Системное программное обеспечение

Саранча Сергей Николаевич, к. т.н., доцент каф ЭВМ ХНУРЭ

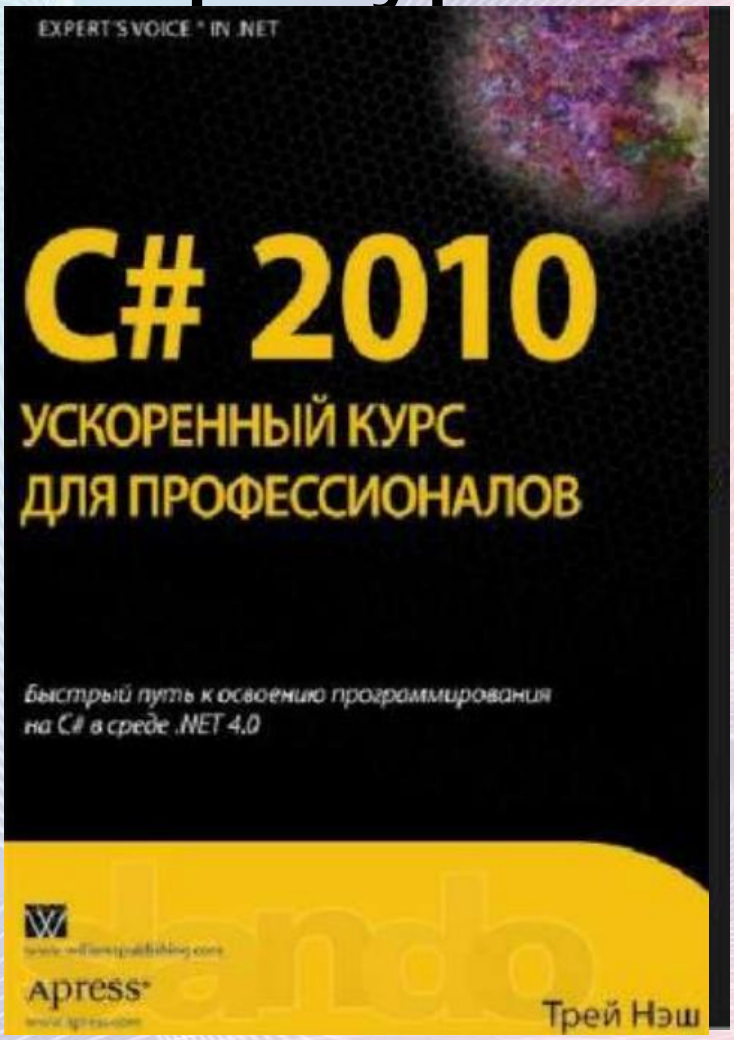
Лекция 2 – Основы языка C#

Системное программное обеспечение - 2012

Содержание лекции

- Библиография
- Анатомия класса в C#
- Композиция приложения в C#
- Системные типы данных
- Условия и циклы
- Массивы
- Работа со строками
- Пользовательские типы данных – перечисления и структуры

Литература


EXPERT'S VOICE * IN .NET
C# 2010
УСКОРЕННЫЙ КУРС
ДЛЯ ПРОФЕССИОНАЛОВ
Быстрый путь к освоению программирования
на C# в среде .NET 4.0
apress
Трей Нэш


Программистам от программистов
Прилагается компакт-диск!
C# 4
и платформа .NET 4
для профессионалов
Кристиан Нейгел, Билл Ивьен, Джей Глинн, Карли Уотсон, Морган Скиннер

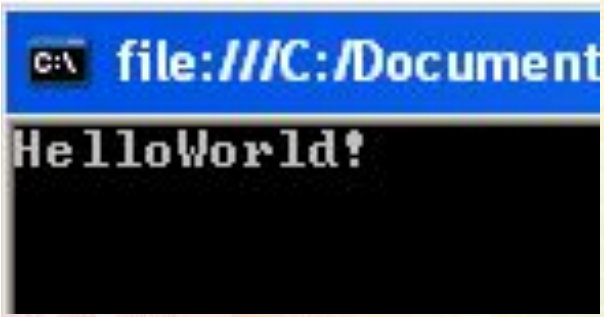
Простейшая программа на C#

Program.cs

ConsoleApplication1.HelloClass

```
using System;

namespace ConsoleApplication1
{
    class HelloClass
    {
        static void Main(string[] args)
        {
            Console.WriteLine("HelloWorld!");
            Console.ReadKey();
        }
    }
}
```



