БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Лабораторная работа № 3

по дисциплине «Структуры и алгоритмы обработки данных»

Вариант № 4

Выполнил студент: Ефименко Павел Викторович,

группа 981063

Минск 2020

**Условие:**

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

**Листинг:**

public class DoubleLinkedList

{

public int Count { get; private set; }

public bool IsEmpty { get { return Count == 0; } }

public DoubleNode<int> Head { get; set; }

public DoubleNode<int> Tail { get; set; }

public void PrintDoubleLinkedList()

{

var node = Head;

if (Head == Head?.Next || Head == null)

{

System.Console.WriteLine("The end of linked list");

}

else

{

while (node != null)

{

System.Console.WriteLine(node.Data.ToString());

node = node.Next;

}

}

}

public void PrintCircleDoubleLinkedList()

{

var node = Head;

if (Head == Head.Next)

{

System.Console.WriteLine("The end of linked list");

}

else

{

do

{

System.Console.WriteLine(node.Data.ToString());

node = node.Next;

} while (node != Head);

}

}

public void Add(int data)

{

var node = new DoubleNode<int>(data);

if (Head == null)

Head = node;

else

{

Tail.Next = node;

node.Previous = Tail;

}

Tail = node;

Count++;

}

public void AddFirst(int data)

{

var node = new DoubleNode<int>(data);

var temp = Head;

node.Next = temp;

Head = node;

if (Count == 0)

Tail = Head;

else

temp.Previous = node;

Count++;

}

public bool Remove(int data)

{

var current = Head;

while (current != null)

{

if (current.Data.Equals(data))

{

break;

}

current = current.Next;

}

if (current != null)

{

if (current.Next != null)

{

current.Next.Previous = current.Previous;

}

else

{

Tail = current.Previous;

}

if (current.Previous != null)

{

current.Previous.Next = current.Next;

}

else

{

Head = current.Next;

}

Count--;

return true;

}

return false;

}

public void Clear()

{

Head = null;

Tail = null;

Count = 0;

}

public bool Contains(int data)

{

var current = Head;

while (current != null)

{

if (current.Data.Equals(data))

return true;

current = current.Next;

}

return false;

}

public void ToRing()

{

Tail.Next = Head;

Head.Previous = Tail;

}

public DoubleLinkedList GetEvenLinkedList()

{

var evenLinkedList = new DoubleLinkedList();

var node = Head;

if (Head == Head.Next)

{

System.Console.WriteLine("The end of linked list");

}

else

{

while (node != null)

{

var value = node.Data;

if (value.IsEven())

{

evenLinkedList.Add(value);

}

node = node.Next;

}

}

evenLinkedList.ToRing();

return evenLinkedList;

}

public DoubleLinkedList GetOddLinkedList()

{

var oddLinkedList = new DoubleLinkedList();

var node = Head;

if (Head == Head.Next)

{

System.Console.WriteLine("The end of linked list");

}

else

{

while (node != null)

{

var value = node.Data;

if (!value.IsEven())

{

oddLinkedList.Add(value);

}

node = node.Next;

}

}

oddLinkedList.ToRing();

return oddLinkedList;

}

}

public class DoubleNode<T>

{

public DoubleNode(T data)

{

Data = data;

}

public T Data { get; set; }

public DoubleNode<T> Previous { get; set; }

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

}

class Program

{

static void Main(string[] args)

{

var linkedList = new DoubleLinkedList();

linkedList.Add(1);

linkedList.Add(2);

linkedList.Add(3);

linkedList.Add(4);

linkedList.PrintDoubleLinkedList();

var evenLinkedList = linkedList.GetEvenLinkedList();

var oddLinkedList = linkedList.GetOddLinkedList();

Console.WriteLine();

evenLinkedList.PrintCircleDoubleLinkedList();

Console.WriteLine();

oddLinkedList.PrintCircleDoubleLinkedList();

Console.ReadKey();

}

}

**Выполнение:**

****