DSA SERIES

- Learn Coding

Topic to be Covered today

Queue

LETS START TODAY’S LECTURE

Queue

- Linear Data Structure
- First in First Out

Operations :

- **Enqueue ()**
- **Dequeue()**
- **Front()**
- **Rear()**
- **isEmpty()**
- **isFull()**

Linear Queue Implementation using the Array

```
#include <iostream>
using namespace std;

class Queue
{
    int *arr;
    int front;
    int rear;
    int size;
    int capacity;

public:
    // Constructor
    Queue(int cap)
    {
        capacity = cap;
        arr = new int[capacity];
        front = 0;
        rear = 0;
        size = 0;
    }
```

```
//  Function to maintain / manage  the queue

void eneque(int val)
{
    if (size == capacity)
    {
        cout << "Queue is full \n";
        return;
    }

    arr[rear] = val;
    rear++;
    size++;
}

void dequeue()
{

    if (size == 0)
    {
        cout << "Queue is empty \n";
        return;
    }
    front++;
    size--;
}

int getFront()
{
    if (size == 0)
    {
        cout << "Queue is empty\n";
        return -1;
    }
    return arr[front];
}

int getBack()
{
    if (size == 0)
    {
        cout << "Queue is empty\n";
        return -1;
    }
    return arr[rear - 1];
}
```

```cpp
    bool isEmpty()
    {
        return size == 0;
    }

    bool isFull()
    {
        return size == capacity;
    }

    ~Queue()
    {
        delete[] arr;
    }
};

int main()
{
    Queue q(5);
```

```
  q.eneque(10);
    q.eneque(20);
    q.eneque(30);
    q.eneque(40);

    cout << "Front :" << q.getFront() << endl;
    // cout<<"back :"<<q.getBack()<<endl;
    q.dequeue();

    cout << "Front :" << q.getFront() << endl;
    // cout<<"back :"<<q.getBack()<<endl;
    q.eneque(70);
    q.eneque(80);
    q.eneque(90);

    cout << "back :" << q.getBack() << endl;

    return 0;
}
```

Circular Queue Implementation using the Array

```
#include <iostream>
using namespace std;

class CircularQueue
{

    int *arr;
    int front;
    int rear;
    int size;
    int capacity;
    public:

    CircularQueue(int cap)
    {
        capacity = cap;
        arr = new int[capacity];

        front = 0;
        rear = 0;
        size = 0;
    }
```

```
void enqueue(int val)
{
    if (size == capacity)
    {
        cout << "Queue if Full \n";
        return;
    }

    arr[rear] = val;
    rear = (rear + 1) % capacity;
    size++;
}

void dequeue()
{
    if (size == 0)
    {
        cout << "Queue is empty";
        return;
    }
    front = (front+1)%capacity;
    size--;
}
```

```
int getFront(){
    if(size==0){
          cout << "Queue is empty";
        return -1;
    }
    return arr[front];
}

bool isEmpty(){
    return size==0;
}

bool isFull(){
    return size==capacity;
}

void display(){
    if(isEmpty()){
        cout<<"Queue is empty\n";
        return;
    }
    cout<<"Queue elements are : ";
        for(int i = 0;i<size;i++){
            int index = (front+i)%capacity;
            cout<<arr[index]<<" ";
        }
        cout<<endl;
    }


};


int main(){

    CircularQueue q(5);

    q.enqueue(10);
    q.enqueue(20);
    q.enqueue(30);
    q.enqueue(40);

    q.display();
        cout<<"Front element : "<<q.getFront()<<endl;
```

```
q.dequeue();
    q.display();

    cout<<"Front element : 
"<<q.getFront()<<endl;


}
```

Queue Implementation using the STL

```
#include<iostream>
#include<queue>

using namespace std;

int main(){
    queue<int> q;
    q.push(10);
    q.push(20);
    q.push(30);
    q.push(40);

    cout<<"Front : "<<q.front()<<endl;
    cout<<"Back : "<<q.back()<<endl;
    cout<<"Size : "<<q.size()<<endl;

    return 0;
}
```

Learn coding

THANK YOU