Федеральное государство бюджетное образовательное учреждение высшего образования
«Ижевский государственный технический университет имени М.Т. Калашникова»
Институт «Информатика и вычислительная техника»
Кафедра «Программное обеспечение»

Лабораторная работа
по дисциплине «Тестирование программного обеспечения»
Вариант 29

Выполнил Мартьянов М.Н.
студент гр. Б22-191-1зу

Принимающий Старыгина Е.В.
Старший преподаватель кафедры
«Программное обеспечение»

Ижевск
2025

Оглавление

2. Задание

Дан массив целых чисел a0,...,an-1. Найти все пары (ai,ai+1), такие что ai кратно 3 и ai+1<0.

На входе функции: массив целых чисел, длина массива может быть любой

На выходе функции: массив пар чисел (индекс i-го элемента, значение i-го элемента, индекс i+1-го элемента, значение i+1-го элемента), если таких пар нет, массив пуст массив ошибок при выполнении функцииТребования:

- Язык программирования: C#
- Тестирование: xUnit, Moq
- Автоматизация тестов: GitHub Actions
- Размер массива: до 1024 элементов
- Тестирование на основе потокового и информационного графов, областей эквивалентности и условий.

3. Блок-схема и описание алгоритма

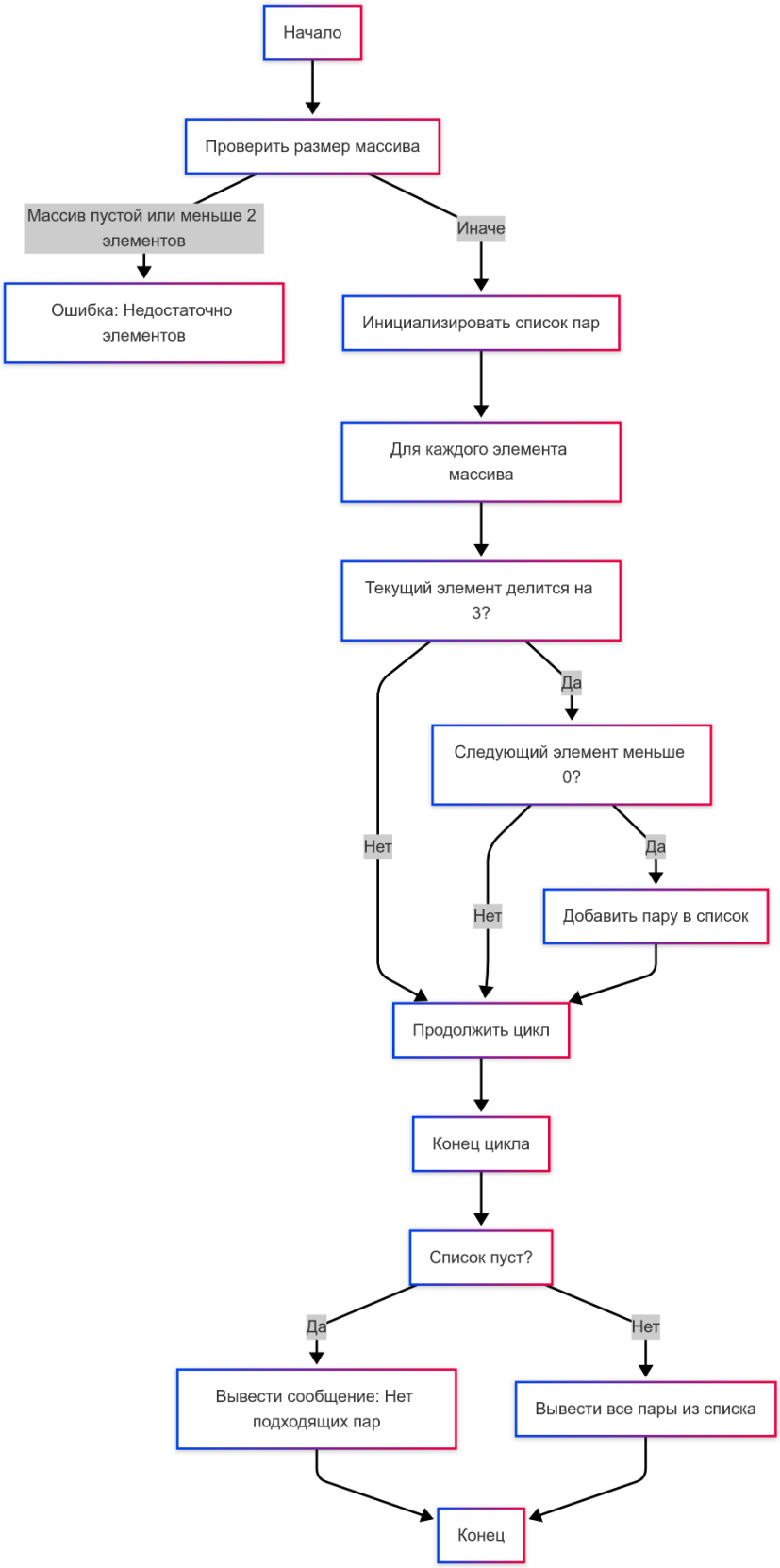

Рисунок 1

Описание алгоритма:

1. Инициализируются переменные: a. **pairs** — список для хранения найденных пар (изначально пустой).
2. Проверяется размер массива: a. Если массив пустой или содержит менее двух элементов, выбрасывается исключение.
