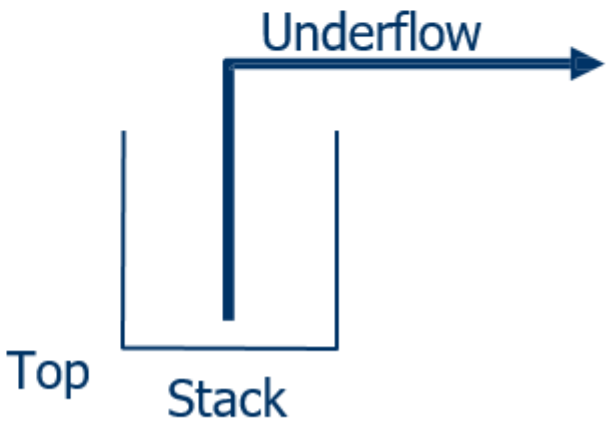


Các thao tác cơ bản của Stack



Lấy ra phần tử ở đỉnh stack

Trường hợp Stack cạn



Danh sách kiểu ngăn xếp

Thao tác	Output	Stack
create()	-	[ ]
push(5)	-	[5]
push(3)	-	[5,3]
pop()	3	[5]
push(7)	-	[5,7]
top()	7	[5,7]
pop()	7	[5]
pop()	5	[]
isEmpty()	true	[]
push(9)	-	[9]
push(8)	-	[9,8]
push(7)	-	[9,8,7]
size()	3	[9,8,7]

Ứng dụng của Stack

- Lưu trữ các trang web đã từng được duyệt trên Web browser
- Cài đặt thao tác Undo trong các phần mềm soạn thảo
- Lưu danh sách các lời gọi hàm trong Java Virtual Machine

Lưu trữ kế tiếp của Stack

- Stack có thể được lưu trữ bởi một vector lưu trữ S, gồmn ô nhớ kế tiếp nhau
- Đỉnh stack được xác định bởi một chỉ số T
  - T sẽ được cập nhật nếu có thao tác bổ sung hay loại bỏ được thực hiện trên stack

Lưu trữ kế tiếp của Stack

- Giải thuật bổ sung một phần tử vào Stack được lưu trữ kế tiếp

```
Procedure PUSH(S,T,X)
Begin
{S: vector lưu trữ có n ô nhớ; T: chỉ số của phần tử đỉnh stack
    hiện thời; X là giá trị cần thêm vào }
1. if T >= n then begin
      write('STACK TRÀN');
      return;
    end;
 2. T:= T+1;
    S[T] := X;
End
```

Lưu trữ kế tiếp của Stack

- Giải thuật lấy ra phần tử ở đỉnh của Stack được lưu trữ kế tiếp

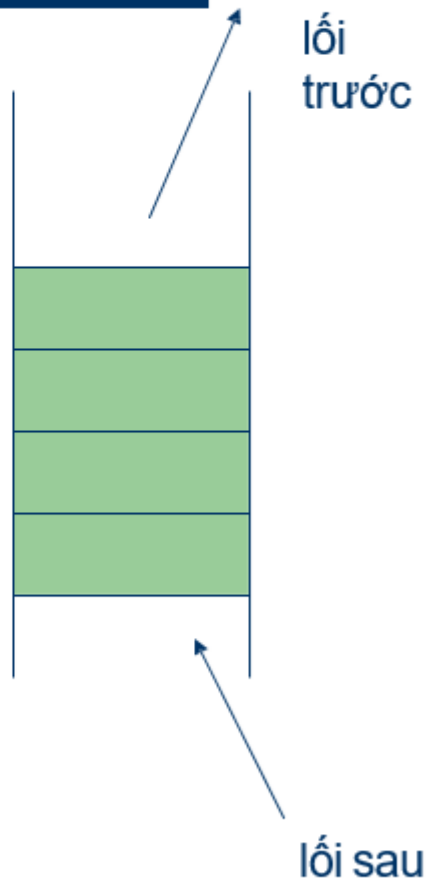
```
Procedure POP(S,T, Y)
Begin
{S: stack đang xét ; T: chỉ số của phần tử tại đỉnh stack hiện thời;
Phần tử được lấy ra sẽ được bảo lưu sử dụng biếnY  }
1.    if T = 0 then begin
        write('STACK CẠN'); return;
      end;
2. Y:= S[T];
     S[T] := null;
     T:= T-1;
End
```