Варианты определения метода Main

```
using System;

namespace ConsoleApplication1
{
    class HelloClass
    {
        static int Main(string[] args)
        {
            Console.WriteLine("HelloWorld!");
            Console.ReadKey();
            return 0;
        }
    }
}
```

```
using System;

namespace ConsoleApplication1
{
    class HelloClass
    {
        static int Main()
        {// main без параметров командной строки
            Console.WriteLine("HelloWorld!");
            Console.ReadKey();
            return 0;
        }
    }
}
```

```
using System;

namespace ConsoleApplication1
{
    class HelloClass
    {
        static void Main()
        {// main без параметров командной строки
            Console.WriteLine("HelloWorld!");
            Console.ReadKey();
        }
    }
}
```

Обработка параметров командной строки

```
using System;

namespace ConsoleApplication1
{
    class HelloClass
    {
        static void Main(string[] args)
        {// обработка аргументов командной строки
            for (int i = 0; i < args.Length; i++)
            {
                Console.WriteLine("Argument {0}: {1}", i + 1, args[i]);
            }
            Console.ReadKey();
        }
    }
}
```

```
>ConsoleApplication1.exe Copy From_source To_dest
Argument 1: Copy
Argument 2: From_source
Argument 3: To_dest
```

Важнейшие пространства имен .NET

Пространство имен	Назначение
System	Множество низкоуровневых классов для работы с простыми типами, выполнение математических операций, сборка мусора и т.д.
System.Collections	Контейнерные классы ArrayList, Queue, SortedList
System.Data System.Data.Common System.Data.OleDb System.Data.SqlClient	Интерфейс баз данных
System.Diagnostics	Классы для трассировки и отладки программного кода
System.Drawing System.Drawing.Drawing2D System.Drawing.Printing	Классы для примитивов GDI+
System.IO	Классы поддержки ввода-вывода

Важнейшие пространства имен .NET

Пространство имен .NET	Назначение
System.Net	Классы для передачи данных по сети (запрос/ответ, создание сокетов)
System.Reflection System.Reflection.Emit	получение дополнительной информации о типах данных во время выполнения)
System.Runtime.InteropServices System.Runtime.Remoting	Взаимодействие с «традиционным» кодом (Win32 DLL, COM-серверы). Типы данных для удаленного доступа (перекрыты в WCF)
System.Security	Классы для работы с разрешениями, криптографией и т.д.
System.Threading	Классы для работы с потоками
System.Web	Классы для работы с ASP.NET
System.Windows.Forms	Классы для работы с элементами интерфейса Windows
System.XML	Классы для работы с данными в формате XML

Использование пространств имен в коде приложения

```
using System;
using System.Collections;

namespace ConsoleApplication1
{
    class HelloClass
    {
        static void Main(string[] args)
        {// обращение к коллекции ArrayList пространства System.Collections
            ArrayList MyList = new ArrayList();
            MyList.Add(10);
            MyList.Add(20);


            for (int i = 0; i < MyList.Count; i++)
            {
                Console.WriteLine("Element {0}: {1}", i + 1, MyList[i]);
            }
            Console.ReadKey();
        }
    }
}
```

