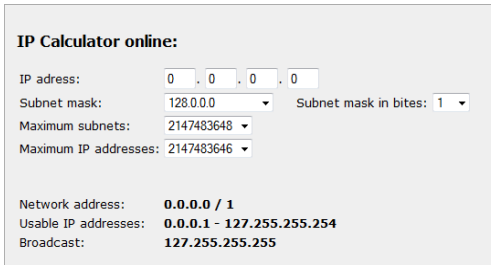
Codage C++ sous Qt

Projet Calculatrice IPv4

Étude de L'interface Homme - Machine

L'objectif de cet exercice est de concevoir une calculatrice IPv4 permettant de déterminer à partir d'une adresse IPv4 de son masque de réseau, ou de son suffixe, ou d'un nombre d'hôtes ou encore d'un nombre de sous réseaux :

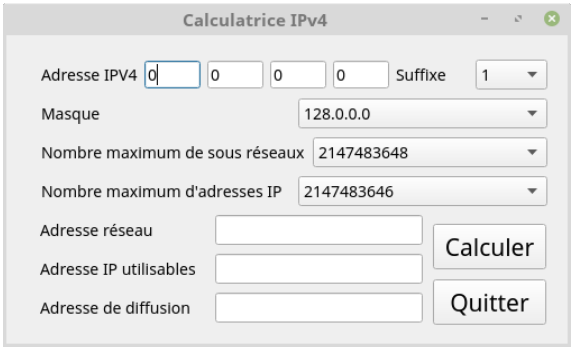
- l'adresse réseau à laquelle elle appartient,
- l'adresse de broadcast du réseau
- et les adresses utilisables dans ce réseau.



Un exemple est proposé avec la calculatrice en ligne :http://ipcalc.nmonitoring.com/ comme le montre la figure ci-dessus.

1. Réalisation de l'IHM

1. Dans un projet nommé **CalculatriceIP** de type ***Application graphique Qt*** réalisez le **Widget** suivant :



Le nom de l'objet gérant la fenêtre principale est également **CalculatriceIP**.

L'adresse IPv4 est composé de 4 zones d'édition. Leur largeur mini et maxi sont fixées à 50 pixels.

Le suffixe, le masque, le nombre maximum de sous réseaux et le nombre maximum d'adresses IP sont représentés par des ***comboBox***.

Les trois autres zones d'éditions sont en lecture seule, elles serviront à afficher les résultats.

Le plus grand soin sera porté au nom de variables associé à chaque widget. Il respectera la convention suivante : **typeDObjet_Fonction** exemple **lineEdit_Adresse1**

Le bouton Quitter servira à sortir de l'application. Grâce à l'éditeur de signaux et de slots, il sera associé au slot ***close()*** de la classe CalculatriceIP (Qwidget).

Attention les données des **comboBox** seront calculées par programme.

2. Le constructeur de la classe CalculatriceIPv4 doit initialiser les différents **comboBox** avec les valeurs appropriées :

Désignation	Première valeur	Dernière valeur	Remarques
Le Suffixe	1	30	
Le Masque	128.0.0.0	255.255.255.252	
Nombre de sous réseau	2147483848	4	2^n avec n le nombre de bits empruntés.
Nombre d'adresses IP utilisables	2147483846	2	2^n -2 avec n le nombre de bits pour définir les hôtes.

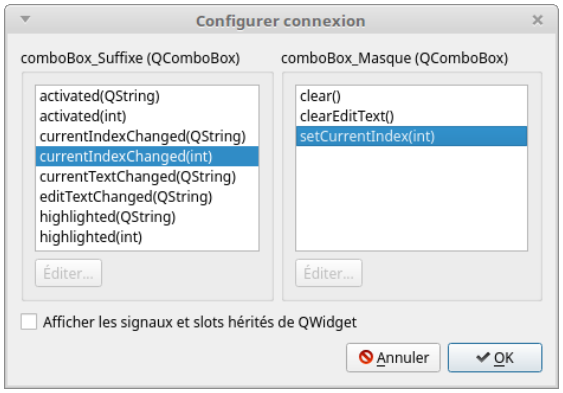
Par calcul, remplissez les différents éléments des **comboBox** en complétant l'extrait de code suivant :

```
CalculatriceIP::CalculatriceIP(QWidget *parent) :
Qwidget(parent),
ui(new Ui::CalculatriceIP)
{
   ui→setupUi(this);
   int suffixe ;
   quint64 nbMaxSousReseau ;
   quint8 masque[4]={0,0,0,0};
   QString leMasque;
   for(suffixe = 0 ; suffixe < 30 ; suffixe++)
   {
      nbMaxSousReseaux = static_cast<quint64>  (qPow(2, 32-suffixe-1)) ;
      masque[suffixe / 8] += static_cast<quint8>  (qPow(2,(7-(suffixe % 8))) ;
      leMasque.clear();
      for(int indice = 0 ; indice < 4; indice++)
      {
         // fabrication de la chaîne correspondant au masque

      }
      // Ajout des valeurs dans les 4 comboBox

   }
}
```

3. Dans un premier temps, il est nécessaire de faire correspondre le suffixe et le masque, chaque fois que l'utilisateur agit sur le suffixe, le masque correspondant s'affiche. Pour cela, utilisez « **l'éditeur Signals/slots** » touche **F4** du créateur d'interface et configurer la connexion du signal ***currentIndexChanged*** du comboBox dédier au suffixe au slot ***setCurrentIndex*** du comboBox dédier au Masque

