ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

C#. Масиви

Лекція 03

План

1. Створення одновимірного масиву
2. Доступ до елементів масиву
3. Методи класу Array
4. Створення багатовимірного масиву
5. Генерація псевдовипадкових чисел
6. Зубчасті масиви

Одновимірний масив

Одновимірний масив – це впорядкована колекція даних одного типу, які зберігаються послідовно в пам'яті. Кожному елементу масиву відповідає унікальний індекс, за яким можна отримати доступ до відповідного значення.

11	7	-3	4	5	6	2	8	-9	15
0	1	2	3	4	5	6	7	8	9

Синтаксис оголошення масиву

тип[ ] ім'я_масиву;

тип[ ] ім'я_масиву = {список_ініціалізаторів};

тип[ ] ім'я_масиву = new тип [довжина_масиву];

тип[ ] ім'я_масиву = new тип[ ] {список ініціалізаторів};

тип[ ] ім'я_масиву = new тип[довжина_масиву] {список_ініціалізаторів};

Синтаксис оголошення масиву

// Оголошення масиву цілих чисел розміром 10

```
int[] numbersI = new int[10];
```

```
int x = 10;
double[] numbersD = new double[x];
```

Ініціалізація масиву

```
int[] num = new int[4];
num[0] = 1;
num[1] = 2;
num[2] = 3;
num[3] = 5;

// або
int[] num = new int[] { 1, 2, 3, 5 };

//або
int[] num = { 1, 2, 3, 5 };
```

Розмір створеного масиву змінити не можна!!!!

Ініціалізація масиву

```
string[] names = {"Volvo", "BMW", "Ford", "Mazda"};

char[] symbol = new char[4] { 'X', 'Y', 'Z', 'M' };

 int[] number = new int[5] { 1, 2, 3 };
```

Доступ до елементів масиву

```
// Отримання значення елемента за індексом

int[] numbers = { 10, 20, 30, 40, 50 };

Console.WriteLine(numbers[2]);

// Зміна значення елемента

numbers[4] = 50;

// Змінюємо перший елемент на 100

numbers[0] = 100;
```

Доступ до елементів масиву

Доступ до елементів одновимірного масиву виконується так само як і в C:

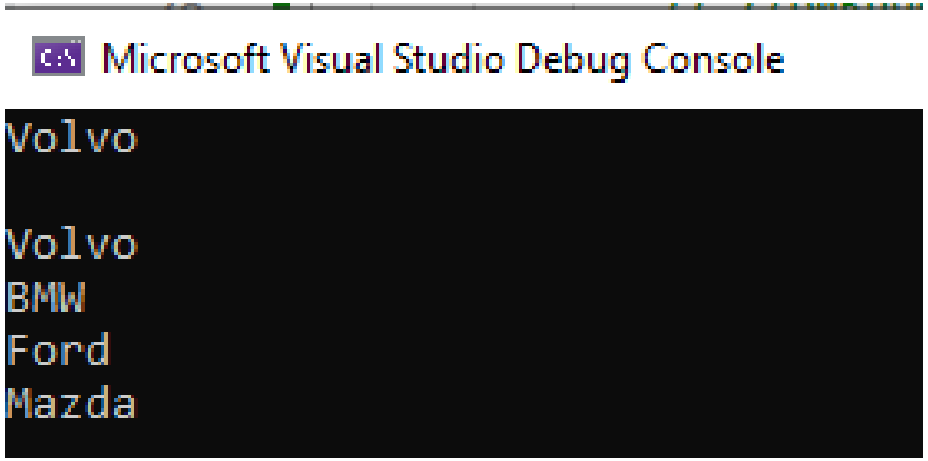
```
int[] nums = new int[3];
for (int i = 0; i < 5; i++)
  nums[i] = int.Parse(Console.ReadLine());
```

