Тема 6. Массивы

# Парадигмы программирования. C#

# План

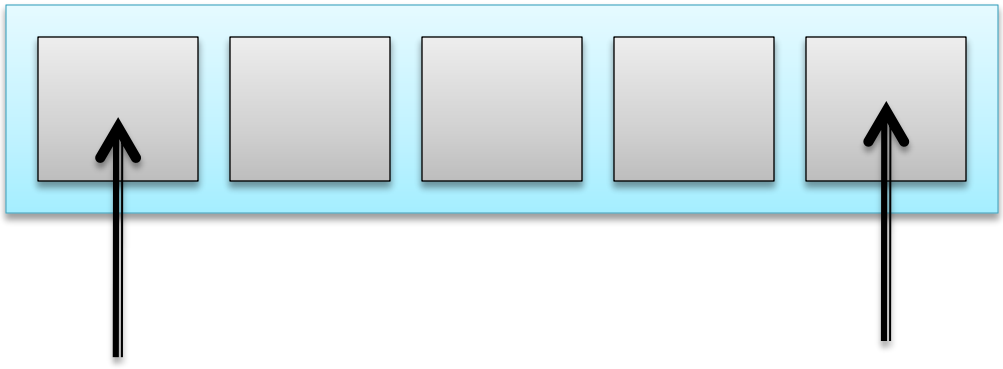
- Введение в массивы
- Создание массивов
- Использование массивов
- Коллекции

# Введение в массивы

Часть 1

# Что такое массив?

- Массив – это именованный набор однотипных переменных, расположенных в памяти непосредственно друг за другом (в отличие от списка), доступ к которым осуществляется по индексу
  - Все элементы массива должны иметь одинаковый тип данных
  - Доступ к каждому элементу осуществляется по его целочисленному индексу



Целочисленный индекс 0

Целочисленный индекс 4

# Синтаксис массивов в C#

- Объявление массива содержит
  - Тип элементов массива
  - Размерность массива
  - Идентификатор переменной

```
Type[] arrayName;
```

# Размерность массива

- Размерность – это количество индексов массива
- Одномерный массив

```
int[] row;
```

- Двумерный массив

```
int[,] table;
```

# Доступ к элементам массива

- Доступ осуществляется указанием индекса каждой размерности
- Индексы начинаются с 0
- Одномерный массив

```
int[] row;
…
row[4] = 5;
```

- Двумерный массив

```
int[,] table;
…
table[2, 3] = 8;
```

# Проверка границ массива

- Все попытки доступа к элементам массива обязательно контролируются на выход за пределы границ массива
  - При неправильно указанном индексе выдаётся исключение `IndexOutOfRangeException`
  - Для проверки правильности индекса используйте свойство `Length` и метод `GetLength`

# Статические и динамические массивы

- Длина обычного массива не может быть изменена после его инициализации, т.е. обычный массив в C# является статическим
- Объект класса ArrayList – это динамический массив, т.е. массив с переменной длиной
- В массиве хранятся элементы одного типа
- В ArrayList можно хранить элементы разных типов
- Массивы работают быстрее ArrayList, но являются менее гибкими
- ArrayList рассмотрен в части 4

# Создание массивов

Часть 2

# Создание экземпляра массива

- Объявление массива не приводит к созданию массива
- Массивы – это ссылочные типы данных и содержат данные не внутри себя, а в куче, что требует их явного создания
- Для создания экземпляра массива следует использовать ключевое слово `new`
- После создания экземпляра массива его элементы неявно принимают значение нуля
- Пример
  - Одномерный массив

```
int[] row = new int[5];
```

  - Двумерный массив

```
int[,] table = new int [4, 6];
```

# Инициализация элементов массивов

- Элементы массива можно задавать явно
- Примеры эквивалентной инициализации массивов

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

# Инициализация элементов многомерных массивов

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

- или

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

- Или

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

# Создание массивов с рассчитываемой длиной

- Размер массива должен быть известен на момент инициализации во выполнения программы
- Нет необходимости использовать константы
- Любое правильное целочисленное выражение подойдёт

```
string s = Console.ReadLine();
int size = Int32.Parse(s);
int[] row = new int[size * 8];
```

# Копирование массивов

- Копирование переменной массива приводит к копировании только ссылки
  - Данные массива не копируются
  - Две переменные массива могут обозначать одни и те же данные

```
int[] row = { 0, 1, 2, 3, 4 };
int[] nextRow = row;
nextRow[0] = 8;
```

# Использование массивов

Часть 3

# Свойства массивов

- Любой массив неявно является наследником класса Array
  - Длина массива `Length`
  - «Длинная» длина массива `LongLenght`
  - Размерность массива `Rank`
- Дополнительная информация http://msdn.microsoft.com/en-us/library/system.array_members(VS.100).aspx

# Методы массивов

- `Sort` – сортирует элементы одномерного массива
- `Clear` – устанавливает значения элементов в `0` или `null`
- `Clone` – создаёт копию массива
- `GetLength` – возвращает длину массива
- `IndexOf` – возвращает индекс первого найденного элемента с заданным значением

# Возврат массивов из методов

- Методы могут возвращать массивы

```
class Example
{
    static int[] CreateArray(int size)
    {
        int[] created = new int[size];
        return created;
    }
    static void Main()
    {
        int[ ] array = CreateArray(28);
    }
}
```

# Передача массивов в качестве параметров

- Параметр метода является переменной массива, т.е. ссылкой на данные массива
- При передаче массива в качестве параметра метода данные массива не копируются
- Изменения данных массива внутри метода будут видны после завершения его работы

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

# Аргументы командной строки

- Аргументы командной строки передаются в качестве параметра метода Main
- Параметр метода Main – это массив строк
- Название исполняемого файла не включается в этот массив

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

# Использование массивов в цикле foreach

- Массивы поддерживают работу с циклом foreach

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

# Коллекции

Часть 4

# Особенности динамических массивов

- Коллекции – это различные структуры с памяти, связанные с хранением набора данных
- Коллекции размещены в области имён System.Collections http://msdn.microsoft.com/en-us/library/system.collections.aspx
- Примеры: ArrayList, SortedList, Stack, Queue, BitArray, Hashtable

# Работа с ArrayList

```
ArrayList list = new ArrayList();
list.Add("Hello");
list.Add(123);
list.Add(true);
list.AddRange(new int[] { 1, 2, 3 });
list.RemoveAt(0);
list.RemoveRange(3, 2);
int listCurrentSize = list.Count;
list.BinarySearch(3);
list.Clear();
```

- Дополнительная информация http://msdn.microsoft.com/en-us/library/system.collections.arraylist.aspx

# Выводы

- Обычные массивы в C# являются статическими
- C# поддерживает одномерные и многомерные массивы
- На самом деле массивы – это наследники класса Array, поэтому могут пользоваться его методами и свойствами
- Для работы с динамическими массивами используйте ArrayList
- Для работы с другими типовыми конструкциями в памяти (FIFO, LIFO, хэш-таблицы) используйте готовые типы данных

# Заключение

- Спасибо за внимание!