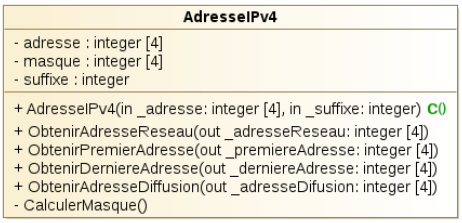


4. Procéder de la même manière avec les autres **comboBox**, puisque toutes les valeurs sont liés entres elles.

5. Écrire le code de la méthode permettant de convertir un tableau d'octets en **QString** en ajoutant les points nécessaires entre les décimales dont le prototype est : **QString CalculatriceIP::ConvertirTabOctetEnQString(const quint8 tabOctet[]) ;**

2. Classe AdresseIPv4

1. Réalisez le prototype de la classe AdresseIPv4 en respectant le schéma suivant :



2. Pour le codage de la classe, les indications suivantes sont données :
   - Le constructeur reçoit l'adresse IP dans un tableau d'octets et le suffixe dans une deuxième variable de type entier sous la forme d'un octet (**quint8** sous Qt inclure **QtGlobal**). Il est chargé d'initialiser les attribut de la classe.
   - La méthode **void CalculerMasque**() est appelée par le constructeur pour déterminer la valeur du masque en valeur décimale pointée à partir du suffixe.
   - Les autres méthodes, dont le nom est suffisamment explicite, ont comme paramètre de sortie chacune un tableau d'octets contenant la valeur désirée. Ces éléments sont calculés à partir de l'adresse IP fournie et du masque réseau.

   Rappel :
   - L'adresse réseau est calculée à partir de l'opération binaire ET entre l'adresse IP et le masque.
   - La première adresse utilisable est l'adresse réseau + 1.
   - L'adresse de diffusion, est calculée à partir de l'opération binaire OU entre l'adresse réseau et l'inverse du masque.
   - La dernière adresse utilisable est l'adresse de diffusion -1.

3. Intégration

- Compléter l'affichage dans la fenêtre principale en utilisant la classe AdresseIPv4. Vérifier le fonctionnement de l'application. Lors de l'appui sur le bouton « **Calculer** », les paramètres du réseau sont mis à jour.
- Compléter le constructeur pour que les zones d'édition utilisées lors de la saisie de l'adresse IP n'acceptent que des nombres entre 0 et 255. Pour cela vous pouvez utiliser l'objet QintValidator. Lien https://doc.qt.io/qt-5/qintvalidator.html
- Toujours dans le constructeur, réalisez la connexion du signal QlineEdit::textChanged au slot utilisé par le bouton « **Calculer** » de chaque zone d'édition afin que l'affichage des résultats se fasse à chaque changement. Faites de même avec une des **comboBox**.
- Ajoutez un bouton « **A propos** » sur votre interface. Ce bouton doit faire apparaître une boîte de message avec le nom de l'application, son auteur et la date de création.

Annexes

Utilisation des différents widget

1. Les zones d'édition :

Sous Qt les zones d'édition sont représentées par des ***QLineEdit***. La méthode ***setText()*** permet d'afficher un ***QString*** à l'intérieur et ***Text()*** affecte un ***QString*** avec le contenu de la zone. Exemples :

```
ui->lineEdit->setText("Bonjour") ;   // pour afficher
QString leTexte ;
leTexte = ui->lineEdit->text () ;      // pour récupérer la valeur
```

Dans notre problème, Il est parfois nécessaire de convertir un entier en **QString** et vis et versa. Le site de développez.com http://qt.developpez.com/doc/4.7/qstring/#toint ou celui de Qt https://doc.qt.io/Qt-5/qstring.html nous propose la méthode :

int QString::toInt(bool *ok = 0, int base = 10) ;

Cette méthode convertit une chaîne de caractères en entier en fonction de la base sélectionnée, par défaut en base 10, le booléen ok est mis à jour par la fonction si l'opération à pu être réalisé correctement.

Pour la fonctionnalité inverse, il existe deux méthodes possibles setNum(int n,int base=10) ou number(int n,int base=10). La deuxième est une méthode ***static*** et n'a donc pas besoin d'une instance pour être utilisée.

Voir http://qt.developpez.com/doc/4.7/qstring/ ou https://doc.qt.io/Qt-5/qstring.html

Exemples :

```
int a = 63;
QString s = QString::number(a, 16);             // s == "3f"
QString t = QString::number(a, 16).toUpper();   // t == "3F"
```

ou

```
QString str;
str.setNum(1234);      // str == "1234"
```

2. Les ComboBox

Les comboBox ou boîtes combinés permettent la saisie dans une zone d'édition ou la sélection d'un item dans une liste. Le remplissage de la liste est réalisé par la méthode :

void QComboBox::addItem (const QString &*text*, const QVariant &*userData*=QVariant()) ;

Dans un premier temps, seul le paramètre ***text*** est utile. Il permet d'ajouter une chaîne de caractère sous la forme d'une **QString** à la liste. Les entiers devront être transformé en chaîne de caractères avant d'être ajoutés.

Le signal ***currentIndexChanged*** est émit chaque fois que l'utilisateur sélectionne un élément de la liste. Il existe sous deux formes, la première transmet l'index d'un item lors d'une sélection, la seconde transmet le texte de l'item sélectionné.

```
void    currentIndexChanged ( int index )
void    currentIndexChanged ( const QString & text )
```

Associés à un slot, ils peuvent chacun traiter les choix de l'utilisateur. Les index varient de 0 à nb-1 éléments, la valeur -1 correspond à une valeur erronée ou vide. La valeur index ou ***text*** suivant le cas est transmise au slot qui réagit au signal.

Le slot public void **setCurrentIndex**(int index) est utilisé pour positionner l'élément visible de la liste. Le slot public void **clear**() est utilisé pour vider la liste de son contenu.