**=====================================================================================**

static T Max<T>(T[] source) where T : IComparable

{

if (source.Length == 0)

return default(T);

var max = source[0];

foreach (var item in source)

{

if (item.CompareTo(max) > 0) max = item;

}

return max;

}

**=====================================================================================**

static T Max<T>(T[] source)

{

if (source.Length == 0)

return default(T);

return source.Max();

}

**=====================================================================================**

static T Max<T>(T[] source) where T : IComparable

{

return source.Any() ? source.Max() : default(T);

}

**=====================================================================================**

public bool MoveNext()

{

if (currentItem == null)

currentItem = myQueue.head;

else

currentItem = currentItem.Next;

return currentItem != null;

}

**=====================================================================================**

Реализация IEnumerator

class MyQueue<T> : IEnumerable<T>

{

private Item<T> head;

private Item<T> tail;

public MyQueue()

{

head = null;

tail = null;

}

public void Enqueue(T value)

{

if (head == null)

{

head = tail = new Item<T> { Next = null, Value = value };

}

else

{

var item = new Item<T> { Next = null, Value = value };

tail.Next = item;

tail = item;

}

}

public T Dequeue()

{

var temp = head;

head = temp.Next;

return temp.Value;

}

public IEnumerator<T> GetEnumerator()

{

return new MyQueueEnumerator(this);

}

IEnumerator IEnumerable.GetEnumerator()

{

return GetEnumerator();

}

class MyQueueEnumerator : IEnumerator<T>

{

private MyQueue<T> myQueue;

private Item<T> currentItem;

public MyQueueEnumerator(MyQueue<T> myQueue)

{

this.myQueue = myQueue;

currentItem = null;

}

public T Current => currentItem.Value;

object IEnumerator.Current => throw new NotImplementedException();

public void Dispose()

{

}

public bool MoveNext()

{

if (currentItem == null)

currentItem = myQueue.head;

else

currentItem = currentItem.Next;

return currentItem != null;

}

public void Reset()

{

}

}

class Item<T>

{

public T Value { get; set; }

public Item<T> Next { get; set; }

}

}

**=====================================================================================**