**Техническое задание №1**Линейный однонаправленный список.  
Вариант 6

Организовать однонаправленный линейный список с функцией поиска минимального элемента в списке.

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.

Листинг файла:

using System;

using System.Collections.Generic;

namespace Laba\_1

{

class Program

{

public static Deque deque = new Deque();

static void Main(string[] args)

{

Console.WriteLine("Введите числа:");

for (int i = 0; i < 10; i++)

{

Console.Write($"{i+1}):");

deque.AddElement(ref deque.\_current, Convert.ToInt32(Console.ReadLine()));

Console.Write("\n");

}

while (true)

{

Console.WriteLine("1) вставка нового элемента\n" +

"2) просмотр всего списка\n" +

"3) поиск элемента\n" +

"4) удаление элемента\n" +

"5) очистка всего списка\n" +

"6) поиск минимального элемента в списке");

Console.Write("Введите значение: ");

int answer = Convert.ToInt32(Console.ReadLine());

Console.Clear();

switch (answer)

{

default:

Console.WriteLine("Такого варианта ответа нет!");

break;

case 1:

Console.Write("Введите число: ");

deque.AddElementRight(Convert.ToInt32(Console.ReadLine()));

Console.WriteLine("Число добавлено!\n");

break;

case 2:

Console.Write("Все элеметы: ");

List<int> list = deque.Show();

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

{

Console.Write($" {list[i]}");

}

Console.WriteLine();

break;

case 3:

Console.WriteLine("Введите число для поиска");

int numberSearch = Convert.ToInt32(Console.ReadLine());

if (deque.Search(numberSearch) != null)

Console.WriteLine(deque.Search(numberSearch).Value);

else

Console.WriteLine("Число не найдено");

break;

case 4:

Console.Write("Введите число для удаления: ");

int numberRemove = Convert.ToInt32(Console.ReadLine());

if (deque.Search(numberRemove) != null)

{

deque.RemoveElement(deque.Search(numberRemove));

Console.WriteLine("Число удалено!\n");

}

else

Console.WriteLine("Число не найдено");

break;

case 5:

deque.Clear();

Console.WriteLine("Список чист!");

break;

case 6:

Console.WriteLine($"Минимальное значение в списке: {deque.FindMinValue()} \n");

break;

}

}

}

}

}

using System.Collections.Generic;

using System.Linq;

namespace Laba\_1

{

class DequeNode

{

public int Value;

public DequeNode Next;

public DequeNode(DequeNode next, int value)

{

Next = next;

Value = value;

}

}

class Deque

{

public DequeNode \_head = null;

public DequeNode \_current = null;

public Deque() { }

public void AddFirstElement(int value)

{

DequeNode tmp = new DequeNode(null, value);

if (\_head == null)

\_head = tmp;

else

{

\_head.Next = \_head;

\_head = tmp;

}

}

public void AddElement(ref DequeNode current, int value)

{

if (\_head == null)

{

AddFirstElement(value);

\_current = \_head;

}

else

{

DequeNode tmp = new DequeNode(null, value);

tmp.Next = \_current.Next;

\_current.Next = tmp;

\_current = tmp;

}

current = \_current;

}

public List<int> Show()

{

List<int> list = new List<int>();

DequeNode tmp = \_head;

while (tmp != null)

{

list.Add(tmp.Value);

tmp = tmp.Next;

}

return list;

}

public DequeNode Search(int value)

{

DequeNode tmp = \_head;

while (tmp != null)

{

if (tmp.Value == value)

return tmp;

tmp = tmp.Next;

}

return null;

}

public void RemoveElement(DequeNode target)

{

if (\_head == target)

\_head = \_head.Next;

else

{

DequeNode tmp = \_head;

while (tmp.Next != target)

tmp = tmp.Next;

tmp.Next = target.Next;

}

}

public void Clear()

{

DequeNode tmp;

while (\_head != null)

{

tmp = \_head.Next;

\_head = tmp;

}

}

public void Replace(DequeNode target, int number)

{

target.Value = number;

}

public void AddElementRight(int value)

{

if (\_head == null)

{

AddFirstElement(value);

\_current = \_head;

}

else

{

DequeNode node = new DequeNode(null, value);

DequeNode tmp = \_head.Next;

\_head.Next = node;

node.Next = tmp;

}

}

public int FindMinValue()

{

List<int> list = Program.deque.Show();

return list.Min();

}

}

}

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.

