Лабораторная работа №3
«Функции и параметры, перегрузка, делегаты»

Цель: получить практический опыт разработки и использования функций в C#, понять различия способов передачи аргументов (по значению, по ссылке `ref`, выходные параметры `out`, параметр-массив `params`), освоить перегрузку функций.

Задание:

1. Напишите две функции `Max2(int a, int b)` и `Min2(int a, int b)`, возвращающие соответственно максимум и минимум из двух целых чисел. Считайте пару чисел из консоли, выведите результат обеих функций.

Пример:

Вход: 7, -3

Выход: Max = 7, Min = -3

Функция (метод) – это именованный блок кода, который можно использовать (вызывать) многократно. Функция может принимать аргументы и возвращать значение.

Аргумент – значение/переменная, передаваемая при вызове функции.

Возвращаемое значение – результат работы функции, отдаваемый через `return`.

Для выполнения 1 задания вам понадобятся конструкции:

1. **Console.ReadLine** – для считывания числа из консоли
2. **Console.WriteLine** – для вывода на экран сообщений с переходом на новую строку
3. **Convert.ToInt32/int.Parse** – для преобразования строк в int
4. **if/else** – для реализации логики функции
5. **return** – для возврата значения

2. Реализуйте функцию `Swap(ref int x, ref int y)`, которая принимает значения по ссылке, меняет местами значения двух переменных без использования дополнительных структур данных. Продемонстрируйте работу: прочитайте из консоли два числа, вызовите `Swap`, выведите результат.

Пример:

Вход: a = 7, b = -3

Выход: b = 7, a = -3

«Передать по значению» - обычно функция получает **копию** значения параметра. Изменения внутри не влияют на переменную «вне» функции, даже если называются они одинаково. Т.е. в функцию передаётся значение переменной, а не сама переменная.

«Передать по ссылке» - при использовании ключевого слова `ref` перед типом данных, например, `int`, функция получает ссылку на место в памяти компьютера, где хранится значение переменной. Значение не копируется и изменения влияют на то, что «вне» функции. Т.е. в функцию передаётся сама переменная.

Для выполнения 2 задания вам понадобятся конструкции:

1. **Console.ReadLine** – для считывания числа из консоли
2. **Console.WriteLine** – для вывода на экран сообщений с переходом на новую строку
3. **ref** – для передачи аргумента по ссылке. Перед использованием переменная **должна быть инициализирована**.
4. **void** – тип возвращаемого значения «ничего». При таком определении не требуется использование **return**

3. Напишите функцию с сигнатурой `void PartitionBySign(int[] xs, out int[] negatives, out int[] positives, out int zeros)`, которая разбивает входной массив `xs` (заполненный с консоли) на два **новых** массива: отрицательные числа и положительные; отдельно записывается количество нулей в массиве. Подсказка: размеры выходных массивов следует вычислять внутри функции (первый проход), затем заполнять (второй проход). Замечание: когда вы передаёте массив в метод **без `ref`**, по умолчанию передаётся **копия ссылки на массив** (а не копия всех элементов). И вызывающий, и метод указывают на **один и тот же объект-массив**. Поэтому любые **изменения элементов** (`a[i] = ...`) видны снаружи, кроме случаев `a = new int[10]` (в таком случае копия ссылки заменяется на ссылку на другой массив)

Пример:

Вход: N = 8, xs = {-3, 0, 5, -1, 0, 2, -4, 7}

Выход: neg = {-3, -1, -4}; pos = {5, 2, 7}; zer = 2.

«Сигнатура функции» - имя функции + упорядоченный список типов её параметров (возвращаемый тип в сигнатуру не входит).

Для выполнения 3 задания вам понадобятся конструкции:

1. **Console.ReadLine** – для считывания числа из консоли
2. **Console.WriteLine** – для вывода на экран сообщений с переходом на новую строку

3. **Convert.ToInt32/int.Parse** – для преобразования строк в int
4. **if/else** – для реализации логики функции
5. **out** – как **ref**, но переменная **может быть неинициализирована** до вызова, но внутри функции ей **обязательно** присваивается значение перед выходом.
6. **for** – цикл для заполнения и чтения массивов
7. **void** – тип возвращаемого значения «ничего». При таком определении не требуется использование **return**

4. Реализуйте 2 перегруженные функции суммирования с переменным числом аргументов, с одинаковыми названиями для суммирования целых чисел и для суммирования вещественных чисел.

Пример:

Вход: Sum(1, 2, 3), Sum(2, 3, 4, 5), Sum(1.3, 2.7, 1.0)

Выход: 6, 14, 5

«Перегруженная функция» - функция, у которой есть альтернативные варианты исполнения (с тем же названием, но другой сигнатурой).