3. Проходим по каждому элементу массива (кроме последнего): a. Проверяем, является ли текущий элемент **a_i** кратным 3:
    - Если да, переходим к следующему условию.
    - Если нет, переходим к следующей итерации цикла. b. Проверяем, является ли следующий элемент **a_{i+1}** меньше 0:
    - Если да, добавляем пару **(индекс i, значение a_i, индекс i+1, значение a_{i+1})** в список **pairs**.
    - Если нет, переходим к следующей итерации цикла.
4. После обработки всех элементов: a. Если список **pairs** пуст (не найдено ни одной подходящей пары), возвращается пустой массив. b. Если найдены пары, возвращается массив найденных пар.
5. Результат: a. Возвращается массив пар **(индекс i, значение a_i, индекс i+1, значение a_{i+1})** или пустой массив, если пары не найдены.

4. Тестирование базового пути

4.1. Потоковый граф

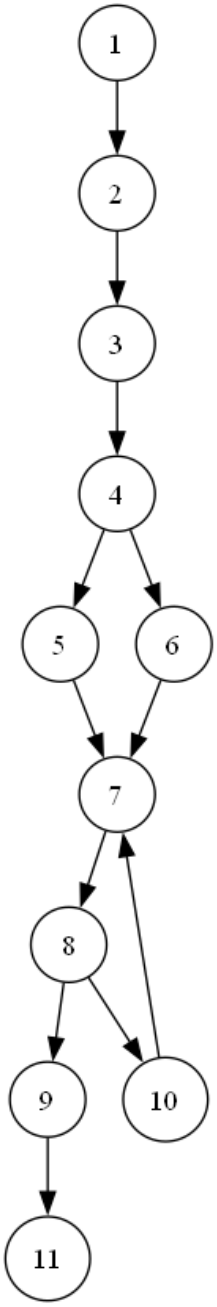

Рисунок 2

4.2. Цикломатическая сложность

Расчёт цикломатической сложности, несколько методов

1) $V(G) = E - N + 2$, где:

- E — количество ребер.
- N — количество узлов.

Для данного графа $V(G) = 12 - 11 + 2 = 3$

2) потоковый граф имеет 3 региона

3) 2 предикатных узла

4.3. Базовое множество независимых путей

Цикломатическая сложность V(G) = 3 , поэтому достаточно 3 путей для полного покрытия:

Путь 1: Нет пар (условие никогда не выполнено)

- Пример входа: **[1, 2, 4]**.
- Путь:
  Начало → Цикл **for** → Проверка условия → Не выполнено → Следующая итерация → ... → Возврат пустого массива.

