Practical-5

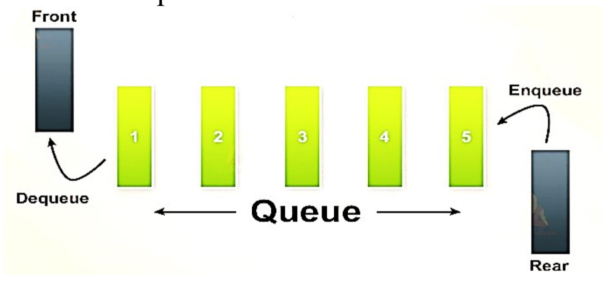
Implementing different types of Queue and its Operations

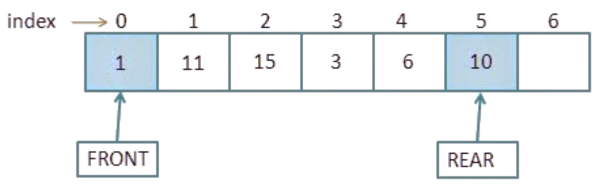
Like Stack, Queue is a linear structure which follows a particular order in which the operations are performed. The order is **F**irst **I**n **F**irst **O**ut (FIFO). A good example of queue is any queue of consumers for a resource where the consumer that came first is served first. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

Operations on Queue:

Mainly the following four basic operations are performed on queue:

- **Enqueue:** Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.
- **Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.
- **Front:** Get the front item from queue.
- **Rear:** Get the last item from queue.





Queue using Array

```
#define size 3
int que[size],front = -1, rear= -1;
void enqueue(int Q[],int y)
{  if(rear==size-1){
    printf("\nQueue overflow.");
    }
  else
   { rear=rear+1;
     Q[rear] = y;
    }
   if(front == -1)
     front=0;
 }
int dequeue(int Q[])
  {
    int val;
   if(front == -1)
     {   printf("Queue is underflow");
        return -1;
     }
   else
     {val = Q[front];
```

```
        Q[front] = 0;

        if (front == rear) //reset the queue
        {
         front= -1;
         rear = -1;
        }
        else
         front=front+1;
       }
      return val;
   }
```

Circular Queue using Array

```
#include<stdio.h>
# define MAX 5
int cqueue_arr[MAX];
int front = -1;
int rear = -1;
void insert(int item)
{
    if((front == 0 && rear == MAX-1) || (front == rear+1)) //check overflow
    {
        printf("Queue Overflow \n");
        return;
    }
    if(front == -1)
    {
        front = 0;
        rear = 0;
    }
    else
    {
        if(rear == MAX-1) // increment rear pointer
            rear = 0;
        else
            rear = rear+1;
    }
    cqueue_arr[rear] = item ;
}
void deletion()
{
    if(front == -1)  //check underflow
    {
        printf("Queue Underflow\n");
        return ;
    }
    printf("Element deleted from queue is : %d \n",cqueue_arr[front]);
    if(front == rear) //reset pointers if last element
    {
        front = -1;
        rear=-1;
    }
    else //increment front pointer
    {
        if(front == MAX-1)  //if front is at last element
             front = 0;
        else
            front = front+1;
    }
}
```

```
int main()
{
    int choice,item;
    do
    {   printf("1.Insert\n");
        printf("2.Delete\n");
        printf("3.Quit\n");
        printf("Enter your choice : ");
        scanf("%d",&choice);
        switch(choice)
        {   case 1 :
                printf("Input the element for insertion in queue : ");
                scanf("%d", &item);
                insert(item);
                break;
            case 2 :
                deletion();
                break;
            case 3:
                break;
            default:
                printf("Wrong choice\n");
        }
    }while(choice!=3);
    return 0;
}
```

