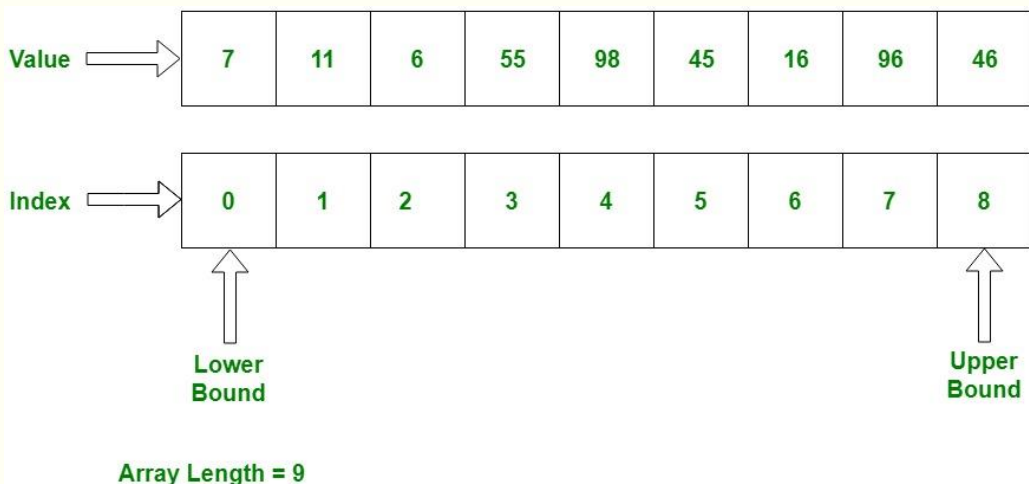




РОБОТА З МАСИВАМИ

Питання 2.3.

Масиви в C#



■ **Масив** — це набір елементів даних, для доступу до яких використовується числовий індекс.

- Масив може бути одновимірним, багатовимірним або зубчастим (масивом масивів, jagged array).
- Кількість вимірів та довжина кожного з них задаються при створенні екземпляру масива та не можуть бути зміненими протягом існування цього екземпляру.
- За умовчанням числові елементи масиву рівні 0, а посилальні елементи – NULL.
- У зубчастому масиві елементи є посилальними типами.
- Масив з n елементів індексується від 0 до $n-1$.
- Елементи масиву можуть мати довільний тип, у тому числі й тип масиву.
- Типи масивів — це посилальні типи, породжені від абстрактного базового типу `Array`. Оскільки цей тип реалізує інтерфейси `IEnumerable` та `IEnumerable<T>`, існує можливість ітерувати по елементах будь-якого масиву за допомогою оператора `foreach`.

Оголошення та ініціалізація масивів

```
// Створюємо одновимірний масив з 5 цілих чисел.  
int[] array1 = new int[5];  
  
// Оголошуємо та ініціалізуємо елементи масиву.  
int[] array2 = new int[] { 1, 3, 5, 7, 9 };  
  
// Альтернативний синтаксис.  
int[] array3 = { 1, 2, 3, 4, 5, 6 };  
  
// Оголошуємо двовимірний масив.  
int[,] multiDimensionalArray1 = new int[2, 3];  
  
// Оголошуємо та ініціалізуємо елементи масиву.  
int[,] multiDimensionalArray2 = { { 1, 2, 3 }, { 4, 5, 6 } };  
  
// Оголошуємо зубчастий (jagged) масив.  
int[][] jaggedArray = new int[6][];  
  
// Ініціалізуємо перший масив у структурі зубчастого масиву.  
jaggedArray[0] = new int[4] { 1, 2, 3, 4 };
```

