



6. Структури от данни в .NET

Част първа
Масиви, колекции, списъци

Мирослав Недялков



Съдържание (1/2)

- Въведение
- Малко теория - колекциите
- Масиви в .NET
 - Едномерни
 - Многомерни
 - Назъбени (jagged)
 - Класът System.Array



Съдържание (2/2)

- Колекции и списъци в .NET – интерфейсите
 - I Enumerable
 - I Collection
 - I List
 - I Dictionary
- Колекции и списъци в .NET – пространството от имена System.Collection



Какво е структура от данни?

- Дефиниция
- Типове структури от данни
 - Статични и динамични
 - Хомогенни и хетерогенни
 - Линейни и разклонени
- Примери за структури от данни
- Защо е полезно да използваме структурите от данни в .NET Framework?



Какво са колекциите?

- Структури от данни, които представляват множество обекти
- Обектите в колекциите обикновенно са от един и същ вид
- Колекциите могат да са
 - за писане и четене или само за четене
 - с фиксиран размер или с променлив размер
 - с произволен или последователен достъп до елементите



А масивите?

- Колекции с фиксиран размер, позволяващи промяна на елементите и произволен достъп до тях
- Най-често срещаните колекции вградени директно в езиците за програмиране



Има ли масиви в .NET?

- Разбира се!
 - Имат посочените до тук характеристики на масивите по принцип
 - Делят се на няколко вида:
 - Едномерни
 - Многомерни
 - Масиви от масиви
 - Имат специален синтаксис за деклариране и създаване



Масиви в .NET (1/2)

- Съдържат еднотипни обекти, като типа се определя при създаване на масива
- Референтен тип данни
- Наследници на **System.Array**
- Имплементират интерфейсите **ICloneable**, **IList**, **ICollection** и **IEnumerable**



Масиви в .NET (2/2)

- Достъп до елементите – по индекс
- Номерацията обикновенно започва от 0
- Достъпът до елементите е проверен



Едномерни масиви (1/2)

- Декларация

```
int[] integerArray;
```

- Инстанциране

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



Едномерни масиви (2/2)

- Достъпът до елементите на масивите
 - Индексиране - []
 - За четене и писане

Пример:

```
int[] integerArray = new int[10];
integerArray[0] = 1; //писане
int firstOne = integerArray[3]; //чтение
```



Многомерни масиви (1/3)

- Какво е ранг?
- Размерност на едномерните масиви
- Масиви с по-голям ранг
 - Декларация
 - Създаване
 - Достъп до елементите



Многомерни масиви (2/3)

- Декларация

```
int[,] integerArray; //двумерен масив от цели числа
```

- Създаване

```
int[,] integerArray =  
    new int[3,3,3]; //тристимерен масив от цели числа
```

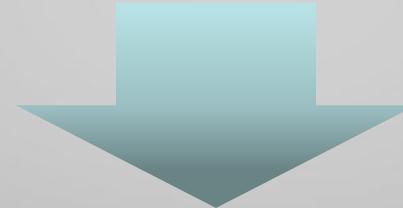
```
integerArray[0,1,1] = 1; //писане  
int firstOne = integerArray[2,2,0]; //чертене
```



Многомерни масиви (3/3)

- ## • Разположение в паметта

```
int[,] integerArray = new int[3,3];
```





Инициализация

- Автоматична инициализация
- Инициализация с конкретни стойности

Пример:

```
// Едномерен масив с 3 елемента, който съдъжа
// числата 1, 2 и 3:
int[] arr = new int[] { 1, 2, 3};

// Двумерен масив с два реда, всеки от които съдъжа
// по 2 елемента (1, 2 и 3, 4)
int[,] arr1 = new int[,] { {1, 2}, {3, 4} };
```



(еди
чи)





Масиви от масиви (1/4)

- Декларация:

```
int[][] jaggedArray;
```

- Създаване:

```
// Декларация и създаване на основния масив.  
// Стойностите на елементите му по подразбиране са  
// null  
int[][] jaggedArray = new int[2][];  
// Създаване на двата масива  
jaggedArray[0] = new int[3];  
jaggedArray[1] = new int[2];
```



Масиви от масиви (2/4)