Hiệu năng và Hạn chế

- Hiệu năng
  - n là số phần tử của stack
  - Không gian lưu trữ : O(n)
  - Các thao tác cơ bản có độ phức tạp O(1)
- Hạn chế
  - Kích thước tối đa phải được xác định trước và không được thay đổi
  - Xảy ra tràn stack

Danh sách kiểu hàng đợi - Queue

- Queue
  - Queue (Hàng đợi) là một kiểu danh sách tuyến tính đặc biệt
  - Phép bổ sung và loại bỏ hoạt động theo cơ chế “vào trước ra trước” (first in first out) ; bổ sung ở một đầu thì loại bỏ ở đầu kia



Danh sách kiểu hàng đợi - Queue

- Hai hàm cơ bản đối với danh sách kiểu hàng đợi
  - enqueue(Element e)
  - Element dequeue()
- Các hàm khác
  - create():
  - size() :
  - isEmpty():
  - Element front()

Danh sách kiểu hàng đợi – Queue

Thao tác	Output	Queue
create()	-	[ ]
enqueue(5)	-	[5]
enqueue(3)	-	[5,3]
dequeue()	5	[3]
enqueue(7)	-	[3,7]
front()	3	[3,7]
dequeue()	3	[7]
dequeue()	7	[]
isEmpty()	true	[]
enqueue(9)	-	[9]
enqueue(8)	-	[9,8]
enqueue(7)	-	[9,8,7]
size()	3	[9,8,7]

Ứng dụng của Queue

- Hàng đợi trong các phòng bán vé
- Truy nhập vào các thiết bị dùng chung tại văn phòng (ví dụ máy in)

Lưu trữ kế tiếp đối với Queue

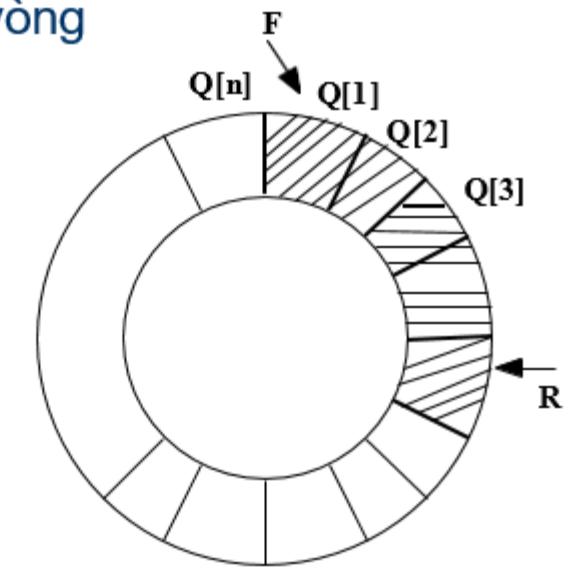
- Sử dụng một vector lưu trữ Q gồm n ô nhớ kế tiếp nhau để biểu diễn một Queue
- Cần nắm được hai chỉ số
  - R: Chỉ số của phần tử nằm ở lối sau của Q
  - F: Chỉ số của phần tử ở lối trước của Q