Звернення до елемента масиву

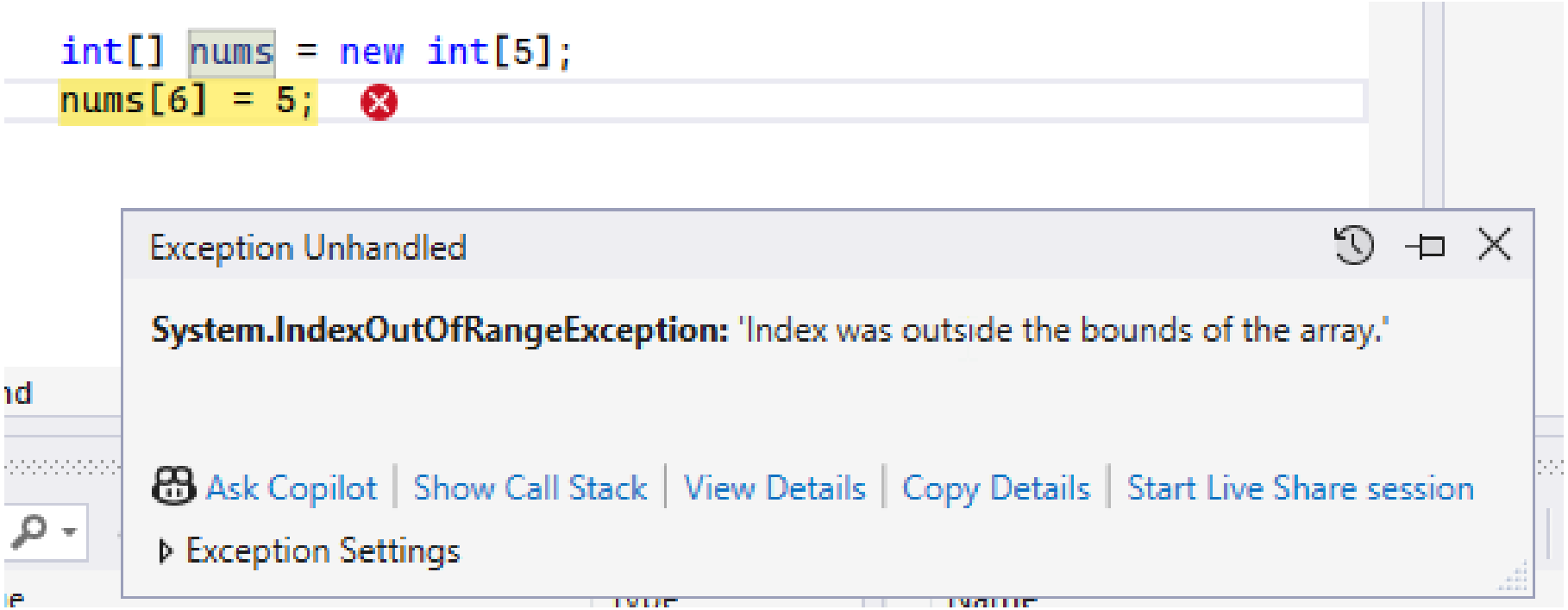
При виході за межі масиву генерується виключення IndexOutOfRangeException

Приклад

```
static void Main(string[] args)
{
    string[] cars = { "Volvo", "BMW", "Ford", "Mazda" };
    Console.WriteLine(cars[0]);

    Console.WriteLine();


    for (int i = 0; i < 4; i++)
      Console.WriteLine(cars[i]);


}
```



Одновимірні масиви



Спроба переглянути елемент (звернутися до елементу) з індексом, що виходить за межі масиву, призведе до виключення *IndexOutOfRangeException*.

Методи класу Array

методи для операцій над масивами:

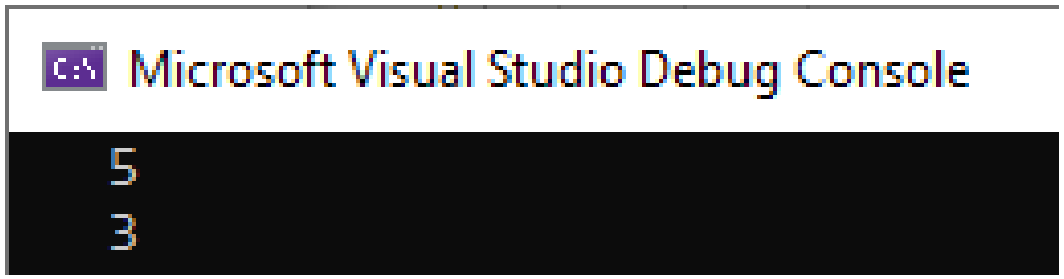
Length - надає загальну кількість елементів у масиві
Sort() - Сортує елементи масиву.
Reverse() - Інвертує порядок елементів масиву.
IndexOf() - Знаходить індекс першого входження заданого елемента, або -1, якщо елемент не знайдено.
LastIndexOf() - Повертає індекс останнього входження зазначеного елемента.
Contains() - Перевіряє, чи містить масив заданий елемент.
Copy() - Копіює елементи одного масиву в інший.
Clone() - Створює копію масиву.

Отримання довжини масиву

.Length в C# надає загальну кількість елементів у масиві, тобто повертає ціле число, що представляє кількість елементів.

