Федеральное государственное бюджетное образовательное учреждение высшего образования
«Сибирский государственный университет телекоммуникаций и информатики»
(СибГУТИ)

Институт информатики и вычислительной техники

09.03.01 "Информатика и вычислительная техника"
профиль "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Кафедра прикладной математики и кибернетики

Современные технологии программирования

Лабораторная работа №3

Модульное тестирование программ на языке C++ средствами Visual Studio

Вариант 1

Выполнил:

студент гр.ИП-213 Дмитриев Антон Александрович
ФИО студента

«__» _____ 2025 г.

Проверил:

Преподаватель

ФИО преподавателя

«__» _____ 2025 г. Оценка_____

Новосибирск 2025 г.

1. **Задание**

Разработайте на языке C++ класс, содержащий набор функций в соответствии с вариантом задания.

Разработайте тестовые наборы данных по критерию C2 для тестирования функций класса.

Протестировать функции с помощью средств автоматизации модульного тестирования Visual Studio.

Провести анализ выполненного теста и, если необходимо отладку кода.

Написать отчёт о результатах проделанной работы.

Вариант 1

Функция упорядочивает значения переменных x,y,z в порядке убывания их значений, так чтобы x>=y>=z.

Функция получает два положительных целых числа a и b. Вычисляет и возвращает их наибольший общий делитель.

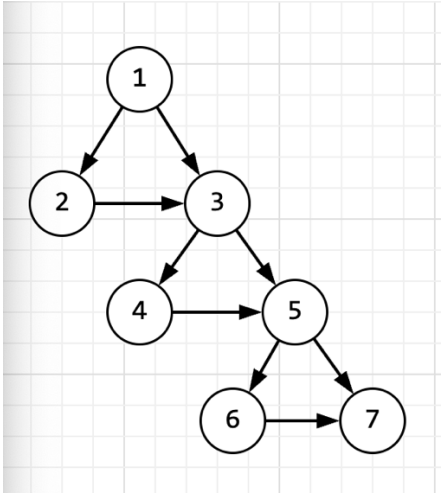
Функция получает целое число a. Формирует и возвращает целое число b, составленное из разрядов целого числа a с чётными значениями. Например: a = 12345, b = 24.

Функция получает двумерный массив целых переменных A. Отыскивает и возвращает сумму нечётных значений компонентов массива, лежащих выше главной диагонали.

2. УГП и тестовые наборы данных для тестирования функций класса

1. Метод SortDescending(ref int x, ref int y, ref int z)

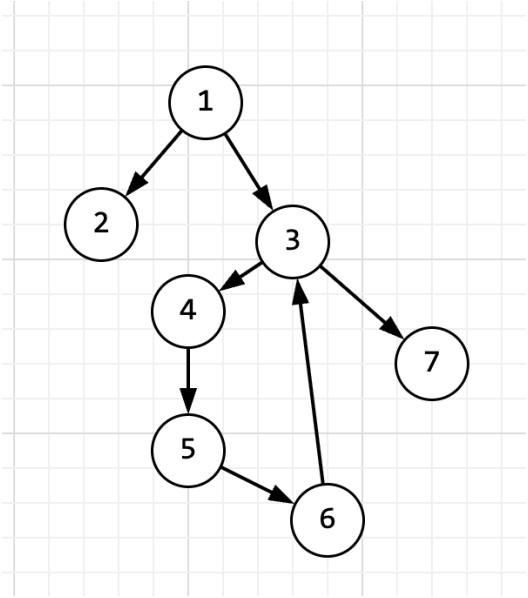
```
1  if (x < y)
2      Swap(ref x, ref y);
3  if (x < z)
4      Swap(ref x, ref z);
5  if (y < z)
6      Swap(ref y, ref z);
7  // Конец метода
```



№	Входные данные (x, y, z)	Ожидаемый результат (x, y, z)	Пройденный путь / Условия
1	(3, 2, 1)	(3, 2, 1)	Все условия `false` (уже отсортировано)
2	(1, 2, 3)	(3, 2, 1)	Все условия `true` (полная перестановка)
3	(2, 3, 1)	(3, 2, 1)	Только `x<y` = true
4	(2, 3, 4)	(4, 3, 2)	Только `x<z` = true
5	(3, 1, 2)	(3, 2, 1)	Только `y<z` = true

2. Метод GCD(int a, int b)

