Compiler un code C ou C++ utilisant Cplex sous Visual Studio

Hélène Toussaint, juillet 2018

Objectif : Ce document explique comment compiler un code C ou C++ qui utilise la librairie Cplex sous Visual Studio (toute version confondue). Les exemples et captures d'écran concernent Cplex 12.7 et Visual Studio 2015 mais la marche à suivre s'applique à toutes les versions.

1 Généralités

1.1 *CPLEX Callable Library* ou *Concert Technology* ?

Déjà il faut savoir que la librairie Cplex existe sous plusieurs formes (qui sont toutes inclues dans les répertoires d'installation de Cplex) pour pouvoir s'adapter à plusieurs langages de programmation. Celles qui nous intéressent ici sont :

- la ***CPLEX Callable Library*** (pour les développeurs C),
- la ***Concert Technology*** (pour les développeurs C++).

A partir d'un code C on ne peut appeler bien sûr que la *Callable Library***.** A partir d'un code C++ on peut appeler au choix la *Callable Library* ou la *Concert Technology.* La *Concert Technology* possède des fonctions plus haut niveau et une interface orientée objet qui la rendent plus facile à utiliser (notamment pour la gestion des contraintes) mais qui nécessitent d'avoir quelques notions de programmation orientée objet.

1.2 Compatibilité des versions de Cplex et Visual Studio

Toutes les versions de Cplex ne sont pas compatibles avec toutes les versions de Visual Studio. Si vous essayez de compiler un code utilisant Cplex avec une version non compatible de Visual Studio vous risquez fort de vous retrouver avec des erreurs incompréhensibles au moment du *link*.
Pour vous assurer que vous avez des versions compatibles il faut regarder dans le répertoire **`cplex/lib`** (qui se trouve dans le répertoire d'installation de Cplex) le nom des répertoires contenant les **`.lib`** pour Visual Studio. Dans le nom du répertoire se trouve la version de Visual Studio compatible. Par exemple sur la Figure 1 on voit que sont compatibles Visual Studio 2013 et Visual Studio 2015 avec la version de Cplex installée (12.7)

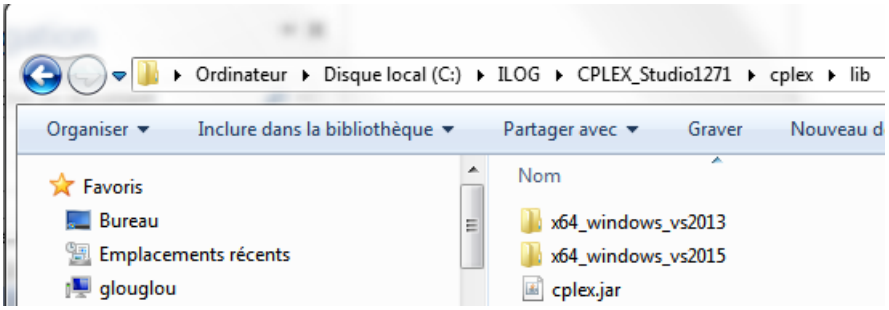

Figure 1 : voir les versions de Visual Studio compatibles avec la librairie Cplex installée

2 Modifications à apporter au projet Visual

Afin de compiler un code utilisant Cplex dans Visual Studio il est nécessaire d'apporter des modifications au projet Visual. Comme il est plus difficile de compiler la *Concert Technology* (C++) que la *callable library* (C) (dans la mesure où il y a plus de modifications à apporter au projet Visual) on montrera la marche à suivre pour la *Concert Technology,* en précisant les étapes inutiles pour la *callable library*.

NB. On suppose que le lecteur est familier de Visual Studio, qu'il sait au moins créer un projet (sinon voir [1]) et accéder aux propriétés du projet.

Quand on crée un projet Visual Studio les valeurs des options du projet sont celles par défaut. Quand on modifie une de ces valeurs elle s'affiche alors en gras dans la "page de propriétés du projet". Pour savoir quelles options doivent être modifiées il suffit donc d'ouvrir avec Visual Studio un des exemples fourni par Cplex et de regarder quelles options sont en gras.

Dans la suite on détaille la marche à suivre pour compiler un code utilisant Cplex 12.7 avec VS 2015. Il est possible qu'avec d'autres versions il y ait des différences, il faut donc toujours se référer aux exemples fournit dans votre répertoire d'installation de Cplex. Supposons que votre répertoire d'installation soit **`C:\ILOG\CPLEX_Studio1271`**, vous trouverez les projets exemples dans **`C:\ILOG\CPLEX_Studio1271\cplex\examples\x64_windows_vs2015\stat_mda`** (voir Figure 2).

