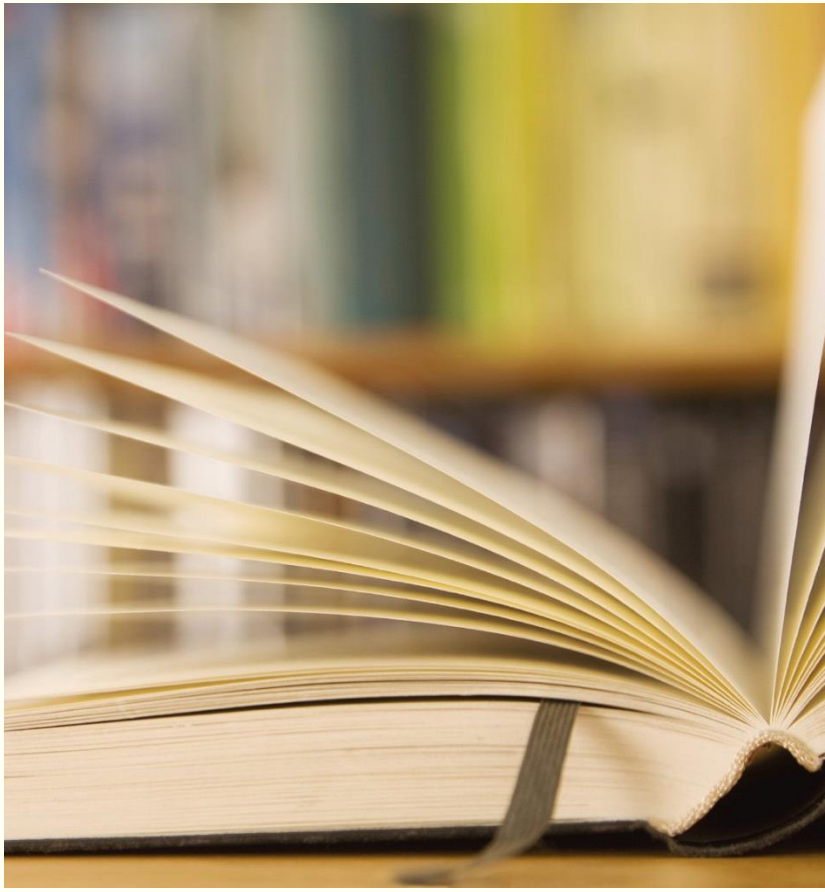
HÀNG ĐỢI



Mục tiêu

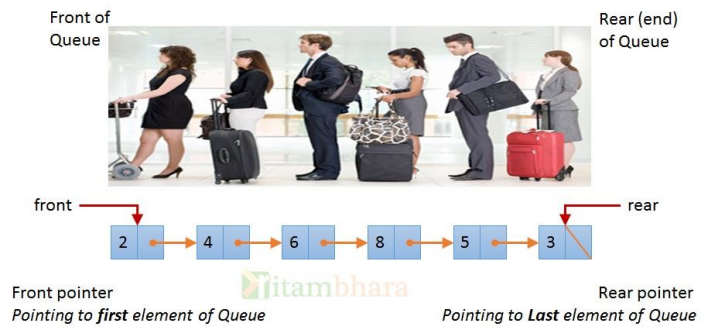
- Hiểu được khái niệm cấu trúc dữ liệu hàng đợi
- Biết cách cài đặt hàng đợi bằng mảng và con trỏ
- Biết cách vận dụng hàng đợi để giải một số bài toán

Nội dung

- Khái niệm
- Các phép toán
- Cài đặt bằng mảng
- Cài đặt bằng con trỏ

Khái niệm

- Hàng đợi là một danh sách
- Các phép toán chèn, xoá phần tử được giới hạn ở các điểm cuối của danh sách
- Phép toán chèn phần tử mới được thực hiện ở một điểm cuối - gọi là đuôi của hàng đợi
- Phép loại bỏ một phần tử được thực hiện ở đầu kia của hàng đợi - gọi là đầu của hàng đợi



Các phép toán cơ bản

- Khởi tạo hàng đợi rỗng
- Kiểm tra hàng đợi rỗng hay không?
- Kiểm tra hàng đợi đầy hay không?
- Chèn phần tử vào đuôi hàng đợi
- Tìm và lấy ra phần tử ở đầu hàng đợi

Cài đặt hàng đợi bằng mảng

```
Struct Queue{
  int front,rear;
  item data[max];
  int count;
};
```

Cài đặt hàng đợi bằng mảng (tiếp…)

- Khởi tạo hàng đợi rỗng

```
void Init( Queue &Q){
  Q.front: =0;
  Q.rear :=-1;
   Q.count=0;
End;
```

Cài đặt hàng đợi bằng mảng (tiếp…)

- Kiểm tra hàng đợi rỗng ?
- Hàm isEmpty :
  - Input: Hàng đợi Q
  - Output:
    - 1: Nếu hàng đợi rỗng
    - 0: Hàng đợi đã có phần tử

Cài đặt hàng đợi bằng mảng (tiếp…)

```
int isEmpty( Queue Q) {
  if (Q.count==0)
        return 1;
  return 0;
}
```