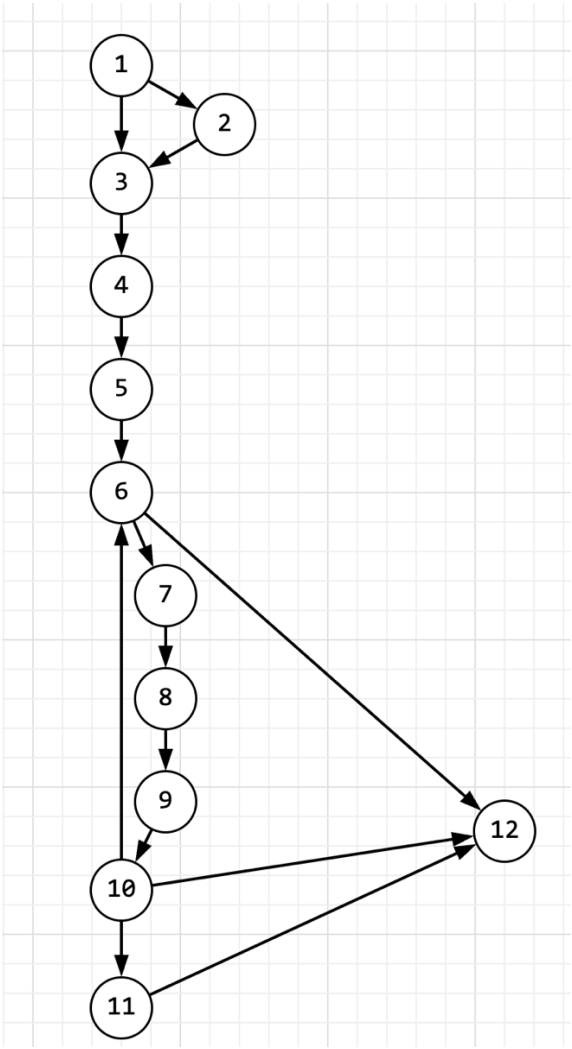
```
1  if (a <= 0 || b <= 0)
2      throw new ArgumentException("Числа должны быть положительными");
3  while (b != 0) {
4      int temp = b;
5      b = a % b;
6      a = temp;
   }
7  return a;
```



№	Входные данные (a, b)	Ожидаемый результат	Пройденный путь / Особенность
1	(0, 5)	Исключение ArgumentException	Проверка входных значений (неположительные числа)
2	(7, 7)	7	Без итераций, сразу b == 0
3	(8, 6)	2	1 итерация цикла
4	(1071, 462)	21	2 итерации цикла
5	(17, 13)	1	Несколько итераций (взаимно простые числа)

3. Метод GetEvenDigits(int number)

```
1  if (a < 0)
2      a = -a;
3  int result = 0;
4  int position = 0;
5  int temp = a;
6  while (temp > 0) {
7      int digit = temp % 10;
8      temp /= 10;
9      position++;
10     if (position % 2 == 0)
11         result = result * 10 + digit;
   }
12 return ReverseNumber(result);
```

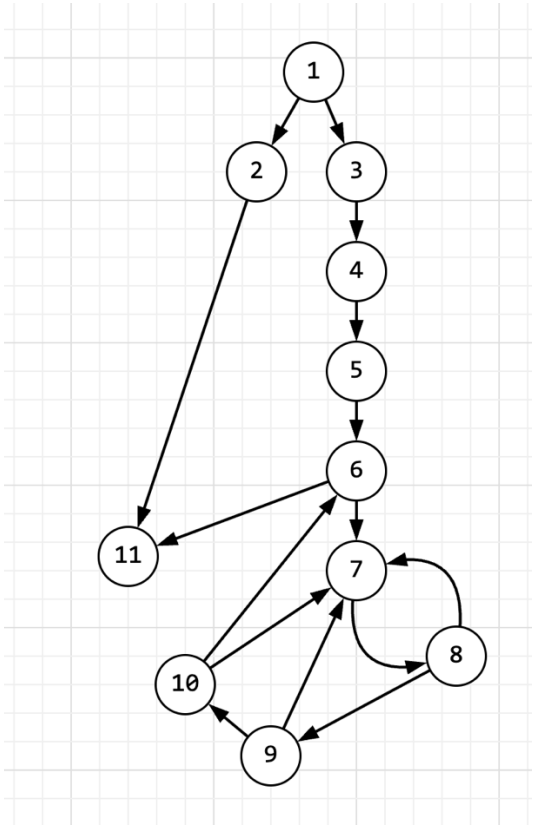


№	Входные данные	Ожидаемый результат	Пройденный путь / Особенность
1	5	0	1 цифра, нечетная позиция
2	45	4	Четная позиция — только вторая цифра
3	1234	13	Цифры на 2-й и 4-й позициях: (2 и 4) → результат 13
4	1050	15	Нули пропущены, остаются 1 и 5
5	-1234	13	Отрицательное число → берется абсолютное значение

