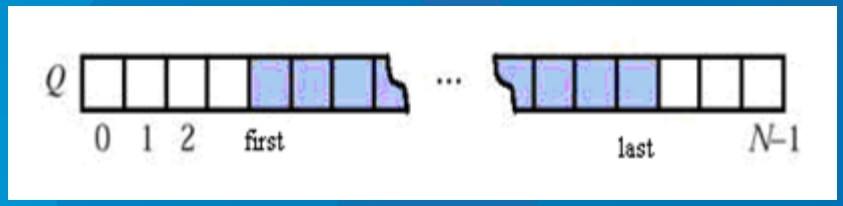
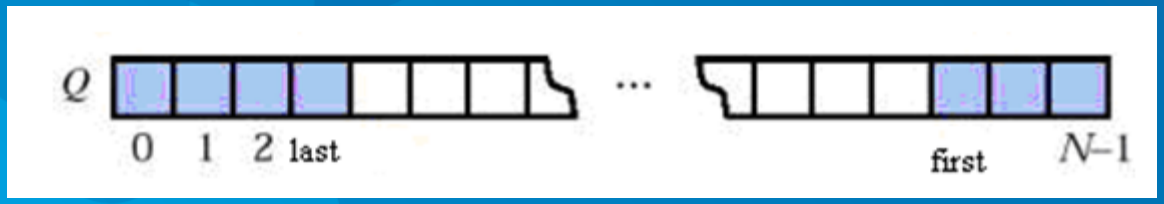
Cài đặt hàng đợi bởi mảng

- Trạng thái hàng đợi lúc bình thường



- Trạng thái hàng đợi lúc xoay vòng



Cấu trúc dữ liệu

```
#define MaxSize 100
struct Queue
        {
                unsigned int  first, last;
                int nodes[MaxSize];
        };
struct  Queue  Q;
```

Khởi tạo hang đợi

- Lúc này firrt và last không trỏ đến vị trí hợp lệ nào trong mảng, cho firt và last đều bằng -1

```
void initialize(struct queue *Q)
{
    Q->first = -1;
    Q->last = -1;
}
```

Kiểm tra hàng đợi rỗng

```
int isEmpty(struct queue Q)
{
      return ((Q.last+1)%MaxSize == Q.first);
}
```

Kiểm tra hang đợi đầy

```
int IsFull(struct Queue Q)
{
    return ((Q.last+2)% MaxSize == Q.first);
}
```

Xác định kích thước hang đợi

```
int sizeQueue(struct queue Q)
{
      if(isEmpty(Q)) return 0;
      else if(Q.first<=Q.last) return (Q.last-Q.first+1);
      else return (Q.last-Q.first+maxSize+1);
}
```

Cài đặt hàng đợi bởi mảng

```
void addQueue(int x, struct queue *Q)
{
    if(!isFull(*Q))
    {
        if(isEmpty(*Q)) Q->first=0;
        Q->last=(Q->last +1)% maxSize;
        Q->nodes[Q->last]=x;
    }
    else printf("\nHàng đợi đầy!");
}
```

Lấy ra một phần tử từ hàng đợi

```
void delQueue(struct queue *Q, int *x)
{
    if(!isEmpty(Q))
    {
        *x=Q->nodes[Q->first];
        Q->first=(Q->first+1)% maxSize;
    }
    else printf("\nHang doi rong!");
}
```

Bài toán quản lý kho

- Quản lý kho hàng có hai cửa:
  - lấy hàng
  - và cửa đưa hàng vào
- Hàng được xếp theo thứ tự, hàng đưa vào trước sẽ được lấy ra trước.
- Cài đặt bằng mảng một chiều

Bài toán quản lý kho

```
#include "stdio.h"
#include "conio.h"
#define maxSize 100

typedef struct hangHoa
{
    int mh;
    char tenMH[30];
};

struct queue
{
    unsigned int first, last;
    hangHoa nodes[maxSize];
};
```

Bài toán quản lý kho

```
//Nguyên mẫu hàm (prototype)
void initialize(struct queue *Q);
bool isEmpty(struct queue Q);
bool isFull(struct queue Q);
int sizeQueue(struct queue Q);
void addQueue(hangHoa x, struct queue *Q);
void delQueue(struct queue *Q, hangHoa *x);
void traverse(struct queue Q);
```

