БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Лабораторная работа № 2

по дисциплине «Структуры и алгоритмы обработки данных»

Вариант № 4

Выполнил студент: Ефименко Павел Викторович,

группа 981063

Минск 2020

**Условие:**

Создать однонаправленный список, состоящий из n целых чисел. Преобразовать его в два списка: первый список должен содержать только четные числа, второй − нечетные.

**Листинг:**

public class LinkedList<T> : IEnumerable<T>

{

public bool IsEmpty => Count == 0;

public Node<T> Head { get; set; }

public Node<T> Tail { get; set; }

public int Count { get; set; }

public void Add(T data)

{

var node = new Node<T>(data);

if (Head == null)

Head = node;

else

Tail.Next = node;

Tail = node;

Count++;

}

public bool Remove(T data)

{

var current = Head;

Node<T> previous = null;

while (current != null)

{

if (current.Data.Equals(data))

{

if (previous != null)

{

previous.Next = current.Next;

if (current.Next == null)

Tail = previous;

}

else

{

Head = Head.Next;

if (Head == null)

Tail = null;

}

Count--;

return true;

}

previous = current;

current = current.Next;

}

return false;

}

public void Clear()

{

Head = null;

Tail = null;

Count = 0;

}

public bool Contains(T data)

{

var current = Head;

while (current != null)

{

if (current.Data.Equals(data))

return true;

current = current.Next;

}

return false;

}

public T GetByIndex(int index)

{

var current = Head;

var ind = 0;

while (current != null)

{

if (index == ind)

return current.Data;

current = current.Next;

ind++;

}

return default;

}

public void AppendFirst(T data)

{

var node = new Node<T>(data)

{

Next = Head

};

Head = node;

if (Count == 0)

Tail = Head;

Count++;

}

IEnumerator<T> IEnumerable<T>.GetEnumerator()

{

Node<T> current = Head;

while (current != null)

{

yield return current.Data;

current = current.Next;

}

}

public IEnumerator GetEnumerator()

{

return ((IEnumerable)this).GetEnumerator();

}

}

public class Node<T>

{

public Node(T data)

{

Data = data;

}

public T Data { get; set; }

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

}

class Program

{

static void Main(string[] args)

{

new Task1().Execute();

}

class Task1

{

public void Execute()

{

Console.WriteLine($"{this.GetType().FullName}\n");

var linkedList = new LinkedList<int> { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };

Console.WriteLine($"Исходный список:\t{string.Join(" ", linkedList)}");

var evenList = CreateLinkedListWithOnlyEvenItems(linkedList);

Console.WriteLine($"Исходный список EVEN:\t{string.Join(" ", evenList)}");

var oddList = CreateLinkedListWithOnlyOddItems(linkedList);

Console.WriteLine($"Исходный список ODD:\t{string.Join(" ", oddList)}");

}

public LinkedList<int> CreateLinkedListWithOnlyEvenItems(LinkedList<int> list)

{

var newList = new LinkedList<int>();

for (int i = 0; i < list.Count; i++)

{

var item = list.GetByIndex(i);

if (item.IsEven())

{

newList.Add(item);

}

}

return newList;

}

public LinkedList<int> CreateLinkedListWithOnlyOddItems(LinkedList<int> list)

{

var newList = new LinkedList<int>();

for (int i = 0; i < list.Count; i++)

{

var item = list.GetByIndex(i);

if (!item.IsEven())

{

newList.Add(item);

}

}

return newList;

}

public void RemoveEachSecondElem(LinkedList<int> ts)

{

var failedItem = new List<int>();

for (int i = 0; i < ts.Count; i++)

{

if ((i + 1) % 2 == 0)

{

var val = ts.GetByIndex(i);

failedItem.Add(val);

}

}

for (int i = 0; i < failedItem.Count; i++)

{

ts.Remove(failedItem[i]);

Console.WriteLine(string.Join(" ", ts));

}

}

}

}

**Выполнение:**

****