4. Метод SumOddAboveMainDiagonal(int[,] array)

```
1  if (A == null)
2      throw new ArgumentNullException(nameof(A));
3  int sum = 0;
4  int rows = A.GetLength(0);
```

```
5   int cols = A.GetLength(1);
6   for (int i = 0; i < rows; i++) {
7       for (int j = 0; j < cols; j++) {
8           if (j > i)
9               if (A[i, j] % 2 != 0)
10                  sum += A[i, j];
        }
    }
11  return sum;
```



№	Входные данные (матрица)	Ожидаемый результат	Пройденный путь / Особенность
1	null	Исключение ArgumentNullException	Проверка на null
2	[[5]]	0	1×1 матрица, нет элементов выше диагонали
3	[[1, 3], [5, 7]]	3	Один элемент выше диагонали — нечетный
4	[[1, 2, 3], [4, 5, 6], [7, 8, 9]]	3	Только 3 — нечетное выше диагонали
5	[[1, 5, 3], [4, 5, 6], [7, 8, 9]]	8	Два нечетных выше диагонали: 5 + 3

№	Входные данные (матрица)	Ожидаемый результат	Пройденный путь / Особенность
6	[[2, 4, 6], [8, 10, 12], [14, 16, 18]]	0	Все элементы четные
7	[[1, 3, 5, 7], [9, 11, 13, 15]]	43	Неквадратный массив (3 + 5 + 7 + 13 + 15)

3. Исходные тексты программ на языке C#.

Class.cs:

```
using System;

namespace PathCoverage
{
   public static class PathOperations
   {
      // 1. Упорядочивание x, y, z в порядке убывания
      public static void SortDescending(ref int x, ref int y, ref int z)
      {
         if (x < y)
         {
            Swap(ref x, ref y);
         }
         if (x < z)
         {
            Swap(ref x, ref z);
         }
         if (y < z)
         {
            Swap(ref y, ref z);
         }
      }

      private static void Swap(ref int a, ref int b)
      {
         int temp = a;
         a = b;
         b = temp;
      }

      // 2. Наибольший общий делитель (алгоритм Евклида)
      public static int GCD(int a, int b)
      {
         if (a <= 0 || b <= 0)
            throw new ArgumentException("Числа должны быть положительными");

         while (b != 0)
         {
            int temp = b;
            b = a % b;
            a = temp;
         }
```

```
    return a;
}

// 3. Формирование числа из четных разрядов
public static int GetEvenDigits(int a)
{
    if (a < 0) a = -a; // Работаем с модулем

    int result = 0;
    int position = 0;
    int temp = a;

    while (temp > 0)
    {
        int digit = temp % 10;
        temp /= 10;

        position++;
        if (position % 2 == 0) // Четная позиция (считая с младшего разряда)
        {
            result = result * 10 + digit;
        }
    }

    // Переворачиваем результат, так как собирали с младших разрядов
    return ReverseNumber(result);
}

private static int ReverseNumber(int n)
{
    int reversed = 0;
    while (n > 0)
    {
        reversed = reversed * 10 + n % 10;
        n /= 10;
    }
    return reversed;
}

// 4. Сумма нечетных значений выше главной диагонали
public static int SumOddAboveMainDiagonal(int[,] A)
{
    if (A == null)
        throw new ArgumentNullException(nameof(A));
```

```
            int sum = 0;
            int rows = A.GetLength(0);
            int cols = A.GetLength(1);

            for (int i = 0; i < rows; i++)
            {
                for (int j = 0; j < cols; j++)
                {
                    if (j > i) // Выше главной диагонали
                    {
                        if (A[i, j] % 2 != 0) // Нечетное значение
                        {
                            sum += A[i, j];
                        }
                    }
                }
            }
            return sum;
        }
    }
}
```

Tests.cs:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace PathCoverage.Tests
{
    [TestClass]
    public class PathOperationsC2Tests
    {
        // ==================== SortDescending ====================

        [TestMethod]
        public void SortDescending_AlreadySorted_NoChanges()
        {
            // Путь: все условия false
            int x = 3, y = 2, z = 1;
            int expectedX = 3, expectedY = 2, expectedZ = 1;

            PathOperations.SortDescending(ref x, ref y, ref z);

            Assert.AreEqual(expectedX, x);
            Assert.AreEqual(expectedY, y);
            Assert.AreEqual(expectedZ, z);
```

```
}

[TestMethod]
public void SortDescending_ReverseOrder_FullSwap()
{
    // Путь: все условия true
    int x = 1, y = 2, z = 3;
    int expectedX = 3, expectedY = 2, expectedZ = 1;