- Якщо масив оголошено, проте явно не заповнено по кожному індексу, його елементи отримують стандартне значення для відповідного типу даних.
 - Наприклад, елементам масива `bool` буде присвоєно `false`, а елементам `int`-масива — `0`).
- Якщо оголошений розмір і кількість ініціалізаторів не співставні, то на етапі компіляції виникне помилка.
 - // Недповідність розміру та кількості елементів
`int[] intArray = new int[2] { 20, 22, 23, 0 };`
`int[] intArray = new int[2] { 20, 22, 23, 0 };`
- Спрацьовує неявна типізація масивів:
 - `var a = new int[4] { 20, 22, 23, 0 };`
- Явне заповнення масивів:
 - `array1[0] = 100;`
 - `multiDimensionalArray1[1, 2] = 300;`

```

// ітерування по одновимірному масиву
int[] numbers = { 4, 5, 6, 1, 2, 3, -2, -1, 0 };
foreach (int i in numbers)
{
    System.Console.WriteLine("{0} ", i);
}
Console.WriteLine();

// ітерування по багатовимірному масиву
int[,] numbers2D = new int[3, 2] { { 9, 99 }, { 3, 33 }, { 5, 55 } };
// або коротка форма:
// int[,] numbers2D = { { 9, 99 }, { 3, 33 }, { 5, 55 } };

foreach (int i in numbers2D)
{
    Console.WriteLine("{0} ", i);
}
Console.WriteLine();

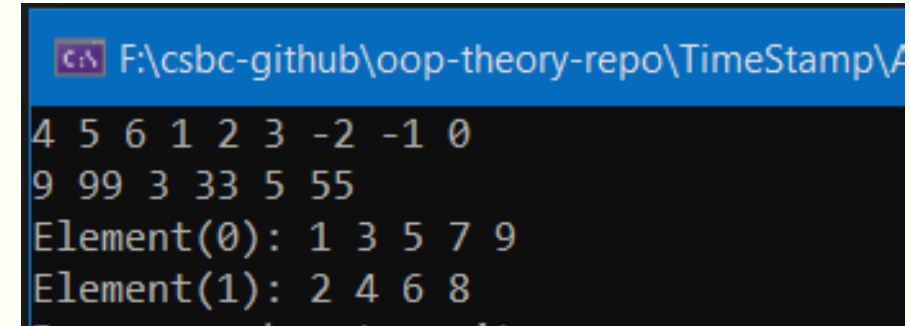
// ітерування по зубчастому масиву за допомогою вкладеного for.
int[][] arr = new int[2][];
arr[0] = new int[5] { 1, 3, 5, 7, 9 };
arr[1] = new int[4] { 2, 4, 6, 8 };

for (int i = 0; i < arr.Length; i++)
{
    Console.WriteLine("Element({0}): ", i);

    for (int j = 0; j < arr[i].Length; j++)
    {
        Console.Write("{0}{1}", arr[i][j],
                        j == (arr[i].Length - 1) ? "" : " ");
    }
    Console.WriteLine();
}

```

Ітерування по масивах



```

F:\csbc-github\oop-theory-repo\TimeStamp\A
4 5 6 1 2 3 -2 -1 0
9 99 3 33 5 55
Element(0): 1 3 5 7 9
Element(1): 2 4 6 8

```

Масиви об'єктів

```
static void ArrayOfObjects()
{
    Console.WriteLine("=> Array of Objects.");
    // Массив объектов может содержать все что угодно.
    object[] myObjects = new object[4];
    myObjects[0] = 10;
    myObjects[1] = false;
    myObjects[2] = new DateTime(1969, 3, 24);
    myObjects[3] = "Form & Void";
    foreach (object obj in myObjects)
    {
        // Вывести тип и значение каждого элемента в массиве.
        Console.WriteLine("Type: {0}, Value: {1}", obj.GetType(), obj);
    }
    Console.WriteLine();
}
```

- Якщо визначити масив типу `System.Object`, то його елементи можуть представляти що завгодно.
 - Під час ітерування по вмісту масива `myObjects` відбувається виведення типу, що лежить в основі, для кожного елемента.

