Практическая работа 16
Получение сетевых параметров компьютера

Цель занятия: Получить практический опыт разработки модулей для определения параметров сети

Перечень оборудования и программного обеспечения

Персональный компьютер
Microsoft Office (Word, Visio)
Microsoft Visual Studio

Краткие теоретические сведения

Пространство имён System.Net содержит сетевые классы для поиска IP-адресов, сетевой аунтификации, разрешений, отправки и получения данных.

Получение параметров сети через System.Net.NetworkInformation

```
using System.Net.NetworkInformation;
System.Net.IPHostEntry host;
host = System.Net.Dns.GetHostEntry("yandex.ru");
foreach (System.Net.IPAddress ip in host.AddressList)
MessageBox.Show(ip.ToString());
```

Получение списка сетевых дисков

Библиотека WSH позволяет быстро получить список сетевых дисков, принтеров. Библиотека WSH позволяет также работать с дисками, папками, файлами, реестром. Для работы с библиотекой необходимо присоединить библиотеки Interop.IWshRuntimeLibrary.dll и Interop.Shell32.dll через ссылки.

```
using IWshRuntimeLibrary;
using Shell32;
using System.Collections;

private void butNetworkDrives_Click(object sender, EventArgs e)
    {
      WshNetwork network = new WshNetwork();
      foreach (IEnumerable driver in network.EnumNetworkDrives())
      {
        MessageBox.Show(driver.ToString());
      }
    }
    private void butCreateShortcut_Click(object sender, EventArgs e)
```

```
        {
            // Создадим ярлык на Рабочем столе
            object shortDesktop = (object)"Desktop";
            WshShell shell = new WshShell();
            // Путь к ярлыку
            string shortcutAddress = (string)shell.SpecialFolders.Item(ref
shortDesktop) + @"\Блокнотик.lnk";

            // Создаем объект ярлыка
            IWshShortcut shortcut =
(IWshShortcut)shell.CreateShortcut(shortcutAddress);
            // Задаем свойства для ярлыка
            // Описание ярлыка в всплывающей подсказке
            shortcut.Description = "Ярлык для текстового редактора";
            // Горячая клавиша
            shortcut.Hotkey = "Ctrl+Shift+N";
            // Путь к самой программе Блокнот
            shortcut.TargetPath =
Environment.GetFolderPath(Environment.SpecialFolder.System) +
@"\notepad.exe";

            // Все готово. Можно создавать ярлык
            shortcut.Save();
        }
        private void butGetPrinters_Click(object sender, EventArgs e)
        {
            WshNetwork network = new WshNetwork();

            foreach (IEnumerable printer in network.EnumPrinterConnections())
            {
                listBox1.Items.Add(printer);
            }
        }
        private void butSetDefaultPrinter_Click(object sender, EventArgs e)
        {
            WshNetwork network = new WshNetwork();
            // Получим список принтеров
            IWshCollection Printers = network.EnumPrinterConnections();
            if (Printers.Count() > 0)
            {
                // Выбираем индекс устанавливаемого принтера
                object index = (object)"1";
                // Устанавливаем выбранный принтер как принтер по умолчанию
                network.SetDefaultPrinter((string)Printers.Item(ref index));
            }
```

```
}
```

WMI

WMI (Windows Management Instrumentation)— это специально разработанный компанией Microsoft интерфейс управления Windows, основанный на определенных стандартах. Технология WMI широко используется системными администраторами при помощи специально написанных сценариев.

