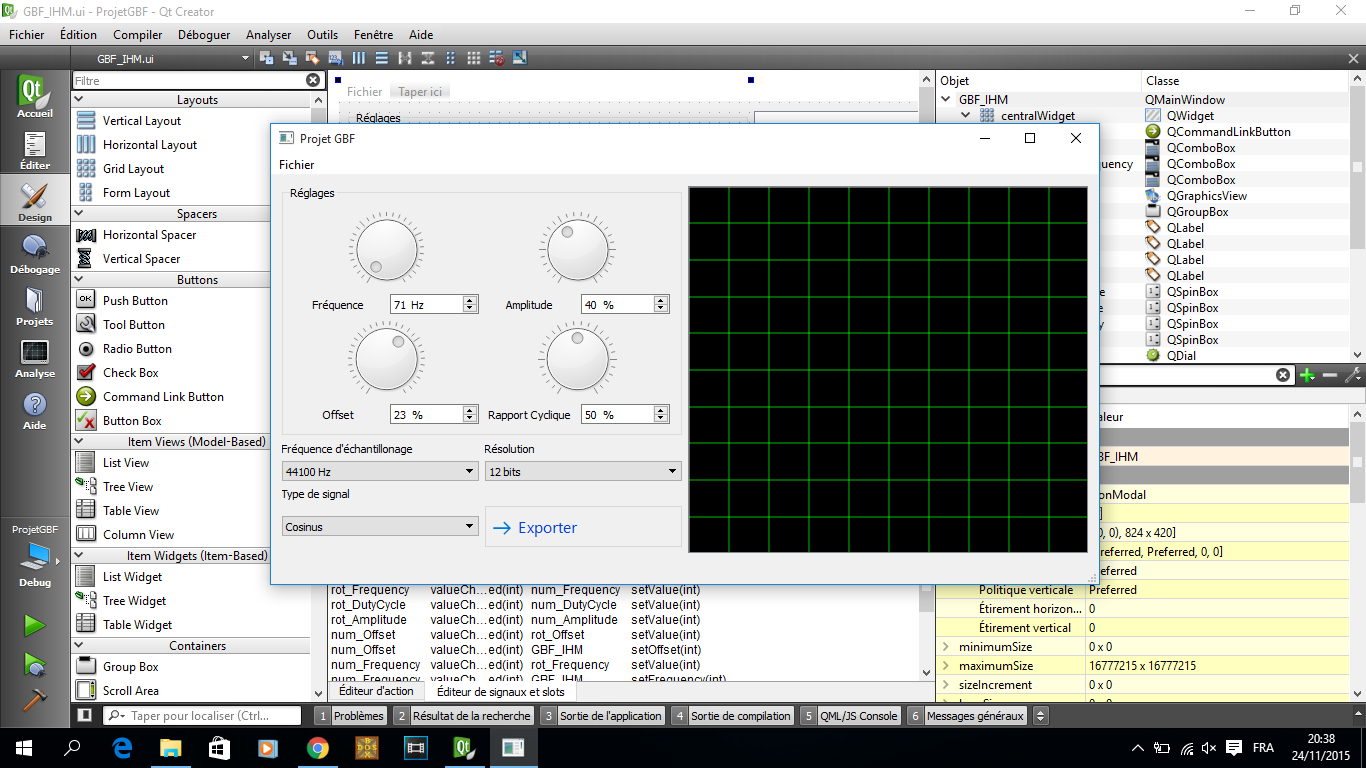
Solution à mettre en œuvre

**Objectif :** Créer une application permettant de générer des fichiers wave contenant des signaux simples (sinus, cosinus, carré, triangle)

## Interface homme machine

### L’interface homme machine



### Explications des relations de l’IHM

## Explication mathématique des signaux

### Généralité sur les signaux

Dans le projet, cinq classes ont été développées pour la partie « création des signaux ». La classe « GBF\_Signal » est la classe mère. Elle contient tous les attributs et les méthodes liées à la création des signaux. Les attributs de cette classe sont les suivants :