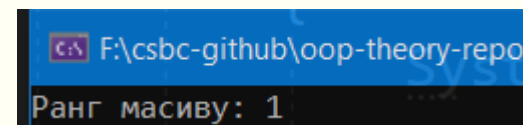
Вивід програми:

```
=> Array of Objects.
Type: System.Int32, Value: 10
Type: System.Boolean, Value: False
Type: System.DateTime, Value: 3/24/1969 12:00:00 AM
Type: System.String, Value: Form & Void
```

Тип Array та його можливості

- Надає можливості для створення, обробки, пошуку та сортування масивів.
 - Клас Array вважається колекцією, оскільки базується на інтерфейсі IList, проте він не є частиною простору імен System.Collections.
- Властивості класу:
 - *IsFixedSize* –перевірочне значення, чи є масив фіксованого розміру.
 - *IsReadOnly* –перевірочне значення, чи є масив доступним тільки для зчитування.
 - *IsSynchronized* –перевірочне значення, чи синхронізований (потокобезпечний, thread safe) доступ до масиву.
 - *Length* – загальна кількість елементів у всіх вимірах масиву.
 - *LongLength* – 64-розрядне ціле число, яке представляє довжину масиву.
 - *Rank* – ранг (кількість вимірів) масиву. Для зубчастих масивів дорівнює 1.
 - *SyncRoot* – об'єкт, який може застосовуватись для синхронізації доступу до масиву.

```
Console.WriteLine("Ранг масиву: {0}", arr.Rank);
```

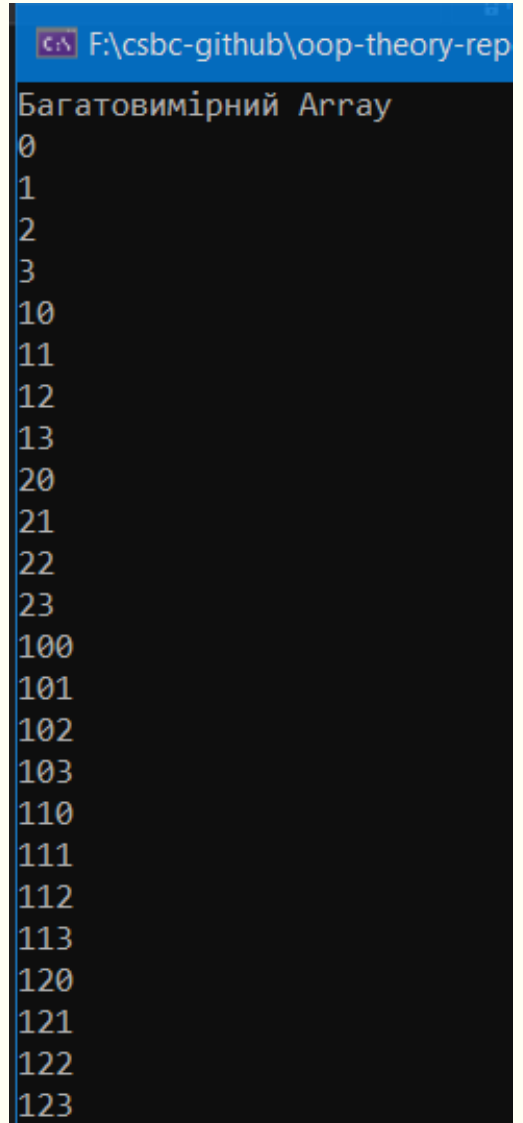


Створення масивів за допомогою класу Array

```
Array stringArray = Array.CreateInstance(typeof(string), 3);
stringArray.SetValue("Іван Петрович", 0);
stringArray.SetValue("Степан Іванович", 1);
stringArray.SetValue("Олексій Степанович", 2);

Console.WriteLine("Багатовимірний Array");
Array intArray3D = Array.CreateInstance(typeof(Int32), 2, 3, 4);
for (int i = intArray3D.GetLowerBound(0); i <= intArray3D.GetUpperBound(0); i++)
    for (int j = intArray3D.GetLowerBound(1); j <= intArray3D.GetUpperBound(1); j++)
        for (int k = intArray3D.GetLowerBound(2); k <= intArray3D.GetUpperBound(2); k++)
        {
            intArray3D.SetValue((i * 100) + (j * 10) + k, i, j, k);
        }
foreach (int ival in intArray3D)
{
    Console.WriteLine(ival);
}
```

■



```
F:\csbc-github\oop-theory-rep
Багатовимірний Array
0
1
2
3
10
11
12
13
20
21
22
23
100
101
102
103
110
111
112
113
120
121
122
123
```

