**Отчет по лабораторной работе №1**

На тему «Структуры данных»

Вариант №17

Выполнил: ст. группы 2221121

Хайрутдинов А.А.

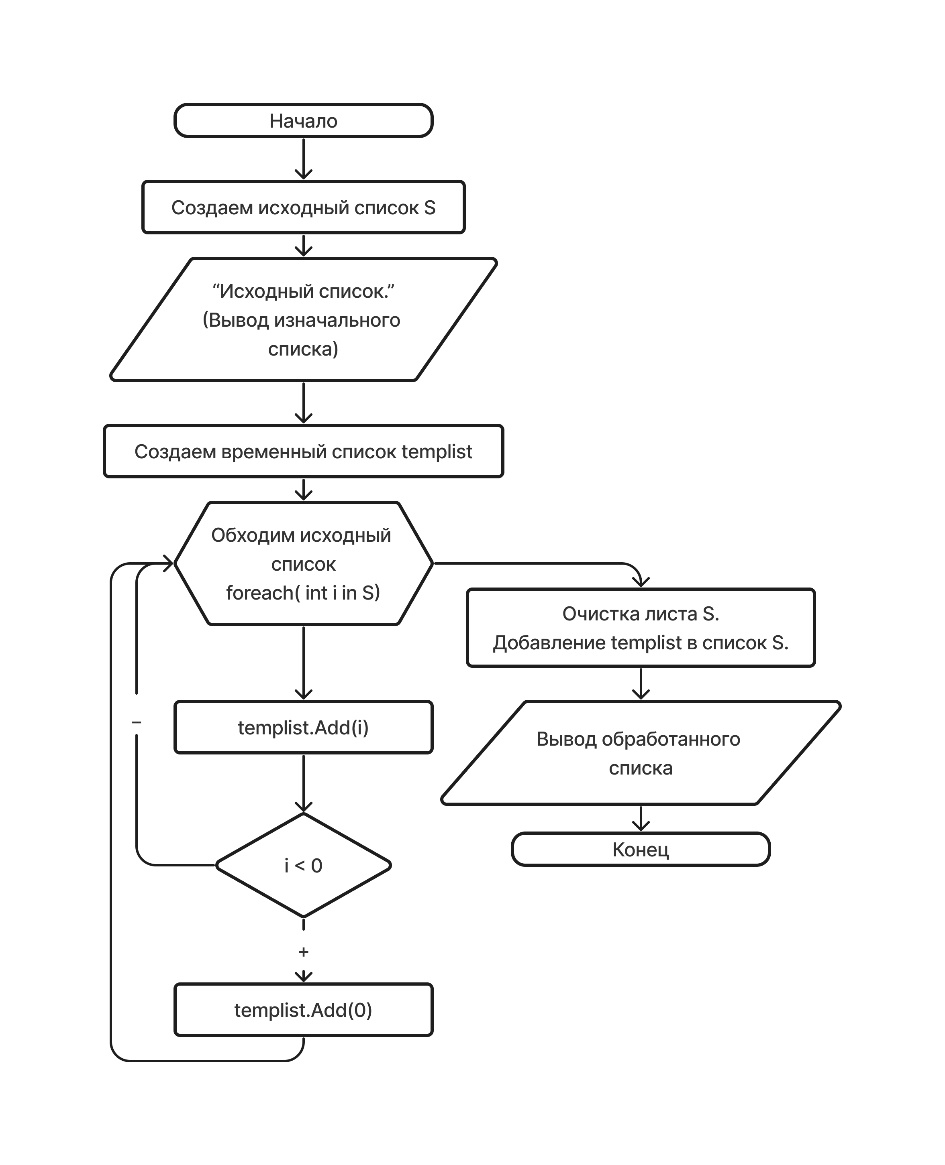
Проверил:

Хузятова Л.Б.

# Задание 1:

Составить программу для обработки очередей и стеков.Составить программу, создающуюсписок Sи выводящую его после некоторой обработки.S– цепочка целых чисел. Добавить после каждого отрицательного элемента новый нулевой элемент(Вар 5).