    PathOperations.SortDescending(ref x, ref y, ref z);

    Assert.AreEqual(expectedX, x);
    Assert.AreEqual(expectedY, y);
    Assert.AreEqual(expectedZ, z);
}

[TestMethod]
public void SortDescending_OnlyFirstSwap_FirstConditionTrue()
{
    // Путь: x<y=true, x<z=false, y<z=false
    int x = 2, y = 3, z = 1;
    int expectedX = 3, expectedY = 2, expectedZ = 1;

    PathOperations.SortDescending(ref x, ref y, ref z);

    Assert.AreEqual(expectedX, x);
    Assert.AreEqual(expectedY, y);
    Assert.AreEqual(expectedZ, z);
}

[TestMethod]
public void SortDescending_OnlySecondSwap_SecondConditionTrue()
{
    // Путь: x<y=false, x<z=true, y<z=false
    // Нужно: после обмена x и z, чтобы y >= z
    int x = 2, y = 3, z = 4;
    int expectedX = 4, expectedY = 3, expectedZ = 2;

    PathOperations.SortDescending(ref x, ref y, ref z);

    Assert.AreEqual(expectedX, x);
    Assert.AreEqual(expectedY, y);
    Assert.AreEqual(expectedZ, z);
}
```

```
[TestMethod]
public void SortDescending_OnlyThirdSwap_ThirdConditionTrue()
{
    // Путь: x<y=false, x<z=false, y<z=true
    int x = 3, y = 1, z = 2;
    int expectedX = 3, expectedY = 2, expectedZ = 1;

    PathOperations.SortDescending(ref x, ref y, ref z);

    Assert.AreEqual(expectedX, x);
    Assert.AreEqual(expectedY, y);
    Assert.AreEqual(expectedZ, z);
}

// ==================== GCD ====================

[TestMethod]
[ExpectedException(typeof(ArgumentException))]
public void GCD_NonPositiveNumbers_ThrowsException()
{
    // Путь: исключение без входа в цикл
    PathOperations.GCD(0, 5);
}

[TestMethod]
public void GCD_EqualNumbers_ReturnsSameNumber()
{
    // Путь: цикл не выполняется (b=0 сразу)
    int result = PathOperations.GCD(7, 7);
    Assert.AreEqual(7, result);
}

[TestMethod]
public void GCD_OneIteration_SimpleCase()
{
    // Путь: цикл выполняется 1 раз
    int result = PathOperations.GCD(8, 6);
    Assert.AreEqual(2, result);
}

[TestMethod]
public void GCD_TwoIterations_ComplexCase()
{
    // Путь: цикл выполняется 2 раза
    int result = PathOperations.GCD(1071, 462); // НОД = 21
```

```
    Assert.AreEqual(21, result);
}

[TestMethod]
public void GCD_PrimeNumbers_ReturnsOne()
{
    // Путь: цикл выполняется несколько раз до b=1
    int result = PathOperations.GCD(17, 13);
    Assert.AreEqual(1, result);
}

// ==================== GetEvenDigits ====================

[TestMethod]
public void GetEvenDigits_SingleDigit_ReturnsZero()
{
    // Путь: цикл выполняется 1 раз, позиция нечетная
    int result = PathOperations.GetEvenDigits(5);
    Assert.AreEqual(0, result);
}

[TestMethod]
public void GetEvenDigits_TwoDigits_ReturnsSecondDigit()
{
    // Путь: цикл выполняется 2 раза
    // 1 итерация: позиция=1 (нечетная) - пропускаем
    // 2 итерация: позиция=2 (четная) - добавляем
    int result = PathOperations.GetEvenDigits(45); // Должно вернуть 4
    Assert.AreEqual(4, result);
}

[TestMethod]
public void GetEvenDigits_FourDigits_ReturnsEvenPositions()
{
    // Путь: цикл выполняется 4 раза
    // Позиции: 1(пропуск), 2(добавить), 3(пропуск), 4(добавить)
    int result = PathOperations.GetEvenDigits(1234); // Должно вернуть 13
    Assert.AreEqual(13, result);
}

[TestMethod]
public void GetEvenDigits_AllEvenPositionsZero_ReturnsZero()
{
    // Путь: цикл с добавлением нулей
    int result = PathOperations.GetEvenDigits(1050); // Должно вернуть 00 → 0
```

```
        Assert.AreEqual(15, result);
    }

    [TestMethod]
    public void GetEvenDigits_NegativeNumber_AbsoluteValue()
    {
        // Путь: обработка отрицательного числа
        int result = PathOperations.GetEvenDigits(-1234);
        Assert.AreEqual(13, result);
    }

