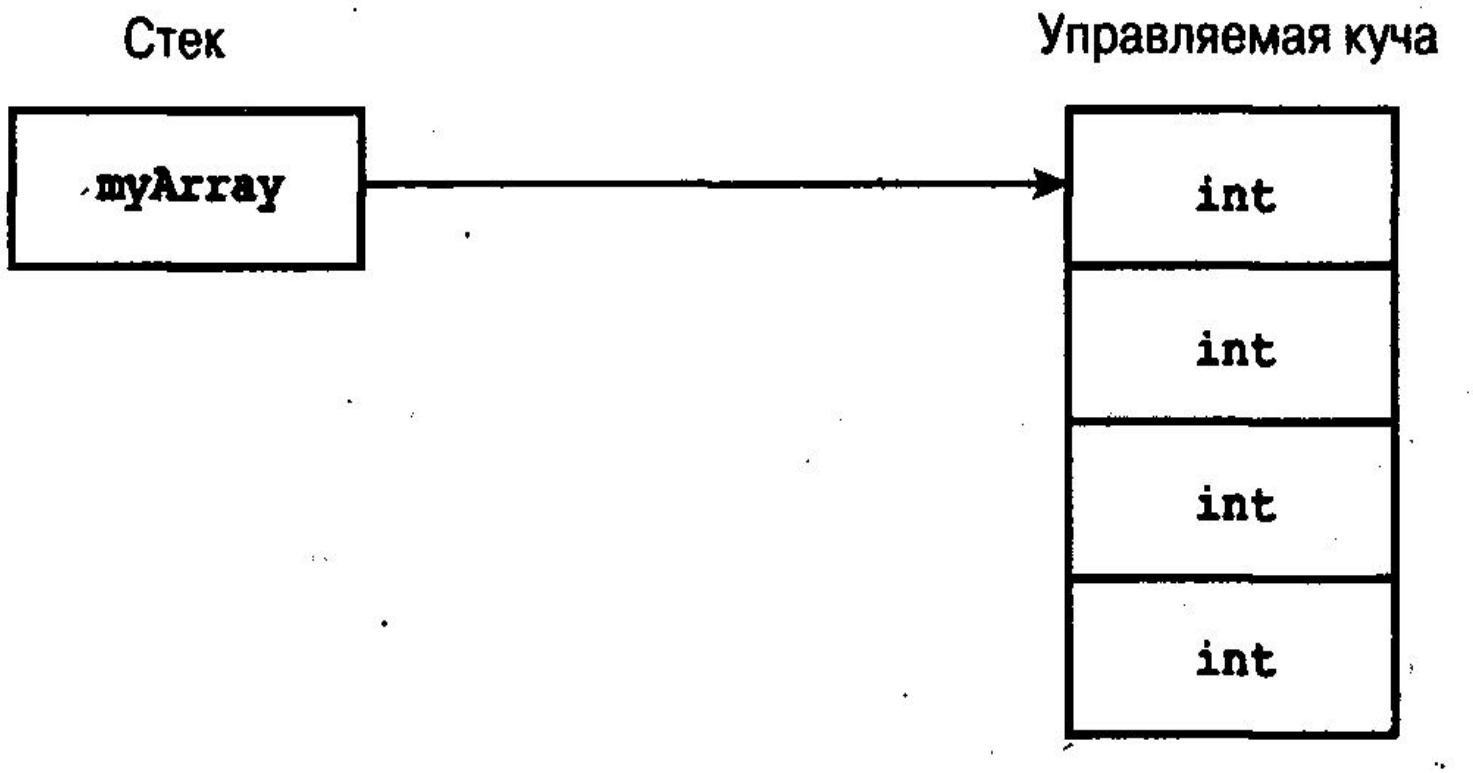
1) int[] myArray;
2) myArray = new int [4] ;
3)



Переменная myArray ссылается на четыре целочисленных значения в управляемой куче

4) int[] myArray = new int[4];

5) int [] myArray = new int[4] {4, 7, 11, 2 };

6) int[] myArray = new int[] {4, 7, 11, 2};

7) int[] myArray = {4, 7, 11, 2};

8)

```
int[] myArray = new int[] {4, 7, 11, 2};
int v1 = myArray[0]; // читать первый элемент
int v2 = myArray[1]; // читать второй элемент
myArray [3] = 44; // изменить четвертый элемент
```