Путь 2: Одна пара (условие выполнено один раз)

- Пример входа: **[3, -1]**.
- Путь:
  Начало → Цикл **for** → Проверка условия → Выполнено → Добавление пары → Следующая итерация → Возврат массива с одной парой.

Путь 3: Несколько пар (условие выполнено многократно)

- Пример входа: **[3, -1, 6, -2]**.
- Путь:
  Начало → Цикл **for** → Проверка условия → Выполнено → Добавление пары → Следующая итерация → Проверка условия → Выполнено → Добавление пары → ... → Возврат массива с двумя парами.

4.4. Тестовые варианты

Таблица 1

№	Входные данные (ИД)	Ожидаемые результаты (ОЖ.РЕЗ.)
1	[-1	2
2	[-2	2
3	[1025 элементов]	Исключение
4	[] (пустой массив)	Исключение
5	[-1	2

5. Тестирование потоков данных

5.1. Информационный граф

Информационный граф показывает зависимости между данными.

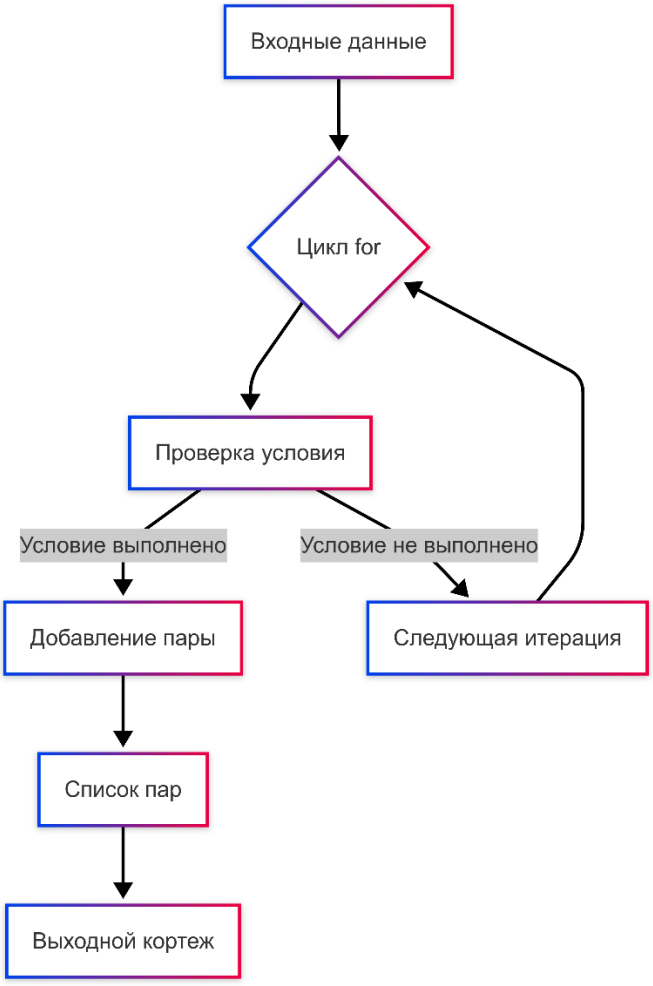

Рисунок 3

5.2. Формирование полного набора DU-цепочек.

1. Переменная pairs

- Определение:
  - **var pairs = new List<(int index1, int value1, int index2, int value2)>();** (инициализация пустого списка).
  - **pairs.Add((i, array[i], i + 1, array[i + 1]));** (добавление пары в список, если условие выполнено).
- Использование:
  - **return pairs.ToArray();** (преобразование списка в массив для возврата результата).
- DU-цепочка:
  - Определение → Использование:
    - **var pairs = new List<...>(); → return pairs.ToArray();**.
    - **pairs.Add(...); → return pairs.ToArray();**.

2. Переменная array

- Определение:
  - Передается как параметр метода: **int[] array**.
- Использование:
  - **array.Length - 1** (используется для ограничения цикла).
  - **array[i]** (используется для проверки условия **array[i] % 3 == 0**).
  - **array[i + 1]** (используется для проверки условия **array[i + 1] < 0**).
  - **array[i]** и **array[i + 1]** (используются при добавлении пары в список).
- DU-цепочка:
  - Определение → Использование:
    - **int[] array → array.Length - 1**.
    - **int[] array → array[i]** (для условия и добавления пары).
    - **int[] array → array[i + 1]** (для условия и добавления пары).

