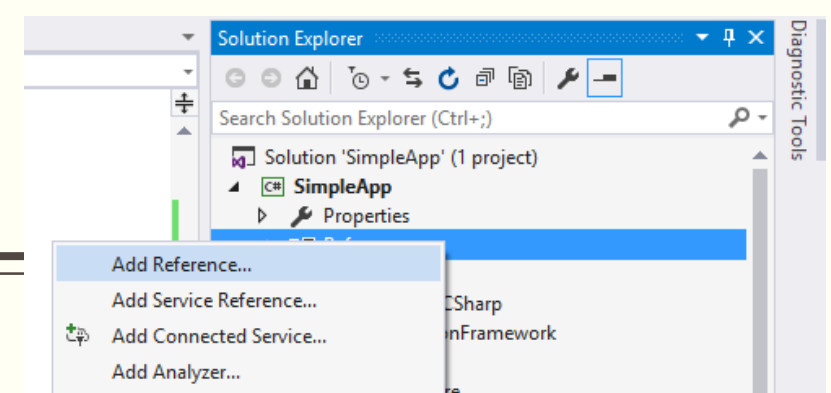




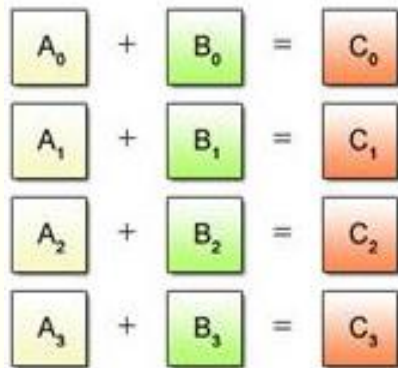
# МАТЕМАТИЧНІ ОБЧИСЛЕННЯ ТА КЛАС MATH

Питання 2.2.

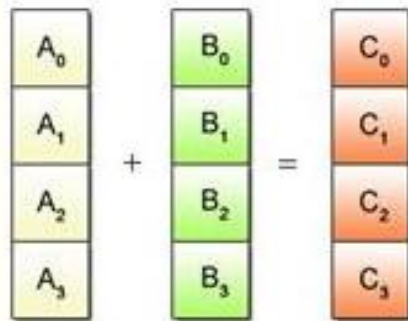
# Простір імен System.Numerics



(a) Scalar Operation



(b) SIMD Operation



## ■ Класи:

- BitOperations – забезпечує службові методи для операцій з бітами, за доступності, з використанням апаратних можливостей платформи. Інакше застосовуються програмні оптимізації.
- Vector – надає колекцію статичних методів для створення та обробки даних щодо узагальнених векторів.
- SIMD-прискорені числові типи (починаючи з .NET Framework 4.6) :
  - Vector2, Vector3, Vector4 - представляють вектори з 2, 3 та 4 Single-значеннями відповідно.
  - Matrix3x2, Matrix4x4, представляють матриці Single-значень відповідних розмірностей.
  - Plane представляє площину в тривимірному просторі за допомогою Single-значень.
  - Quaternion відображає вектор, який використовується для запису тривимірних фізичних обертань, застосовуючи Single-значення.
  - Vector<T> представляє вектор визначеного числового типу та забезпечує оператори з підтримкою SIMD-операцій. Кількість екземплярів Vector<T> фіксована протягом роботи додатку та залежить від ЦП комп'ютера, на якому запускається код.

# Простір імен System.Numerics

---

- Пропонує ще 2 структури:
  - BigInteger – може використовуватись для представлення величезних числових значень, не обмежених фіксованими нижньою або верхньою межами.
  - Complex – представляє комплексні числа.
- Найбільше ціле число, яке можна зберегти в типах .NET Standard, що мають псевдоніми в мові C#, дорівнює близько 18,5 квінтільйона.

```
static void Main(string[] args)
{
    var largestLong = ulong.MaxValue;
    WriteLine($"{largestLong,40:N0}");
    var atomsInTheUniverse = BigInteger.Parse("123456789012345678901234567890");
    WriteLine($"{atomsInTheUniverse,40:N0}");
}
```

