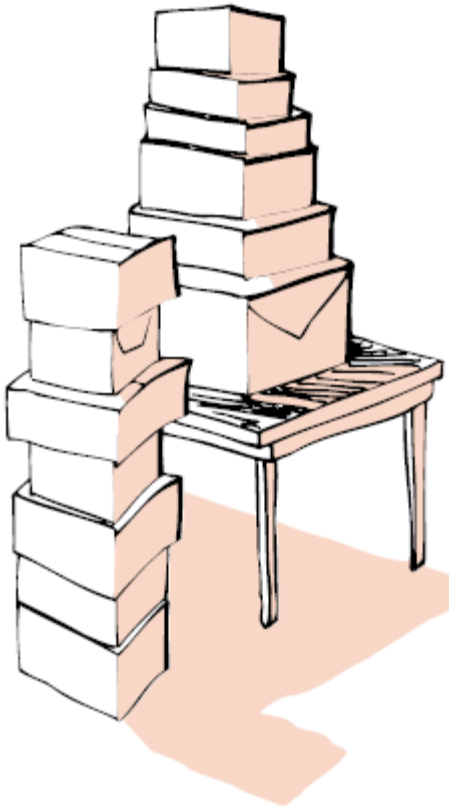
C Programming Basic – week 5

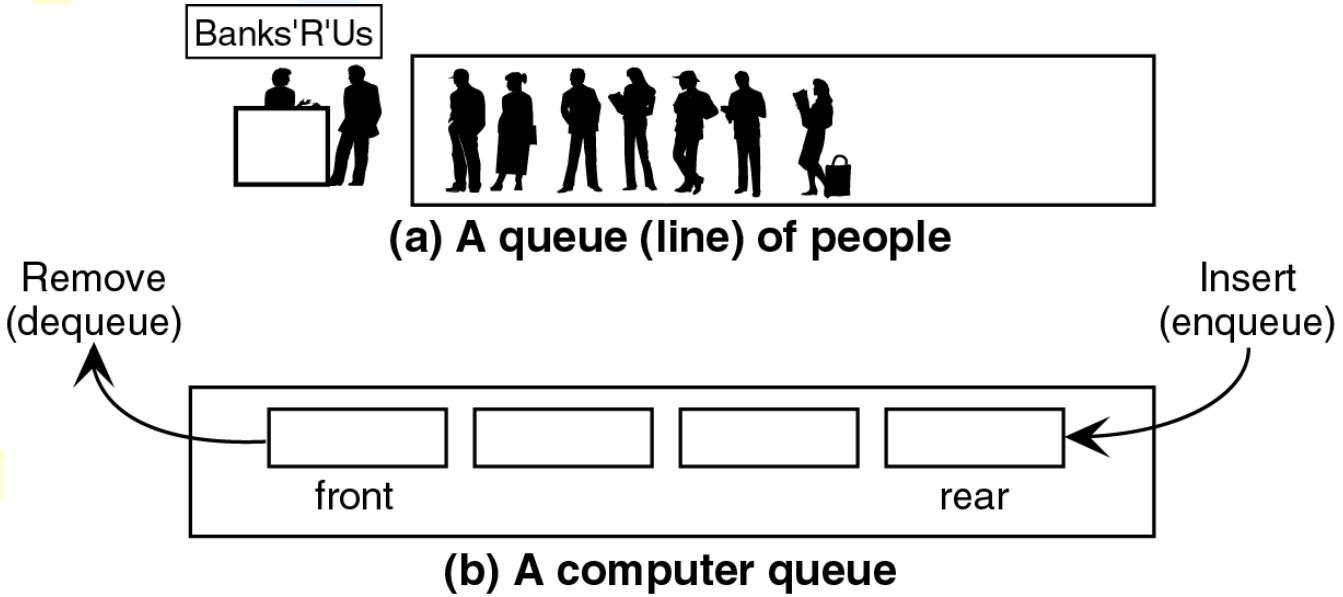
Chủ đề

- Hàng đợi
  - Cài đặt sử dụng mảng
  - Cài đặt sử dụng danh sách liên kết
- Bài tập



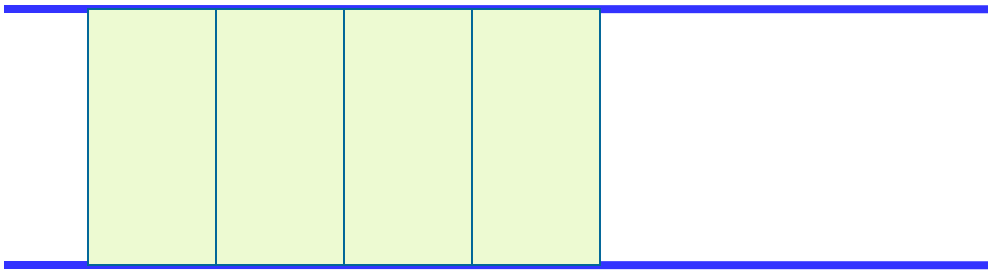
Hàng đợi

- Hai đầu đều được sử dụng: Một đầu để thêm và một đầu để bớt
- Dữ liệu được thêm ở đuôi và được bớt ở đầu



