Untitled-1

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Runtime.Intrinsics.Arm;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace AP8_PROGRAMME_VALLEE_ESTEBAN
{
    public partial class resultatIPB : Form
    {
        public resultatIPB()
        {
            InitializeComponent();
        }

        private void resultatIPB_Load(object sender, EventArgs e)
        {
            TB_IPSHOW.Text = Global.IPV4.ToString(); // Mise de valeurs dans les textboxs
            TB_CIDRSHOW.Text = Global.CIDR.ToString();
            TB_IPBINAIRE.Text = Global.IPV4binaire;
            TB_MASQUE.Text = "non traitée";
            if (Global.CIDRBINAIRE == null) //mise en valeur conditionnée (seulement si une valeur a été calculée)
            {
                TB_CIDRDECIMALE.Text = "non traitée";
            }
            else
            {
                TB_CIDRDECIMALE.Text = Global.CIDRBINAIRE;
            }

            if (Global.CIDRMASQUE == null)
            {
                TB_MASQUE.Text = "non traitée";
            }
            else
            {
                TB_MASQUE.Text = Global.CIDRMASQUE;
            }

            if (Global.CLASSIP is null)
            {
                TB_CLASSE.Text = "non traité";
            }

            else
            {
```

```
                TB_CLASSE.Text = Global.CLASSIP;
            }

            if (Global.nbrhotes == 0)
            {
                TB_Hotes.Text = "non traité";
            }

            else
            {
                TB_Hotes.Text = Global.nbrhotes.ToString();
            }

            if (Global.nbsousres == 0)
            {
                TB_NBRreseaux.Text = "non traité";
            }

            else
            {
                TB_NBRreseaux.Text = Global.nbsousres.ToString();
            }
        }

        private void BTN_TRAITEMENT_Click(object sender, EventArgs e)
        { ///Vérifier au chargement si des données ne sont pas déjà disponibles
            MessageBox.Show("L'IP et le masque ont étés traités avec succès !"); //démarrage du
traitement
            int cidr = Global.CIDR;
            string bits = new string('1', cidr).PadRight(32, '0');
            string CIDRBINAIRE = //Décomposition du CIDR binaire en 4 parties séparés d'un
point
                bits.Substring(0, 8) + "." +
                bits.Substring(8, 8) + "." +
                bits.Substring(16, 8) + "." +
                bits.Substring(24, 8);
            Global.CIDRBINAIRE = CIDRBINAIRE; //Enregistrement de la variable en global
            TB_CIDRDECIMALE.Text = Global.CIDRBINAIRE;

            String[] CIDRBINAIRETRAITEMENT = CIDRBINAIRE.Split("."); //Conversion du CIDR en
binaire en Masque entier
            Int16 c1 = Convert.ToByte(CIDRBINAIRETRAITEMENT[0], 2); //Traitement des parties
une à une pour exclure les points
            Int16 c2 = Convert.ToByte(CIDRBINAIRETRAITEMENT[1], 2);
            Int16 c3 = Convert.ToByte(CIDRBINAIRETRAITEMENT[2], 2);
            Int16 c4 = Convert.ToByte(CIDRBINAIRETRAITEMENT[3], 2);
            Global.CIDRMASQUE = c1 + "." + c2 + "." + c3 + "." + c4; //Enregistrement de la
réponse dans un string
            TB_MASQUE.Text = Global.CIDRMASQUE;
            if (Global.IPnb1 <= 127 && Global.IPnb1 >= 0) // On donne les classes des IPs,
selon les plages connues.
            {
                Global.CLASSIP = "A";
            }
```

```
            else if (Global.IPnb1 <= 191 && Global.IPnb1 >= 128)
            {
                Global.CLASSIP = "B";
            }
            else if (Global.IPnb1 <= 223 && Global.IPnb1 >= 192)
            {
                Global.CLASSIP = "C";
            }
            else if (Global.IPnb1 <= 239 && Global.IPnb1 >= 224)
            {
                Global.CLASSIP = "D";
            }

            else
            {
                Global.CLASSIP = "E";
            }
            TB_CLASSE.Text = Global.CLASSIP;
            ///Calcul du nombre d'hotes (on compte le nombre de zéro dans le masque)
            int nbrzero = 0;
            for (int i = 0; i < Global.CIDRBINAIRE.Length; i++) //On va compter avec un index le nombre de zero
            {
                if (Global.CIDRBINAIRE[i] == '0')
                {
                    nbrzero = nbrzero + 1; //Compteur
                }
            }
            Global.nbrhotes = ((int)Math.Pow(2, nbrzero) - 2); // 2 puissance nombre de zéro - 2 (broadcast et adresse réseau)
            TB_Hotes.Text = Global.nbrhotes.ToString(); //Affichage sur TB


            ////////////////////////////////
            //Determination Classe Réseaux//
            ////////////////////////////////

            Int16 cidrplage = 0;
            if (Global.CIDR>=8 && Global.CIDR<16) //On vérifie que le CIDR commence par 255
            {
                Global.CLASSCIDR = "A";
                cidrplage = 8;
            }
            else if (Global.CIDR >= 16 && Global.CIDR < 24) //On vérifie 255.255
            {
                Global.CLASSCIDR = "B";
                cidrplage = 16;
            }
            else if (Global.CIDR >=24 && Global.CIDR < 32) //On vérifie 255.255.255
            {
                Global.CLASSCIDR = "C";
                cidrplage = 24;
            }
            else
```

```
152             {
153                 Global.CLASSCIDR = "D/E"; //0 sous réseaux pour ceux-ci
154                 cidrplage = 32;
155             }
156 
157             Global.nbsousres = (int)Math.Pow(2, Global.CIDR-cidrplage); // 2 puissance nombre de 1 rajoutés entre cidr et plages de masques habituelles.
158             TB_NBRreseaux.Text = Global.nbsousres.ToString(); //on met la valeur dans le tableau
159         }
160 
161         private void BTN_RETOUR_Click(object sender, EventArgs e)
162         {
163             Form1 page1 = new Form1(); //Ouverture d'une page, fermeture d'une autre
164             this.Close();
165             page1.Show();
166         }
167 
168         private void TB_QUITTER_Click(object sender, EventArgs e)
169         {
170             this.Close();
171         }
172 
173         private void BTN_CLCNBRRESEAUX_Click(object sender, EventArgs e)
174         {
175 
176         }
177     }
178 }
```