МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ И ФИЗИКО-МАТЕМАТИЧЕСКИХ НАУК

КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Отчёт по лабораторной работе №1 по дисциплине «Протоколы компьютерного взаимодействия»

**IP – Адресация**

|  |  |
| --- | --- |
| Выполнил: студент группы ФИб-4302-51-00 | / Д.А. Савин / |
| Проверил: преподаватель каф. ПМиИ | / А.Б. Белиц / |

Киров 2021

Задание №1

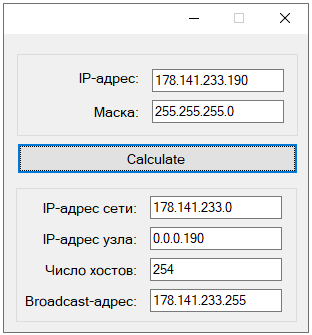
В ходе лабораторной работы необходимо разработать программу, которая по IP-адресу и маске выведет:

IP-адрес сети, IP-адрес узла, максимальное количество узлов в сети, IP-адрес для широковещательной рассылки.

Код:

|  |
| --- |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace zadanie\_1  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private static List<string> convertIPToBinaryIP(string ipAddress)  {  string[] ipArr = ipAddress.Split('.');  int[] ipA = {1,1,1,1};  for (int i=0; i<ipArr.Length; i++)  {  try { ipA[i] = int.Parse(ipArr[i]); } catch { }    }  List<string> binaryIP = new List<string>();    for (var i = 0; i < ipArr.Length; i++)  {  var binaryNo = Convert.ToString(ipA[i], 2);  if (binaryNo.Length == 8)  {  binaryIP.Add(binaryNo);  // binaryIP.push(binaryNo);  }  else  {  var diffNo = 8 - binaryNo.Length;  var createBinary = "";  for (var j = 0; j < diffNo; j++)  {  createBinary += '0';  }  createBinary += binaryNo;  binaryIP.Add(createBinary);  }  }  return binaryIP;  }  private static string reverse(string str)  {  string res = "";  for (int i=0; i<str.Length; i++)  {  if (str[i]=='0')  {  res += '1';  }  else  {  res += '0';  }  }  return res;  }  private static Dictionary<string, uint> Fill(Dictionary<string, uint> dC)  {  dC.Add("128.0.0.0", 2147483648);  dC.Add("192.0.0.0", 1073741824);  dC.Add("224.0.0.0", 536870912);  dC.Add("240.0.0.0", 268435456);  dC.Add("248.0.0.0", 134217728);  dC.Add("252.0.0.0", 67108864);  dC.Add("254.0.0.0", 33554432);  dC.Add("255.0.0.0", 16777216);  dC.Add("255.128.0.0", 8388608);  dC.Add("255.192.0.0", 4194304);  dC.Add("255.224.0.0", 2097152);  dC.Add("255.240.0.0", 1048576);  dC.Add("255.248.0.0", 524288);  dC.Add("255.252.0.0", 262144);  dC.Add("255.254.0.0", 131072);  dC.Add("255.255.0.0", 65536);  dC.Add("255.255.128.0", 32768);  dC.Add("255.255.192.0", 16384);  dC.Add("255.255.224.0", 8192);  dC.Add("255.255.240.0", 4096);  dC.Add("255.255.248.0", 2048);  dC.Add("255.255.252.0", 1024);  dC.Add("255.255.254.0", 512);  dC.Add("255.255.255.0", 256);  dC.Add("255.255.255.128", 128);  dC.Add("255.255.255.192", 64);  dC.Add("255.255.255.224", 32);  dC.Add("255.255.255.240", 16);  dC.Add("255.255.255.248", 8);  dC.Add("255.255.255.252", 4);  dC.Add("255.255.255.254", 2);  dC.Add("255.255.255.255", 1);  return dC;  }  private void calcBt\_Click(object sender, EventArgs e)  {  ipAdNetTb.Text = "";  ipAdHostTb.Text = "";  maxHtb.Text = "";  macTb.Text = "";  if (ipTb.Text != "" && maskTb.Text != "" )  {  ipAdNetTb.Text = "";  ipAdHostTb.Text = "";  maxHtb.Text = "";  macTb.Text = "";  Dictionary<string, uint> dC = new Dictionary<string, uint>();  dC = Fill(dC);  int[] ipAdres=new int [4];  int[] maska = new int[4];  string[] str = ipTb.Text.Split('.');  string[] ipAd = new string[4];  StringBuilder ip = new StringBuilder();  for (int i = 0; i < str.Length; i++)  {  try  {  ipAdres[i] = int.Parse(str[i]);  ip.Append(Convert.ToString(ipAdres[i], 2));  }  catch  {  ipAdNetTb.Enabled = false; ipAdNetTb.Text = "";  ipAdHostTb.Enabled = false; ipAdHostTb.Text = "";  maxHtb.Enabled = false; maxHtb.Text = "";  macTb.Enabled = false; macTb.Text = "";  }  }  Array.Clear(str, 0, str.Length);  str = maskTb.Text.Split('.');  for (int i = 0; i < str.Length; i++)  {  try { maska[i] = int.Parse(str[i]); }  catch  {  ipAdNetTb.Enabled = false; ipAdNetTb.Text = "";  ipAdHostTb.Enabled = false; ipAdHostTb.Text = "";  maxHtb.Enabled = false; maxHtb.Text = "";  macTb.Enabled = false; macTb.Text = "";  }  }  int[] hostAd = new int[4];  int[] hostAd2 = new int[4];  //IP1  int[] invMask = new int[4];  List<string> maskaBin = convertIPToBinaryIP(maskTb.Text.ToString());  for (int i=0; i< maskaBin.Count; i++)  {  invMask[i] = int.Parse(Convert.ToInt32(reverse(maskaBin.ElementAt(i)), 2).ToString());  }    for (int i = 0; i < ipAdres.Length; i++)  {  hostAd[i] = ipAdres[i] & maska[i];  ipAdNetTb.Text += hostAd[i].ToString();  hostAd2[i] = invMask[i] & ipAdres[i];    ipAdHostTb.Text += (hostAd2[i]).ToString();  if (i < ipAdres.Length - 1)  {  ipAdNetTb.Text += ".";  ipAdHostTb.Text += ".";  }  }  //  //Broadcast  List<string> sub = convertIPToBinaryIP(maskTb.Text);  string[] subNet = new string[4];  int[] sn = new int[4];  bool cancel = false;  for (int i = 0; i < subNet.Length; i++)  {  try  {  subNet[i] = reverse(sub.ElementAt(i));  sn[i] = int.Parse(Convert.ToInt32(subNet[i], 2).ToString());  }  catch {  ipAdNetTb.Enabled = false; ipAdNetTb.Text = "";  ipAdHostTb.Enabled = false; ipAdHostTb.Text = "";  maxHtb.Enabled = false; maxHtb.Text = "";  macTb.Enabled = false; macTb.Text = "";  cancel = true;  }  }  if (!cancel)  {  int[] MAC = new int[4];  for (int i = 0; i < hostAd.Length; i++)  {  MAC[i] = hostAd[i] | sn[i];  macTb.Text += MAC[i].ToString();  if (i < ipAdres.Length - 1)  {  macTb.Text += ".";  }  }  //  }  //Max Hosts  try  {  uint hosts = dC[maskTb.Text];  maxHtb.Text = (hosts - 2).ToString();  ipAdNetTb.Enabled = true;  ipAdHostTb.Enabled = true;  maxHtb.Enabled = true;  macTb.Enabled = true;  }  catch { maxHtb.Text = "Неверная маска"; }  }  }  private void ipTb\_TextChanged(object sender, EventArgs e)  {  ipAdNetTb.Enabled = false; ipAdNetTb.Text = "";  ipAdHostTb.Enabled = false; ipAdHostTb.Text = "";  maxHtb.Enabled = false; maxHtb.Text = "";  macTb.Enabled = false; macTb.Text = "";  }  private void maskTb\_TextChanged(object sender, EventArgs e)  {  ipAdNetTb.Enabled = false; ipAdNetTb.Text = "";  ipAdHostTb.Enabled = false; ipAdHostTb.Text = "";  maxHtb.Enabled = false; maxHtb.Text = "";  macTb.Enabled = false; macTb.Text = "";  }  }  } |