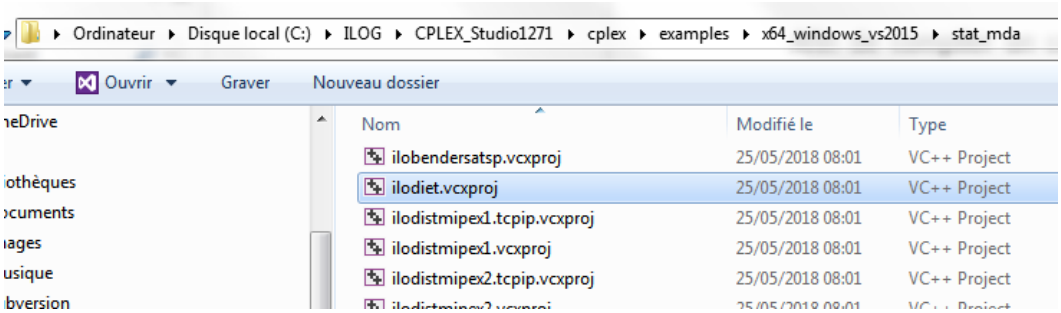

Figure 2. Choix d'un projet visual dans les exemples fournis par Cplex

Les projets qui commencent par "**`ilo`**" sont des projets C++. On ouvre par exemple **`ilodiet.vcxproj`**, et on entre dans les propriétés du projet (onglet **`projet/propriétés`**).
Dans toute la suite on suppose que le répertoire d'installation de Cplex est stocké dans la variable d'environnement **`$(CPLEX_STUDIO_DIR1271).`** En général Cplex crée cette variable à l'installation, si ce n'est pas le cas vous pouvez la créer vous-même ou la remplacer par le chemin complet (ici **`C:\ILOG\CPLEX_Studio1271\`**).

Attention : dans les propriétés du projet il faut choisir pour quelle configuration (**Release** ou **Debug**) on fait les modifications. Certaines modifications s'appliquent aux 2 configurations, certaines à une seule (ceci est précisé dans le titre de chacune des sous-sections suivantes).

2.1 Plateforme win32 ou x64 ?

Avant toute chose assurez-vous d'avoir bien créer un projet Visual Studio avec la même plateforme cible (voir Figure 3) que les projets donnés en exemple par Cplex.

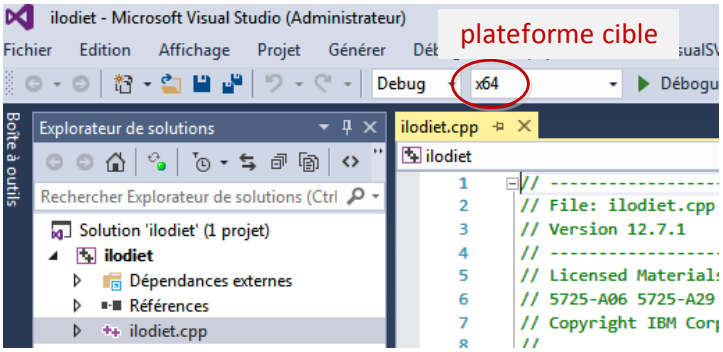

Figure 3. Choix de la plateforme cible

2.2 Répertoires include (dans les conf. **release** et **debug**)

Dans les propriétés du projet, onglet "**`C/C++ / Général`**" ajoutez les chemins vers les répertoires *include* (Figure 4) (ils donnent accès aux .h) : **`$(CPLEX_STUDIO_DIR1271)\cplex\include;`** **`$(CPLEX_STUDIO_DIR1271)\concert\include`**.

Remarque : **`$(CPLEX_STUDIO_DIR1271)\concert\include`** ne concerne que la *Concert Technology*

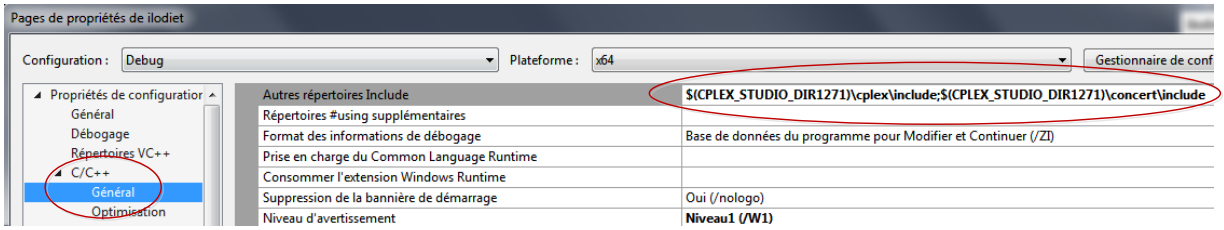

Figure 4. Ajout des répertoires include

2.3 Désactiver les vérifications SDL (conf. **release** et **debug**)

Si vous utilisez la *Concert Technology*, il faut désactiver les vérifications SDL : toujours dans l'onglet "**`C/C++ / Général`**" choisir non pour le paramètre Vérifications SDL (Figure 5).