**Техническое задание №2**Линейные циклические структуры данных.  
Вариант 6

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.  
Тип структуры и остальные функции реализовать в соответствии с вариантом.  
Организовать циклический двунаправленный линейный список с функцией вывода на экран только четных элементов списка.

Листинг файла:

using System;

namespace Laba\_2

{

class Program

{

public static DoublyLinkedList doublyLinkedList = new DoublyLinkedList();

static void Main(string[] args)

{

Console.WriteLine("Введите числа:");

for (int i = 0; i < 10; i++)

{

Console.Write($"{i + 1}): ");

int number = Convert.ToInt32(Console.ReadLine());

doublyLinkedList.AddLastNode(number);

}

while (true)

{

Console.WriteLine("1) вставка нового элемента\n" +

"2) просмотр всего списка\n" +

"3) поиск элемента\n" +

"4) удаление элемента\n" +

"5) очистка всего списка\n" +

"6) вывести на экран четные элементы списка");

Console.Write("Введите значение: ");

int answer = Convert.ToInt32(Console.ReadLine());

Console.Clear();

switch (answer)

{

default:

Console.WriteLine("Такого варианта ответа нет!");

break;

case 1:

Console.Write("Введите число: ");

doublyLinkedList.AddLastNode(Convert.ToInt32(Console.ReadLine()));

Console.WriteLine("Число добавлено!\n");

break;

case 2:

doublyLinkedList.Show();

break;

case 3:

Console.WriteLine("Введите число для поиска");

int numberSearch = Convert.ToInt32(Console.ReadLine());

if (doublyLinkedList.Search(numberSearch) != null)

Console.WriteLine($"Найденное число: {doublyLinkedList.Search(numberSearch).Value}");

else

Console.WriteLine("Число не найдено");

break;

case 4:

Console.Write("Введите число для удаления: ");

int numberRemove = Convert.ToInt32(Console.ReadLine());

if (doublyLinkedList.Search(numberRemove) != null)

{

doublyLinkedList.RemoveNode(doublyLinkedList.Search(numberRemove));

Console.WriteLine("Число удалено!\n");

}

else

Console.WriteLine("Число не найдено");

break;

case 5:

doublyLinkedList.Clear();

Console.WriteLine("Список чист!");

break;

case 6:

doublyLinkedList.Sort(doublyLinkedList);

break;

}

}

}

}

}

using System;

namespace Laba\_2

{

class DoublyLinkedListNode

{

public int Value { get; set; }

public DoublyLinkedListNode Next { get; set; }

public DoublyLinkedListNode Prev { get; set; }

public DoublyLinkedListNode(int value)

{

Value = value;

}

}

class DoublyLinkedList

{

public DoublyLinkedListNode Head = null;

public DoublyLinkedListNode Current = null;

public DoublyLinkedListNode Tail = null;

public int Count;

public DoublyLinkedList() { }

public void AddLastNode(int value)

{

DoublyLinkedListNode node = new DoublyLinkedListNode(value);

if (Head == null)

Head = node;

else

{

Tail.Next = node;

node.Prev = Tail;

}

Tail = node;

Count++;

}

public void Show()

{

Console.WriteLine("Вывод всех элементов двунаправленного цикличного линейного списка:");

DoublyLinkedListNode tmp = Head;

while (tmp != null)

{

Console.Write(tmp.Value + " ");

tmp = tmp.Next;