Cấu trúc FIFO

- Các phần tử được bớt theo đúng thứ tự được thêm vào
  - Cấu trúc FIFO: First in, First out



front rear

Thao tác trên hàng đợi

- *Queue* CreateQ(*max_queue_size*) ::=
  tạo ra hàng đợi rỗng có kích thước tối đa là *max_queue_size*

- *Boolean* IsFullQ(*queue, max_queue_size*) ::=
  if(number of elements in *queue* == *max_queue_size*)
  return *TRUE*
  else return *FALSE*

Thao tác trên hàng đợi (2)

- *Queue* EnQ(*queue, item*) ::=
  if (IsFullQ(*queue*)) *queue_full*
  else insert *item* at rear of *queue* and return *queue*
- *Boolean* IsEmptyQ(*queue*) ::=
  if (*queue* ==CreateQ(*max_queue_size*))
  return *TRUE*
  else return *FALSE*
- *Element* DeQ(*queue*) ::=
  if (IsEmptyQ(*queue*)) **return**
  else remove and return the *item* at front of queue.

Cài đặt sử dụng mảng và cấu trúc

```
#define MaxLength 100
typedef ... ElementType;
typedef struct {
ElementType Elements[MaxLength];
//Store the elements
int Front, Rear;
} Queue;
```

Khởi tạo và kiểm tra

```
void MakeNull_Queue(Queue *Q){
  Q->Front=-1;
  Q->Rear=-1;
}
int Empty_Queue(Queue Q){
 return Q.Front==-1;
}
int Full_Queue(Queue Q){
return (Q.Rear-Q.Front+1)==MaxLength;
}
```

Enqueue

```
void EnQueue(ElementType X,Queue *Q){
if (!Full_Queue(*Q)){
  if (Empty_Queue(*Q)) Q->Front=0;
  Q->Rear=Q->Rear+1;
  Q->Element[Q->Rear]=X;
}
else printf("Queue is full!");
}
```

Dequeue

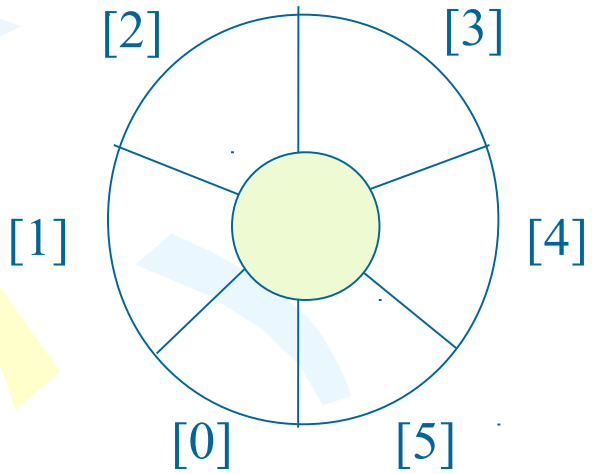
```
void DeQueue(Queue *Q){
  if (!Empty_Queue(*Q)){
     Q->Front=Q->Front+1;
     if (Q->Front > Q->Rear)
  MakeNull_Queue(Q);
     // Queue become empty

}
else printf("Queue is empty!");
}
```

Implementation 2: hàng đợi vòng xoay

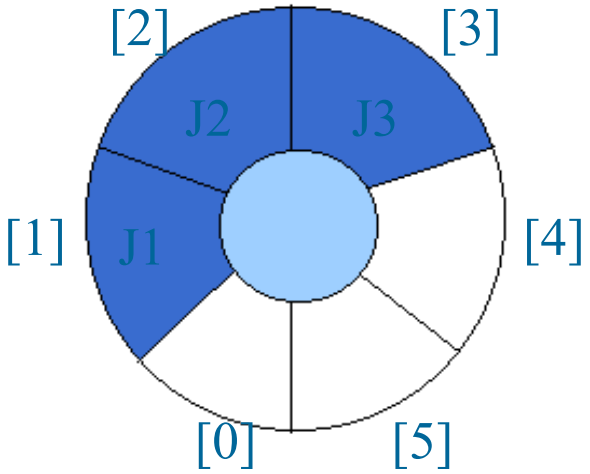
front: vị trí liền trước phần tử đầu tiên ngược chiều kim đồng hồ