## Блок-схема:



# Код программы

static void Main()

{

// Создаем исходный список S

List<int> S = new List<int> { 1, -2, 3, -4, 5 };

Console.WriteLine("Исходный список S:");

PrintList(S);

// Обработка списка: добавление нового нулевого элемента после каждого отрицательного элемента

ProcessList(S);

// Вывод обработанного списка

Console.WriteLine("\nСписок S после обработки:");

PrintList(S);

Console.ReadLine();

}

static void ProcessList(List<int> list)

{

// Создаем временный список для хранения результатов обработки

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

// Обходим исходный список

foreach (int num in list)

{

// Добавляем текущий элемент в новый список

processedList.Add(num);

// Если текущий элемент отрицателен, добавляем новый нулевой элемент

if (num < 0)

{

processedList.Add(0);

}

}

// Очищаем исходный список и добавляем в него все элементы из обработанного списка

list.Clear();

list.AddRange(processedList);

}

static void PrintList(List<int> list)

{

foreach (int num in list)

{

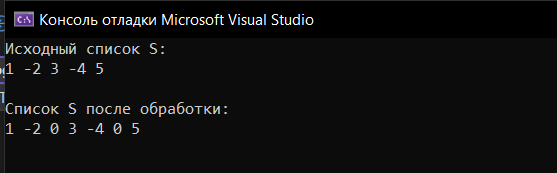
Console.Write(num + " ");

}

Console.WriteLine();

}

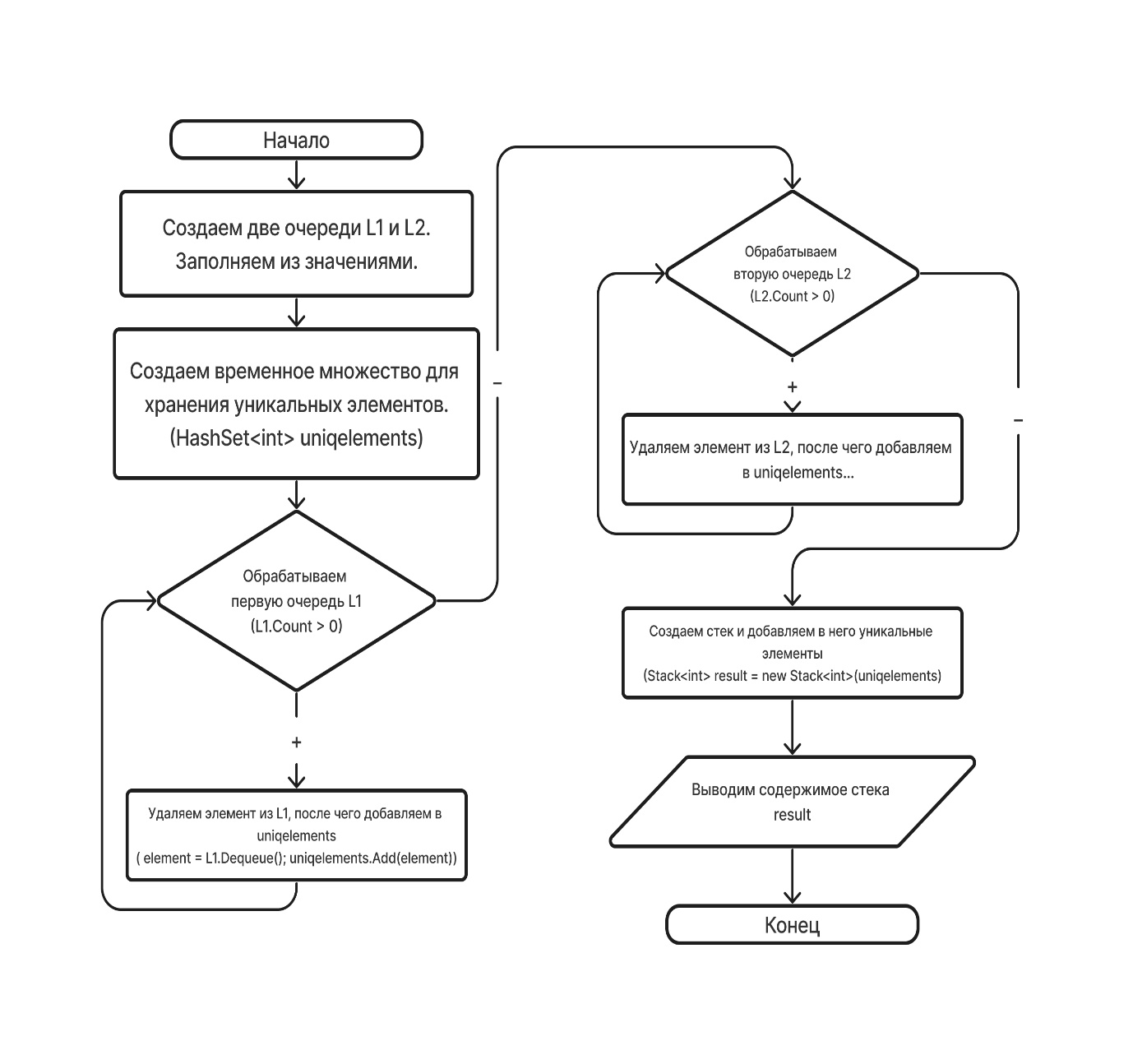
# Вывод:



# Задание 2:

Составить программу для обработки очередей и стеков.Составить программу, формирующую стек Sиз всех различных элементов очередей L1и L2.(Вар 10)

## Блок-схема:



# Код программы

class Program

{

static void Main()

{

// Создаем очереди L1 и L2

Queue<int> L1 = new Queue<int>();

Queue<int> L2 = new Queue<int>();

// Заполняем очереди произвольными значениями

L1.Enqueue(1);

L1.Enqueue(22);

L1.Enqueue(31);

L2.Enqueue(3);

L2.Enqueue(42);

L2.Enqueue(22);

// Формируем стек S из всех различных элементов очередей L1 и L2

Stack<int> S = MergeQueuesToStack(L1, L2);

// Выводим содержимое стека S

Console.WriteLine("Стек S из всех различных элементов очередей L1 и L2:");

PrintStack(S);

Console.ReadLine();

}

static Stack<int> MergeQueuesToStack(Queue<int> queue1, Queue<int> queue2)

{

// Создаем временное множество для хранения уникальных элементов

HashSet<int> uniqueElements = new HashSet<int>();

// Обрабатываем очередь L1

while (queue1.Count > 0)

{

int element = queue1.Dequeue();

uniqueElements.Add(element);

}

// Обрабатываем очередь L2

while (queue2.Count > 0)

{

int element = queue2.Dequeue();

uniqueElements.Add(element);

}

// Создаем стек и добавляем в него уникальные элементы

Stack<int> resultStack = new Stack<int>(uniqueElements);

return resultStack;

}

static void PrintStack(Stack<int> stack)

{

foreach (int element in stack)

{

Console.Write(element + " ");

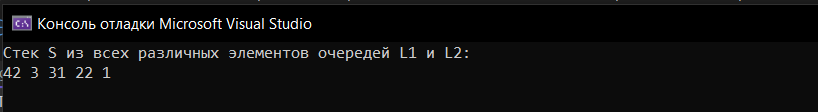
}

Console.WriteLine();