9)

```
for (int i = 0; i < myArray.Length; i++)
{
      Console.WriteLine(myArray[i]);
}
```

10)

```
foreach (int  val in  myArray)
{
  Console.WriteLine(val);
}
```

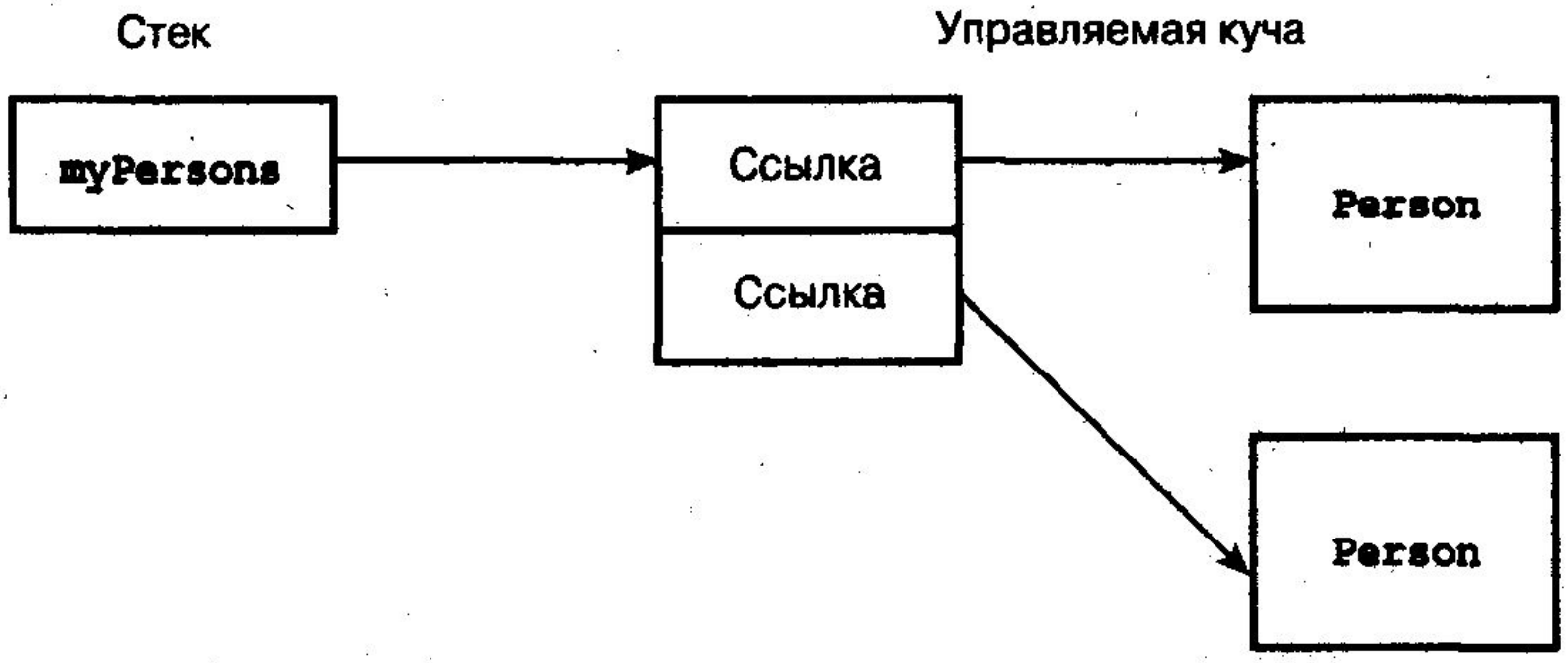
11)

```
public class Person
{
   public string FirstName { get; set; }
   public string LastName { get; set; }
   public override string ToString()
   {
      return String.Format("{0} {1}", FirstName, LastName);
   }
}
```

12) Person[] myPersons = new Person[2];

13)

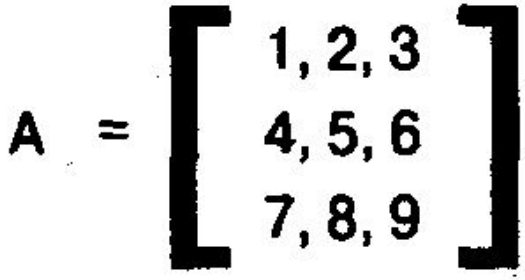
```
new Person { FirstName="Ayrton", LastName="Senna" };
new Person { FirstName="Michael", LastName="Schumacher" };
```