    // ==================== SumOddAboveMainDiagonal
====================

    [TestMethod]
    [ExpectedException(typeof(ArgumentNullException))]
    public void SumOddAboveMainDiagonal_NullArray_ThrowsException()
    {
        // Путь: исключение без циклов
        PathOperations.SumOddAboveMainDiagonal(null);
    }

    [TestMethod]
    public void SumOddAboveMainDiagonal_1x1Array_ReturnsZero()
    {
        // Путь: внешний цикл 1 итерация, внутренний 0 итераций
        int[,] array = { { 5 } };
        int result = PathOperations.SumOddAboveMainDiagonal(array);
        Assert.AreEqual(0, result);
    }

    [TestMethod]
    public void SumOddAboveMainDiagonal_2x2Array_OneOddElement()
    {
        // Путь: проверка условия j>i и нечетности
        int[,] array = {
            { 1, 3 },  // 3 - выше диагонали, нечетное
            { 5, 7 }
        };
        int result = PathOperations.SumOddAboveMainDiagonal(array);
        Assert.AreEqual(3, result);
    }

    [TestMethod]
    public void SumOddAboveMainDiagonal_3x3Array_MultipleOddElements()
```

```
{
    // Путь: множественные итерации с разными условиями
    int[,] array = {
        { 1, 2, 3 },  // 2(четное-пропуск), 3(нечетное-добавить)
        { 4, 5, 6 },  // 6(четное-пропуск)
        { 7, 8, 9 }
    };
    int result = PathOperations.SumOddAboveMainDiagonal(array);
    Assert.AreEqual(3, result); // Только 3
}

[TestMethod]
public void SumOddAboveMainDiagonal_3x3Array_AddMultipleElements()
{
    // Путь: множественные итерации с разными условиями
    int[,] array = {
        { 1, 5, 3 },  // 5(нечетное-добавить), 3(нечетное-добавить)
        { 4, 5, 6 },  // 6(четное-пропуск)
        { 7, 8, 9 }
    };
    int result = PathOperations.SumOddAboveMainDiagonal(array);
    Assert.AreEqual(8, result); // 3 + 5 = 8
}

[TestMethod]
public void SumOddAboveMainDiagonal_AllEvenElements_ReturnsZero()
{
    // Путь: все элементы четные - условие нечетности всегда false
    int[,] array = {
        { 2, 4, 6 },
        { 8, 10, 12 },
        { 14, 16, 18 }
    };
    int result = PathOperations.SumOddAboveMainDiagonal(array);
    Assert.AreEqual(0, result);
}

[TestMethod]
public void SumOddAboveMainDiagonal_RectangularArray_ValidResult()
{
    // Путь: неквадратный массив
    int[,] array = {
        { 1, 3, 5, 7 },  // 3,5,7 - выше диагонали, нечетные
        { 9, 11, 13, 15 } // 13,15 - выше диагонали, нечетные
    };
```

```
            int result = PathOperations.SumOddAboveMainDiagonal(array);
            Assert.AreEqual(3 + 5 + 7 + 13 + 15, result);
        }
    }
}
```

4. **Результаты выполнения модульных тестов.**

5. Выводы по выполненной работе.

В ходе выполнения лабораторной работы были успешно сформированы и закреплены практические навыки разработки модульных тестов для тестирования функций классов на языке C++ с использованием средств автоматизации Visual Studio.

В ходе выполнения работы было сделано:

Полное покрытие кода тестами (критерий C2) - достигнуто 100% покрытие путей выполнения, что подтверждает корректность разработанных тестовых сценариев

Комплексное тестирование функциональности - разработаны тесты для всех черырёх методов класса:

- SortDescending — проверка сортировки трёх чисел по убыванию для всех комбинаций сравнений.
- GCD — тестирование алгоритма нахождения наибольшего общего делителя, включая различные сценарии выполнения цикла и обработку исключений.
- GetEvenDigits — проверка корректного выделения цифр на чётных позициях, включая отрицательные и нулевые значения.
- SumOddAboveMainDiagonal — тестирование суммирования нечетных элементов выше главной диагонали (в том числе в неквадратных матрицах и при исключительных ситуациях).

Покрытие граничных условий - тесты включают проверку:

- Нулевых и отрицательных входных значений;
- Пустых или некорректных структур данных (включая null);
- Краевых случаев (минимальные и максимальные размеры массивов, отсутствие элементов для суммирования);
- Исключительных ситуаций, обрабатываемых методами.

Использование различных подходов тестирования:

- Тестирование нормального выполнения
- Тестирование исключительных ситуаций
- Тестирование граничных значений