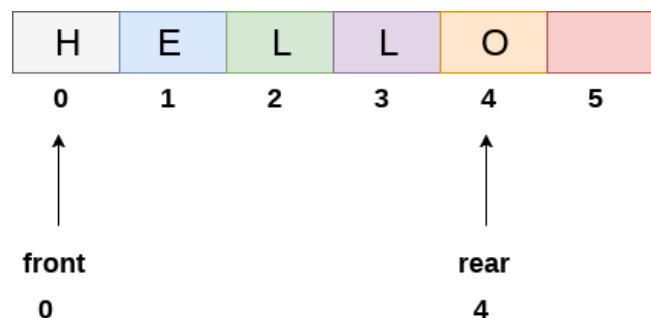


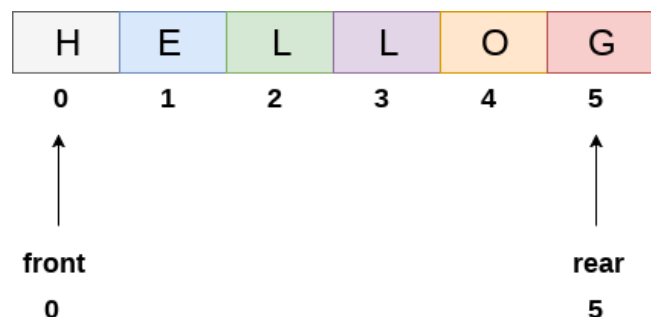
## Array representation of Queue

We can easily represent queue by using linear arrays. There are two variables i.e. front and rear, that are implemented in the case of every queue. Front and rear variables point to the position from where insertions and deletions are performed in a queue. Initially, the value of front and queue is -1 which represents an empty queue. Array representation of a queue containing 5 elements along with the respective values of front and rear, is shown in the following figure.



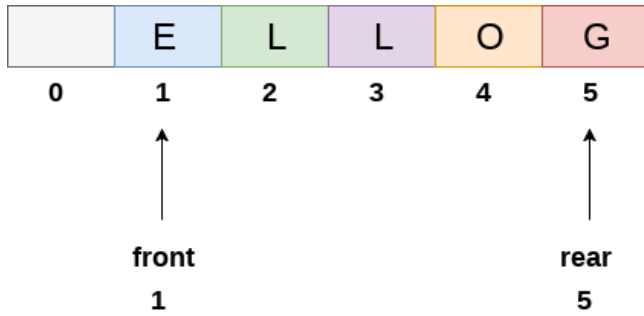
Queue

The above figure shows the queue of characters forming the English word "HELLO". Since, No deletion is performed in the queue till now, therefore the value of front remains -1. However, the value of rear increases by one every time an insertion is performed in the queue. After inserting an element into the queue shown in the above figure, the queue will look something like following. The value of rear will become 5 while the value of front remains same.



Queue after inserting an element

After deleting an element, the value of front will increase from -1 to 0. however, the queue will look something like following.



## Queue after deleting an element

## Algorithm to insert any element in a queue

Check if the queue is already full by comparing rear to max - 1. if so, then return an overflow error.

If the item is to be inserted as the first element in the list, in that case set the value of front and rear to 0 and insert the element at the rear end.

Otherwise keep increasing the value of rear and insert each element one by one having rear as the index.

## Algorithm

- **Step 1:** IF REAR = MAX - 1  
Write OVERFLOW  
Go to step  
[END OF IF]
- **Step 2:** IF FRONT = -1 and REAR = -1  
SET FRONT = REAR = 0  
ELSE  
SET REAR = REAR + 1  
[END OF IF]
- **Step 3:** Set QUEUE[REAR] = NUM
- **Step 4:** EXIT



## C Function

```
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
        {

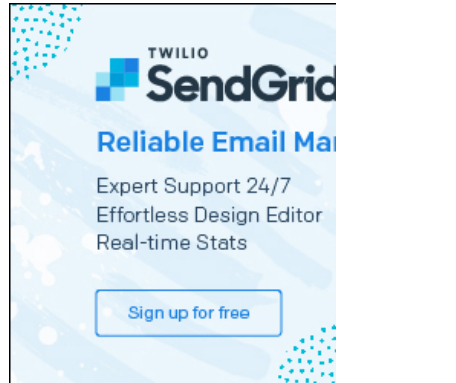
```

```

        rear = rear + 1;
    }
    queue[rear]=item;
}
}

```

## Algorithm to delete an element from the queue



If, the value of front is -1 or value of front is greater than rear , write an underflow message and exit.

Otherwise, keep increasing the value of front and return the item stored at the front end of the queue at each time.