Использование пространств имен в коде приложения

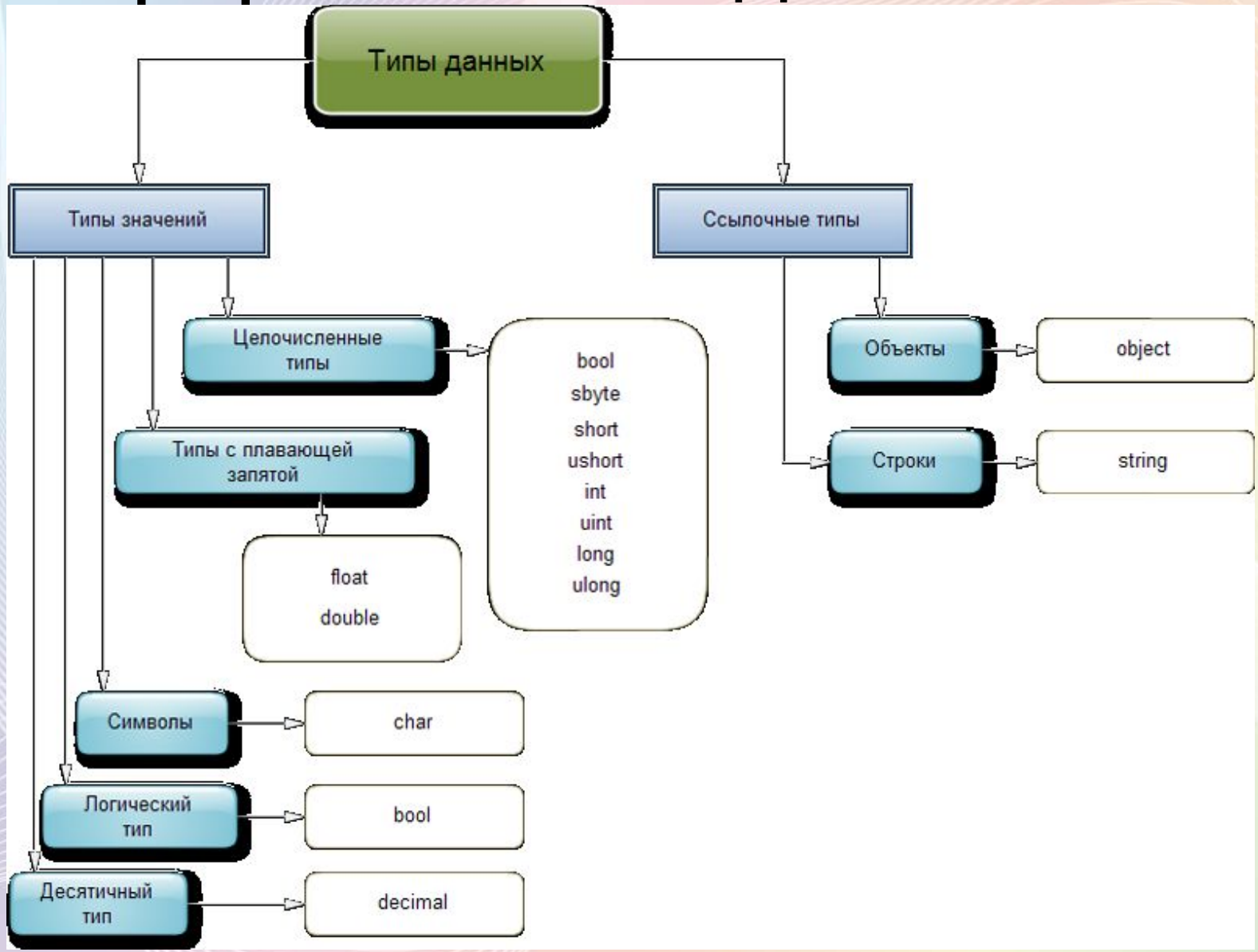
```
using System;

namespace ConsoleApplication1
{
    class HelloClass
    {
        static void Main(string[] args)
        {// пространство System.Collections не подключено
            // требуется явное задание полного пути к классу
            System.Collections.ArrayList MyList =
                new System.Collections.ArrayList();

            MyList.Add(10);
            MyList.Add(20);

            for (int i = 0; i < MyList.Count; i++)
            {
                Console.WriteLine("Element {0}: {1}", i + 1, MyList[i]);
            }
            Console.ReadKey();
        }
    }
}
```

Иерархия типов данных C#



Структурные и ссылочные типы

Характеристика	Структурные типы	Ссылочные типы
Размещение данных	В стеке	В управляемой куче
Представление переменной	В виде локальной копии типа	В виде указателя на область памяти, относящейся к объекту этого типа
Наследование (базовый класс)	Только напрямую от System.ValueType	От System.Object или любого другого незакрытого класса
Наследование – может ли класс быть базовым?	Нет. Структурные типы всегда закрыты и дополнение их не предусмотрено	Да ,если этому не мешают уровни доступа и прочие ограничения