Классы, предназначенные для работы с WMI, находятся в пространстве имен System.Management. С помощью WMI можно узнать множество информации о компьютерах. Часть этой информации можно получить при помощи встроенных средств .NET Framework или функций Windows API. Технология WMI позволяет работать с удаленными компьютерами, и именно эта возможность представляет особый интерес для программистов.

```
private void butOS_Click(object sender, EventArgs e)
    {
      // Соединяемся с удаленной машиной
      //ConnectionOptions options = new ConnectionOptions();
      //options.Username = @"domen\admin";
      //options.Password = "pass";
      ManagementScope scope = new
ManagementScope("\\\\boss\\root\\cimv2");
      scope.Connect();
      // Запрашиваем информацию об операционной системе
      ObjectQuery query = new ObjectQuery(
        "SELECT * FROM Win32_OperatingSystem");
      ManagementObjectSearcher searcher =
        new ManagementObjectSearcher(scope, query);
      ManagementObjectCollection queryCollection = searcher.Get();
      foreach (ManagementObject m in queryCollection)
      {
        // Выводим информацию в текстовое поле
        string nl = Environment.NewLine;

       textBox1.Text = "Имя машины : " + m["CSName"] + nl;
       textBox1.Text += "Операционная система: " + m["Caption"] + nl;
       textBox1.Text += "Версия ОС: " + m["Version"] + nl;
       textBox1.Text += "Язык операционной системы: " + m["OSLanguage"]
+ nl;
       textBox1.Text += "Зарегистрированный пользователь: " +
m["RegisteredUser"] + nl;
       textBox1.Text += "Серийный номер продукта: " + m["SerialNumber"]
+ nl;
```

```
            textBox1.Text += "Время установки: " +
ManagementDateTimeConverter.ToDateTime(m["InstallDate"].ToString()) + nl;
            textBox1.Text += "Папка Windows: " + m["WindowsDirectory"] + nl;
            textBox1.Text += "Системная папка: " + m["SystemDirectory"] + nl;
            textBox1.Text += "Производитель: " + m["Manufacturer"] + nl;
            textBox1.Text += "Доступная физическая память: " +
m["FreePhysicalMemory"] + nl;
            textBox1.Text += "Текущее число процессов: " +
m["NumberOfProcesses"] + nl;
         }
      }
      private void butComputer_Click(object sender, EventArgs e)
      {
         // Соединяемся с удаленной машиной
         //ConnectionOptions options = new ConnectionOptions();
         //options.Username = @"domen\admin";
         //options.Password = "pass";
         ManagementScope scope = new
ManagementScope("\\\\localhost\\root\\cimv2");
         scope.Connect();
         // Запрашиваем информацию о компьютере
         ObjectQuery query = new ObjectQuery(
            "SELECT * FROM Win32_ComputerSystem");
         ManagementObjectSearcher searcher =
            new ManagementObjectSearcher(scope, query);
         ManagementObjectCollection queryCollection = searcher.Get();
         foreach (ManagementObject m in queryCollection)
         {
            // Выводим информацию
            string nl = Environment.NewLine;
            textBox1.Text = "Домен: " + m["Domain"] + nl;
            textBox1.Text += "Изготовитель: " + m["Manufacturer"] + nl;
            textBox1.Text += "Модель: " + m["Model"] + nl;
         }
      }
      private void butProduct_Click(object sender, EventArgs e)
      {
         // Соединяемся с удаленной машиной
         //ConnectionOptions options = new ConnectionOptions();
         //options.Username = @"domen\admin";
         //options.Password = "pass";
         ManagementScope scope = new
ManagementScope("\\\\localhost\\root\\cimv2");
         scope.Connect();
```

```
        ObjectQuery query = new ObjectQuery("SELECT * FROM
Win32_ComputerSystemProduct");
        ManagementObjectSearcher searcher = new
ManagementObjectSearcher(scope, query);
        ManagementObjectCollection queryCollection = searcher.Get();