rear: đuôi hiện tại

EMPTY QUEUE



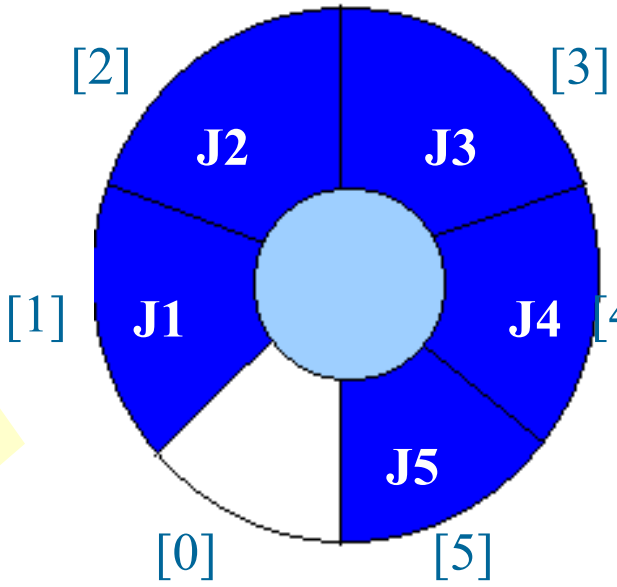
front = 0
rear = 0



front = 0
rear = 3

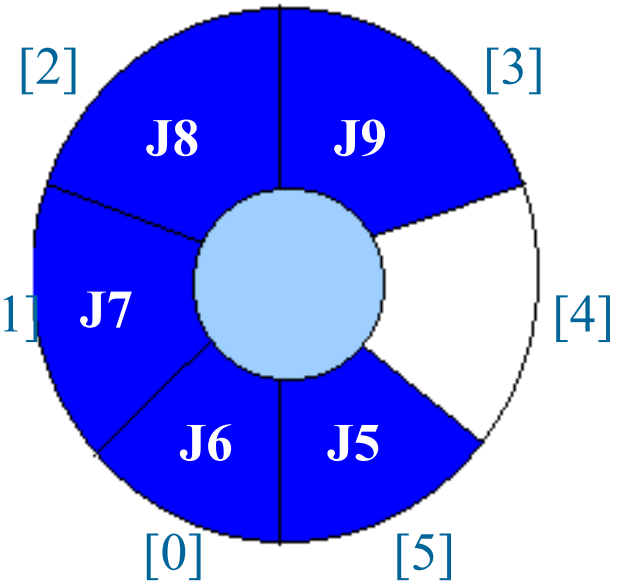
Problem: một ví trí để trống khi hàng đợi đầy

FULL QUEUE



front =0
rear = 5

FULL QUEUE



front =4
rear =3

Kiểm tra hàng đợi đầy

```
int Full_Queue(Queue Q){
 return (Q.Rear-Q.Front+1) %
  MaxLength==0;
}
```

Dequeue

```
void DeQueue(Queue *Q){
if (!Empty_Queue(*Q)){
//if queue contain only one element
  if (Q->Front==Q->Rear) MakeNull_Queue(Q);
  else Q->Front=(Q->Front+1) % MaxLength;
  }
else printf("Queue is empty!");
}
```

Enqueue

```
void EnQueue(ElementType X,Queue *Q){
if (!Full_Queue(*Q)){
  if (Empty_Queue(*Q)) Q->Front=0;
  Q->Rear=(Q->Rear+1) % MaxLength;
  Q->Elements[Q->Rear]=X;
} else printf("Queue is full!");
}
```

Cài đặt sử dụng danh sách

- Sử dụng các thao tác trên danh sách để cài đặt hàng đợi

Khai báo

```
typedef ... ElementType;
typedef struct Node{
  ElementType Element;
  Node* Next; //pointer to next element
 };
 typedef Node* Position;
 typedef struct{
  Position Front, Rear;
  } Queue;
```

Khởi tạo

```
void MakeNullQueue(Queue *Q){
  Position Header;
  Header=(Node*)malloc(sizeof(Node));
  //Allocation Header
  Header->Next=NULL;
  Q->Front=Header;
  Q->Rear=Header;
}
```

Kiểm tra rỗng

```
int EmptyQueue(Queue Q){
return (Q.Front==Q.Rear);
}
```

EnQueue

```
void EnQueue(ElementType X, Queue *Q){
  Q->Rear->Next=
   (Node*)malloc(sizeof(Node));
  Q->Rear=Q->Rear->Next;
  Q->Rear->Element=X;
  Q->Rear->Next=NULL;
}
```

DeqQueue

```
void DeQueue(Queue *Q){
if (!Empty_Queue(Q)){
  Position T;
  T=Q->Front;
  Q->Front=Q->Front->Next;
  free(T);
}
else printf("Error: Queue is empty.");
}
```

Exercise 5.1

- Xây dựng danh bạ điện thoại
- Khai báo cấu trúc "Address" chứa ít nhất "name", "telephone number" và "e-mail address".
- Viết chương trình sao chép danh bạ từ một tệp sang tệp khác sử dụng hàng đợi. Đầu tiên, đọc danh bạ từ một tệp vào hàng đợi. Sau đó lấy các phần tử từ hàng đợi và ghi vào tệp.