Структурные и ссылочные типы

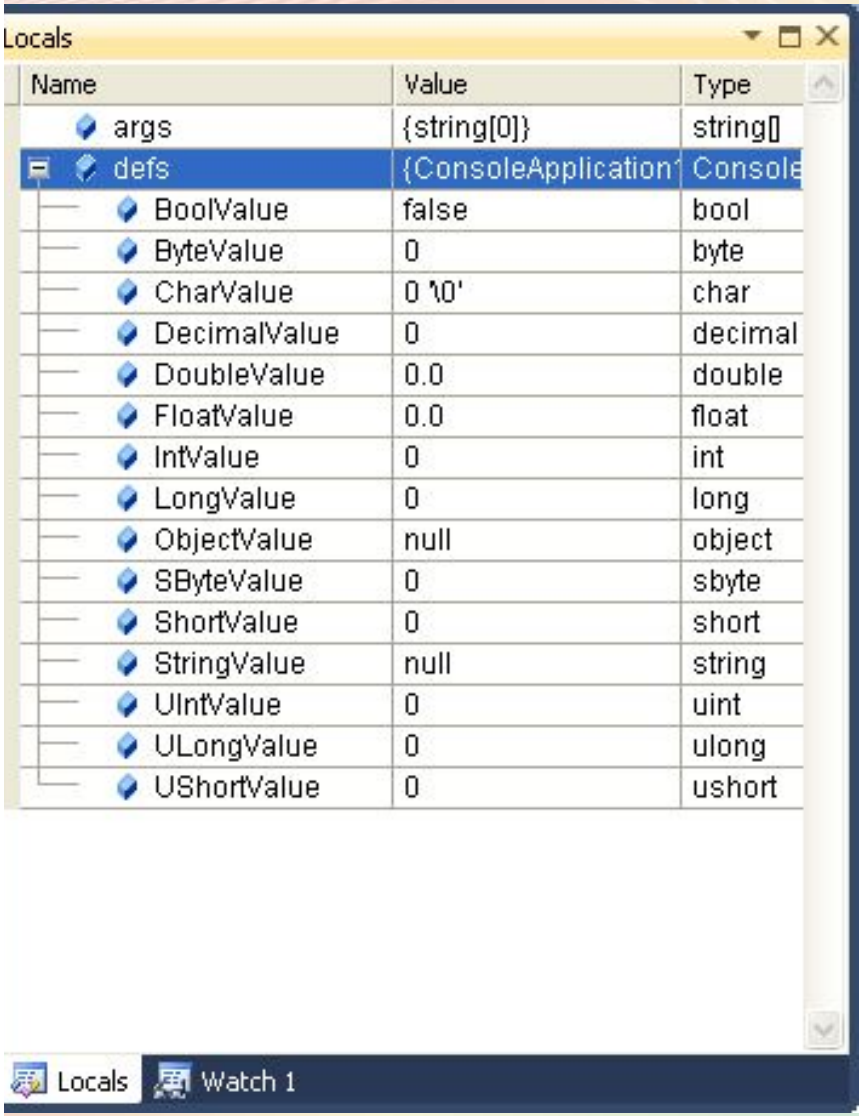
Характеристика	Структурные типы	Ссылочные типы
Способ передачи параметров в подпрограммы	По значению (передаются локальные копии)	По адресу (передается ссылка на область памяти)
Возможность определения конструкторов типа	Есть, кроме конструктора по умолчанию (без параметров)	Есть
Возможность определения деструкторов типа	Нет, структурные типы не размещаются в куче и к ним не применяется функция завершения	Да, но не напрямую (через наследование интерфейса IDisposable)
Момент удаления переменной из памяти	При выходе из области видимости	Во время процесса сборки мусора