Объекты в управляемой куче, относящиеся к массиву Person

15)

```
Person[] myPersons2 =
{
  new Person { FirstName="Ayrton", LastName="Senna"},
  new Person { FirstName="Michael", LastName=”Schumacher"}
};
```



Математическое обозначение двумерного массива

17)

```
int[,]  twodim = new int[3,3] ;
twodim[0, 0] = 1;
twodim[0, 1] = 2;
twodim[0, 2] = 3;
twodim[1, 0] = 4;
twodim[l, 1] = 5;
twodim[1, 2] = 6;
twodim[2, 0] = 7;
twodim[2, 1].= 8;
twodim[2, 2] = 9;
```

18)

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

19)

```
int[,,] threedim = {
                  { { 1, 2 }, { 3, 4 } },
                  { { 5, 6 }, { 7, 8 } },
                  { { 9, 10 }, { 11, 12 } }
                };
Console.WriteLine(threedim[0, 1, 1]);
```

Двумерный массив

1	2	3
4	5	6
7	8	9

Зубчатый массив

1	2				
3	4	5	6	7	8
9	10	11			

Различие между обычным двумерным и зубчатым массивом

21)

```
int[] [ ] jagged = new int[3] [ ] ;
jagged[0] = new int [2] { 1, 2 };
jagged[l] = new int[6] { 3, 4, 5, 6, 7, 8 };
jagged[2] = new int[3] { 9, 10, 11 };
```

22)

```
for (int row = 0; row < jagged.Length; row++)
{
  for (int element = 0; element < jagged[row].Length; element++)
  {
    Console.WriteLine("строка: {0}, элемент: {1}, значение: {2}",
                      row, element, jagged[row][element]);
  }
}
```

23)

```
строка:  0, элемент: 0, значение: 1
строка:  0, элемент: 0, значение: 2
строка:  1, элемент: 0, значение: 3
строка:  1, элемент: 1, значение: 4
строка:  1, элемент: 2, значение: 6
строка:  1, элемент: 3, значение: 1
строка:  1, элемент: 4, значение: 7
строка:  1, элемент: 5, значение: 8
```

строка: 2, элемент: 0, значение: 9
строка: 2, элемент: 1, значение: 10
строка: 2, элемент: 2, значение: 11

24)

```
Array intArray1 = Array.CreateInstance (typeof (int), 5) ;
for (int i = 0;- i < 5; i++)
{
  intArray1.SetValue(33, i);
}
for (int i = 0; i < 5; i++)
{
  Console.WriteLine(intArray1.GetValue(i));
}
```

25) int[] intArray2 = (int[])intArray1;

26)

```
int[] lengths = {2, 3};
int[] lowerBounds = {1, 10};
Array racers = Array.CreateInstance(typeof(Person), lengths,
                                    lowerBounds);
```

27)

```
racers.SetValue (new Person
{
   FirstName = "Alain",
   LastName = "Prost"
}, 1, 10);
racers.SetValue(new Person
{
   FirstName = "Emerson",
   LastName = "Fittipaldi"
}, 1, 11);
racers.SetValue(new Person {
FirstName = "Ayrton",
LastName = "Senna"
}, 1, 12);
racers.SetValue(new Person
{
   FirstName = "Ralf",
   LastName = "Schumacher"
), 2, 10);
```