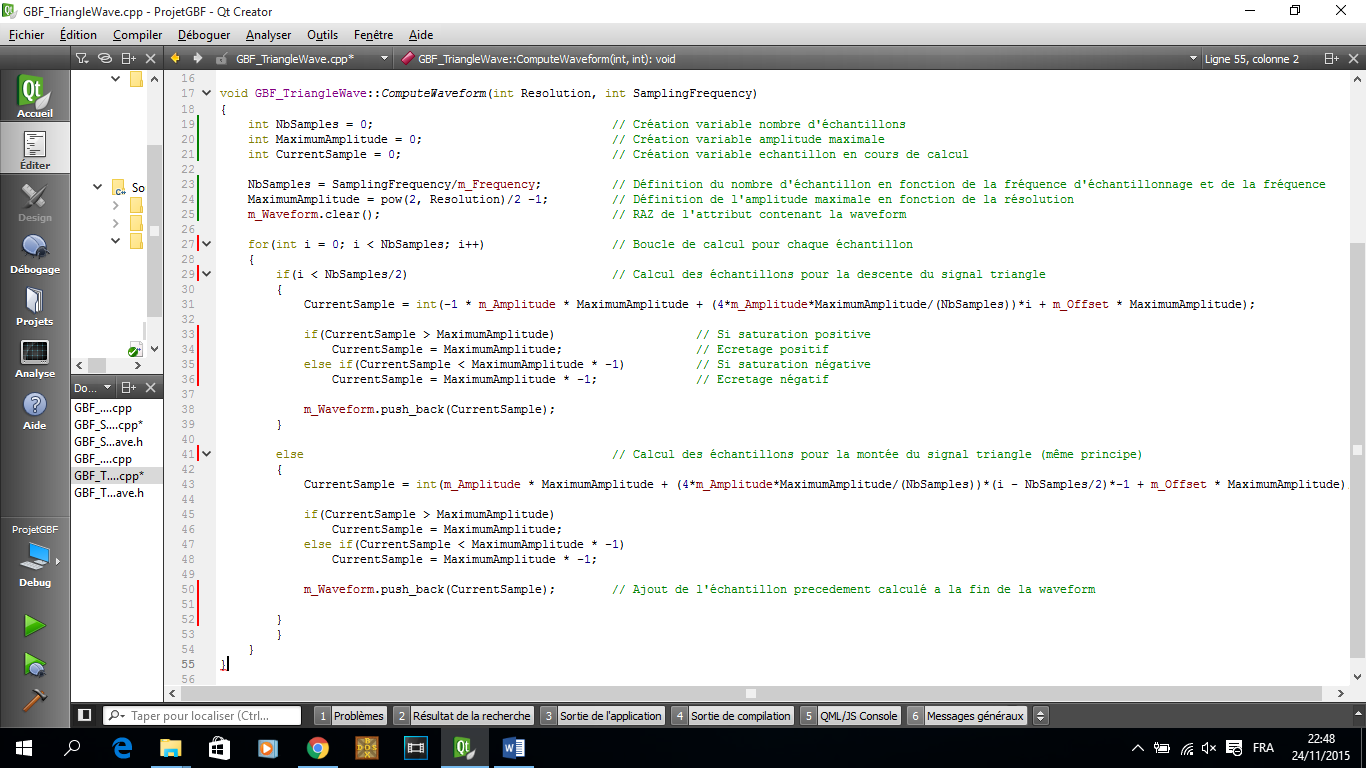
* M\_Frequency : Fréquence en Hz du signal
* M\_Offset : Composante continue du signal
* M\_Amplitude : Amplitude crête à crête du signal
* M\_Waveform : Contient les points calculés du signal
* M\_Type : Contient un enum relatif au type de signal (1 : Carré, 2 : Sinus , 3 : Cosinus, 4 : triangle)

Des classes filles héritent des propriétés et des méthodes de la classe mère. Seule la classe carrée dispose d’un attribut supplémentaire M\_DutyCycle de façon à agir sur le rapport cyclique de ce signal en particulier.

Seulement une période est calculée pour chaque signal généré. Le motif sera ensuite répété en fonction du paramètre «  Temps du signal » lors de la génération du fichier Wave.