Системные типы данных

Имя в C#	CLS	Системный тип	Диапазон
sbyte	X	System.SByte	-128 … 127
byte	V	System.Byte	0 … 255
short	V	System.Int16	-32768 … 32767
ushort	X	System.UInt16	0 … 65535
int	V	System.Int32	-2^{31} … $2^{31}-1$
uint	X	System.UInt32	0… $2^{32}-1$
long	V	System.Int64	-2^{63} … $2^{63}-1$
ulong	X	System.UInt64	0…$2^{64}-1$
char	V	System.Char	U+0000 .. U+FFFF
float	V	System.Single	$1,5*10^{-45}$ .. $3,4*10^{38}$
double	V	System.Double	$5,0*10^{-324}$ … $1,7*10^{308}$
bool	V	System.Boolean	true или false
decimal	V	System.Decimal	0…10^{28} (96-битное знаковое)
string	V	System.String	Ограничено памятью
object	V	System.Object	Базовый класс для всех остальных

Значения по умолчанию

```
class DefaultValues
{ // проверяем начальные значения данных
    public sbyte SByteValue;
    public byte ByteValue;
    public short ShortValue;
    public ushort UShortValue;
    public int IntValue;
    public uint UIntValue;
    public long LongValue;
    public ulong ULongValue;
    public char CharValue;
    public float FloatValue;
    public double DoubleValue;
    public bool BoolValue;
    public decimal DecimalValue;
    public string StringValue;
    public object ObjectValue;
}
class HelloClass
{
    static void Main(string[] args)
    {
        DefaultValues defs = new DefaultValues();
        Console.ReadKey();
    }
}
```



Значения по умолчанию

```
class HelloClass
{
    static int GetValue()
    { // локальная переменная метода - не инициализируется!
        int val;
        return val + 1;
    }
    static void Main(string[] args)
    {
        DefaultV
        Console.
        Console.
    }
}
```

Error List

1 Error | 0 Warnings | 0 Messages

	Description	File	Line	Co...
1	Use of unassigned local variable 'val'	Program.cs	28	20

```
class HelloClass
{
    static int GetValue()
    { // локальная переменная метода инициализируется явно
        int val=0;
        return val + 1;
    }
    static void Main(string[] args)
    {
        DefaultValues defs = new DefaultValues();
        Console.WriteLine(GetValue());
        Console.ReadKey();
    }
}
```

Задание числовых констант

- // две формы представления чисел:
  - int x = 16; // десятичная
  - int x = 0x10; // шестнадцатеричная
  - // восьмеричной формы НЕТ!
  - Следовательно int x = 016; -> int x = 16;
- // задание типа данных для констант
  - uint val = 123U;
  - long L=-123456L;
  - ulong UL =98765432UL;

Задание числовых констант

// вещественные константы по умолчанию определены как double

- const double pi = 3.141592;
- double e=1.6e-19;
- float f = 12.3F;

Тип decimal – для представления десятичных данных (с точностью 28-29 знаков после запятой), без ошибок округления

- decimal sum = 1000.2345678M;

Символьные и логические типы данных

Символьный тип - char

- char ch;
- ch = ‘Z’; // корректно
- ch = ‘\u0041’; // Unicode
- ch = ‘\x0034’; // Unicode
- ch = (char)41; // десятичное с приведением
- ch = 41; // некорректно!

Логический тип – bool

- bool flag = false;
- flag = true; flag = 1;

Специальные символы

```
char ch;
ch='\''; // single quote
ch='\"'; // double quote
ch='\\'; // backslash
ch='\0'; // empty(zero)
ch='\a'; // beep
ch='\b'; // backspace
ch='\f'; // form feed
ch='\t'; // horizontal tabulation
ch='\r'; // carriage return
ch='\n'; // new line
ch='\v'; // vertical tabulation
```

Арифметические операторы

Оператор	Действие
+	Сложение
-	Вычитание, унарный минус
*	Умножение
/	Деление
%	Получение остатка от деления
++	Инкремент
--	Декремент

Операторы отношений

Оператор	Значение
==	Равно
!=	Не равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно

Логические операторы

Оператор	Значение
&	Поразрядное И
|	Поразрядное ИЛИ
^	Поразрядное исключающее ИЛИ
~	Поразрядная инверсия
<<	Поразрядный сдвиг влево
>>	Поразрядный сдвиг вправо
&&	Логическое И
||	Логическое ИЛИ
!	Логическая инверсия

Оператор присваивания

имя_переменной = выражение

- int x, y, z;

