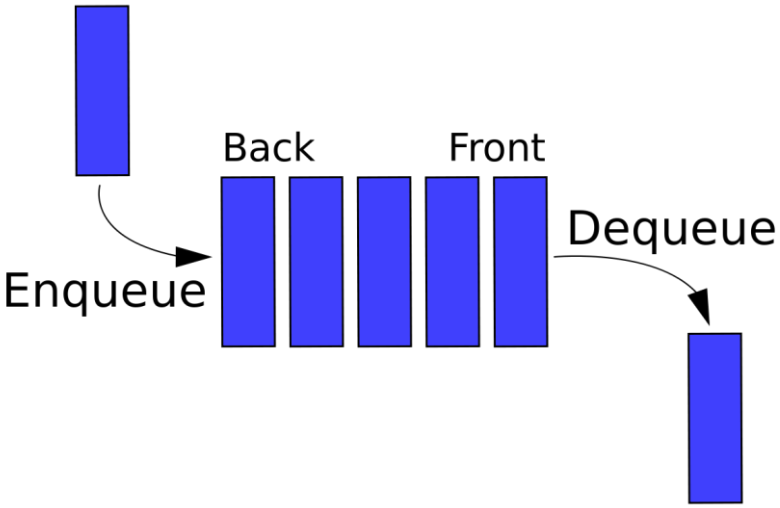
Data Structure

&

Algorithms



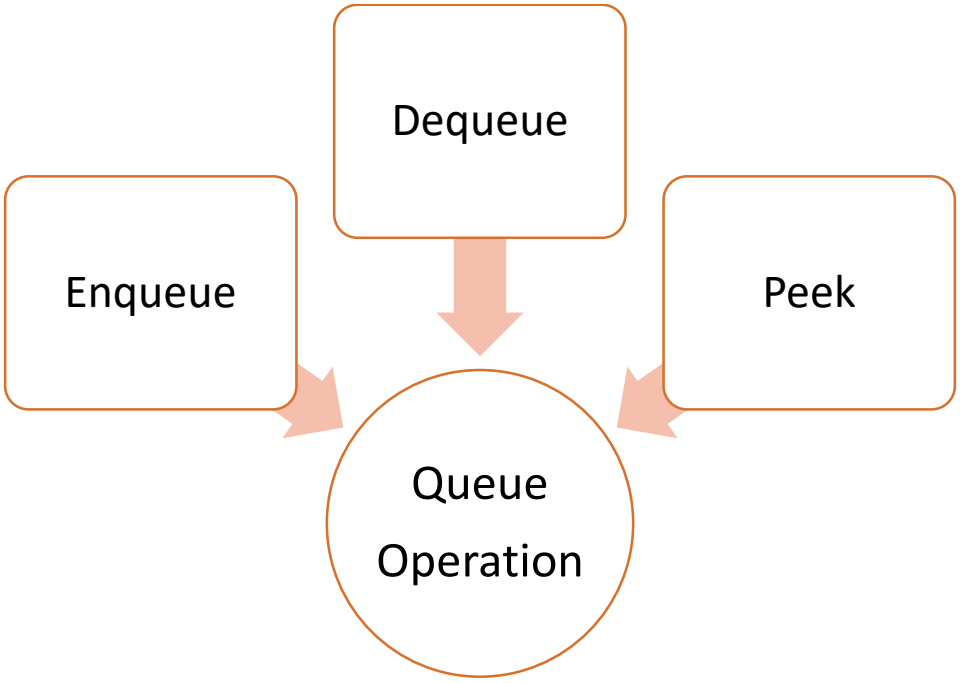
M.Rizwan

Computer Lecturer

Full Stack .NET Developer/Trainer

Queues

Implementation using **Visual C#**



Linear Queue

Implementation using Visual C#

LinearQ.cs

```
using System;
using static System.Console;

namespace LinearQueue
{
    public class LinearQ
    {
        private int[] QU;
        private int F, B;
        public LinearQ(int size)
        {
            QU = new int[size];
            F = B = -1;
        }
        public void Enqueue(int x)  // Insertion
        {
            if (B == QU.Length - 1)
            {
                WriteLine("Queue is Full");
                return;
                // throw new Exception("Queue is Full");
            }
            else
            {
                QU[++B] = x;
                WriteLine($"Value {x} is Inserted!");
            }
            if (F == -1)
                F = 0;
        }
        public void Dequeue()       // Deletion
        {
            if (F == -1)
            {
                WriteLine("Queue is Empty");
                return;
            }
            else
            {
                WriteLine($"Value {QU[F]} is removed!");
                QU[F++] = 0;
            }
            if (F > B)
                F = B = -1;
        }
        public void Display()
        {
            if (F == -1)
            {
```