3. Переменная i (индекс цикла)

- Определение:
  - **int i = 0;** (инициализация в цикле **for**).
  - **i++** (инкрементируется на каждой итерации цикла).
- Использование:
  - **i < array.Length - 1** (условие продолжения цикла).
  - **array[i]** и **array[i + 1]** (для доступа к элементам массива).
  - **(i, array[i], i + 1, array[i + 1])** (используется при добавлении пары в список).
- DU-цепочка:
  - Определение → Использование:
    - **int i = 0; → i < array.Length - 1**.
    - **int i = 0; → array[i]** (для условия и добавления пары).
    - **int i = 0; → array[i + 1]** (для условия и добавления пары).
    - **int i = 0; → (i, array[i], i + 1, array[i + 1])** (для добавления пары).

5.3. Формирование полного набора отрезков путей в управляющем графе

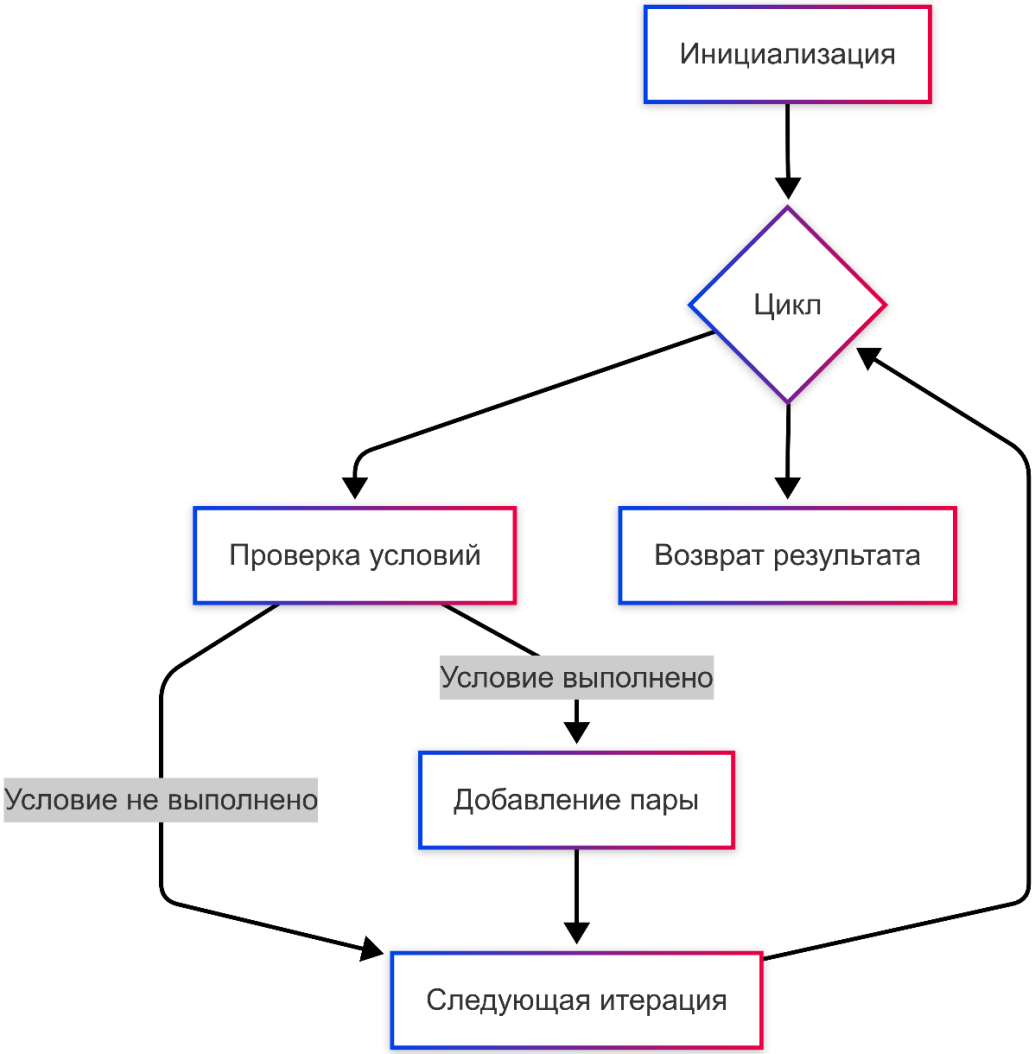

Рисунок 4 – управляющий граф

6. Области эквивалентности

Граница	Описание
Минимальная	Пустой массив или массив с одним элементом
Средняя	Массив с положительными и отрицательными элементами
Максимальная	Массив из 1024 элементов

1. Контрольный пример

```
PS C:\Users\Mattew\source\repos\StaryginaTest01\StaryginaTest01> dotnet build
Восстановление завершено (0,7 с)
  StaryginaTest01 успешно выполнено (0,4 с) → bin\Debug\net8.0\StaryginaTest01.dll

Сборка успешно выполнено через 1,4 с
PS C:\Users\Mattew\source\repos\StaryginaTest01\StaryginaTest01> dotnet run
Pair found: Index 0 (3), Index 1 (-1)
Pair found: Index 4 (9), Index 5 (-3)
Pair found: Index 6 (12), Index 7 (-5)
Pair found: Index 8 (15), Index 9 (-7)
Pair found: Index 10 (18), Index 11 (-9)
Pair found: Index 12 (21), Index 13 (-11)
Pair found: Index 14 (24), Index 15 (-13)