Double ended Queue using Array

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 7
int deque_arr[MAX];
int front=-1;
int rear=-1;

void insert_frontEnd(int item);
void insert_rearEnd(int item);
int delete_frontEnd();
int delete_rearEnd();
void display();
int isEmpty();
int isFull();

int main()
{      int choice,item;
       do
       {      printf("\n\n1.Insert at the front end\n");
              printf("2.Insert at the rear end\n");
              printf("3.Delete from front end\n");
              printf("4.Delete from rear end\n");
              printf("5.Display\n");
              printf("6.Quit\n");
              printf("\nEnter your choice : ");
              scanf("%d",&choice);

              switch(choice)
              {
              case 1:
                     printf("\nInput the element for adding in queue : ");
                     scanf("%d",&item);
                     insert_frontEnd(item);
                     break;
              case 2:
                     printf("\nInput the element for adding in queue : ");
                     scanf("%d",&item);
                     insert_rearEnd(item);
                     break;
                case 3:
                     printf("\nElement deleted from front end is : %d\n",
delete_frontEnd());
                     break;
                case 4:
                     printf("\nElement deleted from rear end is : %d\n",
delete_rearEnd());
                     break;
                case 5:
                     display();
                     break;
                case 6:
                     exit(1);
                default:
                     printf("\nWrong choice\n");
              }/*End of switch*/
              printf("\nfront = %d, rear =%d\n", front , rear);
              display();
       }while(choice!=6);

       /*End of while*/
}/*End of main()*/

void insert_frontEnd(int item)
{
       if( isFull() )
       {
              printf("\nQueue Overflow\n");
              return;
       }
       if( front==-1 )/*If queue is initially empty*/
       {
              front=0;
              rear=0;
       }
       else if(front==0)
              front=MAX-1;
       else
              front=front-1;
       deque_arr[front]=item ;
}/*End of insert_frontEnd()*/

void insert_rearEnd(int item)
{
       if( isFull() )
       {
              printf("\nQueue Overflow\n");
              return;
       }
       if(front==-1)  /*if queue is initially empty*/
       {
              front=0;
              rear=0;
       }
       else if(rear==MAX-1)  /*rear is at last position of queue */
              rear=0;
       else
              rear=rear+1;
       deque_arr[rear]=item ;
}/*End of insert_rearEnd()*/
```

```
int delete_frontEnd()
{
	int item;
	if( isEmpty() )
	{
		printf("\nQueue Underflow\n");
		exit(1);
	}
	item=deque_arr[front];
	if(front==rear) /*Queue has only one element */
	{
		front=-1;
		rear=-1;
	}
	else
		if(front==MAX-1)
			front=0;
		else
			front=front+1;
	return item;
}/*End of delete_frontEnd()*/

int delete_rearEnd()
{
	int item;
	if( isEmpty() )
	{
		printf("\nQueue Underflow\n");
		exit(1);
	}
	item=deque_arr[rear];

	if(front==rear) /*queue has only one element*/
	{
		front=-1;
		rear=-1;
	}
	else if(rear==0)
		rear=MAX-1;
	else
		rear=rear-1;
	return item;
}/*End of delete_rearEnd() */

int isFull()
{
	if ( (front==0 && rear==MAX-1) || (front==rear+1) )
		return 1;
	else
		return 0;
}/*End of isFull()*/

int isEmpty()
{
	if( front == -1)
		return 1;
	else
		return 0;
}/*End of isEmpty()*/

void display()
{	int i;
	if( isEmpty() )
```

```
        {
               printf("\nQueue is empty\n");
               return;
        }
        printf("\nQueue elements :\n");
        i=front;
        if( front<=rear )
        {
            for(i=front;i<=rear;i++)
             { if(i==front)
               printf("\n%d -->front",deque_arr[i]);
                else if(i==rear)
               printf("\n%d --->rear",deque_arr[i]);
               else
                printf("\n%d ",deque_arr[i]);
              }
        }
        else
        {      while(i<=MAX-1)
                      printf("%d ",deque_arr[i++]);
               i=0;
               while(i<=rear)
                      printf("%d ",deque_arr[i++]);
        }
        printf("\n");
}/*End of display() */
```

Exercises

1. Write a C program to implement all operations of normal queue using array.
1. Write a C program to implement all operations of circular queue using array.
2. Write a C program to create a queue where insertion and deletion of elements can be done from both the ends
3. Write a C program to reverse a normal queue.