Пример работы:



**Входные данные:**

IP-адрес: 178.141.233.190

Маска: 255.255.255.0

**Выходные данные:**

IP-адрес сети: 178.141.233.0

IP-адрес узла: 0.0.0.190

Число хостов: 254

Широковещательный адрес: 178.141.233.255

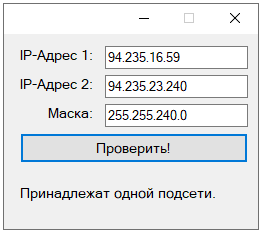
Задание №2

В ходе лабораторной работы необходимо разработать программу, которая определяет, относятся ли два узла к одной подсети (даны IP-адреса двух компьютеров и маска подсети).

Код:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace zadanie\_2  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private void button1\_Click(object sender, EventArgs e)  {  if (ip1Tb.Text!="" && ip2Tb.Text != "" && subnetTb.Text!="")  {  try  {  int[] ipAdres1 = new int[4];  int[] ipAdres2 = new int[4];  int[] maska = new int[4];  string[] str = ip1Tb.Text.Split('.');  string[] ipAd = new string[4];  StringBuilder ip = new StringBuilder();  for (int i = 0; i < str.Length; i++)  {  ipAdres1[i] = int.Parse(str[i]);  //ip.Append(Convert.ToString(ipAdres1[i], 2));  }  Array.Clear(str, 0, str.Length);  str = ip2Tb.Text.Split('.');  for (int i = 0; i < str.Length; i++)  {  ipAdres2[i] = int.Parse(str[i]);  //ip.Append(Convert.ToString(ipAdres1[i], 2));  }  Array.Clear(str, 0, str.Length);  str = subnetTb.Text.Split('.');  for (int i = 0; i < str.Length; i++)  {  maska[i] = int.Parse(str[i]);  }  int[] Tmpip1 = new int[4];  int[] Tmpip2 = new int[4];  bool a = true;  for (int i = 0; i < ipAdres1.Length; i++)  {  Tmpip1[i] = ipAdres1[i] & maska[i];  Tmpip2[i] = ipAdres2[i] & maska[i];  if (Tmpip1[i] != Tmpip2[i])  {  a = false;  }  }  if (a)  {  ansLb.Text = "Принадлежат одной подсети.";  }  else  {  ansLb.Text = "Не принадлежат одной подсети.";  }  }  catch { ansLb.Text = "Ошибка."; }  }  }  private void ip1Tb\_TextChanged(object sender, EventArgs e)  {  ansLb.Text = "";  }  private void ip2Tb\_TextChanged(object sender, EventArgs e)  {  ansLb.Text = "";  }  private void subnetTb\_TextChanged(object sender, EventArgs e)  {  ansLb.Text = "";  }  }  } |