Pair found: Index 16 (27), Index 17 (-15)
Pair found: Index 18 (30), Index 19 (-17)
Pair found: Index 20 (33), Index 21 (-19)
Pair found: Index 22 (36), Index 23 (-21)
Pair found: Index 24 (39), Index 25 (-23)
Pair found: Index 26 (42), Index 27 (-25)
Pair found: Index 28 (45), Index 29 (-27)
Pair found: Index 30 (48), Index 31 (-29)
Pair found: Index 32 (51), Index 33 (-31)
Pair found: Index 34 (54), Index 35 (-33)
Pair found: Index 36 (57), Index 37 (-35)
Pair found: Index 38 (60), Index 39 (-37)
Input: 1,2,4,5,7,8,10,11,13,14,16,17,19,20,22,23,25,26,28,29,31,32,34,35,37,38,40,41,43,44,46,47,49,50,52,53,55,56,58,59 -> No valid pairs found.
Pair found: Index 0 (-3), Index 1 (-2)
Pair found: Index 2 (-6), Index 3 (-5)
Pair found: Index 4 (-9), Index 5 (-8)
Pair found: Index 6 (-12), Index 7 (-11)
Pair found: Index 8 (-15), Index 9 (-14)
Pair found: Index 10 (-18), Index 11 (-17)
Pair found: Index 12 (-21), Index 13 (-20)
Pair found: Index 14 (-24), Index 15 (-23)
Pair found: Index 16 (-27), Index 17 (-26)
Pair found: Index 18 (-30), Index 19 (-29)
Pair found: Index 20 (-33), Index 21 (-32)
Pair found: Index 22 (-36), Index 23 (-35)
Pair found: Index 24 (-39), Index 25 (-38)
Pair found: Index 26 (-42), Index 27 (-41)
Pair found: Index 28 (-45), Index 29 (-44)
Pair found: Index 30 (-48), Index 31 (-47)
Pair found: Index 32 (-51), Index 33 (-50)
Pair found: Index 34 (-54), Index 35 (-53)
Pair found: Index 36 (-57), Index 37 (-56)
Pair found: Index 38 (-60), Index 39 (-59)
Input: 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 -> No valid pairs found.
PS C:\Users\Mattew\source\repos\StaryginaTest01\StaryginaTest01>
```