        foreach (ManagementObject m in queryCollection)
        {
           // Display the remote computer information
           string nl = Environment.NewLine;
           textBox1.Text = "Описание : " + m["Description"] + nl;
           textBox1.Text += "Серийный номер : " + m["IdentifyingNumber"] +
nl;
           textBox1.Text += "Имя : " + m["Name"] + nl;
           textBox1.Text += "Идентификатор продукта : " + m["UUID"] + nl;
           textBox1.Text += "Производитель : " + m["Vendor"] + nl;
        }
     }
     private void butProc_Click(object sender, EventArgs e)
     {
        // Соединяемся с удаленной машиной
        //ConnectionOptions options = new ConnectionOptions();
        //options.Username = @"gamma\admin";
        //options.Password = "mypass";
        ManagementScope scope = new
ManagementScope("\\\\localhost\\root\\cimv2");
        scope.Connect();
        // Делаем запрос к удаленной машине
        WqlObjectQuery query = new WqlObjectQuery("Select * from
Win32_Processor");
        ManagementObjectSearcher find = new
ManagementObjectSearcher(query);
        string nl = Environment.NewLine;
        int i = 0;
        foreach (ManagementObject mo in find.Get())
        {
 textBox1.Text += ("---------- Processor #" + i + " ----------") + nl;
 textBox1.Text += ("Processor address width in bits....." + mo["AddressWidth"])
+ nl;
 textBox1.Text += ("Caption....................." + mo["Caption"]) + nl;
 textBox1.Text += ("Processor address width in bits....." + mo["AddressWidth"])
+ nl;
 textBox1.Text += ("Current clock speed (in MHz)....." +
mo["CurrentClockSpeed"]) + nl;
 textBox1.Text += ("Processor data width....." + mo["DataWidth"]) + nl;
 textBox1.Text += ("Unique string identification....." + mo["DeviceID"]) + nl;
```

```
 textBox1.Text += ("External clock frequency....." + mo["ExtClock"]) + nl;
 textBox1.Text += ("Processor data width....." + mo["DataWidth"]) + nl;
 textBox1.Text += ("L2 cache size....." + mo["L2CacheSize"]) + nl;
 textBox1.Text += ("L2 cache speed....." + mo["L2CacheSpeed"]) + nl;
 textBox1.Text += ("Load percentage(average value for second)" +
mo["LoadPercentage"]) + nl;
 textBox1.Text += ("Manufacturer....." + mo["Manufacturer"]) + nl;
 textBox1.Text += ("Maximum speed (in MHz)....." +
mo["MaxClockSpeed"]) + nl;
 textBox1.Text += ("Name....." + mo["Name"]) + nl;
  textBox1.Text += ("Support for power management.." +
mo["PowerManagementSupported"]) + nl;
 textBox1.Text += ("Unique identificator describing processor." +
mo["ProcessorId"]) + nl;
  textBox1.Text += ("Role (CPU/math)....." + mo["Role"]) + nl;
  textBox1.Text += ("Socket designation....." + mo["SocketDesignation"]) +
nl;
  textBox1.Text += ("Status....." + mo["Status"]) + nl;
  textBox1.Text += ("Processor version....." + mo["Version"]) + nl;
  textBox1.Text += ("Socket voltage....." + mo["VoltageCaps"]) + nl;
                i++;
            }
        }
        private void butVideo_Click(object sender, EventArgs e)
        {
            //ConnectionOptions options = new ConnectionOptions();
            //// ваш домен и учетная запись
            //options.Username = @"domen\administrator";
            //// ваш пароль
            //options.Password = "yourpassword";
            ManagementScope scope = new
ManagementScope("\\\\localhost\\root\\cimv2");
            scope.Connect();
            // Запрашиваем информацию о видеоконтроллере
            ObjectQuery query = new ObjectQuery(
                "SELECT * FROM Win32_VideoController");
            ManagementObjectSearcher searcher = new
ManagementObjectSearcher(scope, query);
            ManagementObjectCollection queryCollection = searcher.Get();
            foreach (ManagementObject m in queryCollection)
            {
                // Выводим информацию в текстовое поле
                string nl = Environment.NewLine;
                textBox1.Text = "Имя : " + m["Name"] + nl;