}

}

public DoublyLinkedListNode Search(int value)

{

DoublyLinkedListNode tmp = Head;

while (tmp != null)

{

if (tmp.Value == value)

return tmp;

tmp = tmp.Next;

}

return null;

}

public void RemoveNode(DoublyLinkedListNode target)

{

if (Head == target)

Head = Head.Next;

else

{

DoublyLinkedListNode tmp = Head;

while (tmp.Next != target)

tmp = tmp.Next;

tmp.Next = target.Next;

}

}

public void Clear()

{

Console.WriteLine("Очищение списка");

DoublyLinkedListNode temp;

while (Head != null)

{

temp = Head.Next;

Head = temp;

}

}

public void Sort(DoublyLinkedList doublyLinkedList)

{

DoublyLinkedListNode tmp = Head;

int localCount = 0;

bool isEven = false;

while (tmp != null)

{

isEven = localCount % 2 == 0;

if (isEven == false)

Console.WriteLine(tmp.Value);

localCount++;

tmp = tmp.Next;

}

}

}

}

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.

**Техническое задание №3**Линейные структуры данных на основе линейных списков.  
Вариант 6

Необходимо организовать линейную структуру, состоящую из 10 элементов, вводимых с клавиатуры.  
Организовать функции:

* просмотр всего списка элементов;
* очистка структуры.

Тип структуры и остальные функции реализовать в соответствии с вариантом.  
Организовать стек с функцией замены максимального элемента на минимальный

Листинг файла:

using System;

namespace Laba\_3

{

class Program

{

public static Deque deque = new Deque();

static void Main(string[] args)

{

Console.WriteLine("Введите числа:");

for (int i = 0; i < 10; i++)

{

Console.Write($"{i + 1}): ");

int number = Convert.ToInt32(Console.ReadLine());

deque.AddLastNode(number);

}

while (true)

{

Console.WriteLine("1) просмотр всех элементов дэка\n" +

"2) вставка элемента в начало\n" +

"3) вставка элемента в конец\n" +

"4) удаление элемента в начале\n" +

"5) удаление элемента в конце\n" +

"6) очистка дека\n" +

"7) заменить максимальный элемент на минимальный и отоброзить\n");

Console.Write("Введите значение: ");

int answer = Convert.ToInt32(Console.ReadLine());

Console.Clear();

switch (answer)

{

default:

Console.WriteLine("Такого варианта ответа нет!");

break;

case 1:

deque.Show();

break;

case 2:

Console.Write("Введите число: ");

deque.AddFirstNode(Convert.ToInt32(Console.ReadLine()));

Console.WriteLine("Число добавлено!\n");

break;

case 3:

Console.Write("Введите число: ");

deque.AddLastNode(Convert.ToInt32(Console.ReadLine()));

Console.WriteLine("Число добавлено!\n");

break;

case 4:

deque.RemoveFirstNode();

Console.WriteLine("Элемент удалён!");

break;

case 5:

deque.RemoveLastNode();

Console.WriteLine("Элемент удалён!");

break;

case 6:

deque.Clear();

Console.WriteLine("Очищен!");

break;

case 7:

deque.ReplaceMaxOnMin();

break;

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

namespace Laba\_3

{

class DequeNode

{

public int Value;

public DequeNode Prev { get; set; }

public DequeNode Next { get; set; }

public DequeNode(int value)

{

Value = value;

}

}

class Deque

{

DequeNode Head;

DequeNode Tail;

int Count;

public void AddFirstNode(int value)

{

DequeNode node = new DequeNode(value);

DequeNode tmp = Head;

node.Next = tmp;

Head = node;

if (Count == 0)

Tail = Head;

else

tmp.Prev = node;

Count++;

}

public void AddLastNode(int value)

{

DequeNode node = new DequeNode(value);

if (Head == null)

Head = node;

else

{

Tail.Next = node;

node.Prev = Tail;

