Очередь – структура, элементы которой обрабатываются по принципу FIFO (first in, first out – первый вошел, первый вышел). Элемент, первым попавший в очередь, обрабатывается первым (чем раньше попал, тем раньше обработается).

Для реализации очереди и других структур в C# существуют стандартные классы коллекций. Обобщенная очередь – значит в нее помещаются элементы обобщенного типа, то есть одного и того же для всех элементов. Этот тип указывается при создании обобщенной очереди, он также может быть и производным типом. Классы обобщенных коллекций в C# находятся в пространстве имен System.Collections.Generic. Queue<T> - класс обобщенной очереди, где вместо T указывается тип элементов.
Создание очереди для целых чисел:

```
Queue<int> q = new Queue<int>();
```

Добавление элементов в очередь:

```
q.Enqueue(3);
q.Enqueue(56);
```

Удаление первого элемента из очереди:

```
q.Dequeue();
```

Метод Dequeue() читает первый элемент и при этом удаляет его. Если на момент вызова этого метода в очереди не осталось элементов, генерируется исключение *InvalidOperationException.* Когда необходимо считать первый элемент и при этом не удалять его оттуда, используется метод Peek():

```
int i = q.Peek();
```

В реализации очереди для .NET можно перебрить элементы при помощи foreach:

```
foreach (int i in q)
{
      Console.WriteLine(i);
}
```

Количество элементов в очереди:

```
int length = q.Count
```

Емкость очереди – количество элементов, которое она может содержать. Когда элементы добавляются в очередь, емкость автоматически увеличивается по мере необходимости. Чтобы установить емкость равной фактическому количеству элементов в очереди, если это количество составляет менее 90 процентов текущей емкости, применяется метод TrimExcess().

Пример работы с очередью:

```
using System;

using System.Collections.Generic;

namespace Collections
{
    class Program
    {
        static void Main(string[] args)
        {
            Queue<int> q = new Queue<int>();

            q.Enqueue(20); // очередь становится 20
            q.Enqueue(7); // очередь становится 20, 7
            q.Enqueue(12); // очередь становится 20, 7, 12

            // получаем первый элемент очереди
            int first = q.Dequeue(); //теперь очередь 7, 12
            Console.WriteLine(first);//выведет 20

            // получаем первый элемент без его извлечения
            int p = q.Peek();
            Console.WriteLine(p);//выведет 7

             Console.WriteLine("Сейчас в очереди {0} элементов",
q.Count);//Сейчас в очереди 2 элемента

            Console.ReadLine();
        }
    }
```