                textBox1.Text += "Процессор : " + m["VideoProcessor"] + nl;
```

```
            textBox1.Text += "Видеопамять: " + m["AdapterRam"] + nl;
            textBox1.Text += "Разрешение: " + m["VideoModeDescription"] + nl;
            textBox1.Text += "Частота обновления : " + m["CurrentRefreshRate"];
        }
    }
    private void butCDRom_Click(object sender, EventArgs e)
    {
        //ConnectionOptions options = new ConnectionOptions();
        //options.Username = @"gamma\admin";
        //options.Password = "mypass";
        ManagementScope scope = new
ManagementScope("\\\\localhost\\root\\cimv2");
        scope.Connect();
        // Запрашиваем информацию о приводах компакт-дисков
        ObjectQuery query = new ObjectQuery(
            "SELECT * FROM Win32_CDROMDrive");
        ManagementObjectSearcher searcher =
            new ManagementObjectSearcher(scope, query);
        ManagementObjectCollection queryCollection = searcher.Get();
        foreach (ManagementObject mo in queryCollection)
        {
            string nl = Environment.NewLine;
            // Выводим информацию с удаленного компьютера
            textBox1.Text = "Описание : " + mo["Description"] + nl;
            textBox1.Text += "Диск: " + mo["Drive"] + nl;
            textBox1.Text += "Тип: " + mo["MediaType"] + nl;
            textBox1.Text += "Статус: " + mo["Status"] + nl;
        }
    }
    private void butAdapter_Click(object sender, EventArgs e)
    {
        //ConnectionOptions options = new ConnectionOptions();
        //options.Username = @"gamma\admin";
        //options.Password = "password";

        ManagementScope scope = new
ManagementScope("\\\\localhost\\root\\cimv2");
        scope.Connect();
        // Запрашиваем информацию о сетевом адаптере
        ObjectQuery query = new ObjectQuery("SELECT * FROM
Win32_NetworkAdapter");
        ManagementObjectSearcher searcher = new
ManagementObjectSearcher(scope, query);
        ManagementObjectCollection queryCollection = searcher.Get();
        foreach (ManagementObject mo in queryCollection)
```

```
            {
                string nl = Environment.NewLine;
                // Выводим информацию с удаленного компьютера
                textBox1.Text = "Производитель : " + mo["Manufacturer"] + nl;
                textBox1.Text += "MACAddress: " + mo["MACAddress"] + nl;
                textBox1.Text += "ProductName: " + mo["ProductName"] + nl;
                textBox1.Text += "AdapterType: " + mo["AdapterType"] + nl;
                textBox1.Text += "CreationClassName: " + mo["CreationClassName"] +
nl;
            }
        }
        private void butMonitor_Click(object sender, EventArgs e)
        {
            //ConnectionOptions options = new ConnectionOptions();
            //options.Username = @"gamma\admin";
            //options.Password = "password";
            ManagementScope scope = new
ManagementScope("\\\\localhost\\root\\cimv2");
            scope.Connect();
            // Запрашиваем информацию о мониторе
            ObjectQuery query = new ObjectQuery("SELECT * FROM
Win32_DesktopMonitor");
            ManagementObjectSearcher searcher = new
ManagementObjectSearcher(scope, query);
            ManagementObjectCollection queryCollection = searcher.Get();
            foreach (ManagementObject mo in queryCollection)
            {
                string nl = Environment.NewLine;
                // Выводим информацию с удаленного компьютера
                textBox1.Text = "Описание : " + mo["Description"] + nl;
                textBox1.Text += "Тип монитора: " + mo["MonitorType"] + nl;
            }
        }
        private void butBoard_Click(object sender, EventArgs e)
        {
            //ConnectionOptions options = new ConnectionOptions();
            //options.Username = @"gamma\admin";
            //options.Password = "password";

