Ligne 16-24 : initialisation.

Nous avons l' initialisation de chaque variable, et tableau que nous utiliserons dans une voir plusieurs fonction/procédure.
Nous commençons avec le tableau conver qui consiste à stocker 8 valeur 128, 64, 32, 16, 8, 4, 2, 1 qui servirons à convertir des Ip_{10} en Ip_2 .
Puis les tableaux IP binaire, Maskbin , et Réseau qui permettrons de stocker l'ip, le Mask et le réseau en Binaire .
un Tableau Ips qui stock l'ip séparer convertie en Décimal.
on définit un separateur '.' ainsi qu'un tableau Ipsep qui stockeras ce que le séparateur auras separer (l'ip mais en string) .
une valeur tempo
Et la valeur Mask qui vas prendre le mask CIDR .

Ligne 29-36 : procédure dans le cas que le bouton « B_Convert_p1_Click » soit utiliser (Bouton Principal) .

L31 : Si l'utilisateur clique sur le buttons alors, nous séparons l'ip qui auras écrit avec le separateur et nous le mettons dans le tableau Ipsep
L32 : on fait une boucle pour Convertir L'ip séparer (Ipsep) en entier dans le tableaux Ips
L34 : on initialise le tempo a 0
L35 : on lance la procédure Convertir().

Ligne 37-64 : Procédure « Convertir() » qui permet de convertir des entier ≤ 255 en $base_2$.

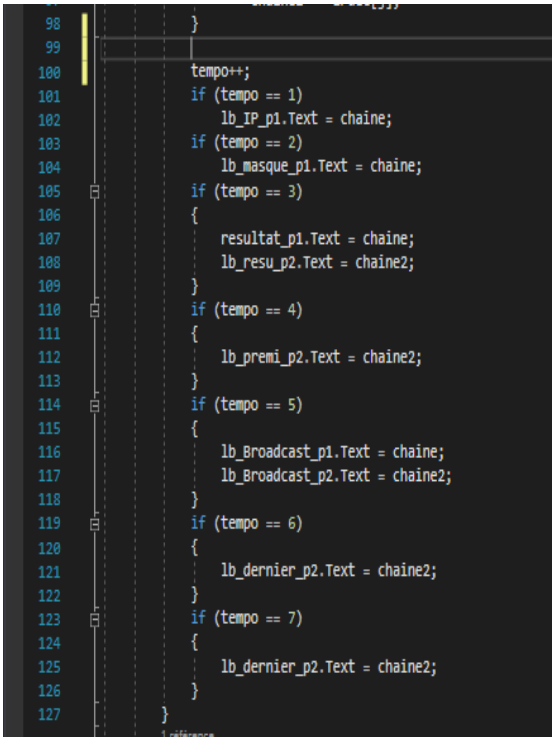
L39 : on initialise la posi a 0.
L40 : on fait une boucle qui vas parcourir le tableaux Ips (longueur 4)
L42/56-57: on vérifie que dans Le tableaux de Ips a la position de la boucle que la valeur en entier et ≤ 255 car une IP ne peut faire que 1octet.Sinon on indique un message d'erreur.
L44 : on fait une autre boucle qui vas parcourir le tableaux conver ou il y a les valeurs prédéfinies
L46-52 : on regarde le tableau de Ips a la position de la boucle si il est ≥ au tableaux conver a la position de la second boucle Si ces le cas on prend la valeur de Ips et on la soustrait a la valeur de conver et dans le tableau Ipbinaire de position de posi on mes 1 Sinon on mais 0 dans le tableaux ipbinaire position de posi puis avant la fin de la second boucle et qu'elle se repette on fait un posi++ pour augmenter la position et avoirs normalement le tableaux Ipbinaire remplis de 32 valeur a la fins des 2 boucle (1^{er} boucle x4 2eme boucle x8 = 32)
L59-60 : on fait en sort de rendre Visible 2 label qui étais invisible
L61 : on lance la procédure affichage avec le tableau de Ipbinaire.
L62 : On vérifie si dans la case ou on indique le masque si une valeur et indiquer
L63 : si c'est le cas on lance la procédure Masque()

L65-127 : procédure affichage(...)

```
6 références
void affichage(int[] IPbinaire)
{
    string chaine = "";
    string chaine2 = "";
    for (int i = 0; i < IPbinaire.Length; i++)
    {
        chaine += "" + IPbinaire[i];

        if ((i == 3) || (i == 11) || (i == 19) || (i == 27))
        {
            chaine += " ";
        }
        if ((i == 7) || (i == 15) || (i == 23))
        {
            chaine += " . ";
        }
    }
    int[] IPdec = { 0, 0, 0, 0 };
    int posi = 0;
    for (int j = 0; j < 4; j++)
    {
        for (int i = 0; i < conver.Length; i++)
        {
            if (IPbinaire[posi] == 1)
            {
                IPdec[j] += conver[i];
            }
            posi++;
        }
        if (j != 3)
            chaine2 += IPdec[j] + ".";
        else
            chaine2 += IPdec[j];
    }
```