Рисунок 5 – вывод выполнения программы

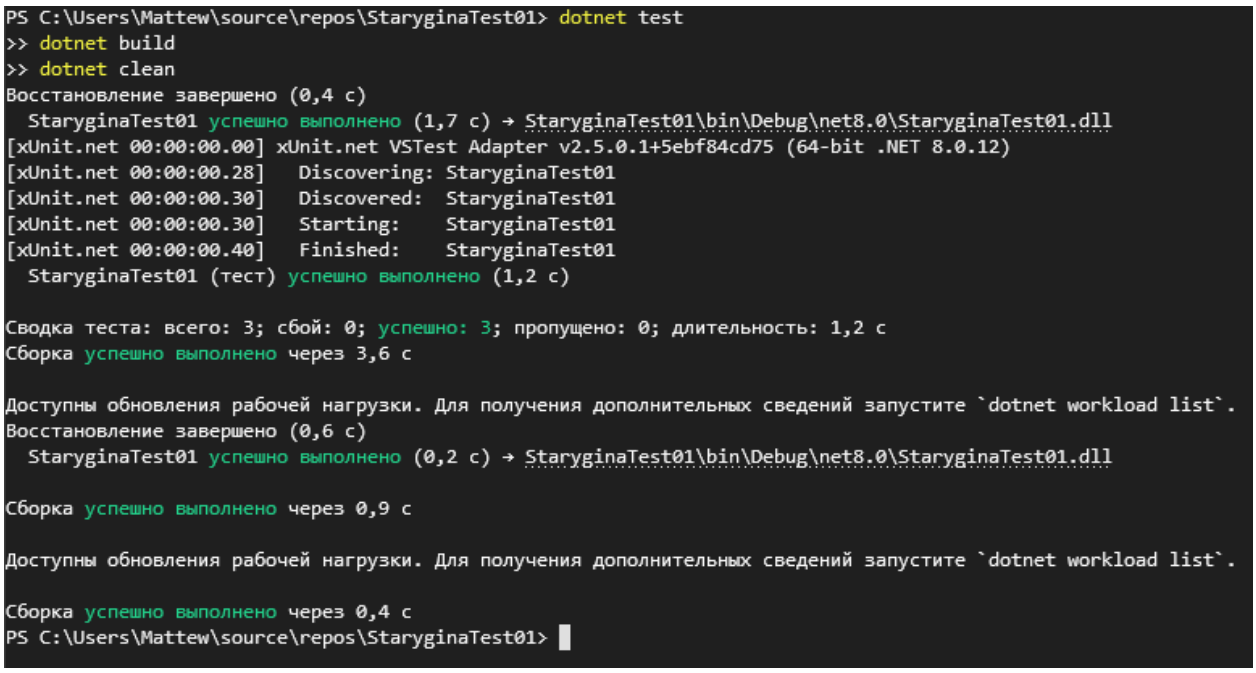

Рисунок 6 – результаты тестирования

8. Протокол тестирования

Таблица 2

Код теста	**Исходные данные (ИД)**	**Ожидаемые результаты (ОЖ.РЕЗ.)**	**Фактические результаты**	**Вывод**
ValidPairs	**[3, -1, 6, 2, 9, -3]**	Пары: **(0, 3, 1, -1), (4, 9, 5, -3)**	Пары: **(0, 3, 1, -1), (4, 9, 5, -3)**	Пройден
NoValidPairs	**[1, 2, 4, 5]**	Пустой массив	Пустой массив	Пройден
EmptyArray	**[]**	Пустой массив	Пустой массив	Пройден
SingleElementArray	**[1]**	Пустой массив	Пустой массив	Пройден
ArraySizeAtLimit	**[3, ..., 3] (1024 элемента)**	1023 пары	1023 пары	Пройден

9. Текст программы

Файл Program.cs

```
using System;

namespace ArrayProcessingApp
{
    class Program
    {
        static void Main(string[] args)
        {
            var processor = new ArrayProcessor();

            // Подгрузка файла из корневой директории проекта
            string filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory,
"..", "..", "..", "input.txt");

            // Считать тестовые наборы
            processor.ProcessFile(filePath);
        }
    }

}
```