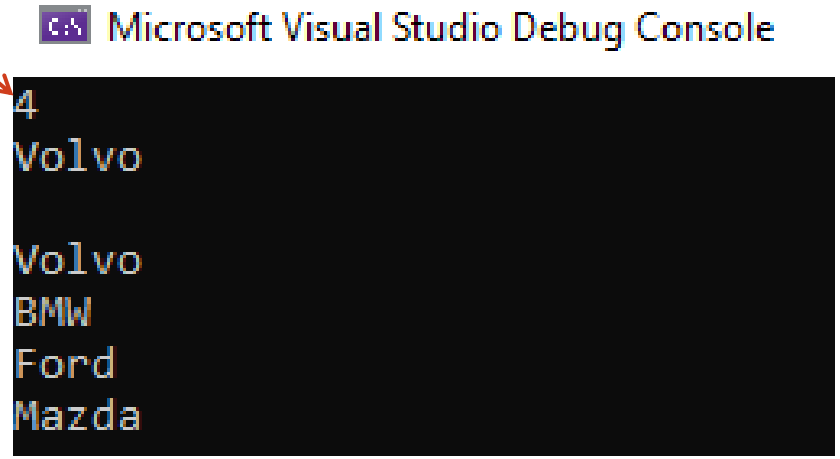
int ім'я_змінної = ім'я_масиву.Length;

```
int[] numbers = { 1, 2, 3, 4, 5 };
    int arrayLength = numbers.Length;
    Console.WriteLine("{0,4:D}", arrayLength);

string[] names = { "Alice", "Ivan", "David" };
    int namesCount = names.Length;
    Console.Write("{0,4:D}", namesCount);
```



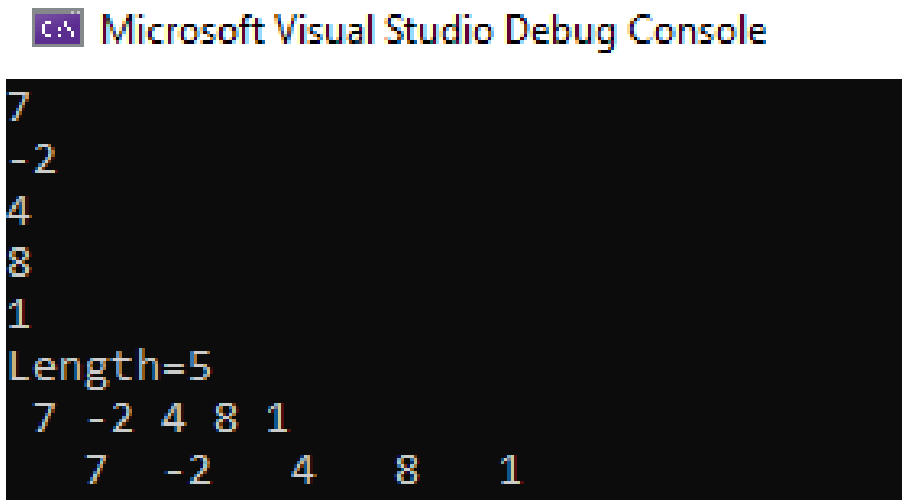
Приклад

```
static void Main(string[] args)
{
    string[] cars = { "Volvo", "BMW", "Ford", "Mazda" };
    Console.WriteLine(cars.Length);

    Console.WriteLine(cars[0]);

    Console.WriteLine();

    for (int i = 0; i < cars.Length; i++)
      Console.WriteLine(cars[i]);


}
```



Приклад

```
static void Main(string[] args)
{
    int x=5;
    int[] num = new int[x];

    for (int i = 0; i < x; i++)
        num[i] = int.Parse(Console.ReadLine());

    Console.WriteLine("Length="+num.Length);

    for (int i = 0; i < num.Length; i++)
        Console.Write($" {num[i]}");

    Console.WriteLine();

    foreach (int n in num)
            Console.Write("{0,4:D}", nums);

}
```

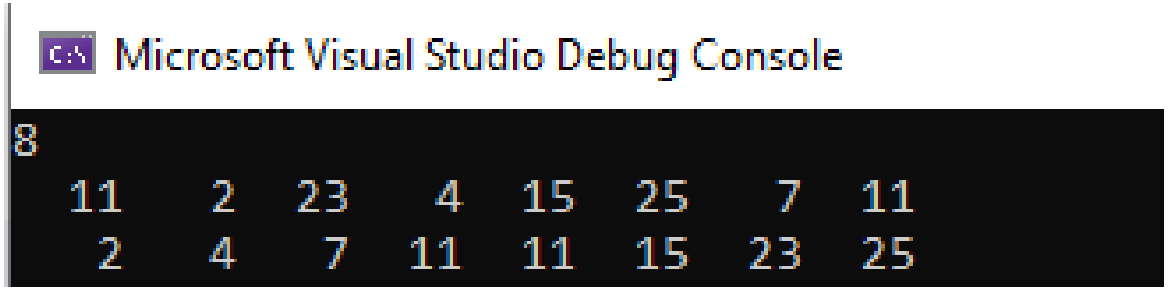


Методи класу Array

C# надає зручний метод *Sort()* для сортування масивів.Синтаксис

Array.Sort(масив);

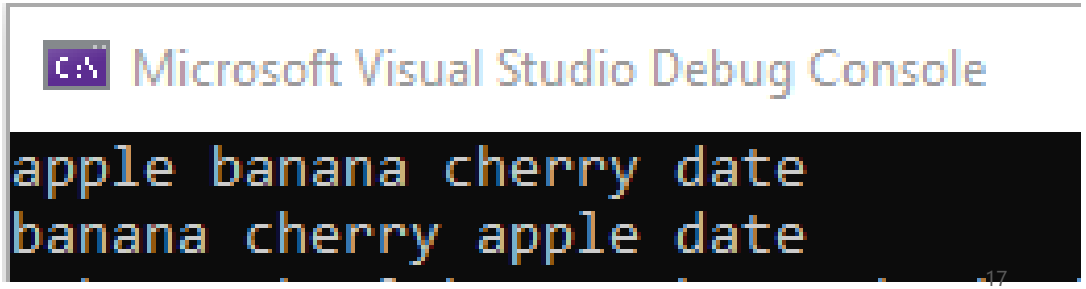
```
int[] numbers = { 11, 2, 23, 4, 15, 25, 7, 11 };
int arrayLength = numbers.Length;
Console.WriteLine(arrayLength);
foreach (int nums in numbers)
        Console.Write("{0,4:D}", nums);

Console.WriteLine();

Array.Sort(numbers);
foreach (int nums in numbers)
        Console.Write("{0,4:D}", nums);
```



Одновимірні масиви

Для сортування масиву можна використовувати метод *CompareTo()*.

```
// Сортування за довжиною рядка за спаданням


string[] words = { "apple", "banana", "cherry", "date" };

foreach( string word  in words)
    Console.Write(word+" ");
Console.WriteLine();

Array.Sort(words, (a, b) => b.Length.CompareTo(a.Length));

foreach (string word in words)
    Console.Write(word + " ");
```

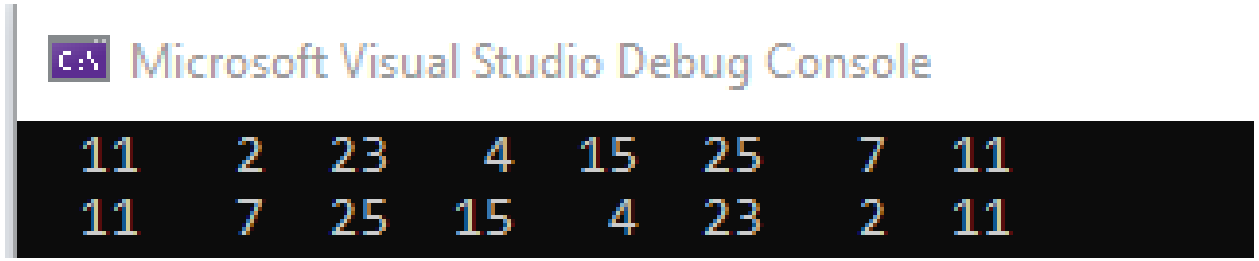


Методи класу Array

Для зміни порядку елементів на протилежну можна використати метод *Array. Reverse*.
Синтаксис

Array. Reverse(масив);

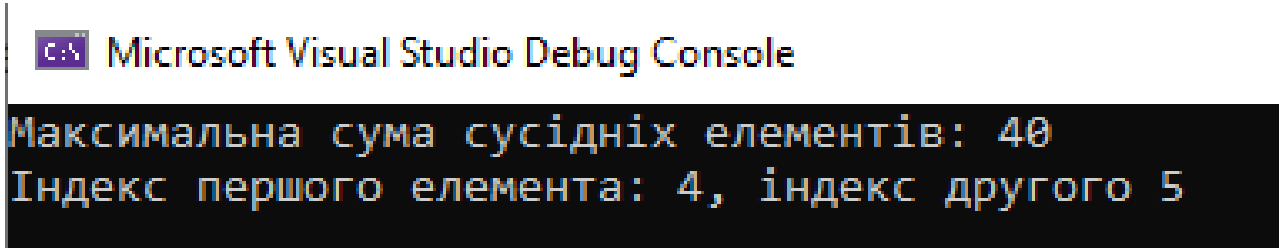
```
int[] numbers = { 11, 2, 23, 4, 15, 25, 7, 11 };
foreach (int nums in numbers)
    Console.Write("{0,4:D}", nums);

Array.Reverse(numbers);

Console.WriteLine();
foreach (int nums in numbers)
    Console.Write("{0,4:D}", nums);
```



Приклад

Завдання. Дано масив розміру N. Знайти два сусідніх елементи, сума яких максимальна, і вивести ці елементи в порядку зростання їхніх індексів.

```
int[] arr = { 11, 2, 23, 4, 15, 25, 7, 11 };
  int sum = int.MinValue;
  int index = 0;
  for (int i = 0; i < arr.Length - 1; i++)
  {
      if (arr[i] + arr[i + 1] > sum)
      {
       sum = arr[i] + arr[i + 1];
       index = i;
      }
   }
```



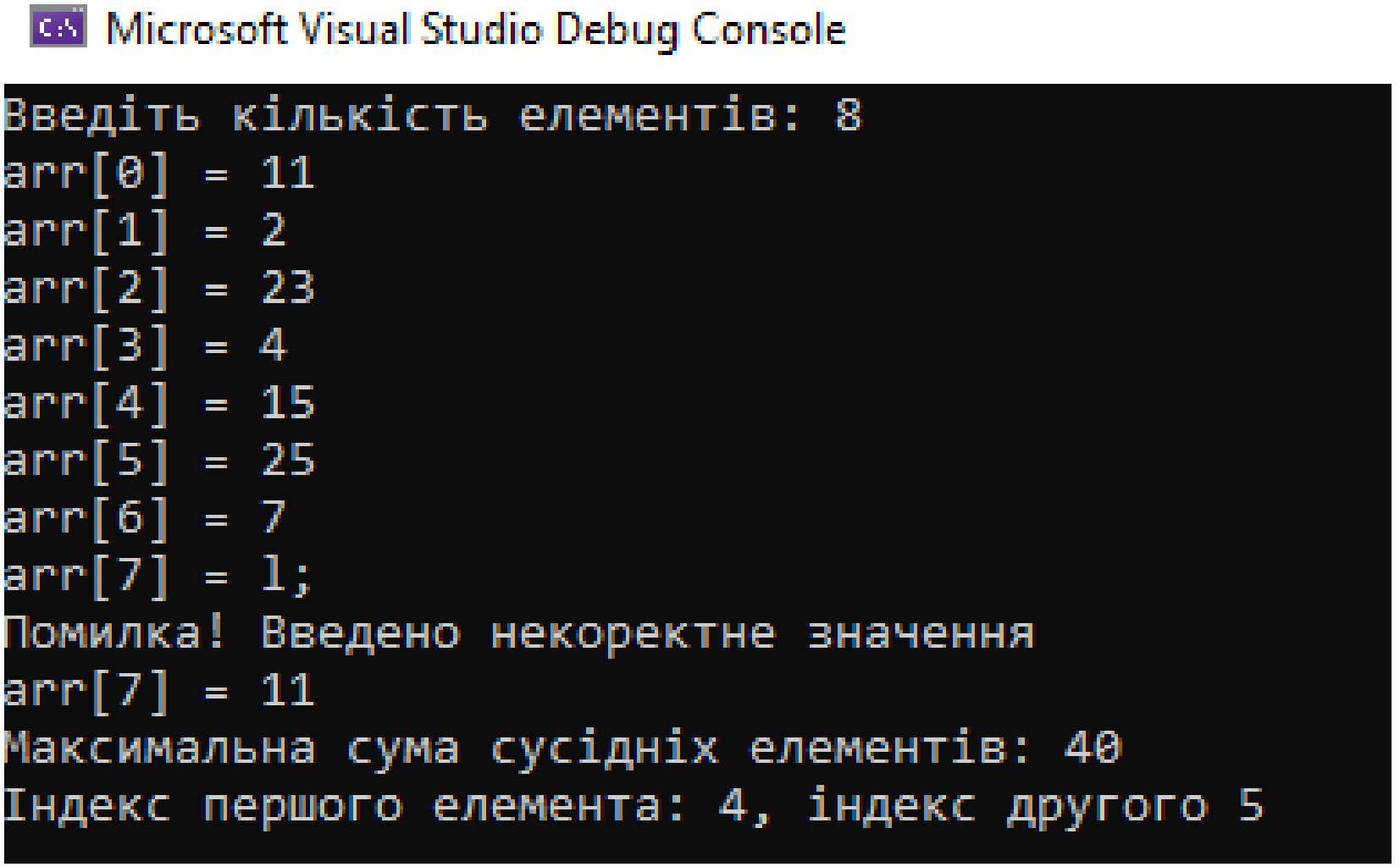

```
Console.WriteLine($"Максимальна сума сусідніх елементів: {sum}");
Console.WriteLine($"Індекс першого елемента: {index}, індекс другого {index+1}");
```

Приклад

```
int n; bool f=true;
do{
    Console.Write("Введіть кількість елементів: ");
    f = int.TryParse(Console.ReadLine(), out n);
    if (f == false)
        Console.WriteLine(" Помилка! Введено некоректне значення");
} while (!f);
int[] arr = new int[n];
for (int i = 0; i < arr.Length; i++){
     do{
     Console.Write("arr[{0}] = ", i);
     f = int.TryParse(Console.ReadLine(), out arr[i]);
     if (f == false)
         Console.WriteLine("Помилка! Введено некоректне значення");
 } while (!f);}
int sum = int.MinValue;  int index = 0;
for (int i = 0; i < arr.Length-1; i++){ … }
Console.WriteLine($"Максимальна сума сусідніх елементів: {sum}");
Console.WriteLine($"Індекс першого елемента: {index}, індекс другого {index + 1}");
```

Приклад

Результат роботи програми:



Багатовимірні масиви

Синтаксис оголошення двовимірного масиву:

тип_даних[,] ім'я_масиву;

тип_даних[,] ім'я_масиву = new тип_даних[к-сть_рядків, к-сть_стовпців];
тип_даних[,] ім'я_масиву = {список_ініціалізаторів};
тип_даних[,] ім'я_масиву = new тип_даних[,] {список_ініціалізаторів};

Приклад

```
// Масив цілих чисел розміром 3 на 4

int[,] numbers = new int[3, 4];

// Масив дробових чисел розміром 3 на 4

double[,] coordinates = new double[3, 4];

// Масив рядків розміром 5 на 5

string[,] names = new string[5, 5];

// Ініціалізація масиву при оголошенні

int[,] matrix1 = new int[,] { { 1, 2, 3 }, { 4, 5, 6 } };

int[,] matrix2 = { { 1, 2 }, { 3, 4 }, { 5, 6 } };
```

Доступу до елементів масиву

Для доступу до конкретного елемента масиву використовується подвійний індекс:

ім'я_масиву[індекс_рядка, індекс_стовпця]

- У квадратних дужках індекси записуються через кому.
- Індексація починається з нуля.

```
// Доступ до елемента двовимірного масиву

int value = matrix[1, 2]
```

Доступу до елементів масиву

Матриця зберігає інформацію про кількість рядків та стовпців:

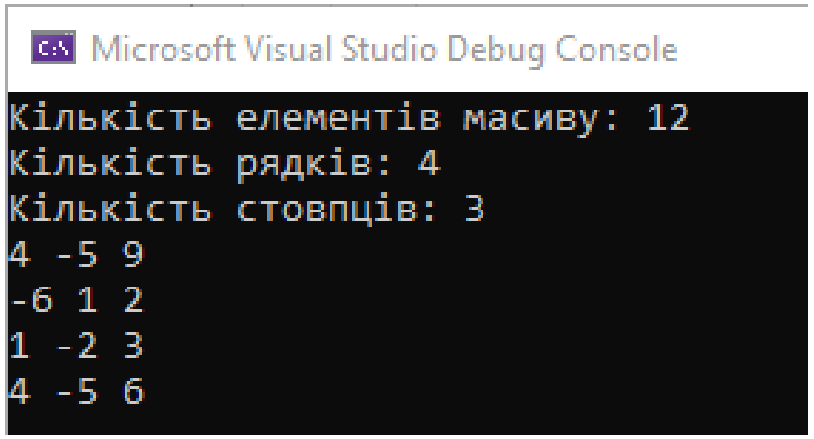
```
int [,] matrix = new int[4,3]

matrix.GetLength(0); //кількість рядків 4

matrix.GetLength(1); //кількість стовпців 3
```

Масив зберігає також інформацію про його розмір:

matr.Rank – для одновимірного = 1, для двовимірного = 2 тощо.

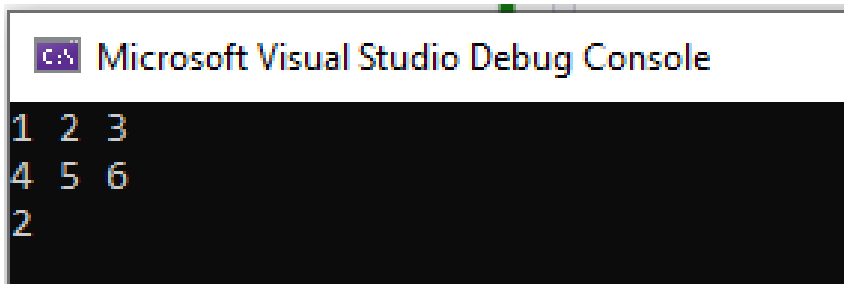
Приклад

```
int [,] matrix = new int [,]{ {4,-5,9 }, {-6,1,2}, {1,-2,3},{4,-5,6} };

int totalElements = matrix.Length;
int rows = matrix.GetLength(0);
int columns = matrix.GetLength(1);

Console.WriteLine($"Кількість елементів масиву: {totalElements}");
Console.WriteLine($"Кількість рядків: {rows}");
Console.WriteLine($"Кількість стовпців: {columns}");

for (int i = 0; i < rows; i++)
{
    for (int j = 0; j < columns; j++)
        Console.Write(matrix[i, j] + " ");
    Console.WriteLine();
}
```



Приклад

```
int[,] matrix = new int[,] { { 1, 2, 3 }, { 4, 5, 6 } };

for (int i = 0; i < matrix.GetLength(0); i++)
{
    for (int j = 0; j < matrix.GetLength(1); j++)
        Console.Write(matrix[i, j] + " ");

    Console.WriteLine();
}
Console.WriteLine(matrix.Rank);
```



Генерація псевдовипадкових чисел

Потрібно один раз оголосити об’єкт типу Random:

```
Random rnd = new Random();
```

Далі, щоб отримати ціле число треба викликати метод Next

Генерація псевдовипадкових чисел

Основні методи класу Random:

Next() - повертає наступне випадкове ціле число в розділі від 0 до Int32.MaxValue.

Next(maxValue) - повертає наступне випадкове ціле число в розділі від 0 до maxValue-1.

Next(minValue, maxValue) - повертає наступне випадкове ціле число в області від minValue до maxValue-1.

NextDouble() - повертає наступне випадкове число з плаваючою комою в діапазоні від 0,0 до 1,0.:

Генерація псевдовипадкових чисел

Щоб отримати дробове число потрібно викликати метод NextDouble:

```
double d = rnd.NextDouble();
```

Число генерується у діапазоні: **[0; 0.99999999999999978]**

Для заокруглення дробового числа d до k-ого знаку:

```
double dNew = Math.Round(d, k);
```

Приклад

```
 Random random = new Random();

// Генерація цілого числа від 0 до 100
int randomNumber = random.Next(101);
Console.WriteLine("Випадкове ціле число: " + randomNumber);

// Генерація числа з плаваючою точкою від 0.0 до 1.0
double randomDouble = random.NextDouble();
Console.WriteLine("Випадкове число з плаваючою комою: " + randomDouble);
```

Приклад

```
int[] array = new int[10];

Random random = new Random();

// Заповнення масиву випадковими числами від -10 до 10
for (int i = 0; i < array.Length; i++)
    array[i] = random.Next(-10, 11);

foreach (int number in array)
    Console.Write(number+" ");
```

Створення багатовимірного масиву

C# дозволяє створювати і тривимірні масиви.

Синтаксис:

тип_даних[, ,] ім'я_масиву = new тип_даних[розмір1, розмір2, розмір3];

Доступ до елементів:

```
int value = numbers[1, 2, 0]; //Отримання значення елемента з індексами 1, 2, 0
```

