

#83 - Doppeltverkettete Liste - Methodensammlung

Eine doppeltverkettete List-Klasse verwendet diese Nodeklasse. Diese darf nicht verändert werden.

```
class Node
{
    public double Number { get; set; }

    public Node Prev { get; set; }
    public Node Next { get; set; }

    public Node(double number)
    {
        Number = number;
    }

    public override string ToString()
    {
        return Number.ToString();
    }
}

class DoubleList
{
    private Node Head; // enthält immer den ersten Knoten
    private Node Tail; // enthält immer auf den letzten Knoten

    public void AddAtEnd(double value)
    {
        Node node = new Node(value);

        if (Head == null)
        {
            Head = Tail = node;
            return;
        }

        Tail.Next = node;
        node.Prev = Tail;

        Tail = node;
    }
    ...
}
```

Implementiere folgende Methoden in der Klasse DoubleList.

1. Methoden bei denen man durch die ganze Liste laufen muss

1.1 string ToString()

Rückgabe eines Strings aller in der Liste gespeicherten Double-Werte durch einen Beistrich getrennt.

1.2 string ToStringReverse()

Rückgabe eines Strings aller in der Liste gespeicherten Double-Werte beginnend mit dem Ende der Liste.
Das Trennzeichen ist ein Beistrich.

1.3 int FindMax()

Rückgabe des Index des größten Werts der Liste. -1 für leere Listen.

1.4 double GetMax()

Rückgabe des größten in der Liste gespeicherten Werts. double.MIN_VALUE für leere Liste.

1.5 (double min, double max) GetMinMax()

Rückgabe des kleinsten und größten in der Liste gespeicherten Werts in einem Value Tuple.

Gib (double.MAX_VALUE, double.MIN_VALUE) für leere Liste zurück.

1.6 double GetAverage()

Rückgabe des Mittelwerts aller in der Liste gespeicherten Werte.

1.7 bool Contains(double value)

Gibt zurück, ob ein Wert in der Liste enthalten ist.

1.8 void Multiply(double factor)

Multipliziere alle Werte der Liste mit dem selben Faktor.

1.9 static DoubleList Multiply(DoubleList list, double factor)

Multipliziere alle Werte von list mit dem selben Faktor.

1.10 Überlade den Operator * so, dass alle Werte einer Liste mit demselben Faktor multipliziert werden.

Achtung: Es soll sowohl list * 5.2, als auch 5.2 * list möglich sein. Tipp: Mehrmals Überladen!

1.11 Überlade ebenso den Operator / für die Division und benutze den Operator von 1.9

Hinweis: Eine Division ist eine Multiplikation mit dem Kehrwert.

1.12 Nur für Ambitionierte! Implementiere für die Klasse einen (double) Cast Operator, der die Summe aller Werte in der Liste zurückgibt. Benutze dafür das implicit Schlüsselwort und implementiere folgende Methode so, dass sie die Summe des Inputparameters list zurückgibt.

```
static implicit operator double(DoubleList list)
{
    ...
}
```

Dann ist folgender Code im Main möglich.

```
DoubleList l = new DoubleList(1.2, 5.6, 3.1);
Console.WriteLine((double)l); // -> 9.9
double sum = l;
Console.WriteLine(sum); // -> 9.9
```

2. Zugriffsmethoden auf einzelne Werte

2.1 double GetAt(int index)

Gibt den Wert an einem bestimmten Index zurück.

2.2 Schreibe einen Indexer, der den Wert an einem bestimmten Index der Liste lesen (get) und schreiben (set) kann.

2.3 double GetFirst()

Gibt den ersten Wert der Liste zurück.

2.4 double GetLast()

Gibt den letzten Wert der Liste zurück.

3. Methoden die Elemente löschen

3.1 void Clear()

Löscht alle Werte der Liste, d. h. die Liste ist hinterher eine leere Liste.

3.2 void RemoveAt(int index)

Löscht den Wert an einer bestimmten Stelle.

3.3 void Remove(double value)

Löscht den ersten Wert mit einem bestimmten Wert.

3.4 void RemoveFirst(int count)

Löscht die ersten count Werte der Liste

3.5 void RemoveLast(int count)

Löscht die letzten count Werte einer Liste.

3.6 void RemoveFrom(int indexFrom, int count)

Löscht ab einem bestimmten Index, die angegebene Anzahl von Werten.

4. Methoden die Elemente hinzufügen

4.1 AddAtStart(double value)

Fügt ein Element am Beginn der Liste ein.

4.2 AddAllAtEnd(params double[] values)

Fügt alle Elemente am Ende der Liste an.

4.3 Insert(int index, double value)

Fügt einen Wert als bestimmten Index ein.

4.4 InsertAll(int index, double[] value)

Fügt alle Werte eines Arrays ab einen bestimmten Index ein.

4.5 InsertAll(int index, DoubleList list)

Fügt alle Werte einer Liste ab einem bestimmten Index ein.

5. Filtermethoden

5.1 `DoubleList FindGreater(double threshold)`

Gibt eine Liste aller Werte zurück, die größer als ein bestimmter Schwellwert ist.

5.2 `DoubleList FindAndRemoveGreater(double threshold)`

Gibt eine Liste aller Werte zurück, die größer als ein bestimmter Schwellwert ist und entfernt diese Werte.

