KỸ THUẬT LẬP TRÌNH C

Chương 6: Hàng đợi

1. Giới thiệu
2. Cài đặt hàng đợi theo kiểu kế tiếp
3. Cài đặt hàng đợi theo kiểu liên kết
4. Ứng dụng hàng đợi

1. Giới thiệu

- Hàng đợi là một danh sách mà thao tác thêm nút được thực hiện ở một đầu gọi là cuối hàng đợi (rear) và thao tác xoá nút được thực hiện ở một đầu khác gọi là đầu hàng đợi (front).
- Trên hàng đợi nút thêm vào trước được lấy ra trước nên hàng đợi còn được gọi là danh sách FIFO (First In – First out).

2. Cài đặt hàng đợi theo kiểu kế tiếp

a. Phương pháp di chuyển tịnh tiến

- **Mô tả:**

Trong cách cài đặt này mỗi khi thực hiện thao tác xoá nút thì dời toàn bộ các nút của hàng đợi xuống một vị trí. Vị trí 0 luôn luôn là đầu hàng đợi.

- **Khai báo hàng đợi:**

```
#define MAX 50
struct QUEUE {
        int rear;                   //Cuối hàng đợi
        int nodes[MAX];       /*Mỗi nút của hàng đợi là một
                                phần tử trên mảng một chiều*/
};
```

- **Các thao tác trên hàng đợi:**

✓ **Khởi động hàng đợi**

```
void Inittialize(QUEUE *queue) {
        queue ->rear = – 1; }
```

✓ **Kiểm tra hàng đợi có bị rỗng không?**

```
int Empty(QUEUE *queue) {
      return (queue->rear == -1 ? 1 : 0); }
```

✓ **Kiểm tra hàng đợi có bị đầy không?**

```
int Full(QUEUE *queue) {
      return (queue->rear == MAX - 1 ? 1 : 0); }
```

✓ Thêm nút có nội dung là x vào cuối hàng đợi

```
void Insert(QUEUE *queue, int x) {
        if(Full(queue))
                printf("hang doi bi day\n");
        else {
                queue->rear++;
                queue->nodes[queue->rear] = x;
        }
}
```

✓ Xoá nút ở đầu hàng đợi

```
int Remove (QUEUE *queue) {
        if(Empty(queue)) {
                printf(“ Hang doi bi rong\n”) ;
                <Có thể trả về một gía trị qui ước>
        }
        else {
                x = queue->nodes[0];
/*Dời các nút còn lại của hàng đợi xuống dưới một vị trí*/
                for(int i = 0; i < queue ->rear; i++)
                queue ->nodes[i] = queue ->nodes[i + 1];
                 queue ->rear--;
                return x;
        }
}
```

a. Phương pháp di chuyển vòng

- **Mô tả:**

Trong cách cài đặt này chúng ta xem mảng như là mảng vòng chứ không phải là mảng thẳng: nút 0 xem như là nút sau của nút MAX – 1.

- **Khai báo cấu trúc hàng đợi:**

```
#define MAX 50
struct queue {
        int front; //Chỉ trước nút đầu hàng đợi
        int rear; //Chỉ ngay nút cuối hàng đợi
        int nodes[MAX]; /*Mỗi phần tử mảng là một nút của
                          hàng đợi*/
};
```

- **Các thao tác trên hàng đợi**

✓ Khởi động hàng đợi

```
void Inittialize(QUEUE *queue) {
        queue->front = queue->rear = MAX – 1; }
```

✓ Kiểm tra hàng đợi có bị rỗng không?

```
int Empty(QUEUE *pq) {
        return (pq->front == pq->rear ? 1 : 0); }
```

✓ Kiểm tra hàng đợi có bị đầy không?

```
int Full(QUEUE *queue) {
        if (queue.front == queue.rear + 1 || (queue.front==0
                && queue.rear==MAX–1))
                return 1 ;
        return 0 ;
}
```