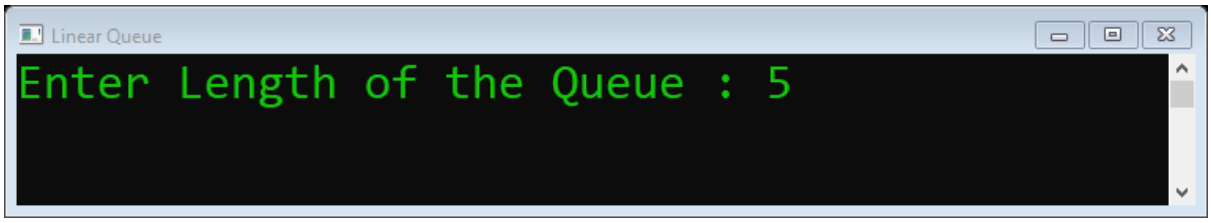
```
                WriteLine("Queue is Empty");
                return;
            }
            WriteLine("Values in queue ...");
            for (int i = F; i <= B; i++)
            {
                WriteLine(QU[i]);
            }
        }
        public int Peek()
        {
            return QU[F];
        }
        public bool IsFull()
        {
            if (B == QU.Length - 1)
                return true;
            else
                return false;
        }
        public bool IsEmpty()
        {
            if (F == -1)
                return true;
            else
                return false;
        }
    }
}
```

Program.cs

```
using System;
using static System.Console;
using static System.Convert;

namespace LinearQueue
{
    class Program
    {
        static void Main(string[] args)
        {
            Write("Enter Length of the Queue : ");
            int length = ToInt32(ReadLine());

            LinearQ linearQ = new LinearQ(length);

            bool flag = true;
            while (flag)
            {
                Clear();
                WriteLine("1 - Insert value into Queue");
                WriteLine("2 - Remove value from Queue");
                WriteLine("3 - Display values from Queue");
                WriteLine("4 - Peek value of the Queue");
                WriteLine("5 - Check if Queue Is-Full");
                WriteLine("6 - Check if Queue Is-Empty");
```

```
                WriteLine("7 - Exit");
                Write("Enter your option : ");
                int op = ToInt32(ReadLine());
                switch (op)
                {
                    case 1:
                        Write("Enter value to Insert : ");
                        int val = ToInt32(ReadLine());
                        linearQ.Enqueue(val);
                        ReadKey();
                        break;
                    case 2:
                        linearQ.Dequeue();
                        ReadKey();
                        break;
                    case 3:
                        linearQ.Display();
                        ReadKey();
                        break;
                    case 4:
                        WriteLine(linearQ.Peek());
                        ReadKey();
                        break;
                    case 5:
                        WriteLine(linearQ.IsFull());
                        ReadKey();
                        break;
                    case 6:
                        WriteLine(linearQ.IsEmpty());
                        ReadKey();
                        break;
                    case 7:
                        flag = false;
                        break;
                    default:
                        WriteLine("Invalid option");
                        ReadKey();
                        break;
                }
            }

            ReadKey(true);
        }
    }
}
```

Output

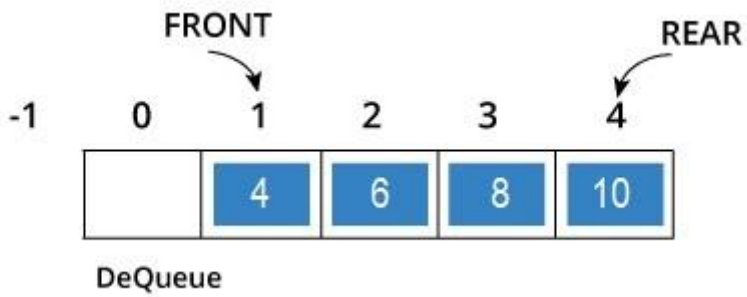


```
Select Linear Queue
1 - Insert value into Queue
2 - Remove value from Queue
3 - Display values from Queue
4 - Peek value of the Queue
5 - Check if Queue Is-Full
6 - Check if Queue Is-Empty
7 - Exit
Enter your option : 1
Enter value to Insert : 2
Value 2 is Inserted!
```

```
Select Linear Queue
1 - Insert value into Queue
2 - Remove value from Queue
3 - Display values from Queue
4 - Peek value of the Queue
5 - Check if Queue Is-Full
6 - Check if Queue Is-Empty
7 - Exit
Enter your option : 3
Values in queue ...
2
4
6
8
10
```

```
Linear Queue
1 - Insert value into Queue
2 - Remove value from Queue
3 - Display values from Queue
4 - Peek value of the Queue
5 - Check if Queue Is-Full
6 - Check if Queue Is-Empty
7 - Exit
Enter your option : 2
Value 2 is removed!
```