Elle permet d'afficher une Ip (celle qui seras donner quand on l'appel)
elle commence afficher les 4 premier bite pour faire un espace puis affiche les 4 suivant pour après l'octet mettre un point et cela jusqu'aux 32bits , après la chaîne fait on vas récupérer l'ip en binaire pour la convertir en décimal et l'utiliser quand cas de besoin .



```
98          }
99
100         tempo++;
101         if (tempo == 1)
102             lb_IP_p1.Text = chaine;
103         if (tempo == 2)
104             lb_masque_p1.Text = chaine;
105         if (tempo == 3)
106         {
107             resultat_p1.Text = chaine;
108             lb_resu_p2.Text = chaine2;
109         }
110         if (tempo == 4)
111         {
112             lb_premi_p2.Text = chaine2;
113         }
114         if (tempo == 5)
115         {
116             lb_Broadcast_p1.Text = chaine;
117             lb_Broadcast_p2.Text = chaine2;
118         }
119         if (tempo == 6)
120         {
121             lb_dernier_p2.Text = chaine2;
122         }
123         if (tempo == 7)
124         {
125             lb_dernier_p2.Text = chaine2;
126         }
127     }
```

Ici nous gérons l'affichage celons le tempo que nous lui avons donner pour afficher celons le bon moment l'ip en binaire dans les labels ou par la suite l'ip qui est convertie en décimal dans des labels aussi.

Dès que le boutons valider et activer (quand on rentre l'ip ou/et le mask) la valeur de tempo sera = 0.

L128-158: procédure Masque()

```
128     void Masque()
129     {
130         string[] MaskCIDR;
131         MaskCIDR = tb_masque_p1.Text.Split('/');
132         Mask = Convert.ToInt32(MaskCIDR[1]);
133         if (Mask > 32)
134         {
135             MessageBox.Show("Mask incunue");
136         }
137         else
138         {
139             Mask--;
140             for (int i = 0; i < 32; i++)
141             {
142                 if (Mask >= i)
143                 {
144                     Maskbin[i] = 1;
145                 }
146                 else
147                 {
148                     Maskbin[i] = 0;
149                 }
150             }
151             Mask++;
152             lb_lmask_p1.Visible = true;
153             lb_masque_p1.Visible = true;
154             lb_ET_p1.Visible = true;
155             affichage(Maskbin);
156             CalcReseau();
157         }
158     }
```

Cette procédure va permettre de traduire le masque en CIDR dans le label en binaire qui par la suite sera utiliser pour calculer l’adresse Ip réseau.
Nous vérifions que le Masque CIDR et bien valide.
Et avant de généré le Masque en Binaire nous faisons un -1 au Mask car nous ferons une boucle qui commencera de 0.

L159-173: Procédure CalcReseau

```
1 référence
void CalcReseau()
{
    egale_p1.Visible = true;
    resultat_p1.Visible = true;
    lb_Re_p1.Visible = true;
    lb_resu_p2.Visible = true;
    lb_re_p2.Visible = true;

    for (int i = 0; i < Maskbin.Length; i++)
    {
        Reseau[i] = IPbinaire[i] & Maskbin[i];
    }
    affichage(Reseau);
    premierPC(Reseau);
}
```

Cette procédure nous permet d'avoir L'Ip réseau en Binaire que nous l'affichons par la suite

on fait simplement une boucle qui fait un ET logique et on enregistre les valeur dans le Tableau Reseau[]

on envois cet Ip en binaire dans une procédure (premierPC()) qui vas calculer l'adresse Ip du premier Pc attribuable Puis le même procéder mais pour ce cous ci le dernier Pc via la procédure dernierPc()

L176-184: Procédure premierPC

```
1 référence
void premierPC(int[] Reseau)
{
    lb_1_p2.Visible = true;
    lb_premi_p2.Visible = true;
    int[] Reseau1 = new int[32];
    Reseau1 = Reseau;
    Reseau1[31] = 1;
    affichage(Reseau1);
}
```

Elle va simplement copier dans un autre tableau l'ip réseau et ajouter à la dernière valeur un 1 qui quand elle sera traduit par la procédure affichage affichera le premier Pc dispos sur ce réseau

L185-203: Procédure dernierPc

```
1 référence
void dernierPC(int[] Reseau)
{
    lb_dernier_p2.Visible = true;
    lb_der_p2.Visible = true;
    lb_iBroadcast_p1.Visible = true;
    lb_iBroadcast_p2.Visible = true;
    lb_Broadcast_p1.Visible = true;
    lb_Broadcast_p2.Visible = true;
    int[] Reseau2 = new int[32];
    Reseau2 = Reseau;
    for (int i = Mask; i < Reseau2.Length; i++)
    {
        Reseau2[i] = 1;
    }
    int[] Broadcast = Reseau2;
    affichage(Broadcast);
    Reseau2[31] = 0;
    affichage(Reseau2);
}
```

Copie dans un autre tableau l'adresse réseau et on commence notre boucle a la position du Mask nous mettons que des 1 dans ce tableau copier précédemment copier et le copier aussi dans un tableau Broadcast et on l'envoie dans la procédure affichage() et à la dernier valeur on ajoute un du tableau Reseau2[] nous mettons 0 ce qui correspondra ainsi a l'ip du dernier Pc de ce réseau
et on l'envoie à la procédure affichage().