Bài toán quản lý kho

```
//============= Các hàm ======================
void initialize(struct queue *Q)
{
    Q->first= -1;
    Q->last= -1;
}

bool isEmpty(struct queue Q)
{
    return ((Q.last+1)% maxSize == Q.first);
}

bool isFull(struct queue Q)
{
    return ((Q.last+2)% maxSize == Q.first);
}
```

Bài toán quản lý kho

```
int sizeQueue(struct queue Q)
{
    if (isEmpty(Q)) return 0;
    else if(Q.first<=Q.last) return (Q.last-Q.first+1);
    else return (Q.last-Q.first + maxSize+1);
}

void addQueue(hangHoa x, struct queue *Q)
{
    if(!IsFull(*Q))
    {
        if(isEmpty(*Q)) Q->first=0;
        Q->last=(Q->last +1)% maxSize;
        Q->nodes[Q->last]=x;
    }
}
```

Bài toán quản lý kho

```
void delQueue(struct queue *Q, hangHoa *x)
{
    if(!isEmpty(*Q))
    {
        *x=Q->nodes[Q->first];
        Q->first=(Q->first+1)% maxSize;
    }
}

void traverse(struct queue Q)
{
    hangHoa hh;
    while(!isEmpty(Q))
    {
        delQueue(&Q,&hh);
        printf("\n \t%d \t\t\t%s",hh.mh,hh.tenMH);
    }
}
```

Bài toán quản lý kho

```
int main(void)
{
    struct queue Q;
    int ch;
    char c;
    hangHoa hh;

    initialize(&Q);
    do
    {
        printf("\nCHUONG TRINH QUAN LY KHO \n");
        printf("\n1. Nhap mot mat hang");
        printf("\n2. Xuat mot mat hang");
        printf("\n3. Xem mat hang moi nhap");
        printf("\n4. Xem mat hang chuan bi xuat");
        printf("\n5. Xem cac mat hang co trong kho");
        printf("\n6. Xuat toan bo kho");
        printf("\n0. Ket thuc chuong trinh ");
        printf("\nLua chon cac chuc nang cua chuong
        trinh\n");
        scanf("%d",&ch);
```

Bài toán quản lý kho

```
switch(ch)
{
    case 1:
        printf("\nMa mat hang: ");
        scanf("%d",&hh.mh);
        printf("\nTen mat hang: ");
        scanf("%s",&hh.tenMH);
        if (!isFull(Q)) addQueue(hh,&Q);
        else printf("\nHang da day kho!");
        break;

    case 2:
        if(!isEmpty(Q))
        {
        delQueue(&Q,&hh);
        printf("\nMat hang xuat la:");
        printf("\nMa    hang:    %d    \t    Ten:
        %s",hh.mh,hh.tenMH);
        }
        else printf("\nKo con hang trong kho!");
        break;
```

Bài toán quản lý kho

```
case 3:
    printf("\n Mat hang moi nhap kho la:");
    if (!isEmpty(Q)) printf("\nMa hang: %d \t
    Ten:                                    %s",
    Q.nodes[Q.last].mh,Q.nodes[Q.last].tenMH)
    ;
    else printf("\n Kho da het hang!");
    break;

case 4:
    int t;
    if (!isEmpty(Q))
    {
        t = Q.first;
        printf("\nHang  chuan  bi  xuat  kho
        la:");
        printf("\nMa  hang:  %d  \t  Ten:  %s",
        Q.nodes[t].mh,Q.nodes[t].tenMH);
    }
    else printf("\n Kho da het hang!");
    break;
```

Bài toán quản lý kho

```
            case 5:
                printf("\nCac mat hang co trong kho
                la:");
                printf("\n\tMa hang \t\t\t Ten ");
                traverse(Q);
                break;

            case 6:
                printf("\nBan co chac la can xoa Queue
                ko?");
                c = getche();
                if (c =='c' || c == 'C') initialize(&Q);
            break;

            default:
                printf("\n Cam on ban da su dung chuong
            trinh!");
                break;
        }
    }while(ch !=0);
    return 0;
}
```