

Aufgabe WindowsFormsNetworkCalculator

In dieser Aufgabe soll ein einfacher IPv4-Netzwerk-Rechner entwickelt werden.

Im Internet ist ein ähnlicher IP-Calculator z.B. unter <http://jodies.de/ipcalc> zu finden.

Orientieren Sie sich an der folgenden Darstellung.

	Address	CIDR
	192.168.87.85	23
	txtAddress	cboCidr
	btnNetzwerkBerechnen	
Address:	192.168.87.85	11000000.10101000.01010111.01010101
Netmask:	255.255.254.0	11111111.11111111.11111110.00000000
Wildcard:	0.0.1.255	00000000.00000000.00000001.11111111
NetAddress:	192.168.86.0	11000000.10101000.01010110.00000000
Broadcast:	192.168.87.255	11000000.10101000.01010111.11111111
Host min:	192.168.86.1	11000000.10101000.01010110.00000001
Host max:	192.168.87.254	11000000.10101000.01010111.11111110
Hosts:	510	

Hinweise:

- Führen Sie einige Berechnungen mit dem **IP-Calculator** aus und versuchen den Algorithmus zu verstehen. Wie lässt sich die Netz-Adresse bestimmen? Wie lässt sich die Broadcast-Adresse bestimmen? Welche Formel kann für die Anzahl der Hosts aufgestellt werden?
- Entwerfen Sie die **WinForms**-Obefläche (txtAddress, cboCidr, btnNetzwerkBerechnen).
- Erstellen Sie eine Klasse **IPv4Helper** mit folgenden **statischen** Methoden:
 - static public long GetDez(string strDezOctet) // Berechne 32Bit-Zahl
 - static public string GetBinOctet(long ip4) // Berechne Binärdarstellung
 - static public string GetDezOctet(long ip4) // Berechne Oktett-Darstellung
 - static public long GetHosts(int cidr) // Berechne Hosts
 - static public long GetNetmaskDez(int cidr) // Bestimme Netzmaske
 - static public long GetWildcardDez(int cidr) // Bestimme Wildcard
 - static public bool CheckDezOctet(string strDezOctet) // Prüfe Oktett
 - static public bool CheckCidr(string strCidr) // Prüfe CIDR
- Implementieren Sie in Form1 eine Methode **WriteString** zum Schreiben in die Listbox:
void WriteString(string bezeichnung, string octet="", string bin="")

- Implementieren Sie die Ereignisbehandlungs-Methode für den Button ‚Netzwerk berechnen‘, indem Sie nacheinander die folgenden Elemente bestimmen und ausgeben:
Address, Netmask, Wildcard, NetAddress, Broadcast, Host min, Host max und Hosts.
- Eine mögliche Implementierung von IPv4Helper:

```
public class IPv4Helper
{
    /// <summary>
    /// Prüft ob die Zeichenkette eine gültige IP4-Adresse
    /// in Oktett-Darstellung ist.
    /// </summary>
    /// <param name="strDezOktet"></param>
    /// <returns></returns>
    static public bool CheckDezOktet(string strDezOktet)
    {
        if (!strDezOktet.Contains("."))
            return false;

        string[] strParts = strDezOktet.Split(new string[] { "." },
                                                StringSplitOptions.RemoveEmptyEntries);

        if (strParts.Length != 4)
            return false;

        for (int i = 0; i < strParts.Length; i++)
        {
            string strDez = strParts[i];
            strDez = strDez.Trim();
            byte bTest;
            if (!Byte.TryParse(strDez, out bTest))
                return false;
        }

        return true;
    }

    /// <summary>
    /// Prüft ob die angegebene Zahl ein gültiges CIDR Suffix ist
    /// </summary>
    /// <param name="strCidr"></param>
    /// <returns></returns>
    static public bool CheckCidr(string strCidr)
    {
        if (String.IsNullOrEmpty(strCidr))
            return false;

        int nCidr;
        if (!Int32.TryParse(strCidr, out nCidr))
            return false;

        if ((nCidr < 0) || (nCidr > 32))
            return false;

        return true;
    }
}
```

```

/// <summary>
/// Berechnet aus der IP4-Oktett-Darstellung die interne 32Bit-Zahl in
/// Dezimaldarstellung. Diese dient als Grundlage für weitere Berechnungen.
/// </summary>
/// <param name="strDezOctet"></param>
/// <returns></returns>
static public long GetDez(string strDezOctet)
{
    long ip4 = 0;

    string[] strParts = strDezOctet.Split(new string[] { "." },
                                           StringSplitOptions.RemoveEmptyEntries);

    //for (int i = 0; i < strParts.Length; i++)
    for (int i = 0; i < 4; i++)
    {
        string strDez = strParts[i];
        strDez = strDez.Trim();
        byte byteOctet = 0;
        if (Byte.TryParse(strDez, out byteOctet))
            ip4 = ip4 * 256 + byteOctet;
    }
    return ip4;
}

/// <summary>
/// Bestimme aus der 32Bit-Zahl die binäre Oktett-Darstellung
/// </summary>
/// <param name="ip4"></param>
/// <returns></returns>
static public string GetBinOctet(long ip4)
{
    string strBin = "";
    long lMod = 0;
    long lDiv = ip4;
    // for Schleife notwendig für führende Nullen
    for (int i = 0; i < 32; i++)
    {
        lMod = lDiv % 2;
        lDiv = lDiv / 2;
        strBin = lMod.ToString() + strBin;
        // Oktettdarstellung -> Punkt einfügen
        if (i % 8 == 7 && i > 0 && i < 31)
            strBin = "." + strBin;
    }
    return strBin;
}

/// <summary>
/// Bestimme die Wildcard aus dem CIDR-Suffix
/// </summary>
/// <param name="cidr"></param>
/// <returns></returns>
static public long GetWildcardDez(int cidr)
{
    long lDez = 0;
    for (int i = 0; i < (32 - cidr); i++) {
        lDez = lDez * 2 + 1;
    }
    return lDez;
}

```

```

/// <summary>
/// Bestimme aus der 32Bit-Zahl die IP4 Oktett-Darstellung
/// </summary>
/// <param name="ip4"></param>
/// <returns></returns>
static public string GetDezOctet(long ip4)
{
    string strDezOctet = "";
    long lMod = 0;
    long lDiv = ip4;

    //while (uDiv > 0)
    for (int i = 0; i < 4; i++)
    {
        lMod = lDiv % 256;
        lDiv = lDiv / 256;
        strDezOctet = lMod.ToString() + strDezOctet;
        if (i < 3)
            strDezOctet = "." + strDezOctet;
    }
    return strDezOctet;
}

/// <summary>
/// Bestimme die Anzahl der Hosts aus dem CIDR-Suffix
/// </summary>
/// <param name="cidr"></param>
/// <returns></returns>
static public long GetHosts(int cidr)
{
    long lhosts = 0;
    if (cidr < 32)
        lhosts = Convert.ToInt64(Math.Pow(2, 32 - cidr) - 2);
    return lhosts;
}

/// <summary>
/// Bestimme die Netzmaske aus dem CIDR-Suffix
/// </summary>
/// <param name="cidr"></param>
/// <returns></returns>
static public long GetNetmaskDez(int cidr)
{
    long lDez = 0;
    for (int i = 0; i < 32; i++)
    {
        if (i < cidr)
            lDez = lDez * 2 + 1;
        else
            lDez = lDez * 2;
    }
    return lDez;
}
}

```