Cài đặt hàng đợi bằng mảng (tiếp…)

- Kiểm tra hàng đợi đầy?
- Xây dựng hàm isFull:
  - Input: Hàng đợi Q
  - Output:
    - 1: Hàng đợi đầy
    - 0: ngược lại

Cài đặt hàng đợi bằng mảng (tiếp…)

```
int isFull(Queue Q){
  if (Q.count ==max)
        return 1;
  else
        return 0;
}
```

Cài đặt hàng đợi bằng mảng (tiếp…)

- Thêm phần tử vào đuôi hàng đợi
- Thuật toán:
  - Nếu hàng đợi đầy thì dừng, ngược lại làm các bước sau:
    - Gán rear = rear +1;
    - Gán phần tử tại vị trí rear bằng phần tử x cần thêm vào hàng đợi

Cài đặt hàng đợi bằng mảng (tiếp…)

- Xây dựng hàm Add(Queue &Q; Item x)
  - Input:
    - Q: Hàng đợi
    - x : Phần tử cần chèn có kiểu Item

Cài đặt hàng đợi bằng mảng (tiếp…)

```
void Push(Queue &Q, item x){
    if (Isfull(Q)) printf("Hang doi day !");
    else    {
        Q.Data[++Q.Rear] = x;
        Q.count++;
    }
}
```

Cài đặt hàng đợi bằng mảng (tiếp…)

- Tìm và lấy ra phần tử ở đầu hàng đợi
- Thuật toán:
  - Nếu hàng đợi rỗng thì dừng, ngược lại làm các bước:
    - Lấy phần tử tại vị trí front
    - Nếu front = rear đánh dấu hàng rỗng bằng cách gán front =1, rear =0 ngược lại front = front+1

Cài đặt hàng đợi bằng mảng (tiếp…)

Xây dựng thủ tục Remove(Queue Q;Item &x)

```
void Remove(Queue Q;Item &x);
Begin
  if(!isEmpty(Q)) {
        x = Q.Data[Q.front];
        if(Q.front== Q.rear) then
              Q.front =1;Q.rear =0;}
        else Q.front = Q.front +1;
   end;
End;
```

Cài đặt hàng đợi bằng mảng vòng tròn

```
Const max = N;
Type Queue = Record
  count,front, rear :Integer;
  element= array[1.. max] of item;
end;
Var Q:Queue;
```

Cài đặt hàng đợi bằng mảng vòng tròn

```
Procedure Init( var Q: Queue);
Begin
  Q.count: =0;Q.front :=1; Q.rear :=0;
End;
Function isEmpty( Q:Queue):boolean;
Begin
  isEmpty :=(Q.count =0);
End;
```

Cài đặt hàng đợi bằng mảng vòng tròn

```
Function isFull(Q:Queue):boolean;
Begin
  isFull:= (Q.count = max);
End;
```

Cài đặt hàng đợi bằng mảng vòng tròn

```
Procedure Add(var Q:Queue; x:Item);
Begin
  if (not isFull(Q)) then
  begin
        if (Q.rear  = max) then Q.rear :=1;
        else  Q.rear: = Q.rear +1;
        Q.element[Q.rear] := x;
        Q.count: = Q.count +1;
   end;
End;
```

Cài đặt hàng đợi bằng mảng vòng tròn

```
Procedure Remove(Q: Queue; var: x: Item );
begin
   if (not isEmpty(Q)) then
   begin
            x := Q.element[Q.front];
            if(Q.front  =Q.rear) then begin
                     Q.front: =1; Q.rear: =0;
            end else if(Q.front =  max) then Q.front: =1;
                     else Q.front := Q.front+1;
            Q.count := Q.count+1;
  end;
End;
```

Cài đặt hàng đợi bằng con trỏ

```
struct Node
{
    item Data;
    Node * Next;
};
struct Queue
{
    Node * Front, *Rear;
   int count;
};
```

Cài đặt hàng đợi bằng con trỏ

```
void Init(Queue &Q)
{
    Q.Front = Q.Rear = NULL;
    Q.count = 0;
}
int Isempty (Queue Q) {
if (Q.count == 0) return 1;
    return 0;
}
```

Cài đặt hàng đợi bằng con trỏ

```
void Push(Queue &Q, item x) {
   Node *P = MakeNode(x);
   if (Isempty(Q))       {
      Q.Front = Q.Rear = P;
      }    else       {
      Q.Rear->Next = P;
      Q.Rear = P;
      }
   Q.count ++ ;
}
```

Cài đặt hàng đợi bằng con trỏ

```
void Pop(Queue Q, item &x) {
   if (Isempty(Q))   {
      printf("Hang doi rong !");
      return 0;
   }  else      {
      item x = Q.Front->Data;
      if (Q.count == 1)
         Init(Q);
      else
                  Q.Front = Q.Front->Next;
      Q.count --;
      return x;
   }
```

Bài tập

- Cho một số n, hãy đưa số dãy số siêu nguyên tố nhỏ hơn hoặc bằng n, các số đã được sắp xếp tăng dần.

Tổng kết

- Danh sách thao tao thêm và loại bỏ phần tử ở hai phía khác nhau
- Danh sách "FIFO"

Tiếp theo…

- Sắp xếp