

WEEK-10

Date:17-09-2025

List of programs:

1. Write a C program to implement Queue operations using arrays.
2. Write a C program to implement Queue operations using linked list.

1. Aim: To write a C program to implement Queue operations using arrays.

Program:

```
#include<stdio.h>

#include<stdlib.h>

#define size 4

int Queue[size];

int front=-1,rear=-1;

void enqueue(int x)

{

    if(rear==size-1)

    {

        printf("queue is full");

    }

    else

    {

        if(front==-1&&rear==-1)

        {

            front=rear=0;

        }

        else

        {
```

```
        rear++;  
    }  
    Queue[rear]=x;  
}  
}  
void dequeue()  
{  
    if(front== -1)  
    {  
        printf("queue is empty");  
    }  
    else  
    {  
        printf("deleted element is %d",Queue[front]);  
        if(front==rear)  
        {  
            front=rear=-1;  
        }  
        else  
        {  
            front++;  
        }  
    }  
}  
void display()  
{  
    if(front== -1)  
    {
```

```
        printf("queue is empty");
    }
    else
    {
        for(int i=front;i<=rear;i++)
        {
            printf("%d\n",Queue[i]);
        }
    }
}

void main()
{
    int ch,num;
    printf("\n:: queue using arrays ::\n");
    while(1)
    {
        printf("\nMAIN MENU:\n1.enqueue\n2.dequeue\n3.DISPLAY\n4.EXIT\n\nENTER YOUR CHOICE:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("ENTER THE QUEUE ELEMENT : ");
                    scanf("%d",&num);
                    enqueue(num);
                    break;
            case 2: dequeue();
                    break;
            case 3: display();
```

```
        break;

    case 4:exit(0);

        break;

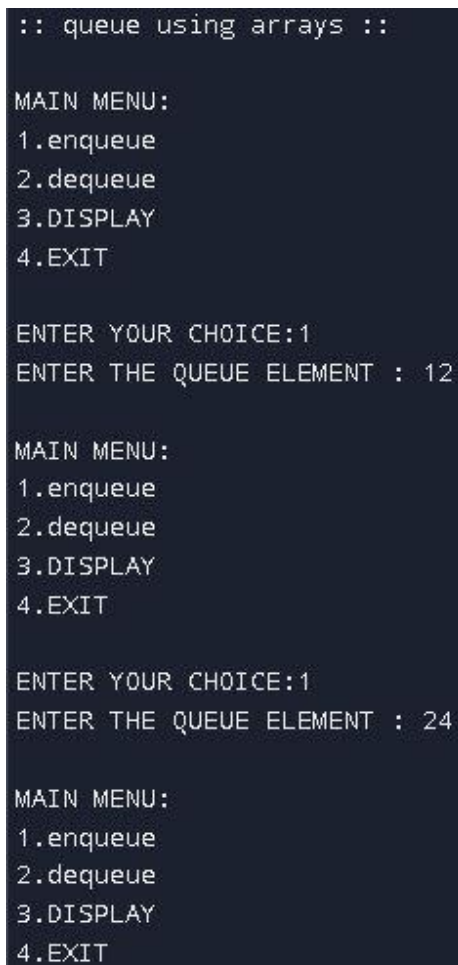
    default:printf("Invalid Choice : ");

}

}

}
```

Output:



```
:: queue using arrays ::

MAIN MENU:
1.enqueue
2.dequeue
3.DISPLAY
4.EXIT

ENTER YOUR CHOICE:1
ENTER THE QUEUE ELEMENT : 12

MAIN MENU:
1.enqueue
2.dequeue
3.DISPLAY
4.EXIT

ENTER YOUR CHOICE:1
ENTER THE QUEUE ELEMENT : 24

MAIN MENU:
1.enqueue
2.dequeue
3.DISPLAY
4.EXIT
```

```
ENTER YOUR CHOICE:1
ENTER THE QUEUE ELEMENT : 36

MAIN MENU:
1.enqueue
2.dequeue
3.DISPLAY
4.EXIT

ENTER YOUR CHOICE:3
12
24
36

MAIN MENU:
1.enqueue
2.dequeue
3.DISPLAY
4.EXIT

ENTER YOUR CHOICE:2
deleted element is 12
MAIN MENU:
1.enqueue
2.dequeue
3.DISPLAY
4.EXIT

ENTER YOUR CHOICE:3
24
36

MAIN MENU:
1.enqueue
2.dequeue
3.DISPLAY
4.EXIT

ENTER YOUR CHOICE:4

=== Code Execution Successful ===
```