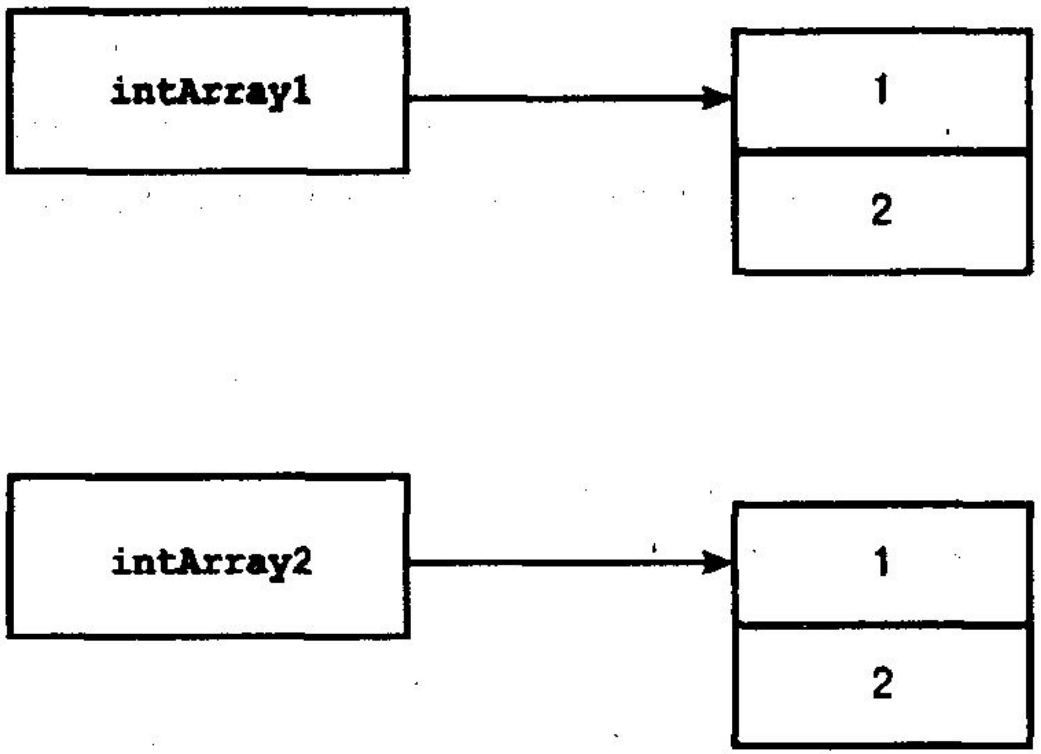
```
racers.SetValue(new Person
{
   FirstName = "Fernando",
   LastName = "Alonso"
}, 2, 11);
racers.SetValue(new Person
{
   FirstName = "Jenson",
   LastName = "Button"
}, 2, 12);
```

28)

```
Person[,] racers2 = (Person[,])racers;
Person first = racers2[l, 10];
Person last = racers2[2, 12];
```

29)

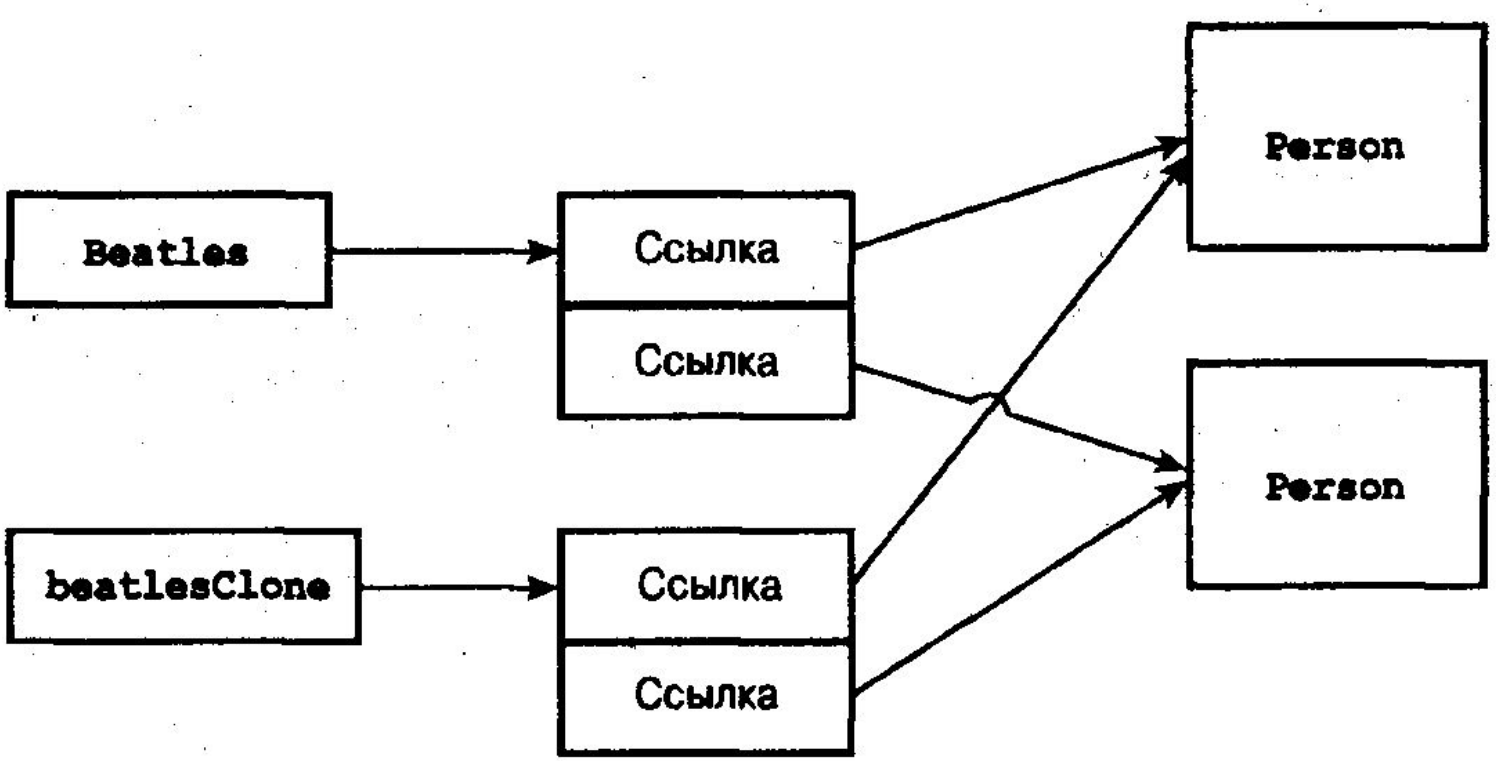
```
int[] intArray1 = {1, 2);
int[] intArray2 = (int[])intArrayl.Clone();
```



