**Лабораторная работа 2**

**Линейные циклические структуры данных.**

Вариант 2

**1. Задание**

Во всех вариантах необходимо организовать линейную циклическую структуру, состоящую из 10 элементов вводимых с клавиатуры. Организовать функцию просмотра всего списка элементов. Организовать функцию очистки структуры и проверки на пустоту. Организовать циклический двунаправленный линейный список с функцией вставки элемента в список справа от текущего элемента.

**2. Листинг программы**

ListElements.cs

public class Item<T>

{

public Item(T node) => Node = node;

public T Node { get; set; }

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

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

}

List.cs

using System.Collections;

using System.Collections.Generic;

namespace LR2

{

public class DoubleLinkedList<T> : IEnumerable<T> // кольцевой двусвязный список

{

Item<T> head { get; set; }

int count { get; set; }

public int Count { get { return count; } }

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

public void Add(T node)

{

Item<T> newNode = new Item<T>(node);

if (head == null)

{

head = newNode;

head.Next = newNode;

head.Previous = newNode;

}

else

{

newNode.Previous = head.Previous;

newNode.Next = head;

head.Previous.Next = newNode;

head.Previous = newNode;

}

count++;

}

// удаление элемента

public bool Remove(T node)

{

Item<T> current = head;

Item<T> removedItem = null;

if (count == 0) return false;

// поиск удаляемого узла

do

{

if (current.Node.Equals(node))

{

removedItem = current;

break;

}

current = current.Next;

}

while (current != head);

if (removedItem != null)

{

// если удаляется единственный элемент списка

if (count == 1)

head = null;

else

{

// если удаляется первый элемент

if (removedItem == head)

{

head = head.Next;

}

removedItem.Previous.Next = removedItem.Next;

removedItem.Next.Previous = removedItem.Previous;

}

count--;

return true;

}

return false;

}

public void Clear()

{

head = null;

count = 0;

}

public bool Contains(T data)

{

Item<T> current = head;

if (current == null) return false;

do

{

if (current.Node.Equals(data))

return true;

current = current.Next;

}

while (current != head);

return false;

}

IEnumerator IEnumerable.GetEnumerator() => ((IEnumerable)this).GetEnumerator();

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

{

Item<T> current = head;

do

{

if (current != null)

{

yield return current.Node;

current = current.Next;

}

}

while (current != head);

}

}

}

Program.cs

using System;

namespace LR2

{

class Program

{

static DoubleLinkedList<object> List = new DoubleLinkedList<object>();

static void Main()

{

while (true)

{

Console.Write("Что Вы хотите сделать?\n" +

"1. Добавить элемент\n" +

"2. Удалить элемент\n" +

"3. Вывести все элементы\n" +

"4. Очистить список\n" +

"5. Выйти из программы\n" +

"Выберите действие: ");

switch (int.Parse(Console.ReadLine()))

{

case 1:

AddElement(); break;

case 2:

RemoveAtList(); break;

case 3:

AllElements(); break;

case 4:

List.Clear(); break;

case 5: break;

}

}

}

static void AllElements()

{

int i = 0;

if (List.IsEmpty)

{

Console.WriteLine("Элементов в списке нет!");

return;

}

Console.Write("Элементы в списке: ");

foreach (var data in List)

{

Console.Write($"{data} ");

i++;

}

Console.WriteLine();

}

static void AddElement()

{

Console.Write("Что Вы хотите добавить: ");

string data = Console.ReadLine();

List.Add(data);

}

static void RemoveAtList()

{

AllElements();

if (List.IsEmpty)

return;

Console.Write("Введите элемент, который хотите удалить: ");

string delete = Console.ReadLine();

List.Remove(delete);

}

}

}

**3. Ответы на контрольные вопросы**

1. Циклический однонаправленный список.  
          Ответ: то же самое, что и линейный однонаправленный список, но последний указатель ссылается на первый элемент, а не на NULL.

2. Операция вставки элемента в циклический однонаправленный список.  
          Ответ: в качестве входных параметров передаются данные для заполнения создаваемого элемента, указатель на начало списка и указатель на текущий элемент в списке, после которого осуществляется вставка.

3. Операция удаления элемента из циклического однонаправленного списка.  
          Ответ: происходит удаление элемента, на который установлен указатель текущего элемента. После удаления указатель текущего элемента устанавливается на следующий за удаляемым элемент списка.

4. Операция просмотра циклического однонаправленного списка.  
          Ответ: то же самое, как и в линейном однонаправленном списке, но просмотр будет идти до тех пор, пока не будет указателя на первый элемент.

5. Циклический двунаправленный список.  
          Ответ: В этом циклическом списке любой элемент имеет два указателя, один из которых указывает на следующий элемент в списке, а второй указывает на предыдущий элемент.

6. Операция вставки элемента в циклический двунаправленный список.  
          Ответ: если список пуст, то созданный элемент будет первым, иначе, последний элемент ссылается на новый как на следующий, а этот новый ссылается как на предыдущий, далее элемент полностью занимает место последнего в списке.

7. Операция удаления элемента из циклического двунаправленного списка.  
          Ответ: пока не указывается на элемент с заданными значениями, идёт продвижение по листу. Далее ссылка как-бы перепрыгивает, игнорируя тот, который нужно удалить, связываясь с тем, что идёт за удалённым.

8. Операция просмотра циклического двунаправленного списка.  
          Ответ: элемент ссылается на следующий, пока не дойдёт до NULL.