Problem …




```
Linear Queue
1 - Insert value into Queue
2 - Remove value from Queue
3 - Display values from Queue
4 - Peek value of the Queue
5 - Check if Queue Is-Full
6 - Check if Queue Is-Empty
7 - Exit
Enter your option : 1
Enter value to Insert : 12
Queue is Full
```

Circular Queue

Implementation using Visual C#

CircularQ.cs

```
using System;
using static System.Console;
namespace CircularQueue
{
    public class CircularQ
    {
        private int[] QU;
        private int F, B;
        public CircularQ(int size)
        {
            QU = new int[size];
            F = B = -1;
        }
        public void Enqueue(int x)
        {
            if (F == (B + 1) % QU.Length)
            {
                WriteLine("CQ is Full");
                return;
            }
            else
            {
                B = (B + 1) % QU.Length;
                QU[B] = x;
                WriteLine($"Value {x} is Inserted!");
            }
            if (F == -1)
                F = 0;
        }
        public void Dequeue()
        {
            if (F == -1 && B == -1)
            {
                WriteLine("CQ is Empty");
                return;
            }
            else
            {
                WriteLine($"Value {QU[F]} is removed!");
                QU[F] = 0;
            }
            if (F == B)
                F = B = -1;
            else
                F = (F + 1) % QU.Length;
        }
    }

}
```

Program.cs

```
using System;
using static System.Console;
using static System.Convert;

namespace CircularQueue
{
    class Program
    {
        static void Main(string[] args)
        {
            Write("Enter Length of the Queue : ");
            int length = ToInt32(ReadLine());

            CircularQ circularQ = new CircularQ(length);

            bool flag = true;
            while (flag)
            {
                Clear();
                WriteLine("1 - Insert value into Queue");
                WriteLine("2 - Remove value from Queue");
                WriteLine("3 - Exit");
                Write("Enter your option : ");
                int op = ToInt32(ReadLine());
                switch (op)
                {
                    case 1:
                        Write("Enter value to Insert : ");
                        int val = ToInt32(ReadLine());
                        circularQ.Enqueue(val);
                        ReadKey();
                        break;
                    case 2:
                        circularQ.Dequeue();
                        ReadKey();
                        break;
                    case 3:
                        flag = false;
                        break;
                    default:
                        WriteLine("Invalid option");
                        ReadKey();
                        break;
                }
            }

            ReadKey(true);
        }
    }
}
```

DEQUE

Implementation using Visual C#

Deque.cs

```
using System;
using static System.Console;

namespace DEQUE
{
    public class Deque
    {
        private int[] QU;
        private int F, R;
        public Deque(int size)
        {
            QU = new int[size];
            F = R = -1;
        }
        public void EnqueueFront(int x)
        {
            if (F == 0 && R == QU.Length-1)
            {
                WriteLine("Deque is Full");
                return;
            }
            if (F == -1 && R == -1)
            {
                F = R = 0;
                QU[F] = x;
            }
            else if (F > 0)
            {
                F--;
                QU[F] = x;
            }
            else
                WriteLine("No space from front side");
        }
        public void EnqueueRear(int x)
        {
            if (F == 0 && R == QU.Length-1)
            {
                WriteLine("Deque is Full");
                return;
            }
            if (F == -1 && R == -1)
            {
                F = R = 0;
                QU[R] = x;
            }
            else if (R < QU.Length-1)
            {
                R++;
```

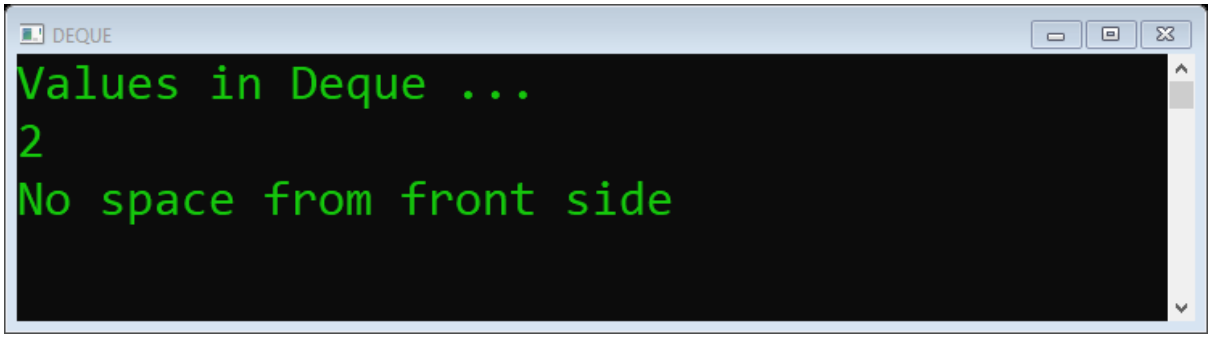
```
                QU[R] = x;
            }
            else
            {
                WriteLine("No space from rear side");
            }
        }
        public void DequeueFront()
        {
            if (F == -1 && R == -1)
            {
                WriteLine("Deque is Empty");
                return;
            }
            else
                QU[F] = 0;
            if (F == R)
                F = R = -1;
            else
                F++;
        }
        public void DequeueRear()
        {
            if (F == -1 && R == -1)
            {
                WriteLine("Deque is Empty");
                return;
            }
            else
                QU[R] = 0;
            if (F == R)
                F = R = -1;
            else
                R--;
        }
        public void Display()
        {
            if (F == -1 && R == -1)
            {
                WriteLine("Deque is Empty");
                return;
            }
            WriteLine("Values in Deque ...");
            for (int i = F; i <= R; i++)
            {
                WriteLine(QU[i]);
            }
        }
    }
}
```