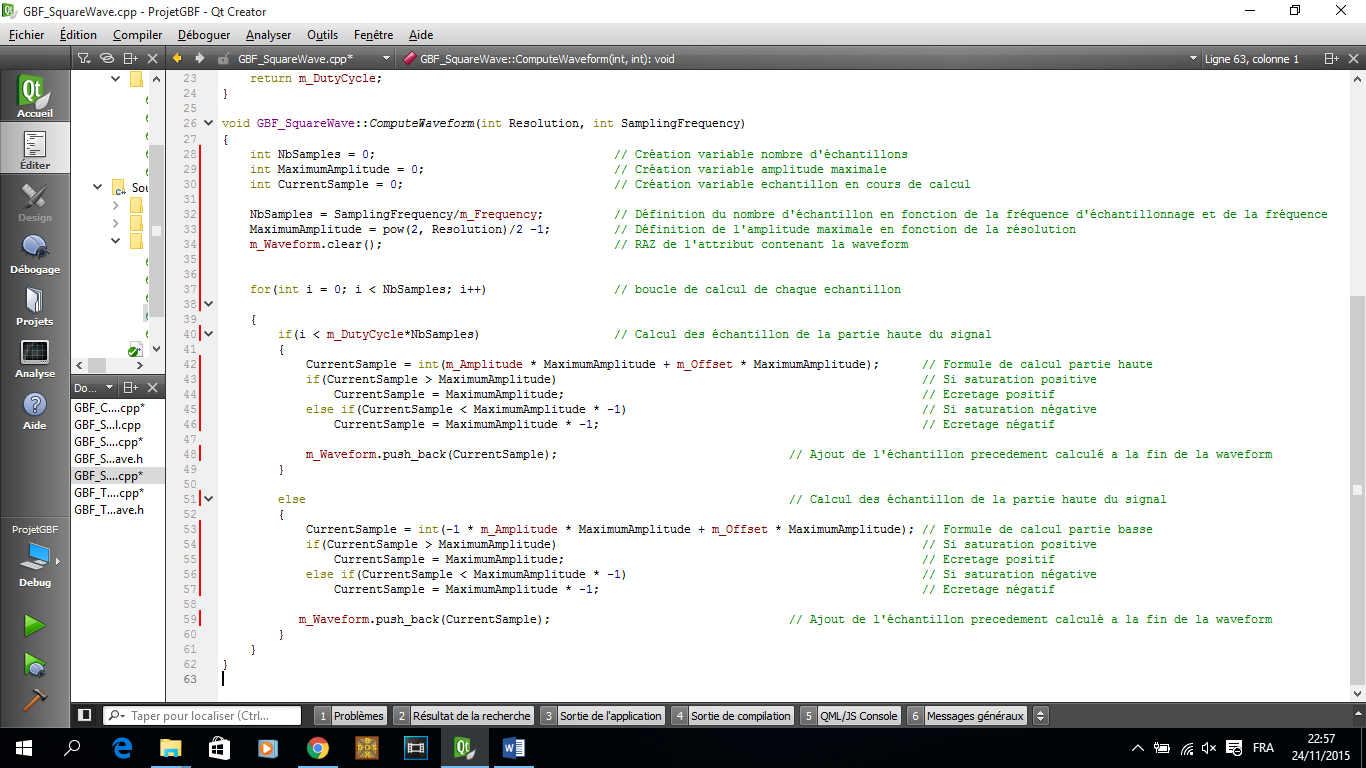
### Le signal triangle

Ci-dessous le code C++ commenté développé pour la fonction triangle.