5.3 `DoubleList SubList(int index)`

Gibt eine Liste zurück, die alle Werte ab einem bestimmten Index enthält.

5.4 `DoubleList SubList(int index, int count)`

Gibt eine Liste zurück, die alle Werte ab einem bestimmten Index enthält, maximal aber count viele Werte.
Hinweis: Das Verhalten soll analog zur Methode `String.SubString` sein.

6. Konstruktoren

6.1 `DoubleList(params double[] values)`

Varargs-Konstruktor: Fügt alle Elemente am Ende der Liste ein.

6.2 `DoubleList(DoubleList original)`

Copy-Konstruktor: Erzeugt eine Liste, die eine Kopie einer anderen Liste ist.

7. Sonstige Methoden

7.1 `void Swap(int index1, int index2)`

Tauscht die Nodes (Nicht die Werte!) an den Positionen `index1` und `index2` aus.

`liste = 10.1 -> 8.4 -> 3.2 -> 7.5 -> 6.4` und `liste.Swap(0,3)` ergibt

`liste = 7.5 -> 8.4 -> 3.2 -> 10.1 -> 6.4`

7.2 `void Swap(int index)`

Tauscht eine Node mit ihrem Nachfolgerknoten an einer bestimmten Position aus.

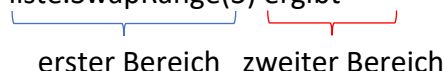
`liste = 10.1 -> 8.4 -> 3.2 -> 7.5 -> 6.4` und `liste.Swap(2)` ergibt

`liste = 10.1 -> 8.4 -> 7.5 -> 3.2 -> 6.4`

7.3 `void SwapRange(int index)`

Tauscht die Bereiche ab einem bestimmten Index aus. Der Index definiert wo der hintere Bereich beginnt.

`v` und `liste.SwapRange(3)` ergibt


erster Bereich zweiter Bereich

`liste = 7.5 -> 6.4 -> 10.1 -> 8.4 -> 3.2`


zweiter Bereich erster Bereich

7.4 Überlade den Operator | so, dass Du die Methode von 7.3 aufrufen kannst.

Hinweis: Eine Möglichkeit ist es den Copy-Konstruktor von 6.2 zu verwenden und eine Kopie der Liste anzulegen und dann SwapRange aufzurufen.

liste = 10.1 -> 8.4 -> 3.2 -> 7.5 -> 6.4 und liste | 3 ergibt

liste = 7.5 -> 6.4 -> 10.1 -> 8.4-> 3.2

7.5 static void Swap(DoubleList a, DoubleList b, int pos, int count)

Die Methode tauscht Werte zwischen zwei Listen aus.

Und zwar ab der Position posFrom, soviele wie count angibt.

Hinweis: Zur Vereinfachung kann davon ausgegangen werden, dass beide Listen gleich lang sind.

Die Methode für unterschiedlich lange Listen, ist aufwendiger.

List.Swap(l1, l2, 2, 3)

l1=	a	b	c	d	e	f
			↕	↕	↕	
l2=	A	B	C	D	E	F

Beispiele:

List.Swap(l1, l2, 0, 1)

l1=	a	b	c	d	e	f
	↕					
l2=	A	B	C	D	E	F

List.Swap(l1, l2, 3, 5)

l1=	a	b	c	d	e	f
			↕	↕	↕	
l2=	A	B	C	D	E	F

List.Swap(l1, l2, 2, 0)

l1=	a	b	c	d	e	f
l2=	A	B	C	D	E	F

List.Swap(l1, l2, 0, l1.Count)

l1=	a	b	c	d	e	f
	↕	↕	↕	↕	↕	↕
l2=	A	B	C	D	E	F

7.6 static DoubleList CreateRandom(int count)

Erzeugt eine Liste mit einer bestimmten Anzahl von Zufallszahlen im Intervall [0,1).

Hinweis: Gehe am Beginn der Methode mit

Random rnd = new Random()

einen Zufallszahlengenerator an und erzeuge mit

rnd.NextDouble()

entsprechend viele Zufallszahlen. Die Werte sind alle im Bereich [0,1).

7.7 static DoubleList CreateRandom(int count, double min, double max)

Erzeugt eine Liste mit einer bestimmten Anzahl von Zufallszahlen, die alle im Bereich [min, max) liegen.

Hinweis: Gehe wie in 7.6 vor, aber skaliere den Bereich [0,1) auf [min,max).

7.8 static double ScalarProduct(DoubleList list1, DoubleList list2)

Die Methode berechnet die Summe der Produkte aller Werte in den Listen (=Skalarprodukt).

Produkt der ersten Werte + Produkt der zweiten Werte + Produkt der dritten Werte + ... usw.

Überprüfe: Die beiden Listen list1 und list2 müssen gleichlang sein.

Wirf ansonsten eine Exception:

```
throw new IllegalArgumentException("Length mismatch!");
```

Beispiel:

list1 = 10.1 -> 8.4 -> 3.2 -> 7.5 -> 6.4

list2 = 2.4 -> 0 -> 3 -> -3.4 -> 8.9

24.24 + 0 + 9.6 + (-25.5) + 56.96 ergibt 65.8

7.9 void Sort()

Sortiert alle Werte der Liste, z. B. mit Bubble-Sort