Пошук елементів у масиві

// методи вимагають аргументи більш конкретного типу, ніж Array

```
string[] array = (string[])stringArray;
```

```
string res = Array.Find(array, ele => ele.StartsWith("C"));
```

```
Console.WriteLine("Результат пошуку методом Find():");
```

```
Console.WriteLine(res);
```

```
string[] res2 = Array.FindAll(array, element => element.Contains("Іван"));
```

```
Console.WriteLine("Результат пошуку методом FindAll():");
```

```
foreach (var result in res2)
```

```
{
```

```
    Console.WriteLine(result);
```

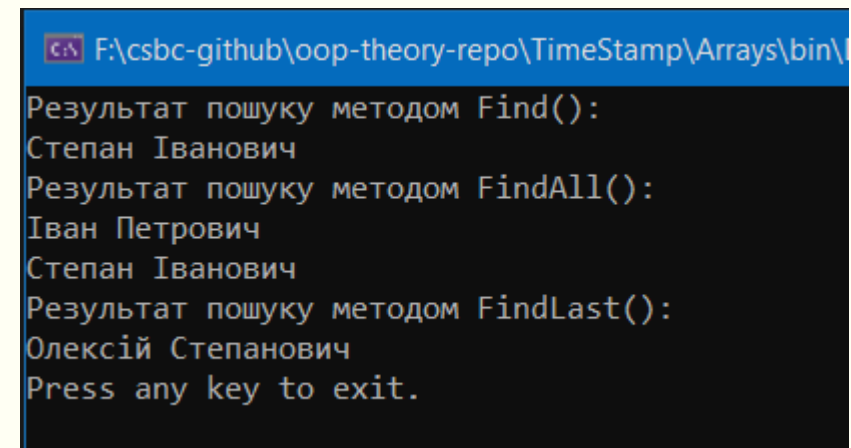
```
}
```

```
string res3 = Array.FindLast(array, element => element.Contains("Степан"));
```

```
Console.WriteLine("Результат пошуку методом FindLast():");
```

```
Console.WriteLine(res3);
```

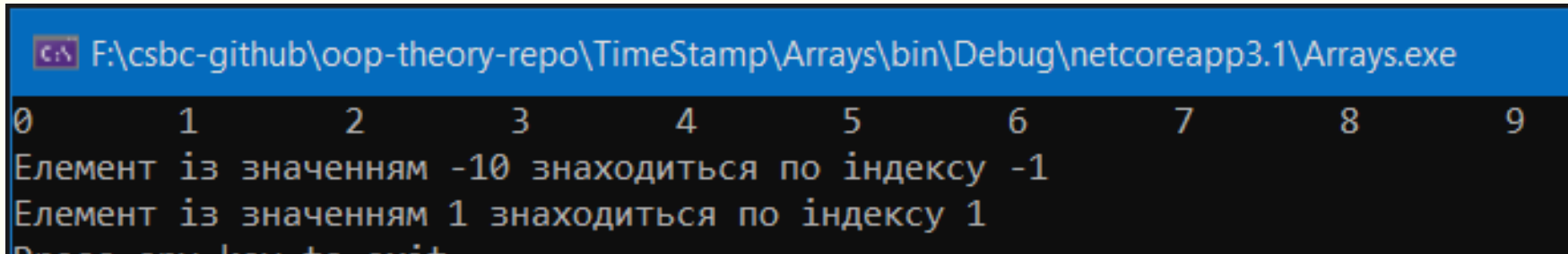
- Застосовується лямбда-вираз, який задає умову пошуку



```
F:\csbc-github\oop-theory-repo\TimeStamp\Arrays\bin\
Результат пошуку методом Find():
Степан Іванович
Результат пошуку методом FindAll():
Іван Петрович
Степан Іванович
Результат пошуку методом FindLast():
Олексій Степанович
Press any key to exit.
```


Сортування елементів та бінарний пошук

```
int[] myNums = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };
Array.Sort(myNums, 0, myNums.Length);
foreach(int num in myNums)
    Console.WriteLine("{0}\t", num);
Console.WriteLine();
Console.WriteLine("Елемент із значенням -10 знаходиться по індексу {0}",
    Array.BinarySearch(myNums, -10));
Console.WriteLine("Елемент із значенням 1 знаходиться по індексу {0}",
    Array.BinarySearch(myNums, 1));
```



The screenshot shows a Windows Command Prompt window with the title bar "F:\csbc-github\oop-theory-repo\TimeStamp\Arrays\bin\Debug\netcoreapp3.1\Arrays.exe". The output of the program is as follows:

0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

Елемент із значенням -10 знаходиться по індексу -1
Елемент із значенням 1 знаходиться по індексу 1
Press any key to exit

- Сортування може відбуватись не за всіма елементами, відповідний діапазон індексів передається в якості останніх параметрів `Array.Sort()`.