}

}

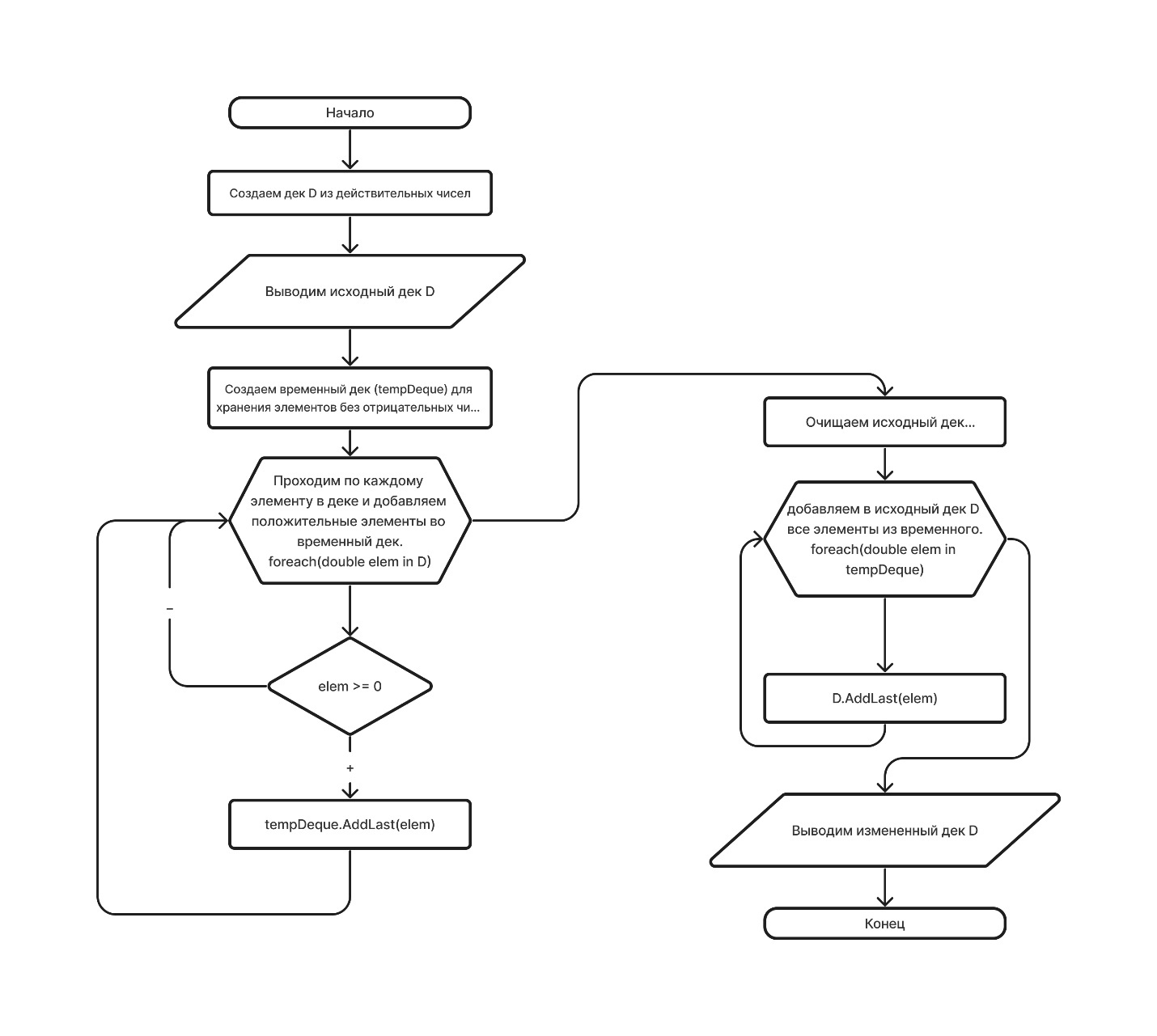
# Вывод:



# Задание 3:

Составить программу, создающую очередь с двухсторонним доступом (дек) D и выводящую его на экран после некоторой обработки. Создание и вывод дека на экран оформить в виде процедур для обработки очередей и стеков.8.D– список действительных чисел. Удалить все отрицательные элементы.(вар 8)

## Блок-схема:



# Код программы

class Program

{

static void Main()

{

// Создаем дек D из списка действительных чисел

LinkedList<double> D = new LinkedList<double>(new[] { 1.5, -2.0, 3.7, -4.2, 5.0 });

// Выводим исходный дек D

Console.WriteLine("Исходный дек D:");

PrintDeque(D);

// Удаляем отрицательные элементы из дека D

RemoveNegativeElements(D);

// Выводим измененный дек D

Console.WriteLine("\nДек D после удаления отрицательных элементов:");

PrintDeque(D);

Console.ReadLine();

}

static void RemoveNegativeElements(LinkedList<double> deque)

{

// Создаем временный список для хранения элементов без отрицательных чисел

LinkedList<double> tempDeque = new LinkedList<double>();

// Проходим по каждому элементу в деке и добавляем положительные элементы во временный дек

foreach (double element in deque)

{

if (element >= 0)

{

tempDeque.AddLast(element);

}

}

// Очищаем исходный дек

deque.Clear();

// Добавляем в исходный дек все элементы из временного дека

foreach (double element in tempDeque)

{

deque.AddLast(element);

}

}

static void PrintDeque(LinkedList<double> deque)

{

foreach (double element in deque)

{

Console.Write(element + " ");

}

Console.WriteLine();

}

}

# Вывод:

