Модуль 5: Массивы

Обзор

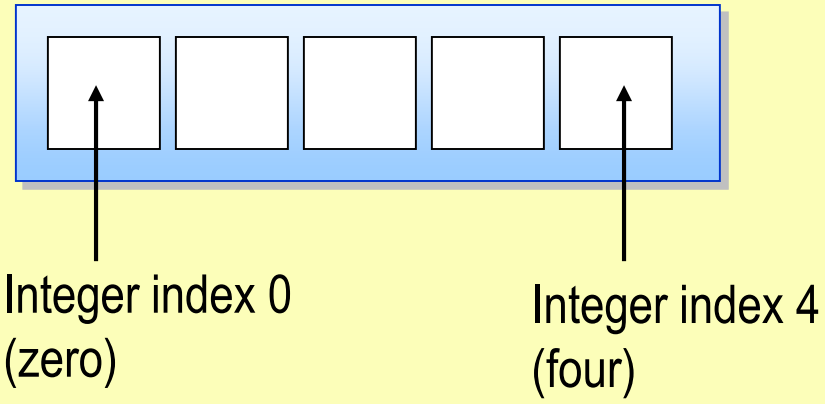
- Обзор массивов
- Создание массивов
- Использование массивов

Обзор массивов

- Что такое массив?
- Форма записи массива в C#
- Ранг массива
- Организация доступа к элементам массива
- Проверка границ массива
- Сравнение массивов и коллекций

Что такое массив?

- **Массив –это последовательный набор элементов**
  - Все элементы массива одного типа
  - Структуры могут хранить элементы разных типов
  - Доступ к конкретным элементам массива происходит через использование индекса

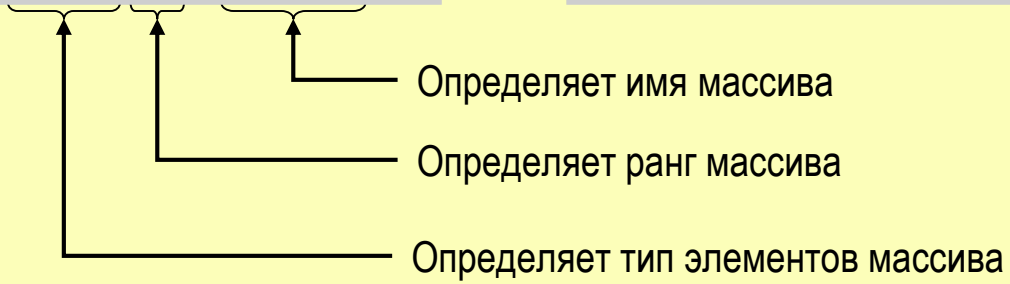


Форма записи массива в C#

- **При объявлении массива необходимо определить следующее:**
  - Тип элементов массива
  - Ранг массива
  - Имя массива

```
type[ ] name;
```

```
int [] arr;
```

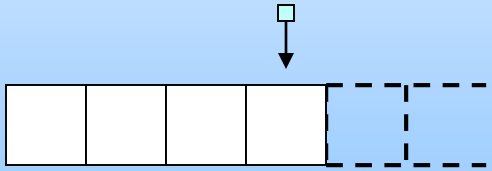


Ранг массива

- **Вместо слова ранг иногда говорят размерность**
- **Количество индексов, определяющих каждый из элементов**

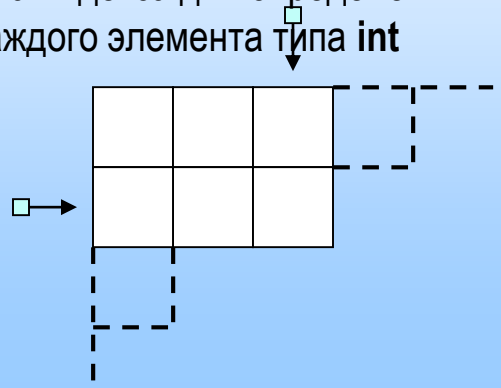
```
long[ ] row;
```

Ранг 1: Одномерный
Один индекс для определения каждого элемента типа **long**



```
int[,] grid;
```

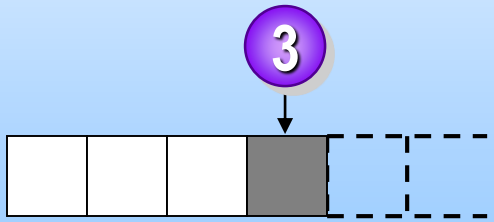
Ранг 2: Двумерный
Два индекса для определения каждого элемента типа **int**



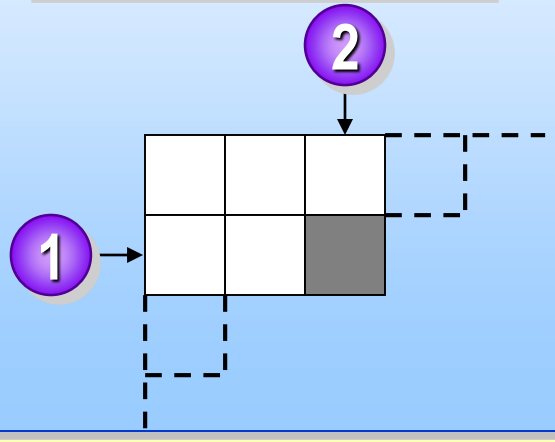
Организация доступа к элементам массива

- **Определяйте индекс для каждой из размерностей**
  - Индекс первого элемента равен нулю

```
long[ ] row;
...
row[3];
```

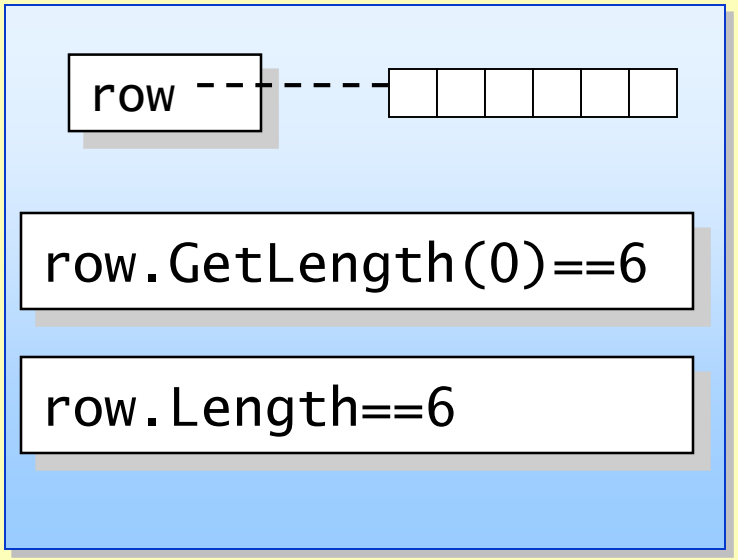


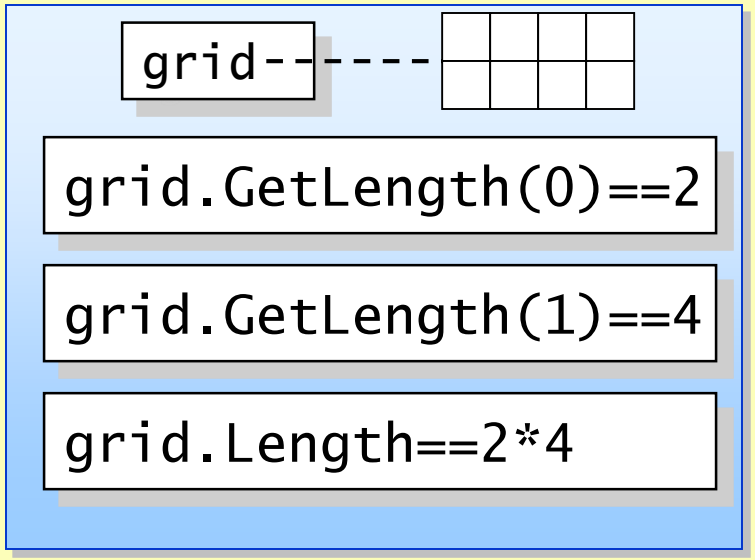

```
int[,] grid;
...
grid[1,2];
```



Проверка на отсутствие нарушения границ массива

- **При попытке доступа к элементу массива производится проверка на отсутствие нарушения границ**
  - При использовании несуществующего индекса выбрасывается исключение IndexOutOfRangeException
  - Используйте свойство **Length** и метод **GetLength**





Выход за границы массива

- **Выход за границы массива в C# расценивается как ошибка, в ответ на которую генерируется исключение - IndexOutOfRangeException**

```
static void Main()
{
  int[] myArr = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
  int i;
  try
  {
for (i = 0; i <= 10; i++) Console.WriteLine(myArr[i]);
  }
  catch (IndexOutOfRangeException)
  {
Console.WriteLine("Exception: Выход за границу");
  }
}
```

Сравнение массивов и коллекций

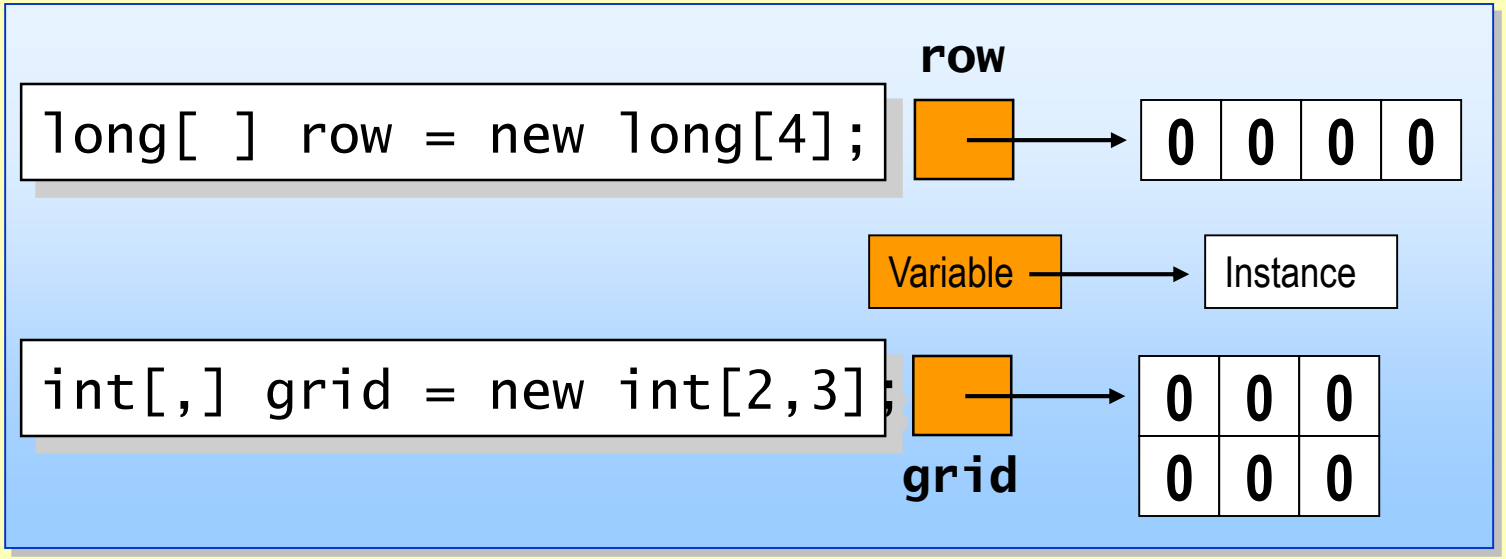
- **Если массив переполниться, его размер нельзя изменить**
  - Класс коллекции, например ArrayList, может менять размеры
- **Массивы используются для хранения однотипных данных**
  - Коллекции могут хранить данные разных типов
  - Коллекция может быть открыта только для чтения
- **Массивы быстрее, но менее гибкие**
  - Коллекции немного медленнее, но более гибкие

Создание массивов

- Создание экземпляров массивов
- Инициализация массивов
- Инициализация многомерных массивов
- Создание массива вычисляемого размера
- Копирование переменных массива

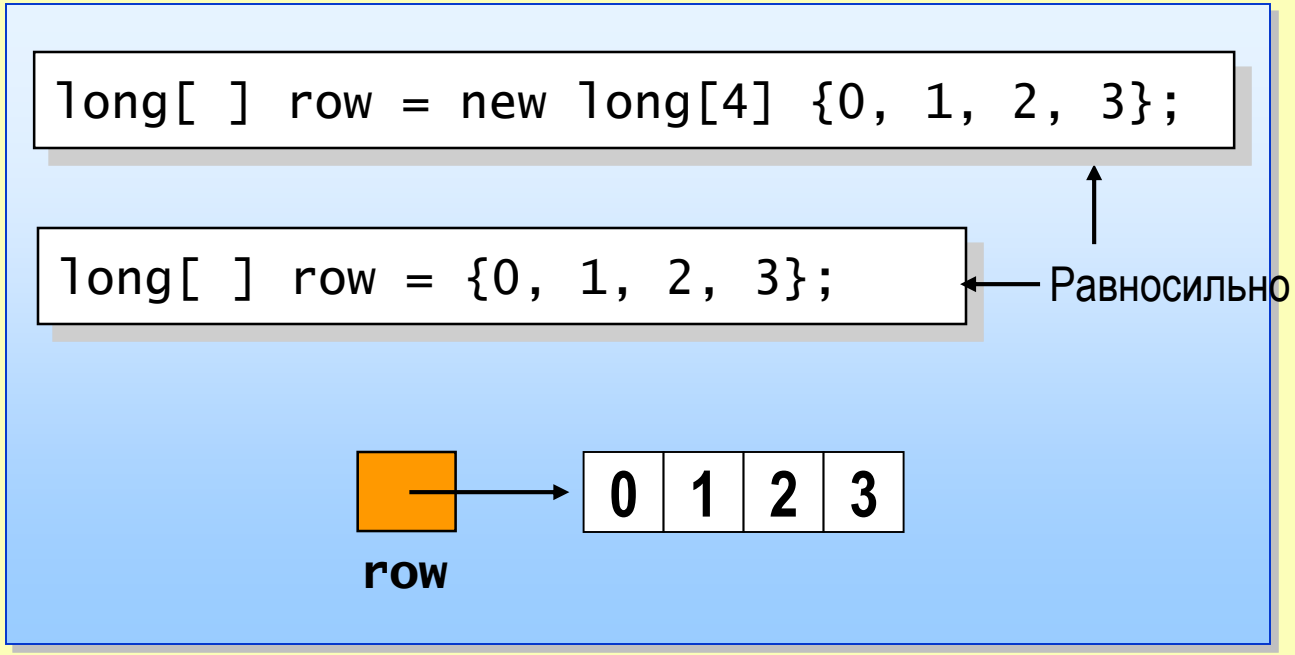
Создание экземпляров массивов

- **Объявление массива не приводит к созданию массива**
  - Для создания массива необходимо использовать слово **new**
  - Все элементы массива имеют значение по умолчанию, равное нулю



Инициализация массивов

- **Можно явным образом инициализировать массивы**
  - Можно использовать удобную краткую запись



Использование при инициализации массива операции new

- **Данная форма инициализации массива может оказаться полезной в случае, когда уже существующей ссылке на массив присваивается ссылка на новый массив.**

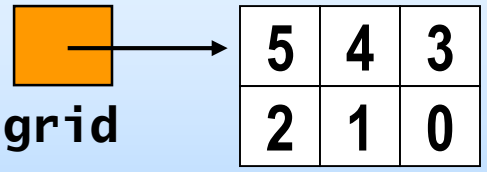
```
static void Main()
{
  int[] myArray = { 0, 1, 2, 3, 4, 5};
  int i;
  for (i = 0; i < 6; i++)
  Console.Write(" "+myArray[i]);
  Console.WriteLine("\nНовый массив: ");
  myArray = new int[] { 9, 1, 10, 8, 7, 3, 16, 5, 87, 4 };
  for (i = 0; i < 10; i++)
  Console.Write(" " + myArray[i]);
}
```

Инициализация многомерных массивов

- **Можно инициализировать и многомерные массивы**
  - Необходимо определить все элементы

```
int[,] grid = {
        {5, 4, 3},
        {2, 1, 0}
};
```

Неявно new int[2,3] массив




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

Создание массивов вычисляемого размера

- **Размер массива не обязан быть константой, определяемой на этапе компиляции**
  - Это может быть любое целочисленное выражение
  - Доступ к элементам будет таким же быстрым

Размер массива, определяемый целочисленной константой на этапе компиляции:

```
long[ ] row = new long[4];
```

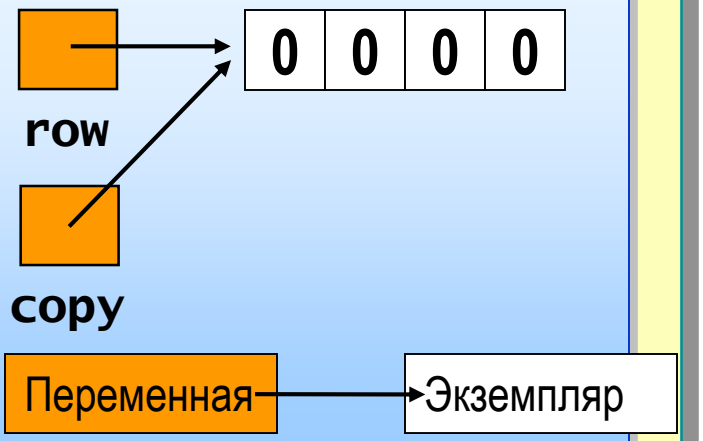
Размер массива, определяемый целочисленным выражением на этапе выполнения:

```
string s = Console.ReadLine();
int size = int.Parse(s);
long[ ] row = new long[size];
```

Копирование переменных массива

- **При копировании переменной массива копируется только переменная массива**
  - Не создается копия экземпляра массива
  - Две переменные массива могут ссылаться на один и тот же экземпляр массива

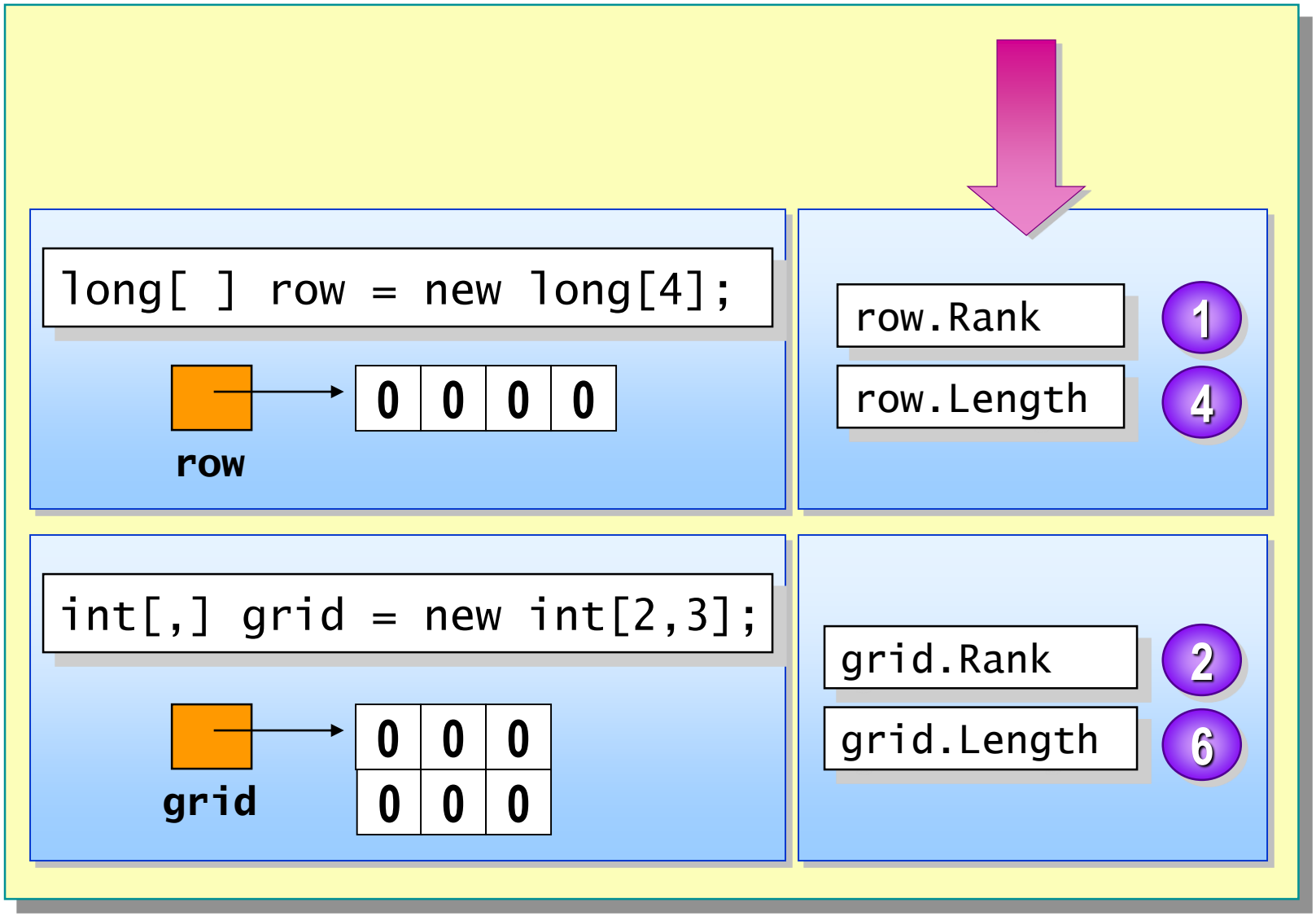
```
long[ ] row = new long[4];
long[ ] copy = row;
...
row[0]++;
long value = copy[0];
Console.WriteLine(value);
```



Использование массивов

- Свойства массива
- Методы массива
- Возвращение массивов из методов
- Передача массивов в качестве параметров
- Аргументы командной строки
- Использование массивов с циклом foreach

Свойства массивов



Методы массивов, класс Array

- **Часто используемые методы**
  - **Sort** – сортирует элементы одномерного массива
  - **Clear** – устанавливает для указанного диапазона элементов значения **0, false** или **null**
  - **Copy** – создает копию массива
  - **IndexOf** – возвращает индекс впервые встречаемого значения

Передача массивов в качестве параметров

- **Параметр-массив является копией переменной массива**
  - Не копией экземпляра массива

```
class Example2 {
    static void Main( ) {
        int[ ] arg = {10, 9, 8, 7};
        Method(arg);
        System.Console.WriteLine(arg[0]);
    }
    static void Method(int[ ] parameter) {
        parameter[0]++;
    }
}
```

Этот метод изменит настоящий экземпляр массива, созданного в Main

Возвращение массивов из методов

- **Можно объявлять методы, возвращающие массивы**

```
class Example {
    static void Main( ) {
        int[ ] array = CreateArray(42);
        ...
    }
    static int[ ] CreateArray(int size) {
        int[ ] created = new int[size];
        return created;
    }
}
```

Аргументы командной строки

- **В процессе выполнения аргументы командной строки передаются в Main**
  - **Main** может принимать в качестве параметра массив из строк
  - Имя программы не является членом массива

```
class Example3 {
    static void Main(string[ ] args) {
        for (int i = 0; i < args.Length; i++) {
            System.Console.WriteLine(args[i]);
        }
    }
}
```

Использование массивов с циклом foreach

- **Цикл foreach позволяет скрыть подробности перебора элементов массива**
- **Синтаксис:**

```
foreach (<тип> <имя> in <группа>)
{<тело цикла>}
```

```
class Example {
    static void Main(string[ ] args)
{
   foreach (string arg in args)
      {
   System.Console.WriteLine(arg);
      }
}
            }
```

Массивы массивов

Jagged arrays - изрезанные (ступенчатые)

- **одномерный массив, элементы которого являются массивами, элементы которых, в свою очередь, снова могут быть массивами, и так может продолжаться до некоторого уровня вложенности.**
- **Объявление ступенчатого массива:**

```
тип [][] имя_массива;
```

Созданием массивов с инициализацией

- **Требуется вызывать конструктор для каждого массива на самом нижнем уровне**

```
int[][] jagger = new int[3][]
{
   new int[] {5,7,9,11},
   new int[] {2,8},
   new int[] {6,12,4}
};
```

  - ИЛИ

```
int[][] jagger1 = new int[3][]
{
   new int[4],
   new int[2],
   new int[3]
};
```