Пример работы:



**Входные данные:**

IP-адрес 1: 94.235.16.59

IP-адрес 2: 94.235.23.240

Маска: 255.255.240.0

**Выходные данные:**

Принадлежат одной подсети.

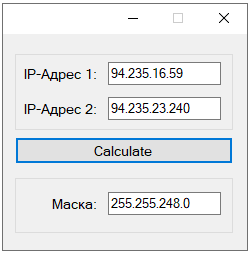
Задание №3

В ходе лабораторной работы необходимо разработать программу, которая определяет минимальную маску подсети если известны IP‑адреса нескольких узлов из этой подсети.

Код:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace zadanie\_3  {  public partial class Form1 : Form  {  public Form1()  {  InitializeComponent();  }  private void Form1\_Load(object sender, EventArgs e)  {  }  private static List<string> convertIPToBinaryIP(string ipAddress)  {  string[] ipArr = ipAddress.Split('.');  int[] ipA = { 1, 1, 1, 1 };  for (int i = 0; i < ipArr.Length; i++)  {  try { ipA[i] = int.Parse(ipArr[i]); } catch { }  }  List<string> binaryIP = new List<string>();  for (var i = 0; i < ipArr.Length; i++)  {  var binaryNo = Convert.ToString(ipA[i], 2);  if (binaryNo.Length == 8)  {  binaryIP.Add(binaryNo);  // binaryIP.push(binaryNo);  }  else  {  var diffNo = 8 - binaryNo.Length;  var createBinary = "";  for (var j = 0; j < diffNo; j++)  {  createBinary += '0';  }  createBinary += binaryNo;  binaryIP.Add(createBinary);  }  }  return binaryIP;  }  private void button1\_Click(object sender, EventArgs e)  {  subnetTb.Text = "";  subnetTb.Enabled = false;  if (ipAd1Tb.Text != "" && ipAd2Tb.Text != "")  {  try  {  subnetTb.Enabled = false;  List<int> SubNet = new List<int>();  List<string> ip1 = new List<string>();  List<string> ip2 = new List<string>();  string subNet = "";  ip1 = convertIPToBinaryIP(ipAd1Tb.Text);  ip2 = convertIPToBinaryIP(ipAd2Tb.Text);  string ipAd1 = ""; string ipAd2 = "";  for (int i=0; i<ip1.Count; i++)  {  ipAd1 += ip1.ElementAt(i);  ipAd2 += ip2.ElementAt(i);  if (i<ip1.Count-1)  {  ipAd1 += "."; ipAd2 += ".";  }  }  bool flag = true;  for (int i=0; i< ipAd1.Length; i++)  {  if (ipAd1[i] == ipAd2[i] && ipAd1[i]!='.' && flag)  {  subNet += "1";  if (i == ipAd1.Length - 1)  {  SubNet.Add(Convert.ToInt32(subNet, 2));  subNet = "";  }  }  else  {  if (ipAd1[i]=='.' || i == ipAd1.Length-1)  {    SubNet.Add(Convert.ToInt32(subNet, 2));  subNet = "";  }  else  {  subNet += "0";  flag = false;  }  }  }  if (SubNet[0] < 128)  {  SubNet[0] = 0;  SubNet[1] = 0;  SubNet[2] = 0;  SubNet[3] = 0;  }  for (int i=0; i<SubNet.Count; i++)  {  subnetTb.Text += SubNet.ElementAt(i).ToString();  if (i< SubNet.Count-1)  {  subnetTb.Text += ".";  }  subnetTb.Enabled = true;  }    }  catch {  subnetTb.Text = "";  subnetTb.Enabled = false;  }  }  }  private void textBox2\_TextChanged(object sender, EventArgs e)  {  subnetTb.Text = "";  subnetTb.Enabled = false;  }  private void ipAd1Tb\_TextChanged(object sender, EventArgs e)  {  subnetTb.Text = "";  subnetTb.Enabled = false;  }  }  } |

Пример работы:



**Входные данные:**

IP-адрес 1: 94.235.16.59

IP-адрес 2: 94.235.23.240

**Выходные данные:**

Маска: 255.255.248.0