Копіювання, клонування, реверс та очищення елементів

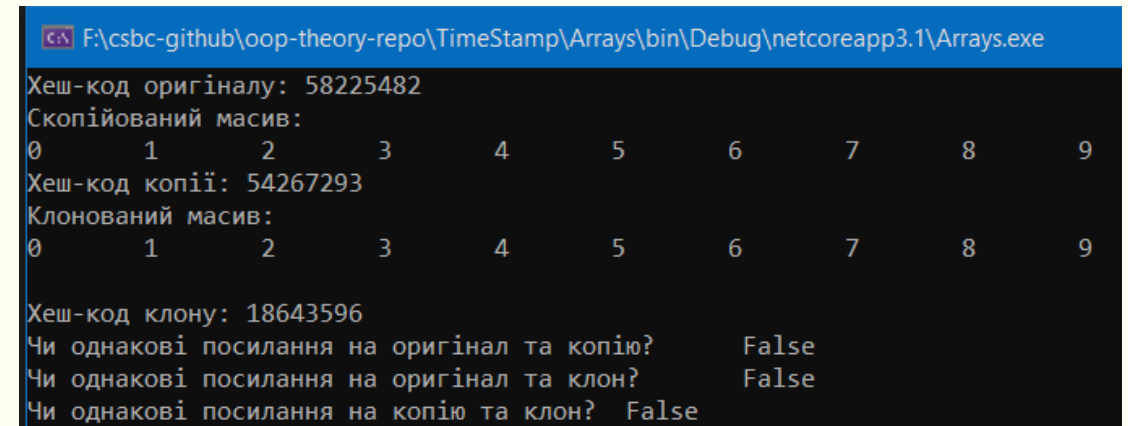
- Метод `Array.Clone()` повертає об'єкт, який необхідно зводити до потрібного типу.
 - не вимагає доступності цільового (target/destination) масиву при виклику, на відміну від методів `Array.CopyTo()` чи `Array.Copy()`.

```
Console.WriteLine("\nХеш-код оригіналу: {0}", myNums.GetHashCode());
```

```
int[] copiedNums = new int[myNums.Length];  
Array.Copy(myNums, myNums.GetLowerBound(0), copiedNums, copiedNums.GetLowerBound(0), 10);  
Console.WriteLine("Скопійований масив: ");  
for (int i = 0; i <= copiedNums.GetUpperBound(0); i++)  
    Console.Write($"{copiedNums.GetValue(i)}\t");  
Console.WriteLine("\nХеш-код копії: {0}", copiedNums.GetHashCode());
```

```
int[] clonedNums = (int[])myNums.Clone();  
Console.WriteLine("Клонований масив: ");  
for (int i = 0; i <= clonedNums.GetUpperBound(0); i++)  
    Console.Write($"{clonedNums.GetValue(i)}\t");  
Console.WriteLine();  
Console.WriteLine("\nХеш-код клону: {0}", clonedNums.GetHashCode());
```

```
Console.WriteLine("Чи однакові посилання на оригінал та копію?\t {0}", object.ReferenceEquals(myNums, copiedNums));  
Console.WriteLine("Чи однакові посилання на оригінал та клон?\t {0}", object.ReferenceEquals(myNums, clonedNums));  
Console.WriteLine("Чи однакові посилання на копію та клон?\t {0}", object.ReferenceEquals(copiedNums, clonedNums));
```



```
F:\csbc-github\oop-theory-repo\TimeStamp\Arrays\bin\Debug\netcoreapp3.1\Arrays.exe  
Хеш-код оригіналу: 58225482  
Скопійований масив:  
0      1      2      3      4      5      6      7      8      9  
Хеш-код копії: 54267293  
Клонований масив:  
0      1      2      3      4      5      6      7      8      9  
Хеш-код клону: 18643596  
Чи однакові посилання на оригінал та копію?      False  
Чи однакові посилання на оригінал та клон?      False  
Чи однакові посилання на копію та клон?      False
```