2. **Aim:** To Write a C program to implement Queue operations using linked list.

Program:

```
#include<stdio.h>

#include<stdlib.h>

struct node
{
    int data;
    struct node *next;
};

struct node *front=NULL,*rear=NULL,*new,*temp;

void enqueue(int x)
{
    new=(struct node*)malloc(sizeof(struct node));
    new->data=x;
    new->next=NULL;
    if(front==NULL&&rear==NULL)
    {
        front=rear=new;
    }
    else
    {
        rear->next=new;
        rear=new;
    }
}

void dequeue()
{

```

```
    if(front==NULL&&rear==NULL)
    {
        printf("queue is empty");
    }
    else
    {
        printf("deleted element is %d",front->data);
        temp=front;
        if(front==rear)
        {
            front=rear=NULL;
        }
        else
        {
            front=front->next;
        }
        free(temp);
    }
}

void display()
{
    temp=front;
    if(front==NULL)
    {
        printf("queue is empty");
    }
    else
    {
```

```
        while(front!=NULL)
        {
            printf("%d->",front->data);
            front=front->next;
        }
        printf("NULL");
    }
}

void main()
{
    int ch,num;
    printf("\n:: queue using linked list ::\n");
    while(1)
    {
        printf("\nMAIN MENU:\n1.enqueue\n2.dequeue\n3.DISPLAY\n4.EXIT\n\nENTER YOUR CHOICE:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("ENTER THE QUEUE ELEMENT : ");
                    scanf("%d",&num);
                    enqueue(num);
                    break;
            case 2: dequeue();
                    break;
            case 3: display();
                    break;
            case 4: exit(0);
        }
    }
}
```



```
        break;

default:printf("Invalid Choice : ");
}
}
}
```

Output:

```
:: queue using linked list ::

MAIN MENU:
1.enqueue
2.dequeue
3.DISPLAY
4.EXIT

ENTER YOUR CHOICE:1|
ENTER THE QUEUE ELEMENT : 10

MAIN MENU:
1.enqueue
2.dequeue
3.DISPLAY
4.EXIT

ENTER YOUR CHOICE:1
ENTER THE QUEUE ELEMENT : 20

MAIN MENU:
1.enqueue
2.dequeue
3.DISPLAY
4.EXIT
```

```
ENTER YOUR CHOICE:1  
ENTER THE QUEUE ELEMENT : 30
```

```
MAIN MENU:
```

```
1.enqueue  
2.dequeue  
3.DISPLAY  
4.EXIT
```

```
ENTER YOUR CHOICE:2  
deleted element is 10
```

```
MAIN MENU:
```

```
1.enqueue  
2.dequeue  
3.DISPLAY  
4.EXIT
```

```
ENTER YOUR CHOICE:3  
20->30->NULL
```

```
MAIN MENU:
```

```
1.enqueue  
2.dequeue  
3.DISPLAY  
4.EXIT
```

```
ENTER YOUR CHOICE:4
```

```
=== Code Execution Successful ===
```

Inferences:

- Working principle for queues using arrays is Queue follows **FIFO (First In, First Out)** order. Elements are inserted at the **rear** and removed from the **front**.
- It is Easy to implement and it has constant time insertion and deletion.
- The main advantage of queues using linked list is:
 1. **Dynamic size** → no overflow (unless memory is full).
 2. No need for shifting elements (as in arrays).
 3. Efficient memory utilization.