            //ManagementScope scope =
            //   new ManagementScope(
            //       "\\\\smena01\\root\\cimv2", options);
            // для локальной машины
            ManagementScope scope = new
ManagementScope("\\\\localhost\\root\\cimv2");
```

```
            scope.Connect();
            // Запрашиваем информацию о материнской плате
            ObjectQuery query = new ObjectQuery("SELECT * FROM
Win32_BaseBoard");
            ManagementObjectSearcher searcher =new
ManagementObjectSearcher(scope, query);
            ManagementObjectCollection queryCollection = searcher.Get();
            foreach (ManagementObject mo in queryCollection)
            {
               string nl = Environment.NewLine;
               // Выводим информацию с удаленного компьютера
               textBox1.Text = "Manufacturer : " + mo["Manufacturer"] + nl;
               textBox1.Text += "Name: " + mo["Name"] + nl;
            }
         }
         private void butShare_Click(object sender, EventArgs e)
         {
            // Соединяемся с удаленной машиной
            ConnectionOptions options = new ConnectionOptions();
            options.Username = @"gamma\admin";
            options.Password = "password";
            //ManagementScope scope =
            //   new ManagementScope(
            //       "\\\\mymachine\\root\\cimv2", options);
            // для локальной машины
            ManagementScope scope = new
ManagementScope("\\\\localhost\\root\\cimv2");
            scope.Connect();
            // Делаем запрос к удаленной машине
            System.Management.ObjectQuery oq = new
System.Management.ObjectQuery("SELECT * FROM Win32_Share");
            ManagementObjectSearcher find = new
ManagementObjectSearcher(scope, oq);
            string nl = Environment.NewLine;
            foreach (ManagementObject mo in find.Get())
            {
               textBox1.Text = ("Список общих ресурсов = " + mo["Name"]) + nl;
            }
         }
         private void butLogDisk_Click(object sender, EventArgs e)
         {
            // Соединяемся с удаленной машиной
            ConnectionOptions options = new ConnectionOptions();
            options.Username = @"gamma\admin";
            options.Password = "password";
```

```
        //ManagementScope scope = //  new ManagementScope( //
"\\\\support\\root\\cimv2", options);
        ManagementScope scope =  new
ManagementScope("\\\\localhost\\root\\cimv2");
        scope.Connect();
        // Делаем запрос к удаленной машине
        string cmiPath = @"\root\cimv2:Win32_LogicalDisk.DeviceID='C:'";
        ManagementObject mo = new ManagementObject(cmiPath);
        // Выводим информацию
        string nl = Environment.NewLine;
        textBox1.Text = "Описание: " + mo["Description"] + nl;
        textBox1.Text += "Файловая система: " + mo["FileSystem"] + nl;
        textBox1.Text += "Свободно: " + mo["FreeSpace"] + nl;
        textBox1.Text += "Размер диска: " + mo["Size"] + nl;
    }
    private void butReboot_Click(object sender, EventArgs e)
    {
        // Соединяемся с удаленным компьютером
        ConnectionOptions options = new ConnectionOptions();
        options.Username = @"domen\admin";
        options.Password = "password";
        ManagementScope scope = new
ManagementScope("\\\\machine\\root\\cimv2", options);
        scope.Connect();
        // Делаем запрос к удаленной машине
        System.Management.ObjectQuery oq = new
System.Management.ObjectQuery("SELECT * FROM
Win32_OperatingSystem");
        ManagementObjectSearcher query1 = new
ManagementObjectSearcher(scope, oq);
        ManagementObjectCollection queryCollection1 = query1.Get();
        foreach (ManagementObject mo in queryCollection1)
        {
            string[] ss ={  "" };
            mo.InvokeMethod("Reboot", ss);
            this.Text = mo.ToString();
        }
    }
```