✓ Thêm nút vào cuối hàng đợi

```
void Insert(QUEUE *queue, int x) {
        if(queue ->rear == MAX – 1)
                queue ->rear = 0
        else queue ->rear++;
        if(Full(queue))
                printf(“Hang doi bi day\n”);
        else     queue ->nodes[queue ->rear] = x;
}
```

✓ Xoá nút ở đầu hàng đợi

```
int Remove (QUEUE *queue) {
        if(Empty(queue)) {
                printf("Hang doi bi rong\n") ;
                <Có thể trả về một gía trị qui ước>
        }
        else {
                if(queue->front == MAX – 1)
                        queue->front = 0;
                else  queue->front++;
                return queue->nodes[queue->front];
        }
}
```

✓ Duyệt hàng đợi từ nút đầu tới nút cuối

```
void Traverse(QUEUE *queue) {
    int i;
    if(Empty(*queue))
        printf(Hang doi bi rong\n”);
    else {
        if(queue->front == MAX – 1)        i = 0;
        else  i = queue->front + 1;
        //In các nút từ đầu đến kế cuối
        while(i != queue->rear) {
            printf(“%d, ”, queue->nodes[i]);
            if(i == MAX – 1)  i = 0;
            else  i++; }
        printf(“%d\n”; queue->nodes[i]); //In nút cuối
        }}
```

3. Cài đặt hàng đợi theo kiểu liên kết

Cài đặt hàng đợi như một danh sách liên kết: con trỏ đầu danh sách liên kết là pqfront chỉ nút tại đầu hang đợi, nút tại đầu chỉ nút thứ hai, …, con trỏ cuối danh sách liên kết là pqrear chỉ nút cuối hàng đợi.

a. Khai báo cấu trúc một nút của hàng đợi

```
struct node {
    int info; //chứa nội dung của nút
    node *next; //Con trỏ chỉ nút kết tiếp trong hàng đợi
}
typedef node *NODEPTR;
```

- **Các thao tác trên hàng đợi**
  - ✓ **Cấp phát biến động làm nút cho hàng đợi**

```
NODEPTR GetNode() {
    NODEPTR p;
    p = (NODEPTR) malloc(sizeof(node));
    return p;
}
```

  - ✓ **Giải phóng biến động đã cấp phát trước đó**

```
void FreeNode(NODEPTR p) {
    free(p);
}
```

- **Khởi động hàng đợi**

```
void Initialize(NODEPTR *pqfront, NODEPTR
      *pqrear) {
      * pqfront = *pqrear = NULL;
}
```

- **Kiểm tra hàng đợi có rỗng hay không**

```
int Empty(NODEPTR pqfront, NODEPTR pqrear) {
      if (pqfront == NULL && pqrear == NULL)
            return 1;
      return 0;
}
```

✓ Thêm nút vào cuối hàng đợi

```
void Insert(NODEPTR *pqfront, NODEPTR *pqrear, int x) {
        NODEPTR p;
        p = GetNode();
        p->info = x;
        p->next = NULL;
        if(Empty(*pqfront, *pqrear))
                *pqfront = *pqrear = p;
        else  {
                (*pqrear)->next = p;
                *pqrear = p;
        }
}
```

✓Xóa nút ở đầu hàng đợi

```
int Remove(NODEPTR *pqfront, NODEPTR *pqrear) {
        int x;
        NODEPTR p;
        x = p->info;
        if(p->next == NULL)
                *pqfront = *pqrear = NULL;
        else *pqfront = p->next;
        FreeNode(p);
        return x;
}
```

✓ Duyệt hàng đợi

```
void Traverse(NODEPTR pqfront, NODEPTR pqrear) {
        NODEPTR p;
        p = pqfront;
        while(p != pqfront) {
                printf("%d\t", p->info);
                p = p->next;
        }
        printf("%d\t", p->info);
}
```

4. Ứng dụng hàng đợi

Hàng đợi thường được dùng để giải quyết các vấn đề có cơ chế FIFO như:

- tổ chức quản lý và điều phối tiến trình trong các hệ điều hành.
- tổ chức bộ đệm bàn phím (nhấn phím -> bộ đệm -> CPU xử lý).
- Xử lý các dịch vụ ngân hàng.
- …

Hết