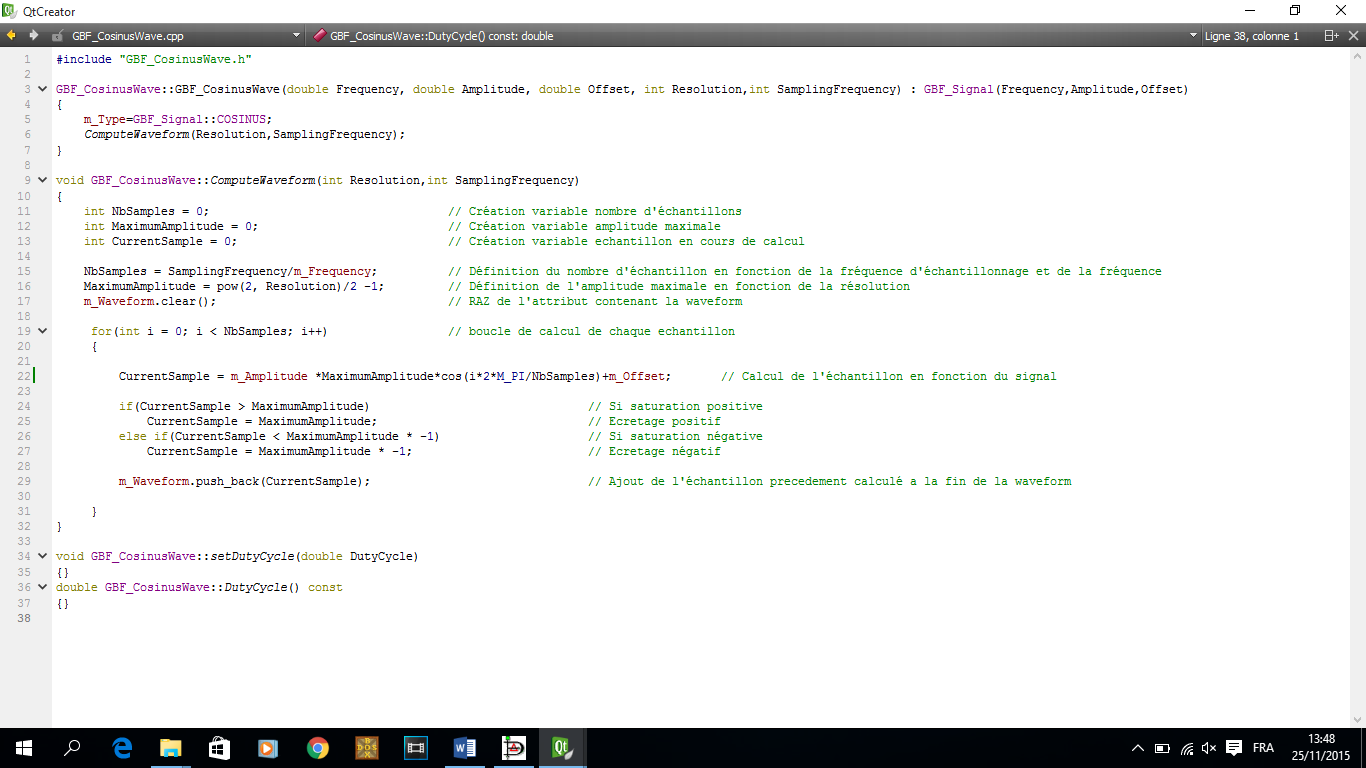
### Le signal carré

Ci-dessous le code C++ commenté développé pour la fonction carré.



### Le signal sinus et cosinus

Ci-dessous le code C++ commenté développé pour la fonction composition du cosinus. La fonction sinus a été développée sur le même modèle en remplaçant simplement le cos par un sinus.



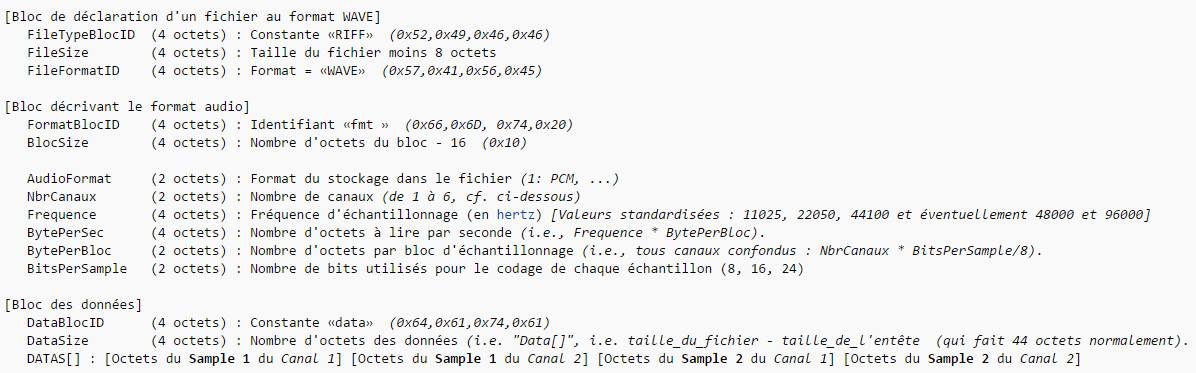
## L’exportation du signal généré en fichier WAV

### La trame à mettre en œuvre

La génération du signal Wave se fait en deux étapes :

* création d'un fichier
* remplissage du fichier selon le format Wave

#### Format du fichier



On retrouve trois blocs distincts qui forment l'en-tête de 44 octets :

* Un bloc de déclaration sur 12 octets
* Un bloc décrivant le format sur 24 octets
* Un bloc de données de 8 octets minimum

### La solution choisie

Lors de l'appui sur le bouton "Enregistrer", la méthode "Export" de la classe GBF\_Generator sera exécutée. Cette méthode recevra en paramètres :

* un string contenant l'adresse du fichier
* un string contenant le nom du fichier
* un objet de type GBF\_Signal passé en référence
* un double pour la durée d'enregistrement

Nous utiliserons utiliser l'attribut de fréquence de l'objet GBF\_Signal pour retrouver sa période et donc calculer le nombre de duplication nécessaire pour atteindre la durée d'enregistrement demandée. Les points du signal qui sont stockés dans un le vector m\_WaveForm, seront elles mises en forme dans la zone réservé du fichier Wave.

Une fois les données du signal copiées, nous pourrons renseigné la taille du fichier (Taille -8octets).

A noté que certains champs de l'en-tête seront fixés :

* Constante "RIFF" en héxa
* Format "WAVE" en héxa
* Identifiant "fmt " en hexa
* Format de stockage dans le fichier : 1 pour PCM
* Nombre canaux = 2
* Format audio = 1 pour PCM
* Constante "data" en héxa