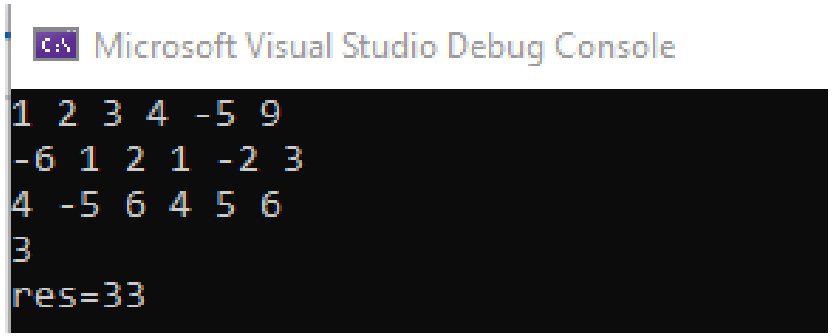
Приклад

```
// Масив цілих чисел розміром 3x4x2
int[,,] numbers = new int[3, 4, 2];

// Масив дробових чисел розміром 2x2x2
double[,,] values = new double[2, 2, 2];

// Масив рядків розміром 5x3x4
string[,,] words = new string[5, 3, 4];
```

```
int[,,] numbers = new int[3, 4, 2]
{
    { {1, 2}, {3, 4}, {5, 6}, {7, 8} },
    { {9, 10}, {11, 12}, {13, 14}, {15, 16} },
    { {17, 18}, {19, 20}, {21, 22}, {23, 24} }
};
```

Приклад

```
int res = 0;
int[,,] matr3D = new int[,,] {{{1,2,3},{4,-5,9}},{{-6,1,2},{1,-2,3}}, {{4,-
5,6},{4,5,6}}};

for (int i = 0; i < matr3D.GetLength(0); i++)
{
    for (int j = 0; j < matr3D.GetLength(1); j++)
        for (int k = 0; k < matr3D.GetLength(2); k++)
            Console.Write(matr3D[i, j, k] + " ");
    Console.WriteLine();
}
Console.WriteLine(matr3D.Rank);
for (int i = 0; i < matr3D.GetLength(0); i++)
    for (int j = 0; j < matr3D.GetLength(1); j++)
        for (int k = 0; k < matr3D.GetLength(2); k++)
            res += matr3D[i, j, k];
Console.WriteLine("res={0}",res);;
```



Зубчасті масиви

Якщо потрібно зберігати різну кількість елементів у рядках матриці, тоді використовують зубчасті масиви (масиви масивів, рвані масиви).

Синтаксис оголошення:

тип_даних[ ][ ] ім'я_масиву;

Зубчасті масиви - це масиви, елементами яких є інші масиви, можливо, різної довжини

Доступ до елементів:

```
int value = myJaggedArray[1][0]; // Отримання значення елемента з індексами 1, 0
```

Зубчасті масиви

Масив масивів

Кількість рядків

```
int[][] myJaggedArray = new int[3][];
  myJaggedArray[0] = new int[3];
  myJaggedArray[1] = new int[2];
  myJaggedArray[2] = new int[7];
```

Кількість елементів у кожному рядку

Ініціалізація зубчастого масиву

```
int[][] myJaggedArray = new int[][] {
new int[] {1, 2, 3},
new int[] {4, 5},
new int[] {6, 7, 8, 9}
```

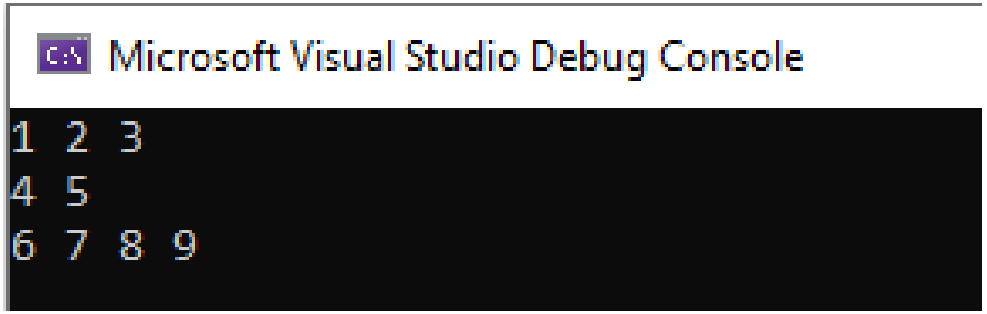
Пряма ініціалізація

Динамічна ініціалізація

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

Приклад

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

for (int i = 0; i < numbers.Length; i++)
{
    for (int j = 0; j < numbers[i].Length; j++)
        Console.Write(numbers[i][j] + " ");
    Console.WriteLine();
}
```



Приклад

```
string[][] students = new string[][] {
    new string[] {"Олександр", "Олексій", "Омелян", "Орест"},
    new string[] {"Діна", "Давід", "Діана"},
    new string[] {"Анісія", "Анна"}
};

for (int i = 0; i < students.Length; i++)
{
    for (int j = 0; j < students[i].Length; j++)
        Console.Write(students[i][j] + " ");

  Console.WriteLine();
}
```