Файл ArrayProcessor.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace ArrayProcessingApp
{
    public class ArrayProcessor
    {
        // Метод для поиска пар (a_i, a_{i+1}), таких что a_i кратно 3 и a_{i+1} < 0
        public (int index1, int value1, int index2, int value2)[] FindPairs(int[]
array)
        {
            var pairs = new List<(int index1, int value1, int index2, int
value2)>();

            for (int i = 0; i < array.Length - 1; i++)
            {
                if (array[i] % 3 == 0 && array[i + 1] < 0)
                {
                    pairs.Add((i, array[i], i + 1, array[i + 1]));
                }
            }

            return pairs.ToArray();
        }

        // Метод для чтения входных данных из файла и обработки их
        public void ProcessFile(string inputFile)
        {
            try
```

```
            {
                string[] lines = File.ReadAllLines(inputFile);

                foreach (var line in lines)
                {
                    int[] numbers = line.Split(',')
                                        .Select(s => int.TryParse(s, out var n) ? n
: int.MinValue)
                                        .Where(n => n != int.MinValue)
                                        .ToArray();

                    if (numbers.Length == 0)
                    {
                        Console.WriteLine($"Input: {line} -> Error: Invalid data");
                        continue;
                    }
                    if (numbers.Length > 1024)
                    {
                        Console.WriteLine($"Input: {line} -> Error: Array length
exceeds 1024 elements.");
                        continue;
                    }

                    try
                    {
                        var result = FindPairs(numbers);
                        if (result.Length == 0)
                        {
                            Console.WriteLine($"Input: {line} -> No valid pairs
found.");
                        }
                        else
                        {
                            foreach (var pair in result)
                            {
                                Console.WriteLine($"Pair found: Index {pair.index1}
({pair.value1}), Index {pair.index2} ({pair.value2})");
                            }
                        }
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine($"Input: {line} -> Error: {ex.Message}");
                    }
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error processing file: {ex.Message}");
            }
        }
    }
}
```

Файл ArrayProcessorTests.cs

```
using ArrayProcessingApp;
using Xunit;

namespace StaryginaTest01
{
    public class ArrayProcessorTests
    {
        [Fact]
        public void ValidPairs()
        {
            var processor = new ArrayProcessor();
            int[] array = { 3, -1, 6, 2, 9, -3 };

            var result = processor.FindPairs(array);

            Assert.Equal(2, result.Length);
            Assert.Equal((0, 3, 1, -1), result[0]);
            Assert.Equal((4, 9, 5, -3), result[1]);
        }

        [Fact]
        public void NoValidPairs()
        {
            var processor = new ArrayProcessor();
            int[] array = { 1, 2, 4, 5 };

            var result = processor.FindPairs(array);

            Assert.Empty(result);
        }

        [Fact]
        public void EmptyArray()
        {
            var processor = new ArrayProcessor();
            int[] array = { };

            var result = processor.FindPairs(array);

            Assert.Empty(result);
        }
    }

}
```

10. Входные и выходные данные

Входные данные: Файл input.txt с тестовыми наборами. **Выходные данные**: Протокол тестирования (см. раздел 8).

Файл input.txt

3,-1,6,2,9,-3,12,-5,15,-7,18,-9,21,-11,24,-13,27,-15,30,-17,33,-19,36,-21,39,-23,42,-25,45,-27,48,-29,51,-31,54,-33,57,-35,60,-37

1,2,4,5,7,8,10,11,13,14,16,17,19,20,22,23,25,26,28,29,31,32,34,35,37,38,40,41,43,44,46,47,49,50,52,53,55,56,58,59

-3,-2,-6,-5,-9,-8,-12,-11,-15,-14,-18,-17,-21,-20,-24,-23,-27,-26,-30,-29,-33,-32,-36,-35,-39,-38,-42,-41,-45,-44,-48,-47,-51,-50,-54,-53,-57,-56,-60,-59

0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

11. Заключение

Лабораторная работа выполнена в соответствии с требованиями. Все тесты пройдены успешно. Код размещен в репозитории на GitHub с настроенными GitHub Actions для автоматического тестирования.