Присваивание транзитивно

- x = y = z = 10; // присвоить значение 10 переменным x, y и z
- x = 2 + (y = 3 * (z = 4));

Укороченные операторы присваивания

+= -= *= /= %= &= |= ^=

Условное присваивание

Выражение1 ? Выражение2 :
Выражение3;

abs_val=(a>=0)?a:-a;

Условия

if (условие)
  оператор (операторы)
else
  оператор (операторы)

```
if (x < 0)
     isNegative = true;

if (x < 0)
{
     isNegative = true;
     x = -x;
}

if (x < 0)
{
     isNegative = true;
     x = -x;
}
else
{
     isNegative = false;
}
```

Оператор switch

```
switch(выражение)
{
    case константа1:
      последовательность операторов
      break;
    case константа2:
      последовательность операторов
      break;
    case константа3:
      последовательность операторов
      break;
    ...
    default:
      последовательность операторов
      break;
}
```

Циклы

for (инициализатор; условие; итератор)
    оператор (операторы)

while (условие)
    оператор (операторы)

do
{
    оператор (операторы)
}
while (условие);

Цикл foreach

```
int[] SingleDim; // Объявление массива
// работа с массивом
// вывод массива
foreach (int value in SingleDim)
{
    Console.Write("{0} \t", value);
}
```

Одномерные массивы в C#

int [] SimpleArray; // объявление массива

SimpleArray = new int[10]; // выделение памяти

int [] simpleArray2 = new int[5];

int [] data = {2,5,7,9}; // создание массива

int[] data – new int[4]{2,5,7,9};

Элементы массива инициализируются начальными значениями!

Многомерные массивы в C#

Две разновидности многомерных массивов:

- Прямоугольные

```
int [,] MyMatrix;
MyMatrix = new int[4,3];
```

Многомерные массивы в C#

- Ломаные (jagged) массивы

```
int[][] MyJaggedMatrix = new int[5][];
for (int k = 0; k < MyJaggedMatrix.Length; k++)
{ // создание ломаного массива
    MyJaggedMatrix[k] = new int[k + 7];
}

for (int k = 0; k < MyJaggedMatrix.Length; k++)
{ // вывод ломаного массива на экран
    Console.Write("Length of row {0} is {1}\t", k, MyJaggedMatrix[k].Length);
    for (int l = 0; l < MyJaggedMatrix[k].Length; l++)
    {
        Console.Write("  {0}", MyJaggedMatrix[k][l]);
    }
    Console.WriteLine();
}
```

```
Length of row 0 is 7    0  0  0  0  0  0  0
Length of row 1 is 8    0  0  0  0  0  0  0  0
Length of row 2 is 9    0  0  0  0  0  0  0  0  0
Length of row 3 is 10   0  0  0  0  0  0  0  0  0  0
Length of row 4 is 11   0  0  0  0  0  0  0  0  0  0  0
```

Массивы – это объекты класса System.Array !!!

- MyArray.Length – получение длины
- Array.Sort(MyArray); // сортировка по возрастанию
- Array.Reverse(MyArray); // изменение порядка следования элементов
- Array.IndexOf(MyArray,value,StartIndex, Count); // поиск позиции элемента

Строки

- string str1 = “System software”;
- string str2 = “2012”;
- string str3 = s1+s2; // конкатенация строк

```
string s1 = "string";
string s2 = s1;
Console.WriteLine("first string: {0}", s1);
Console.WriteLine("second string: {0}", s2);
s1 = "New string";
Console.WriteLine("first string: {0}", s1);
Console.WriteLine("second string: {0}", s2);
```

```
first string: string
second string: string
first string: New string
second string: string
```

Перечисления

```
public enum TimeOfDay
{
    Morning,
    Afternoon,
    Evening
}
TimeOfDay timeOfDay = TimeOfDay.Morning;
Console.WriteLine(timeOfDay);
```

Перечисления

```
string CurrentTime = timeOfDay.ToString();
// обратное преобразование
CurrentTime = “Evening”;
TimeOfDay time2 =
   (TimeOfDay)Enum.Parse(
    typeof(TimeOfDay),
    CurrentTime,
    true
   );
```