## Algorithm

- **Step 1:** IF FRONT = -1 or FRONT > REAR  
Write UNDERFLOW  
ELSE  
SET VAL = QUEUE[FRONT]  
SET FRONT = FRONT + 1  
[END OF IF]
- **Step 2:** EXIT

## C Function

```

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;  
}  
}
```

## Menu driven program to implement queue 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("\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()  
{  
    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]);
    }
}
}
```

**Output:**

```
*****Main Menu*****
```

```
=====
```

```
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
```

```
Enter your choice ?1
```

```
Enter the element
```

```
123
```

```
Value inserted
```

```
*****Main Menu*****
```

```
=====
```

```
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
```

```
Enter your choice ?1
```

```
Enter the element
```

```
90
```

```
Value inserted
```

```
*****Main Menu*****
```

```
=====
```

```
1.insert an element
2.Delete an element
3.Display the queue
4.Exit
```

```
Enter your choice ?2
```



```

value deleted

*****Main Menu*****
=====

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?3

printing values .....

90

*****Main Menu*****
=====

1.insert an element
2.Delete an element
3.Display the queue
4.Exit

Enter your choice ?4

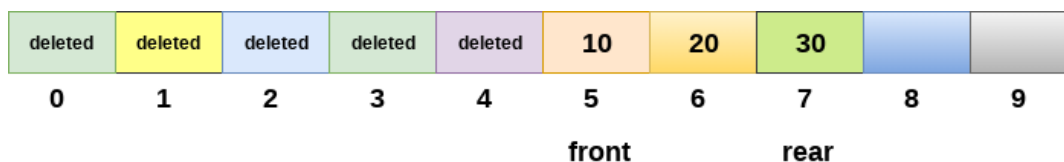
```



## Drawback of array implementation

Although, the technique of creating a queue is easy, but there are some drawbacks of using this technique to implement a queue.

- **Memory wastage** : The space of the array, which is used to store queue elements, can never be reused to store the elements of that queue because the elements can only be inserted at front end and the value of front might be so high so that, all the space before that, can never be filled.



## Limitation of array representation of queue

The above figure shows how the memory space is wasted in the array representation of queue. In the above figure, a queue of size 10 having 3 elements, is shown. The value of the front variable is 5, therefore, we can not reinsert the values in the place of already deleted element before the position of front. That much space of the array is wasted and can not be used in the future (for this queue).

- **Deciding the array size**

One of the most common problem with array implementation is the size of the array which requires to be declared in advance. Due to the fact that, the queue can be extended at runtime depending upon the problem, the extension in the array size is a time taking process and almost impossible to be performed at runtime since a lot of reallocations take place. Due to this reason,

we can declare the array large enough so that we can store queue elements as enough as possible but the main problem with this declaration is that, most of the array slots (nearly half) can never be reused. It will again lead to memory wastage.

[< prev](#)[next >](#)

## Invest in Metals & Oil

Ad Looking for a shelter in turmoil? Switch metals and oil. Try at FBS







FBS

Sign Up





## Please Share







## Learn Latest Tutorials

 Verbal A.	 AWS	 D. Math.	 React Native
 TypeScript	 COA		

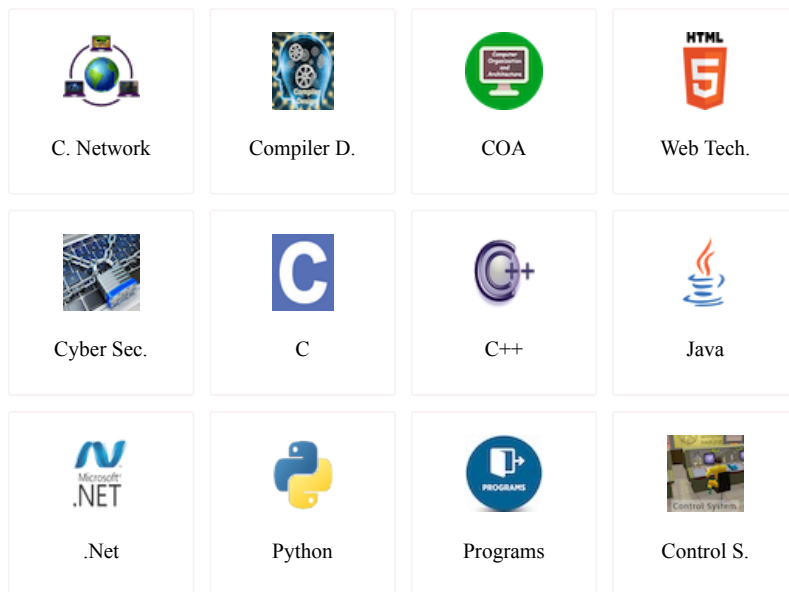
## Preparation

 Aptitude	 Reasoning	 Verbal A.	 Interview
---	--	--	--

## B.Tech / MCA

 DBMS	 DS	 DAA	 OS
---	---	--	---





## Invest in Metals & Oil

Looking for a shelter in turmoil? Switch to good old metals and oil. Try at FBS FBS