Lưu trữ kế tiếp đối với Queue

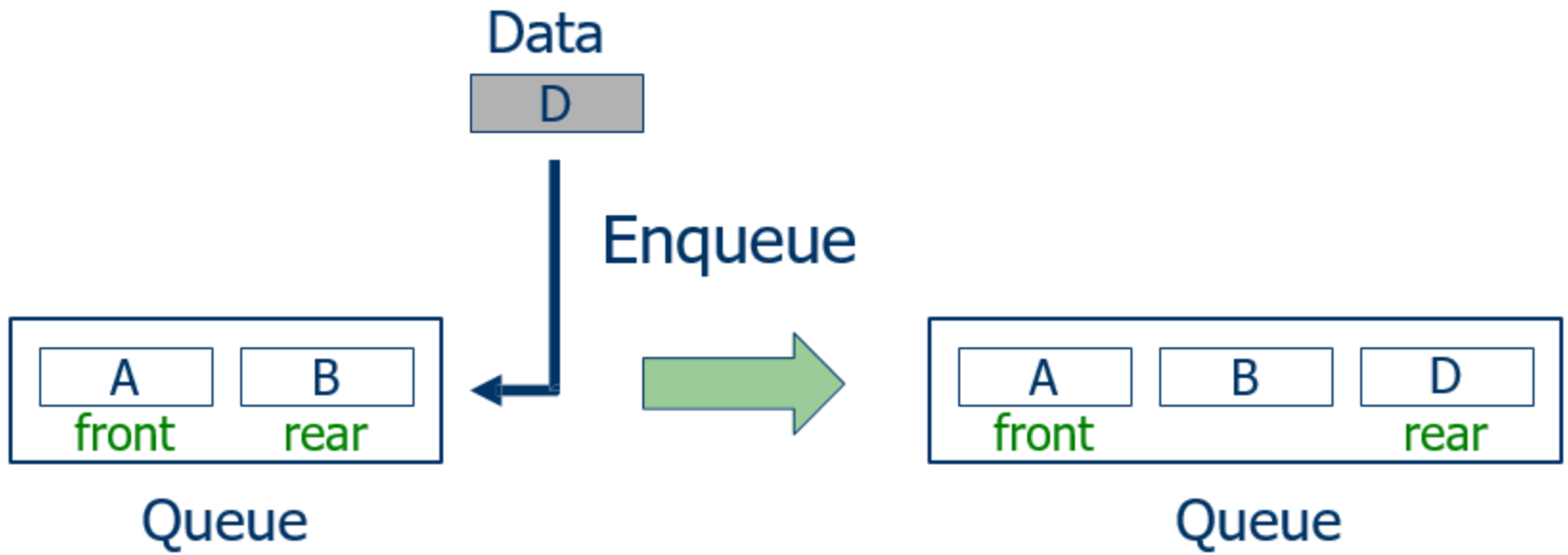
- Khi Queue rỗng thì F = R = 0
- Khi bổ sung thêm một phần tử vào Queue thì R tăng lên 1
- Khi lấy ra một phần tử trong Queue thì F tăng lên 1
- Nhược điểm của cách tổ chức lưu trữ này
  - Các phần tử trong Queue sẽ dịch chuyển khắp không gian nhớ nếu liên tục thực hiện bổ sung rồi loại bỏ
  - Hiện tượng TRÀN vẫn xảy ra khi vector lưu trữ Q vẫn còn chỗ nhưng R = n

Lưu trữ kế tiếp đối với Queue

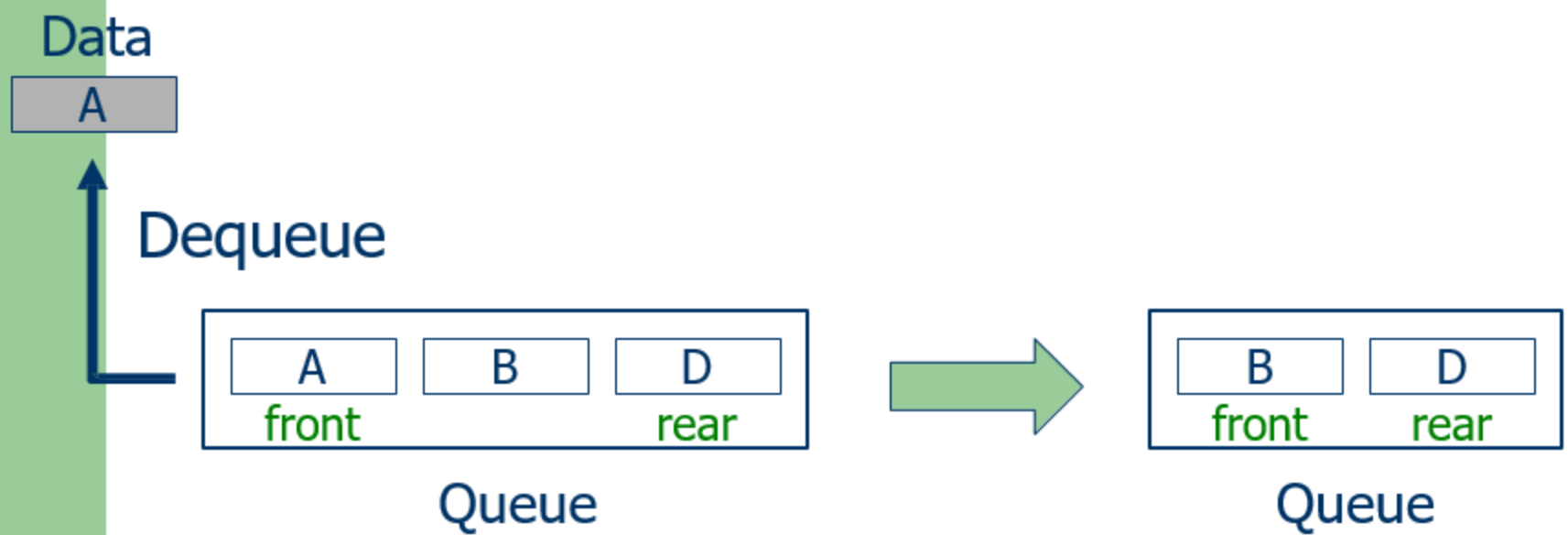
- Khắc phục các vấn đề bằng cách coi vector lưu trữ Queue được tổ chức dưới dạng vòng
  - Q[1] được coi như đứng sau Q[n]