}

Tail = node;

Count++;

}

public void RemoveFirstNode()

{

if (Count == 0)

Console.WriteLine("Дек пуст!");

int number = Head.Value;

if (Count == 1)

Head = Tail = null;

else

{

Head = Head.Next;

Head.Prev = null;

}

Count--;

}

public void RemoveLastNode()

{

if (Count == 0)

Console.WriteLine("Дек пуст!");

int number = Head.Value;

if (Count == 1)

Head = Tail = null;

else

{

Tail = Tail.Prev;

Tail.Next = null;

}

Count--;

}

public void Show()

{

Console.WriteLine("Вывод всех элементов однонаправленного списка:");

DequeNode tmp = Head;

while (tmp != null)

{

Console.Write(tmp.Value + " ");

tmp = tmp.Next;

}

Console.WriteLine();

}

public List<int> GetElement()

{

List<int> list = new List<int>();

DequeNode tmp = Head;

while (tmp != null)

{

list.Add(tmp.Value);

tmp = tmp.Next;

}

return list;

}

public void ReplaceMaxOnMin()

{

List<int> list = GetElement();

int max = list.Max();

int min = list.Min();

DequeNode tmp = Head;

while (tmp != null)

{

if (tmp.Value == max)

tmp.Value = min;

tmp = tmp.Next;

}

Show();

}

public void Clear()

{

while (Head != null)

RemoveLastNode();

}

}

}

Контрольные вопросы:

1. Стек динамическая реализация.  
 Ответ: Стек – это структура данных, в которой новый элемент всегда записывается в ее начало (вершину) и очередной читаемый элемент также всегда выбирается из ее начала. Здесь используется принцип «последним пришел – первым вышел».

2. Очередь динамическая организация.  
 Ответ: Очередь – это структура данных, представляющая собой последовательность элементов, образованная в порядке их поступления. Каждый новый элемент размещается в конце очереди; элемент, стоящий в начале очереди, выбирается из нее первым. Здесь используется принцип «первым пришел – первым вышел» (FIFO: First Input – First Output). Очередь можно реализовывать как динамическую структуру – в виде линейного списка.

3. Дек динамическая организация.  
 Ответ: Дек — это структура данных, представляющая собой последовательность элементов, в которой можно добавлять и удалять в произвольном порядке элементы с двух сторон. Первый и последний элементы дека соответствуют входу и выходу дека.

4. Стек функция добавления элемента.  
 Ответ: для такого списка достаточно хранить указатель вершины стека, который указывает на первый элемент списка. Если стек пуст, то списка не существует и указатель принимает значение NULL.

5. Стек функция извлечения элемента.  
 Ответ: крайний занесённый в список элемент выйдет оттуда первым.

6. Очередь функция постановки в очередь.  
 Ответ: необходимо хранить два указателя – один на начало списка (откуда извлекаем элементы) и один на конец списка (куда добавляем элементы). Если очередь пуста, то списка не существует и указатели принимают значение NULL.

7. Очередь функция извлечения из очереди.  
 Ответ: крайний занесённый в список элемент выйдет оттуда последним.

8. Дек функция добавления в начало.  
 Ответ: берётся первый элемент из дека, добавленный элемент ссылается на него как на следующий.

9. Дек функция извлечения из начала.  
 Ответ: если там лишь 1 элемент, то и голова, и хвост равны NULL, если больше, то головой становится второй элемент, а предыдущий от головы (от второго элемента, то есть первый) становится NULL.

10. Дек функция добавления в конец.  
 Ответ: берётся последний элемент из дека, добавленный элемент ссылается на него как на предыдущий.

11. Дек функция извлечения с конца.  
 Ответ: если там лишь 1 элемент, то и голова, и хвост равны NULL, если больше, то хвостом становится предпоследний элемент, а следующий от головы (от предпоследнего элемента, то есть последний) становится NULL.