Копирование массива с элементами типа значений

31)

```
Person[] beatles = {
            new Person {FirstName="John", LastName=''Lennon"},
            new Person { FirstName="Paul", LastName="McCartney"}
           } ;
Person[] beatlesClone = (Person[])beatles.Clone();
```



Копирование массива с элементами ссылочного типа

33)

```
string [] names = {
    "Christina Aguillera",
    "Shakira",
    "Beyonce",
    "Gwen Stefani"
  };
Array.Sort(names);
foreach (var name in names)
{
   Console.WriteLine(name);
}
```

34)

Beyonce
Christina Aguillera
Gwen Stefani
Shakira

35)

```
public class Person: IComparable<Person>
{
  public int CompareTo(Person other)
  {
    if (other == null) throw new ArgumentNullException("other");
    int result = this.LastName.CompareTo(other.LastName);
    if (result == 0)
    {
      result = this.FirstName.CompareTo( other.FirstName);
    }
    return result;
  }
  //. . .
```

36)

```
Person[] persons = {
new Person   { FirstName="Daraon", LastName="Hill"),
new Person   { FirstName="Niki", LastName="Lauda"),
new Person   { FirstName="Ayrton", LastName="Senna"),
new Person   { FirstName="Graham", LastName=”Hill”)
  };
Array.Sort(persons);
foreach (var p in persons)
{
  Console.WriteLine(p);
}
```

37)

Damon Hill
Graham Hill
Niki Lauda
Ayrton Senna

38)

```
public enum PersonCompareType
{
  FirstName,
  LastName
}
public class PersonComparer: IComparer<Person>
{
  private PersonCompareType compareType;
  public PersonComparer(PersonCompareType compareType)
  {
    this.compareType = compareType;
  }
  public int Compare(Person x, Person y)
  {
    if(x == null) throw new ArgumentNullException("x") ;
    if(y == null) throw new ArgumentNullException("y");
```

```
    switch (compareType)
    {
      case PersonCompareType.Firstname:
        return x.Firstname.CompareTo(y.FirstName);
      case PersonCompareType.Lastname:
        return x.Lastname.CompareTo(y.LastName);
      default:
        throw new ArgumentException(
                        "недопустимый тип для сравнения");
    }
  }
}
```

39)

```
Array.Sort(persons, new PersonComparer(PersonCompareType.Firstname));
foreach (var p in persons)
{
  Console.WriteLine(p);
}
```

40)

Ayrton Senna
Damon Hill
Graham Hill
Niki Lauda

41)

```
static Person[] GetPersons()
{
   return new Person [ ] {
        new Person { FirstName="Damon", LastName="Hill" },
        new Person { FirstName="Niki", LastName="Lauda" },
        new Person { FirstName="Ayrton", LastName="Senna" },
        new Person { FirstName="Graham", LastName="Hill" }
   };
}
```