Работа с электронной почтой

Отправка почты. SmtpClient

Для отправки почты в среде интернет используется протокол SMTP (Simple Mail Transfer Protocol). Данный протокол указывает, как почтовые сервера взаимодействуют при передаче электронной почты.

Для работы с протоколом SMTP и отправки электронной почты в .NET предназначен класс **SmtpClient** из пространства имен **System.Net.Mail**.

Этот класс определяет ряд свойств, которые позволяют настроить отправку:

– **Host:** smtp-сервер, с которого производится отправление почты. Например, smtp.yandex.ru

– **Port:** порт, используемый smp-сервером. Если не указан, то по умолчанию используется 25 порт.

– **Credentials:** аутентификационные данные отправителя

– **EnableSsl:** указывает, будет ли использоваться протокол SSL при отправке

Еще одним ключевым классом, который используется при отправке, является **MailMessage**. Данный класс представляет собой отправляемое сообщение. Среди его свойств можно выделить следующие:

– **Attachments:** содержит все прикрепления к письму

– **Body:** непосредственно текст письма

– **From:** адрес отправителя. Представляет объект MailAddress

– **To:** адрес получателя. Также представляет объект MailAddress

– **Subject:** определяет тему письма

– **IsBodyHtml:** указывает, представляет ли письмо содержимое с кодом html

Используем эти классы и выполним отправку письма:

```
using System;
using System.Net;
using System.IO;
using System.Threading.Tasks;
using System.Net.Mail;

namespace NetConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {

            // отправитель - устанавливаем адрес и отображаемое в письме имя
            MailAddress from = new MailAddress("somemail@gmail.com", "Tom");
            // кому отправляем
            MailAddress to = new MailAddress("somemail@yandex.ru");
            // создаем объект сообщения
            MailMessage m = new MailMessage(from, to);
```

```
            // тема письма
            m.Subject = "Тест";
            // текст письма
            m.Body = "<h2>Письмо-тест работы smtp-клиента</h2>";
            // письмо представляет код html
            m.IsBodyHtml = true;
            // адрес smtp-сервера и порт, с которого будем отправлять письмо
            SmtpClient smtp = new SmtpClient("smtp.gmail.com", 587);
            // логин и пароль
            smtp.Credentials = new NetworkCredential("somemail@gmail.com",
"mypassword");
            smtp.EnableSsl = true;
            smtp.Send(m);
            Console.Read();
        }
    }
}
```

Для отправки применяется метод **Send()**, в который передается объект **MailMessage**.

Также мы можем использовать асинхронную версию отправки с помощью метода **SendMailAsync**:

```
using System;
using System.Net;
using System.IO;
using System.Threading.Tasks;
using System.Net.Mail;

namespace NetConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {

            SendEmailAsync().GetAwaiter();
            Console.Read();
        }

        private static async Task SendEmailAsync()
        {
            MailAddress from = new MailAddress("somemail@gmail.com", "Tom");
            MailAddress to = new MailAddress("somemail@yandex.ru");
            MailMessage m = new MailMessage(from, to);
```

```
            m.Subject = "Тест";
            m.Body = "Письмо-тест 2 работы smtp-клиента";
            SmtpClient smtp = new SmtpClient("smtp.gmail.com", 587);
            smtp.Credentials = new NetworkCredential("somemail@gmail.com",
"mypassword");
            smtp.EnableSsl = true;
            await smtp.SendMailAsync(m);
            Console.WriteLine("Письмо отправлено");
        }
    }
}
```

Добавление вложений

К письму мы можем прикрепить вложения с помощью свойства Attachments. Каждое вложение представляет объект System.Net.Mail.Attachment:

```
MailAddress from = new MailAddress("somemail@gmail.com", "Tom");
MailAddress to = new MailAddress("somemail2@yandex.ru");
MailMessage m = new MailMessage(from, to);
m.Attachments.Add(new Attachment("D://temlog.txt"));
```

Задания

1 Изучить теоретические сведения и задание к работе

2 Разработать отлаженный модуль представления сетевых параметров компьютера.

Порядок выполнения работы

Задание 2

Создадим форму для представления данных, содержащую кнопки, несколько label для размещения информации.

Для получения информации об устройствах включим в исходный код пространство имен и сборку System.Management. Создаем запрос по сетевому адаптеру и выводим на форму его параметры.

```
ManagementObjectSearcher searcher = new ManagementObjectSearcher("select * from Win32_NetworkAdapterConfiguration");
foreach (ManagementObject share in searcher.Get())
{
    label2.Text = "Описание  " +share["Description"].ToString();
    label3.Text = "Физический адрес  " + share["SettingID"].ToString();
    label4.Text = "Номер сетевого адаптера в системе  " + share ["Index"].ToString();
    label5.Text = "Подпись  " + share["Caption"].ToString();
}
```

Теперь нам нужно подключить директиву **System.Net**, которая позволяет программировать, используя различные сетевые протоколы. Подключаем её, используя using, в самом начале кода, среди прочих директив и библиотек.

```
using System.Net;
```

Далее получим наш host и IP следующим образом:

```
string Host = System.Net.Dns.GetHostName();
string IP = System.Net.Dns.GetHostByName(Host).AddressList[0].ToString();
```

То есть при загрузке формы (при запуске программы), данные сразу поступят в неё.