Консоль отладки Microsoft Visual Studio

```
18 446 744 073 709 551 615
123 456 789 012 345 678 901 234 567 890
```



biginteger.cs

# Крупні цілі числа

[illegible]

- Тип даних `BigInteger` реагує на внутрішні математичні операції `C#`, такі як `+`, `-` та `*`.
  - Замість виклику `BigInteger.Multiply()` можна писати:  
`BigInteger reallyBig2 = biggy * reallyBig;`

[illegible]

ЕСЛИ ДИСКРИМИНАНТ  
ОТРИЦАТЕЛЬНЫЙ



КОРЕНЬ МНИМЫЙ  
ИСПОЛЬЗУЙ ТЫ

## Комплексні числа

- Поширені в авіаційних та електротехнічних обчисленнях.
- В інформатиці:
  - Швидке перетворення Фур'є. Застосовується в частотному аналізі сигналів
  - У 2D-графіці – обчислення, пов'язані з поворотами, перенесеннями та відображеннями. Для 3D-графіки комплексні числа узагальнюються до кватерніонів.
  - Моделювання фізичних процесів, зокрема у квантовій механіці.

```
1 using System.Numerics;
2 using static System.Console;
3
4 namespace ConsoleApp
5 {
6     class ComplexNumbers
7     {
8         static void Main(string[] args)
9         {
10             var c1 = new Complex(4, 2);
11             var c2 = new Complex(3, 7);
12             var c3 = c1 + c2;
13             WriteLine($"{c1} added to {c2} is {c3}");
14         }
15     }
16 }
17
```

Консоль отладки Microsoft Visual Studio

(4, 2) added to (3, 7) is (7, 9)

# Клас Math

---

- Поля:
  - `public const double E`
  - `public const double PI`

# Клас Math

---

## ■ Методи:

- Знаходження абсолютного значення (модуля) числа – `Math.Abs()`
- Множення дуже великих цілих чисел – `Math.BigMul()`
- Округлення числа до меншого (`Math.Floor()`), більшого (`Math.Ceiling()`) або найближчого (`Math.Round()`) числа
- Тригонометричні функції – `Math.Sin()`, `Math.Cos()`, `Math.Tan()`
- Гіперболічні тригонометричні функції – `Math.Sinh()`, `Math.Cosh()`, `Math.Tanh()`
- Обернені тригонометричні функції – `Math.Asin()`, `Math.Acos()`, `Math.Atan()`
- Отримання цілої частини та остачі від ділення – `Math.DivRem()`
- Отримання значення експоненти – `Math.Exp()`
- Логарифми – `Math.Log()`, `Math.Log2()` – тільки в .NET Core, `Math.Log10()`
- Мінімум (`Math.Min()`) та максимум (`Math.Max()`)
- Піднесення до степеня – `Math.Pow()`
- Взяття кореня від числа – `Math.Sqrt()`
- Відкидання дробової частини – `Math.Truncate()`

# Генерування випадкових чисел – клас Random

```
static void Main(string[] args)
{
    int seed1 = (int)DateTime.Now.Ticks & 0x0000FFFF;
    Console.WriteLine("Seed = {0}", seed1);
    Random rand1 = new Random(seed1);
    ShowRandomNumbers(rand1);

    Random rand2 = new Random(seed1);
    ShowRandomNumbers(rand2);

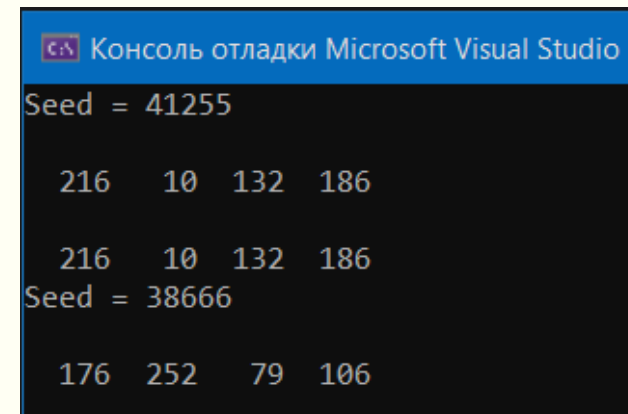
    Thread.Sleep(20);
    int seed3 = (int)DateTime.Now.Ticks & 0x0000FFFF;
    Console.WriteLine("Seed = {0}", seed3);
    Random rand3 = new Random(seed3);
    ShowRandomNumbers(rand3);
}

private static void ShowRandomNumbers(Random rand)
{
    Console.WriteLine();
    byte[] values = new byte[4];
    rand.NextBytes(values);
    foreach (var value in values)
        Console.Write("{0, 5}", value);

    Console.WriteLine();
}
```