Các thao tác cơ bản của Queue



Các thao tác cơ bản của Queue



Lưu trữ kế tiếp đối với Queue

- Giải thuật bổ sung vào Queue được lưu trữ trong vector Q gồmn phần tử và được tổ chức dưới dạng thường

```
Procedure ENQUEUE(Q,F,R,X)
Begin
1.  if (R >= n) then begin
            write('QUEUE TRÀN');
            return;
    end;
2.  {Q rỗng} if F = 0 then F:= R:= 1;
3.else R:= R+ 1;
4. Q[R] := X;  End
```

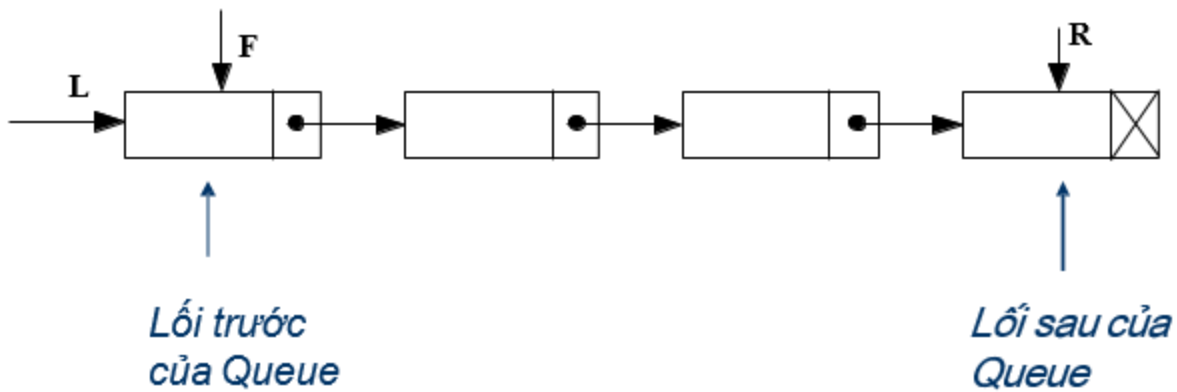
Lưu trữ kế tiếp đối với Queue

- Giải thuật lấy ra (loại bỏ) khỏi Queue

```
Procedure DEQUEUE(Q,F,R, Y)
Begin
{ Y là biến lưu trữ phần tử được lấy ra }
1.  if F = 0 then begin
            write('QUEUE CẠN');
            return;
    end;
2.  Y:= Q[F]; {lưu giá trị của phần tử cần lấy}
3.  if F = R=1 then F:= R:= 0; { Queue chỉ còn một phần tử}
4.else F:= F+ 1;
End
```

