**Лабораторная работа 1**

**Линейный однонаправленный список**

Вариант 2

**1. Задание**

Организовать список, состоящий из 10 элементов, вводимых с клавиатуры, через функцию вставки первого элемента в список. Организовать функцию просмотра всего списка элементов. Организовать функцию вставки элемента справа относительно текущего. Организовать функцию поиска элемента в списке. Организовать функцию удаления элемента из списка. Организовать функцию очистки списка. Организовать однонаправленный линейный список с функцией вставки элемента в список с заменой текущего элемента.

**2. Листинг программы**

ListElement.cs

public class Node<T>

{

public Node(T data) => Data = data;

public T Data { get; set; }

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

}

List.cs

using System.Collections;

using System.Collections.Generic;

namespace LR1

{

public class LinkedList<T> : IEnumerable<T>

{

Node<T> head;

Node<T> tail;

int count;

public void Add(T data)

{

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

if (head == null)

head = node;

else

tail.Next = node;

tail = node;

count++;

}

public void Edit(T data, T value)

{

Node<T> current = head;

while (current != null)

{

if (current.Data.Equals(data))

current.Data = value;

current = current.Next;

}

}

public bool Remove(T data)

{

Node<T> 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 int Count { get { return count; } }

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

public void Clear()

{

head = null;

tail = null;

count = 0;

}

public bool Contains(T data)

{

Node<T> current = head;

while (current != null)

{

if (current.Data.Equals(data))

return true;

current = current.Next;

}

return false;

}

public void AppendFirst(T data)

{

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

node.Next = head;

head = node;

if (count == 0)

tail = head;

count++;

}

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

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

{

Node<T> current = head;

while (current != null)

{

yield return current.Data;

current = current.Next;

}

}

}

}

Program.cs

using System;

namespace LR1

{

class Program

{

static void Main()

{

LinkedList<string> linkedList = new LinkedList<string>();

while (true)

{

Console.Write("\n1. Вывод всех элементов\n" +

"2. Вставка элемента справа\n" +

"3. Поиск элемента\n" +

"4. Удаление элемента\n" +

"5. Очистка списка\n" +

"6. Измeнение элемента\n" +

"7. Выйти\n" +

"Выберите действие: ");

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

{

case 1:

Console.WriteLine();

foreach (var item in linkedList)

Console.Write($"{item} "); break;

case 2:

linkedList.Add(Console.ReadLine()); break;

case 3:

bool isPresent = linkedList.Contains(Console.ReadLine());

Console.WriteLine(isPresent == true ? "Элемент присутствует" : "Элемент отсутствует"); break;

case 4:

linkedList.Remove(Console.ReadLine()); break;

case 5:

linkedList.Clear(); break;

case 6:

Console.Write("Введите элемент, который хотите изменить, затем введите его новое значение: ");

linkedList.Edit(Console.ReadLine(), Console.ReadLine()); break;

default: break;

}

}

}

}

}

**3. Ответы на контрольные вопросы**

1. Однонаправленный линейный список.  
 Ответ: однонаправленный линейный список — это структура данных, представляющая собой логически связанную коллекцию элементов списка. В этом списке любой элемент имеет один указатель, который указывает на следующий элемент в списке или является пустым указателем у последнего элемента.

2. Функция вставки первого элемента в список.  
 Ответ: в случае, если при проверке выявлено, что лист пуст, то новый созданный элемент будет первым.

3. Функция вставки не первого элемента в список.  
          Ответ: в случае, если при проверке выявлено, что лист не является пустым, то необходимо сохранить в новый элемент ссылку текущего, далее за текущим вставляем элемент и делаем его текущим.

4. Функция просмотра всех элементов списка.  
          Ответ: последовательный просмотр всех элементов списка до тех пор, пока ссылка не будет указывать на NULL.

5. Функция удаления элемента из списка.  
          Ответ: если целью удаления является первый элемент, то второй элемент необходимо сделать первым, иначе, указатель текущего элемента устанавливается на предшествующий элемент списка.

6. Функция очистки списка.  
         Ответ: создаётся временный элемент, осуществляется переход к следующему элементу, который удаляется путём перезаписи первого элемента.

7. Функция поиска элемента в списке по информационному полю.  
          Ответ: производится последовательный просмотр всех элементов в списке до тех пор, пока один из них не будет содержать заданное значение или пока не будет достигнут конец списка.