Разберем, что мы написали. Для начала мы объявили переменную с именем Host, у которой тип данных string, т.е. строковой тип. Это означает, что всё, что записано в нашей переменной, представлено в виде обычной строки (неизвестно, находятся ли там внутри буквы, какие-либо символы или числа).

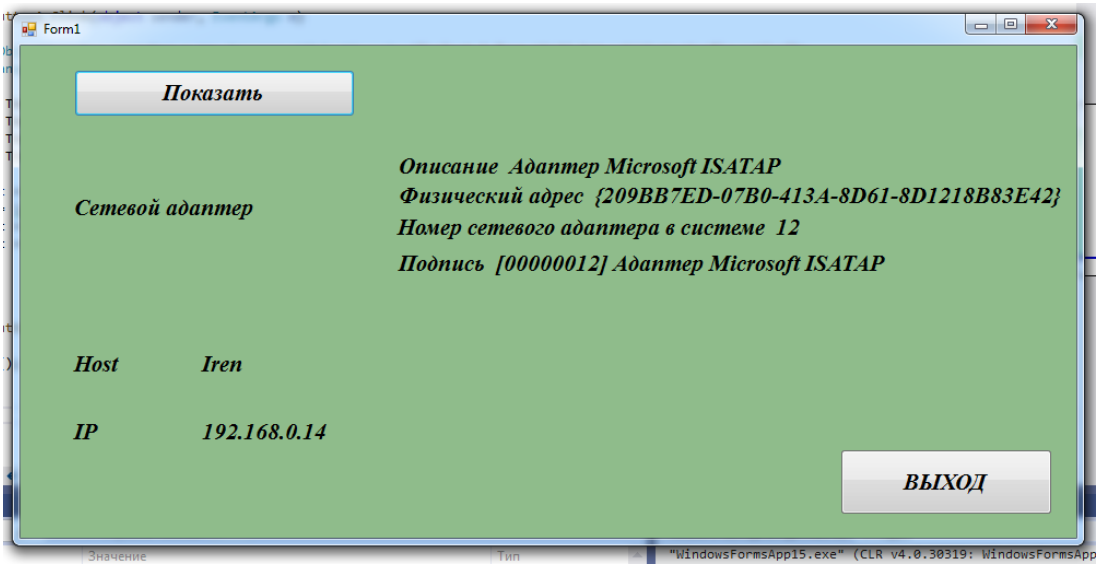
Далее мы говорим программе, что надо записать в эту переменную. Нам нужен хост, поэтому мы прописываем как бы “путь”, по которому мы сможем его получить.

```
}
string Host = System.Net.Dns.GetHostName();
string IP = System.Net.Dns.GetHostByName(Host).AddressList[0].ToString();
label9.Text = Host;
label8.Text = IP;


ссылка: 1
private void button2_Click(object sender, EventArgs e)

    this.Close();
```

Остаётся лишь вывести их в форму для того, чтобы пользователь смог увидеть их.



Определим ip-адреса сетевого ресурса, например mail.ru.

Прежде чем отправить запрос к какому-нибудь ресурсу, компьютер обращается к DNS-северу, чтобы по имени ресурса получить его ip-адрес. И затем уже обращается по этому ip-адресу.

Все ip-адреса представляют 32-битное (протокол IPv4) или 128-битное значение (протокол IPv6), например, 31.170.165.181.

В системе классов .NET ip-адрес представлен классом IPAddress. Этот класс позволяет управлять адресами с помощью следующих свойств и методов:

Метод Parse(): преобразует строковое представление адреса в IPAddress

```
IPAddress ip = IPAddress.Parse("127.0.0.1"); // ip указывает на локальный адрес
```

Статическое свойство Loopback: возвращает объект IPAddress для адреса 127.0.0.1. Аналогично вышеприведенному коду

Статическое свойство Any: возвращает объект IPAddress для адреса 0.0.0.0

Статическое свойство Broadcast: возвращает объект IPAddress для адреса 255.255.255.255

Также для получения адреса в сети используется класс **IPHostEntry**. Он содержит информацию об определенном компьютере-хосте.