10.09.2020

- Конструюється в 2 варіантах:
  - Random() – ініціалізує генератор випадкових чисел із значенням зерна (seed) за умовчанням
  - Random(Int32) - ініціалізує генератор випадкових чисел із заданим значенням зерна
- Об'єкти з однаковим зерном згенерують однакову послідовність випадкових чисел.
  - Корисно при тестуванні додатків з випадковими числами.
  - Генерування різних послідовностей випадкових чисел зазвичай здійснюється за допомогою залежного від часу зерна.



```
Консоль отладки Microsoft Visual Studio
Seed = 41255
  216   10  132  186
  216   10  132  186
Seed = 38666
  176  252   79  106
```



# Зерно генератора за умовчанням

---

- У .NET Framework початкове значення за умовчанням походить від системного годинника, який має скінченну роздільну здатність.
  - Тому різні об'єкти Random, що створюються близько один від одного в часі, мають однакові початкові значення, а отже, створюють ідентичні набори випадкових чисел.
  - Проблема усувається використанням тільки одного об'єкта Random для створення всіх випадкових чисел або створенням власного випадкового початкового значення для конструктора Random(Int32).
- У .NET Core за умовчанням створюється за допомогою статичного ГВЧ потоку, тому вище описане обмеження не застосовується.
  - Після створення екземпляра ГВЧ викликаються окремі методи класу Random, наприклад, Next() або NextDouble(), для створення випадкових чисел.
  - Next() повертає невід'ємне випадкове ціле число.
  - NextDouble() повертає дробове число з діапазону від 0 до 1 (точніше, до 0.99999999999999978).

# Криптографічні генератори випадкових чисел

---

- .NET має абстрактний клас `RandomNumberGenerator`, від якого повинні успадковуватись всі реалізації криптографічних генераторів випадкових чисел (криптоГВЧ).
  - Одна з таких реалізацій у .NET — клас `RNGCryptoServiceProvider`, який використовує кілька постачальників ентропії.
  - Цей «внутрішній» шум може використовуватись не тільки в якості початкового стану, але й між викликами наступних випадкових чисел; таким чином, навіть знаючи поточний стан класу, цього не вистачає для обчислення минулих чи майбутніх згенерованих чисел.
  - Точна поведінка залежить від реалізації. Крім того, Windows може використовувати спеціалізоване апаратне забезпечення, яке буде джерелом «істинних випадковостей» (наприклад, датчик розпаду радіоактивного ізотопу) для генерації ще більш надійних та захищених випадкових чисел.
- У більшості випадків перепоною застосування класу є не його низька продуктивність, а недоречний API.
  - `RandomNumberGenerator` створений *лише* для генерування послідовностей байтів.
  - Методи класу `Random` дозволяють отримати випадкове ціле чи дробове число, набір байтів, зокрема, число з указанного діапазону.
  - виправити ситуацію можна, створивши свою оболонку (вряппер) навколо `RandomNumberGenerator`, яка буде перетворювати випадкові байти в «зручний» результат, проте це нетривіальне рішення.

# Додаткова інформація

---

- Генерація случайных чисел в .NET



# ДЯКУЮ ЗА УВАГУ!

Наступне питання: Робота з масивами