42)

```
static void DisplayPersons(Person[] persons)
{
   //...
```

43)

```
static void DisplayArray(object[] data)     .
{
   //...
}
```

44)

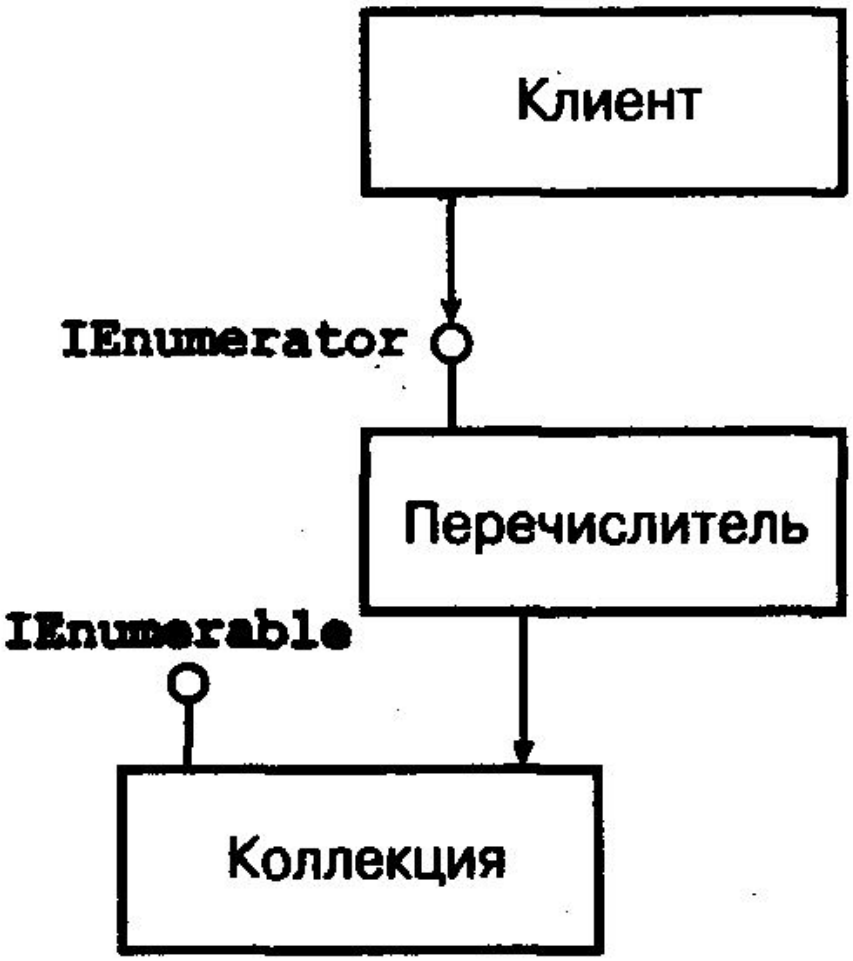
```
static int SumOfSegments(ArraySegment<int>[] segments)
{
   int sum =0;
   foreach (var segment in segments)
   {
      for (int  i=segment.Offset; i<segment.Offset+segment.Count; i++)
      {
         sum += segment.Array[i];
      }
   }
   return sum;
}
```

45)

```
int[] ar1 = { 1, 4, 5, 11, 13, 18 };
int[] ar2 = { 3, 4, 5, 18, 21, 27, 33 };
var segments = new ArraySegment<int>[2]
{
   new ArraySegment<int>(ar1, 0, 3),
   new ArraySegment<int>(ar2, 3, 3)
} ;
var sum = SumOfSegments(segments);
```

47)

```
foreach (var p in persons)
{
   Console.WriteLine(p);
}
```



Отношение между клиентом, вызывающим `foreach`, и коллекцией

48)

```
IEnumerator<Person> enumerator = persons.GetEnumerator();
while (enumerator.MoveNext())
{
  Person p = (Person)enumerator.Current;
  Console.WriteLine(p);
}
```

49)

```
using System;
using System.Collections;

namespace Wrox.ProCSharp.Arrays
{
  public class HelloCollection
  {
    public XEnumerator<string> GetEnumerator()
    {
      yield return "Hello";
      yield return "World";     }
  }
```

50)

```
public void HelloWorld()
{
   var helloCollection = new HelloCollection();
   foreach (string s in helloCollection)
   {
      Console.WriteLine(s);
   }
}
   }
```