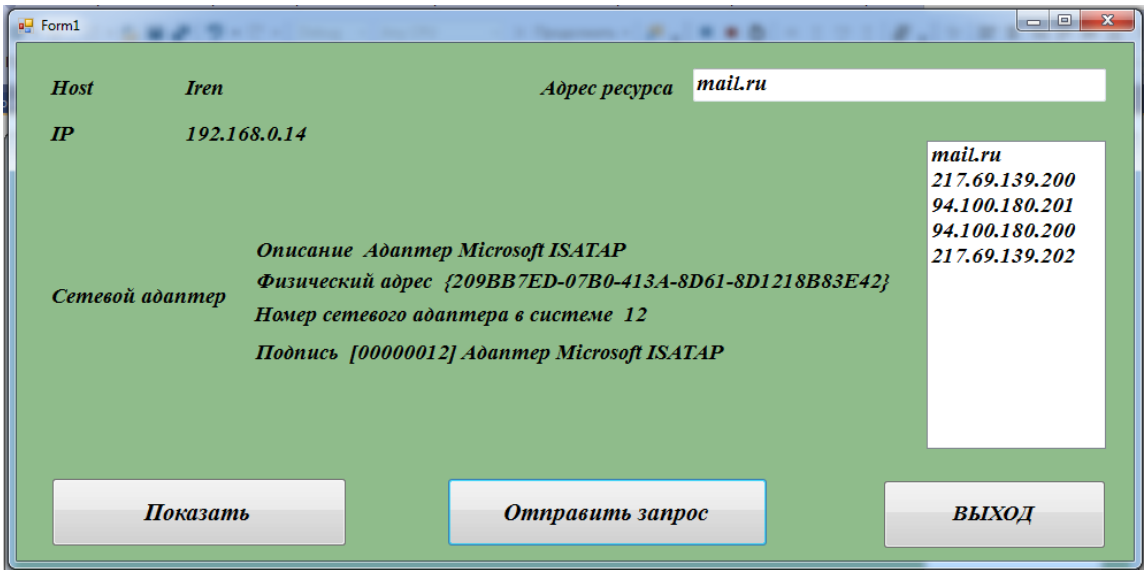
С помощью свойства **HostName** этот класс возвращает имя хоста, а с помощью свойства AddressList - все ip-адреса хоста, так как один компьютер может иметь в сети несколько ip-адресов.

Для взаимодействия с dns-сервером и получения ip-адреса применяется класс **Dns**. Для получения информации о хосте компьютера и его адресах у него имеется метод **GetHostEntry()**:

Внесем изменения в интерфейс приложения, добавим нужные элементы и для кнопки зададим код:

```
ссылка: 1
private void button3_Click(object sender, EventArgs e)
{
    IPHostEntry host = Dns.GetHostEntry(textBox1.Text);
    listBox1.Items.Add(host.HostName);
    foreach (IPAddress ip in host.AddressList)
        listBox1.Items.Add(ip.ToString());
}
```

В результате получим:



Содержание отчета

1 Название работы

2 Цель работы
3 Технические средства обучения
4 Задания (условия задач)
5 Порядок выполнения работы
6 Вывод

Контрольные вопросы

1. Для чего предназначено пространство имён System.Net?
2. Какое пространство имен необходимо подключить в программе для получения информации о сетевых устройствах компьютера?
3. Какое пространство имен необходимо подключить в программе для отправки электронной почты?

Используемая литература

1 Гниденко, И. Г. Технология разработки программного обеспечения: учеб. пособие для СПО / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — М.: Издательство Юрайт, 2017.

2 Шарп Джон Ш26 Microsoft Visual C#. Подробное руководство. 8-е изд. — СПб.: Питер, 2017.

3 Васильев А.Н. Программирование на C# для начинающих. Основные сведения. – Москва: Эксмо, 2018.

4 Васильев А.Н. Программирование на C# для начинающих. Особенности языка. – Москва: Эксмо, 2019.

5 http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx.

6 https://metanit.com/sharp/net/1.2.php

7 https://vscode.ru/prog-lessons/kak-uznat-svoy-host-i-ip-c-sharp.html