Lưu trữ móc nối đối với Queue

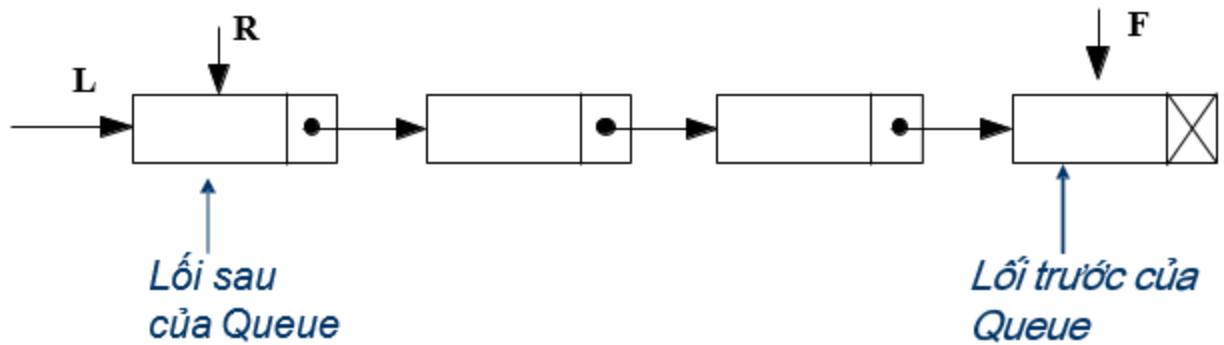
- Cách tiếp cận 1: Sử dụng danh sách nối đơn
  - Lối trước của Queue là đầu danh sách
  - enqueue(o): bổ sung phần tử vào cuối danh sách
  - dequeue() : loại bỏ phần tử ở đầu danh sách



  - Luôn nắm giữ hai con trỏ F trỏ tới phần tử ở lối trước của queue, R trỏ tới phần tử ở lối sau của queue

Lưu trữ móc nối đối với Queue

- Cách tiếp cận 2:
  - Lối sau của Queue là đầu danh sách
  - enqueue(o): bổ sung phần tử vào đầu danh sách
  - dequeue() : loại bỏ phần tử ở cuối danh sách



Lưu trữ móc nối đối với Queue

- Giải thuật bổ sung một phần tử vào Queue lưu trữ trong danh sách móc nối– Bổ sung vào cuối danh sách

```
Procedure ENQUEUE(F,R,X)
Begin
1.   {Khởi tạo nút mới} Call
     New(p);  INFO(p) := X;
     LINK(p) :=Null;
2.   {Danh sách đã cho rỗng} if F = Null then F:= R:= p;
3.else LINK(R) := p; R:= p;
End
```

Lưu trữ móc nối đối với Queue

- Giải thuật loại bỏ phần tử khỏi Queue – Loại bỏ phần tử đầu tiên trong danh sách

```
Procedure DEQUEUE(F,R, Y)
Begin
{ Y là biến lưu trữ phần tử được lấy ra }
1.  p:= F; Y:= INFO(p);
2.  {Danh sách ban đầu chỉ có một phần tử}
    if (F = R) and (F <> Null) then F:= R:= Null;
2.  else F:= LINK(p);
3.Call Dispose(p) ;
End
```

Hàng đợi hai đầu- DEQueue

- DeQueue
  - Hàng đợi hai đầu là một cấu trúc dữ liệu dạng hàng đợi nhưng nó hỗ trợ phép bổ sung và loại bỏ ở cả đầu vàcuối
- Các hàm cơ sở của hàng đợi hai đầu
  - insertFirst(o)
  - insertLast(o):
  - removeFirst()
  - removeLast()
- Các hàm khác
  - first()
  - last()
  - size()
  - isEmpty()
  - create()

Hàng đợi hai đầu- DeQueue

Thao tác	Output	DeQueue
create()	-	[ ]
insertFirst(5)	-	[5]
insertFirst(3)	-	[3,5]
removeFirst()	3	[5]
insertLast(7)	-	[5,7]
removeFirst()	5	[7]
removeLast()	7	[]
removeLast()	error	[]
isEmpty()	true	[]
insertLast(9)	-	[9]
insertFirst(8)	-	[8,9]
insertLast(7)	-	[8,9,7]
size()	3	[8,9,7]

Lưu trữ móc nối với DeQueue

- DeQueue được lưu trữ sử dụng cấu trúc danh sách móc nối kép (Doubly Linked – List)
  - Mỗi nút trong danh sách ngoài trường INFO chứa dữ liệu còn có 2 trường con trỏ
    - PREV
    - NEXT
  - Cần nắm được hai con trỏ, con trỏ L trỏ tới nút cực trái, con trỏ R trỏ tới nút cực phải của danh sách
  - Với danh sách rỗng , L = R = NULL

Lưu trữ móc nối đối với DeQueue

- Giải thuật bổ sung phần tử vào đầu một DeQueue lưu trữ trong một danh sách nối kép
- Giải thuật loại bỏ phần tử đầu một DeQueue lưu trữ trong một danh sách nối kép