Exercise 5.2

- Tạo một hàng đợi chứa các số nguyên. Kích thước hàng đợi là 10 phần tử.
- Đọc các số nguyên từ bàn phím, cách nhau bởi dấu trắng và thêm vào hàng đợi. Khi đọc tới số thứ 11, danh sách đã đầy. Khi đó chương trình xóa số đầu tiên và thêm số 11 vào hàng đợi. In các số nguyên bị xóa ra màn hình
- Xử lý tất cả các số nguyên theo cách này

Exercise 5.3

- Máy báy có 50 hàng chỗ: A B C D E F
- Sử dụng hàng đợi, viết chương trình quản lý đặt vé máy bay với menu có các chức năng thêm,hoãn, sửa thông tin vé từ khách hàng
- Một vé có các trường sau:
  - Flight Number
  - Name of client
  - Booking Time
  - Quantity
  - Seat type: W/C/N (W: C,F C: A, D, N: B,E)
- Kết quả đặt vé: Accept hoặc Refuse. Nếu Accept – hệ thống thực hiện giữ chỗ và thông báo cho người dùng. Trường Booking Time có giá trị là thời gian hệ thống tại thời điểm đặt vé

Exercise 5.4

- Mô phỏng hoạt động của máy tính xử lý các yêu cầu tính toán từ các chương trình
- Đầu vào:
  - Số lượng tiến trình có thể chạy song song
  - Dung lượng bộ nhớ
- Chương trình có menu:
  - Tạo chương trình mới (với ID và dung lượng bộ nhớ cần thiết)
  - Tắt một chương trình
  - Hiển thị trạng thái chạy và chờ đợi của các chương trình

Homework 1

- Ngân hàng phục vụ rút tiền và nạp tiền vào tài khoản
- Viết chương trình phục vụ khách hàng theo số cửa (mỗi cửa là một hàng đợi)
- Chương trình có menu để thêm khách hàng
  - Thời gian vào
  - Thời gian phục vụ mỗi khách hàng là 15 phút
- Output: Khách hàng được đưa vào một cửa với thông tin thời gian chờ
- Chương trình cần tính thời gian chờ của khách hàng
  - Tổng số khách hàng
  - Tổng thời gian chờ
  - Thời gian chờ trung bình của một khách hàng

Hướng dẫn:

- Thông tin khách hàng lưu trong tệp có định dạng:

Time	Number of clients
9:00	2
9:10	1
9:25	3
9:40	2

Giao diện

- BIDV- Hà Thành 17 Tạ Quang Bửu
- Number of queues:2
- 9:00 2
- 1st client goes to Queue 1 : 0
- 2nd client goes to Queue 2 : 0
- 9:10 1
- 1st client goes to Queue 1 : 5 (serve at 9:15)
- 9:25 3
- 1st client goes to Queue 2 : 0
- 2nd client goes to Queue 1: 5
- 3rd client goes to Queue 2: 15
- …

- Average: XYZ clients – total and average waiting time

Cài đặt khác sử dụng mảng

- Queue CreateQ(*max_queue_size*) ::=
  # define MAX_QUEUE_SIZE 100
  typedef struct {
  int key; /* other fields */
  } element;
  element queue[MAX_QUEUE_SIZE];
  int rear = -1;
  int front = -1;
  Boolean IsEmpty(queue) ::= front == rear
  Boolean IsFullQ(queue) ::= rear == MAX_QUEUE_SIZE-1

Enqueue

```
void enq(int *rear, element item)
{
/* add an item to the queue */
    if (*rear == MAX_QUEUE_SIZE_1) {
        queue_full( );
        return;
    }
    queue [++*rear] = item;
}
```

Dequeue

- element deq(int *front, int rear)
  {
  if (*front == rear)
  return queue_empty( );
  /* return an error key */
  return queue [++ *front];
  }

Enqueue

```
void addq(int front, int *rear, element item)
{
    *rear = (*rear +1) % MAX_QUEUE_SIZE;
     if (front == *rear) /* reset rear and print
                                    error */
     return;
   }
     queue[*rear] = item;
}
```

Dequeue

```
element deleteq(int* front, int rear)
{
    element item;

     if (*front == rear)
         return queue_empty( );
        /* queue_empty returns an error key */

    *front = (*front+1) % MAX_QUEUE_SIZE;
     return queue[*front];
}
```