Для выполнения 4 задания вам понадобятся конструкции:

1. **Console.ReadLine** – для считывания числа из консоли
2. **Console.WriteLine** – для вывода на экран сообщений с переходом на новую строку
3. **Convert.ToInt32/int.Parse** – для преобразования строк в int
4. **Convert.ToSingle/float.Parse** – для преобразования строк во float
5. **params** – ключевое слово для определения параметра-массива переменной длины; позволяет передавать произвольное число аргументов, автоматически упакуемых в массив. Должен быть **последним** параметром. Пример использования `static int Sum(params int[] xs)`
6. **return** – для возврата значения

5. С консоли задаётся 2 числа: N – размер квадратной матрицы (двумерного массива), P – количество «рамок». Требуется создать матрицу, заполнить «рамками» из «1» на одинаковом расстоянии друг от друга и вывести в консоль результат. Для P = 1 рисуйте только внешнюю рамку, для P = 2 – внешнюю и центральную. Для больших значений – равномерно заполняйте рамками оставшееся пространство. Для этого реализуйте минимум 3 функции: `int[,] CreateZeroMatrix(int size)` — создаёт и возвращает нулевую матрицу; `void DrawSquareBorder(int[,] a, int layer)` — рисует по периметру слой `layer` (0 — внешний, 1 — следующий и т.д.) единицами; `void PrintMatrix(int[,] a)` — печать матрицы.

Пример:

Вход: N = 5, P = 2; N = 6, P = 2; N = 15; P = 3

Выход:

```
1 1 1 1 1
1 0 0 0 1
1 0 1 0 1
1 0 0 0 1
1 1 1 1 1
```

```
1 1 1 1 1 1
1 0 0 0 0 1
1 0 1 1 0 1
1 0 1 1 0 1
1 0 0 0 0 1
1 1 1 1 1 1
```

```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 1 1 1 1 1 1 1 0 0 0 1
1 0 0 0 1 0 0 0 0 0 1 0 0 0 1
1 0 0 0 1 0 0 0 0 0 1 0 0 0 1
1 0 0 0 1 0 0 1 0 0 1 0 0 0 1
1 0 0 0 1 0 0 0 0 0 1 0 0 0 1
1 0 0 0 1 0 0 0 0 0 1 0 0 0 1
1 0 0 0 1 1 1 1 1 1 1 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Для выполнения 5 задания вам понадобятся конструкции:

1. **Console.ReadLine** – для считывания числа из консоли
2. **Console.WriteLine** – для вывода на экран сообщений с переходом на новую строку
3. **Console.Write** – для вывода на экран сообщений без перехода на новую строку
4. **Convert.ToInt32/int.Parse** – для преобразования строк в int
5. **if/else** – для реализации логики функции (следует ли ставить 1 в точку с координатами row, col)
6. **for** – цикл для заполнения и чтения массивов
7. **void** – тип возвращаемого значения «ничего». При таком определении не требуется использование **return**
8. **return** – для возврата значения

Справочная информация:

Ссылки на описание используемых функций/операторов:

1. **Console.ReadLine** — чтение строки
   https://learn.microsoft.com/ru-ru/dotnet/api/system.console.readline
2. **Console.WriteLine** — вывод с переводом строки
   https://learn.microsoft.com/ru-ru/dotnet/api/system.console.writeline
3. **Console.Write** — вывод без перевода строки
   https://learn.microsoft.com/ru-ru/dotnet/api/system.console.write
4. **Convert.ToInt32** — преобразование к `int`
   https://learn.microsoft.com/ru-ru/dotnet/api/system.convert.toint32
5. **int.Parse (Int32.Parse)** — разбор строки в `int`
   https://learn.microsoft.com/ru-ru/dotnet/api/system.int32.parse
6. **Convert.ToSingle** — преобразование к `float`
   https://learn.microsoft.com/ru-ru/dotnet/api/system.convert.tosingle
7. **float.Parse (Single.Parse)** — разбор строки в `float`
   https://learn.microsoft.com/ru-ru/dotnet/api/system.single.parse
8. **if / else** — условные операторы
   https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/statements/selection-statements#if-else-statement
9. **for** — цикл со счётчиком
   https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/statements/iteration-statements#the-for-statement
10. **while** — цикл с предусловием (если используете)
    https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/statements/iteration-statements#the-while-statement
11. **Массивы в C# (1D)** — обзор
    https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/arrays/
12. **Двумерные и многомерные массивы**
    https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/arrays/multidimensional-arrays
13. **ref** — модификатор аргумента (передача по ссылке)
    https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/ref
14. **out** — выходной параметр
    https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/out-parameter-modifier
15. **params** — параметр-массив переменной длины
    https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/params
16. **void** — «пустой» возвращаемый тип
    https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/void
17. **return** — оператор возврата из метода
    https://learn.microsoft.com/ru-ru/dotnet/csharp/language-reference/statements/jump-statements#the-return-statement
18. **Методы (общие сведения, сигнатура)**
    https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/methods
19. **Перегрузка методов**
    https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/method-overloading