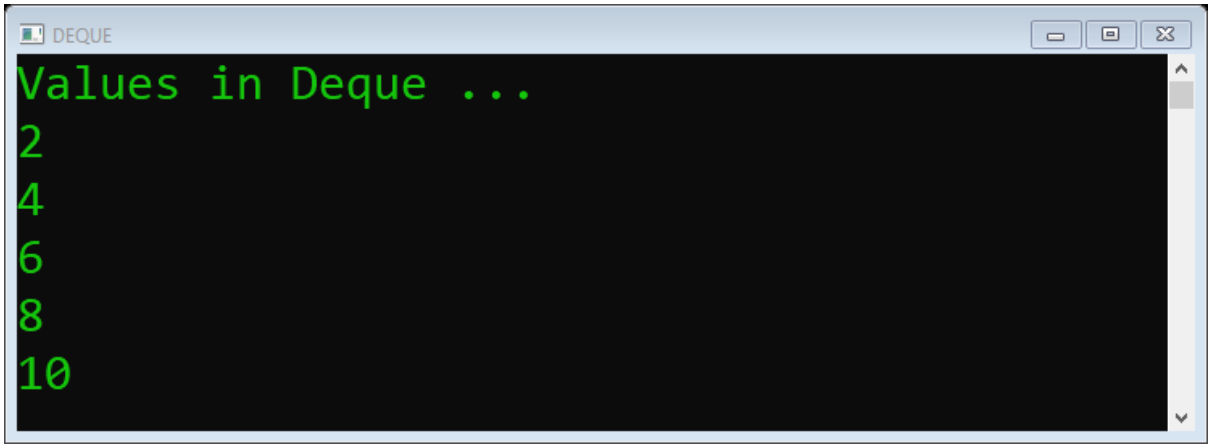
Program.cs

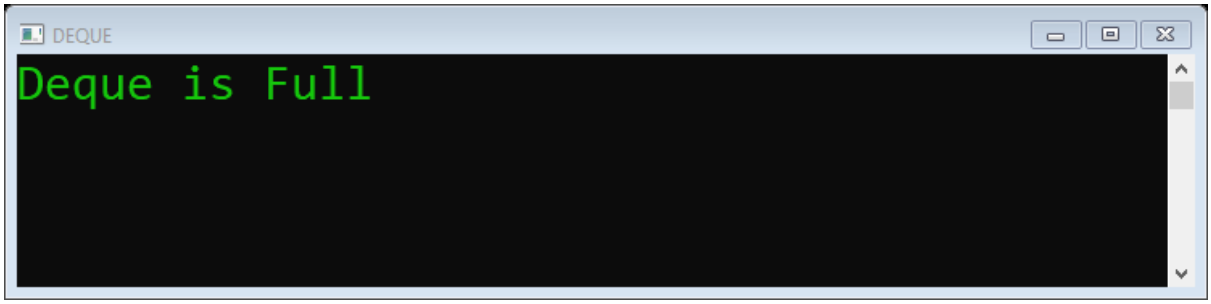
```
using System;
using static System.Console;
using static System.Convert;

namespace DEQUE
{
```

```
    class Program
    {
        static void Main(string[] args)
        {
            Deque dq = new Deque(5);
            dq.EnqueueFront(2);
            //dq.Display();
            //dq.EnqueueFront(4);
            dq.EnqueueRear(4);
            dq.EnqueueRear(6);
            dq.EnqueueRear(8);
            dq.EnqueueRear(10);
            //dq.Display();
            //dq.EnqueueRear(12);
            dq.DequeueFront();
            dq.DequeueFront();
            dq.DequeueFront();
            dq.EnqueueFront(12);
            dq.EnqueueFront(14);
            dq.EnqueueFront(16);
            //dq.EnqueueFront(18);
            //dq.Display();
            dq.DequeueRear();
            dq.DequeueRear();
            dq.Display();
            dq.EnqueueFront(18);

            ReadKey();
        }
    }
}
```

Output







https://github.com/MRizwanSE/Queues-VisualCSharp

Best of Luck 😊