- Достъп до елементите:

```
int[][] jaggedArray = new int[2][];
jaggedArray[0] = new int[] { 1, 2, 3 };
jaggedArray[1] = new int[] { 4, 5 };
jaggedArray[0][1] = 3;
foreach (int[] array in jaggedArray)
{
    foreach (int value in array)
    {
        Console.Write(value + ", ");
    }
    Console.WriteLine();
}
```



Масиви от масиви (3/4)

- Разположение в паметта

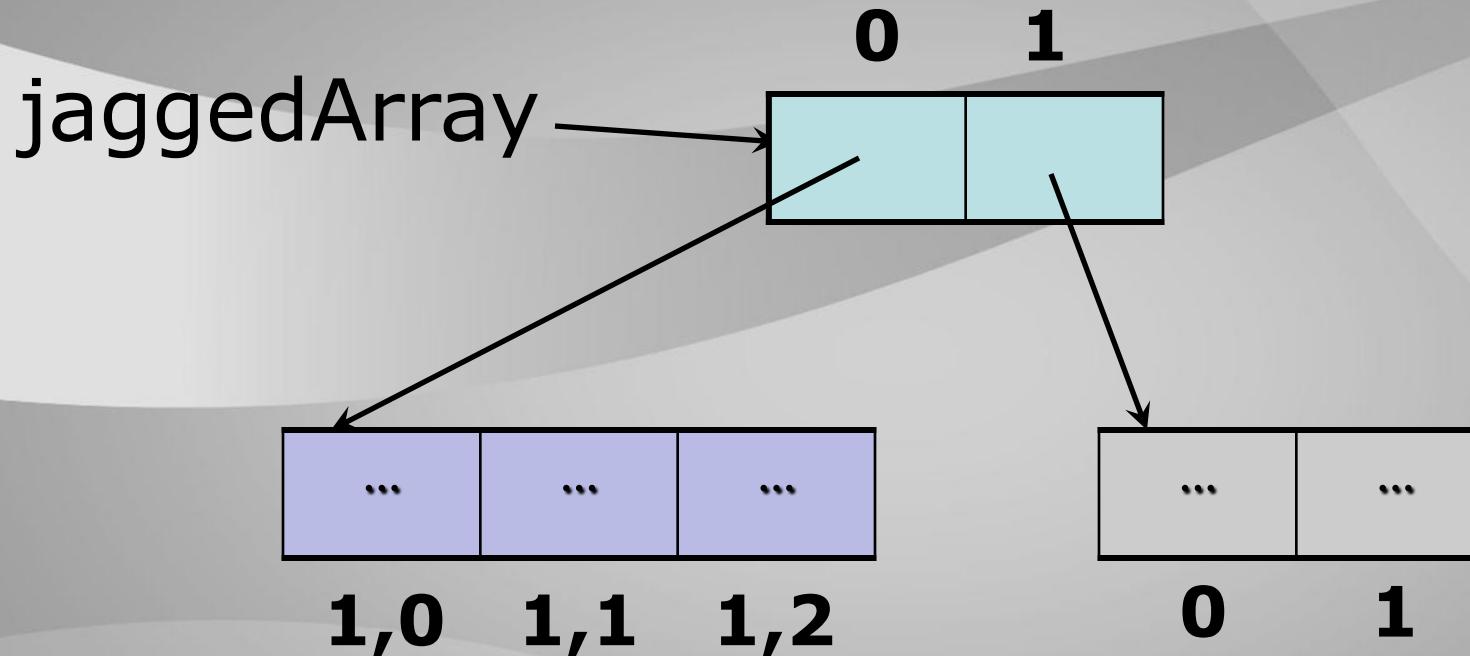
```
int[][] jaggedArray = new int[2][];  
jaggedArray[0] = new int[] { 1, 2, 3 };  
jaggedArray[1] = new int[] { 4, 5 };
```

ИЛИ

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



Масиви от масиви (4/4)





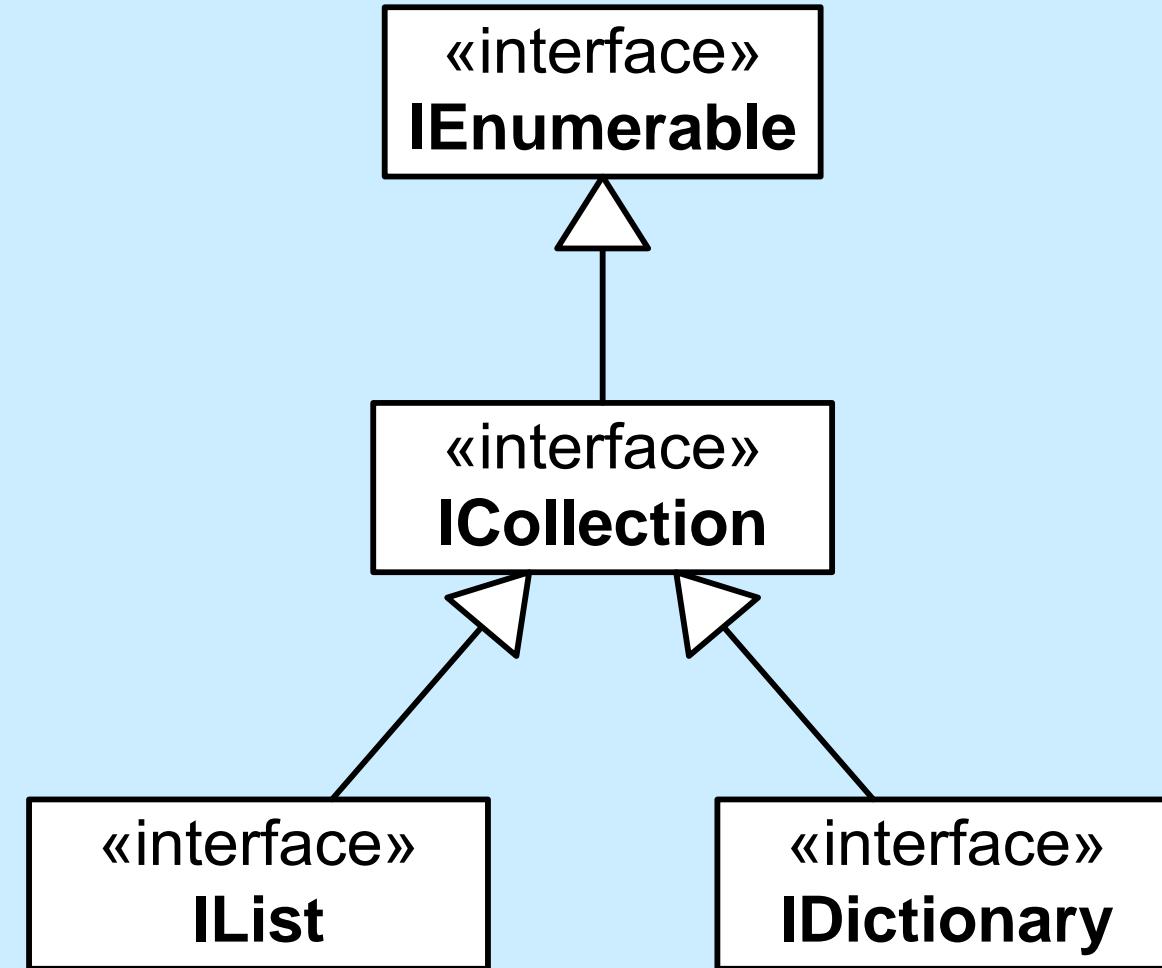


Колекции (1/2)

- Контейнери за обекти
- **ICollection** и **IEnumerable**
- Видове
 - Списъчни - имплементират **IList**
 - Речникови - имплементират **IDictionary**
- Пространство от имена **System.Collections**



Колекции (2/2)





Нетипизирани?!?

- Колекциите, които имплементират **ICollection** са нетипизирани!
 - Несигурни
 - Бавни (boxing + unboxing)
 - Неудобни (преобразуване на типове)
- Има изход!
 - Пространството от имена **System.Collections.Generic**



Какво научихме до тук?

- Колекциите са контейнери за обекти
- Масивите са силно типизирани колекции с достъп до елементите по индекс и фиксиран размер
- Слаботипизираните колекции създават проблеми



Въпроси?