Реверс та очищення

```
Array.Reverse(myNums);
Array.Clear(clonedNums, 4, 6);
myNums[0] = 100;

Console.WriteLine("Початковий масив після реверсу:\t\t ");
for (int i = 0; i <= myNums.GetUpperBound(0); i++)
    Console.WriteLine($"{myNums.GetValue(i)}\t");
Console.WriteLine();

Console.WriteLine("Скопійований масив після реверсу:\t ");
for (int i = 0; i <= copiedNums.GetUpperBound(0); i++)
    Console.WriteLine($"{copiedNums.GetValue(i)}\t");
Console.WriteLine();

Console.WriteLine("Клонований масив після реверсу:\t\t ");
for (int i = 0; i <= clonedNums.GetUpperBound(0); i++)
    Console.WriteLine($"{clonedNums.GetValue(i)}\t");
Console.WriteLine();
```

- Очищення масиву передбачає передачу в метод `Array.Clear()` масиву, індексу початку очищення та кількості елементів для очистки.
 - Порожні місця займаються значеннями за умовчанням.

```
F:\csbc-github\oop-theory-repo\TimeStamp\Arrays\bin\Debug\netcoreapp3.1\Arrays.exe
Початковий масив після реверсу:      100      8      7      6      5      4      3      2      1      0
Скопійований масив після реверсу:    0       1       2       3       4       5       6       7       8       9
Клонований масив після реверсу:      0       1       2       3       0       0       0       0       0       0
```

Технологія LINQ та масиви

```
int[] nums = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };
```

```
var lowNums = from n in nums  
              select n;
```

```
Console.WriteLine("Всі числа: ");  
foreach (var x in lowNums)  
{  
    Console.WriteLine(x);  
}
```

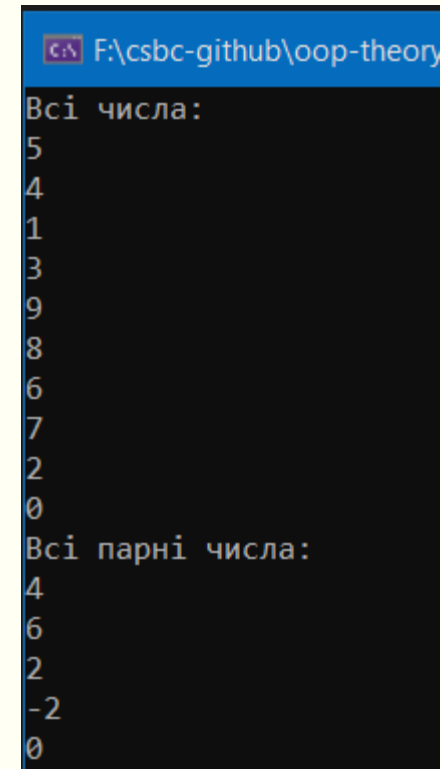
```
Console.WriteLine("Всі парні числа: ");  
lowNums = (from n in numbers  
           where (n % 2 == 0)  
           select n);
```

```
foreach (var x in lowNums)  
{  
    Console.WriteLine(x);  
}
```

- LINQ (Language Integrated Query) – технологія Microsoft для виконання операцій з практично будь-якими джерелами даних.
 - Підтримує масиви, списки, XML, бази даних, плоскі (flat) файли та ін.
 - LINQ-запит може обробляти дані з різних джерел, не змінюючи форму запису запиту.
 - Для застосування потрібно включити простір імен System.Linq.

- Приклади застосування LINQ для масивів.

- Довідка щодо класу Array



The screenshot shows a Windows command prompt window with the title bar 'F:\csbc-github\oop-theory'. The output of the program is as follows:

```
Всі числа:  
5  
4  
1  
3  
9  
8  
6  
7  
2  
0  
Всі парні числа:  
4  
6  
2  
-2  
0
```



ДЯКУЮ ЗА УВАГУ!

Наступне питання: Програмні засоби для роботи з текстом