

Queue

Queue Implementation

Queue can be implemented using Array and Linked List.

Queue implementation using Array Insertion Algorithm

```
void insert (int queue[], int max, int front, int rear, int item)
{
    if (rear + 1 == max)
    {
        printf("overflow");
    }
    else
    {
        if(front == -1 && rear == -1)
        {
            front = 0;
            rear = 0;
        }
        else
        {
            rear = rear + 1;
        }
        queue[rear]=item;
    }
}
```

Queue implementation using Array Deletion Algorithm

```
int delete (int queue[], int max, int front, int rear)

{
    int y;

    if (front == -1 || front > rear)

    {

        printf("underflow");

    }

    else

    {

        y = queue[front];

        if(front == rear)

        {

            front = rear = -1;

        }

        else

            front = front + 1;

    }

    return y;

}
```

1. Queue implementation using Array.

```
#include<stdio.h>

#include<stdlib.h>

#define maxsize 5

void insert();

void delete();

void display();

int front = -1, rear = -1;

int queue[maxsize];

void main ()

{

    int choice;

    while(choice != 4)

    {

        printf("\n*****Main Menu*****\n");

        printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");

        printf("\nEnter your choice ?");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:

                insert();

                break;

            case 2:

                delete();

        }

    }

}
```

```
        break;

    case 3:
        display();
        break;

    case 4:
        exit(0);
        break;

    default:
        printf("\nEnter valid choice??\n");
    }
}

void insert()
{
    int item;
    printf("\nEnter the element\n");
    scanf("\n%d",&item);
    if(rear == maxsize-1)
    {
        printf("\nOVERFLOW\n");
        return;
    }
    if(front == -1 && rear == -1)
    {
        front = 0;
```

```
    rear = 0;

}

else

{

    rear = rear+1;

}

queue[rear] = item;

printf("\nValue inserted ");




}

void delete()

{

    int item;

    if (front == -1 || front > rear)

    {

        printf("\nUNDERFLOW\n");

        return;

    }

    else

    {

        item = queue[front];

        if(front == rear)

        {

            front = -1;
```

```
    rear = -1 ;

}

else

{

    front = front + 1;

}

printf("\nvalue deleted ");

}

}

void display()

{

    int i;

    if(rear == -1)

    {

        printf("\nEmpty queue\n");

    }

    else

    { printf("\nprinting values ..... \n");

        for(i=front;i<=rear;i++)

        {

            printf("\n%d\n",queue[i]);

        }

    }

}
```

```
    }  
}  
}
```

Queue Implementation using Linked List using Insertion Algorithm

```
void insert(struct node *ptr, int item; )  
{   ptr = (struct node *) malloc (sizeof(struct node));  
    if(ptr == NULL)  
    {  
        printf("\nOVERFLOW\n");  
        return;  
    }  
    else  
    {  
        ptr -> data = item;  
        if(front == NULL)  
        {  
            front = ptr;  
            rear = ptr;  
            front -> next = NULL;  
            rear -> next = NULL;  
        }  
        else  
        {  
            rear -> next = ptr;  
            rear = ptr;
```

```
    rear->next = NULL;  
}  
}  
}
```

Queue Implementation using Linked List using Deletion Algorithm

```
void delete (struct node *ptr)
```

```
{
```

```
    if(front == NULL)
```

```
{
```

```
    printf("\nUNDERFLOW\n");
```

```
    return;
```

```
}
```

```
else
```

```
{
```

```
    ptr = front;
```

```
    front = front -> next;
```

```
    free(ptr);
```

```
}
```

```
}
```

2. Queue implementation using linked list

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
struct node
```

```

{
    int data;
    struct node *next;
};

struct node *front;
struct node *rear;

void insert();
void delete();
void display();

void main ()
{
    int choice;
    while(choice != 4)
    {
        printf("\n*****Main
Menu*****\n");

        printf("\n=====|n");
        printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");
        printf("\nEnter your choice ?");
        scanf("%d",& choice);
        switch(choice)
        {
            case 1:
                insert();

```

```
break;

case 2:
    delete();
    break;

case 3:
    display();
    break;

case 4:
    exit(0);
    break;

default:
    printf("\nEnter valid choice??\n");
}

}

}

void insert()
{
    struct node *ptr;
    int item;

    ptr = (struct node *) malloc (sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW\n");
        return;
    }
}
```

```
}

else

{

printf("\nEnter value?\n");

scanf("%d",&item);

ptr -> data = item;

if(front == NULL)

{

    front = ptr;

    rear = ptr;

    front -> next = NULL;

    rear -> next = NULL;

}

else

{

    rear -> next = ptr;

    rear = ptr;

    rear->next = NULL;

}

}

void delete ()

{

struct node *ptr;

if(front == NULL)
```

```
{  
    printf("\nUNDERFLOW\n");  
    return;  
}  
  
else  
{  
    ptr = front;  
    front = front -> next;  
    free(ptr);  
}  
}  
  
void display()  
{  
    struct node *ptr;  
    ptr = front;  
    if(front == NULL)  
    {  
        printf("\nEmpty queue\n");  
    }  
    else  
    {  
        printf("\nprinting values ..... \n");  
        while(ptr != NULL)  
        {  
            printf("\n%d\n",ptr -> data);  
            ptr = ptr -> next;  
        }  
    }  
}
```

}

}

}