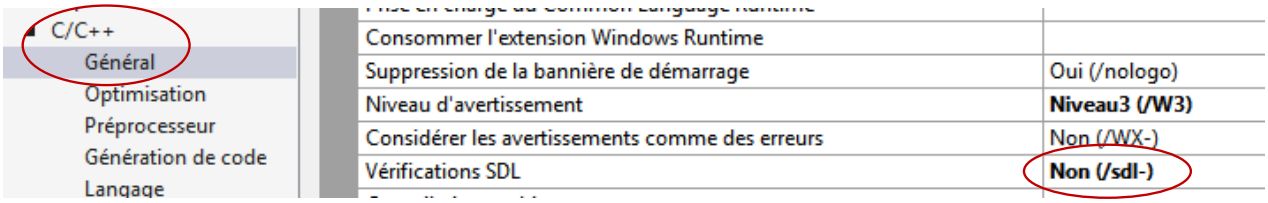

Figure 5. Positionner "vérifications SDL" sur non

2.4 Répertoires des .lib (dans les conf. **release** et **debug**)

Dans les propriétés du projet, onglet "**`Editeur de liens / entrée`**" ajouter les chemins vers les 3 .lib suivants : **`cplex<version>.lib`**, **`ilocplex.lib`** et **`concert.lib`** où *<version>* désigne le numéro de version de cplex utilisé (Figure 6).

Remarque : **`concert.lib`** ne concerne que la *Concert Technology*

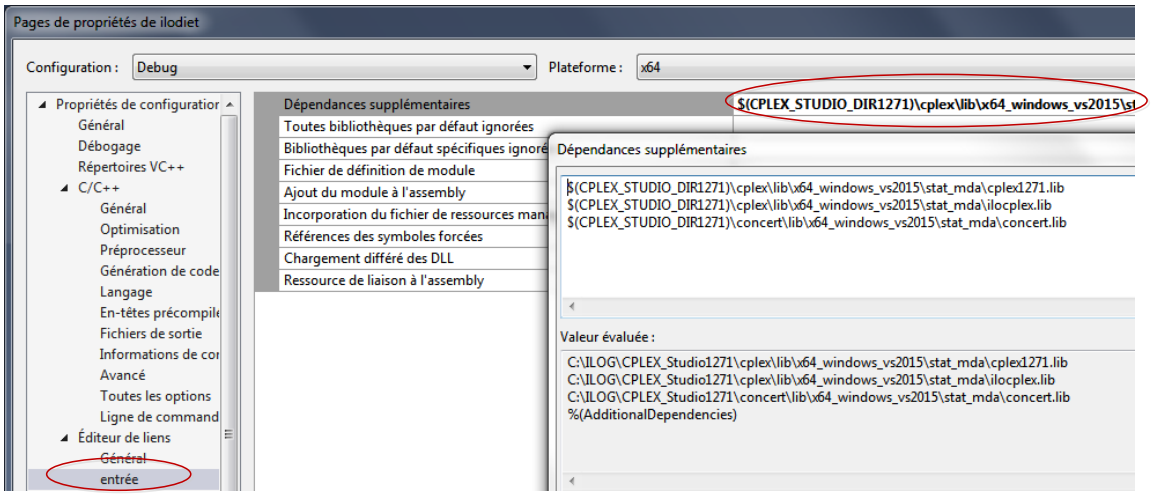

Figure 6. Ajout des répertoires contenant les .lib

2.5 Directives du préprocesseur (dans les conf. **release** et **debug**)

Dans l'onglet "**`C/C++ / Préprocesseur`**" ajouter **`IL_STD`** si vous utilisez la *Concert Technology* (Figure 7).

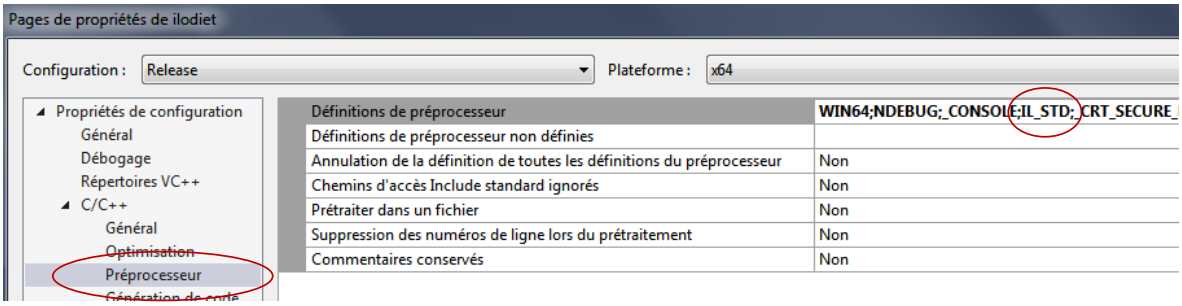

Figure 7. Ajout de directives préprocesseur

2.6 Runtime (dans les conf. release et debug)

Dans l'onglet "**`C/C++ / génération de code`**" mettre la bibliothèque runtime sur **`/MD`** (Figure 8).

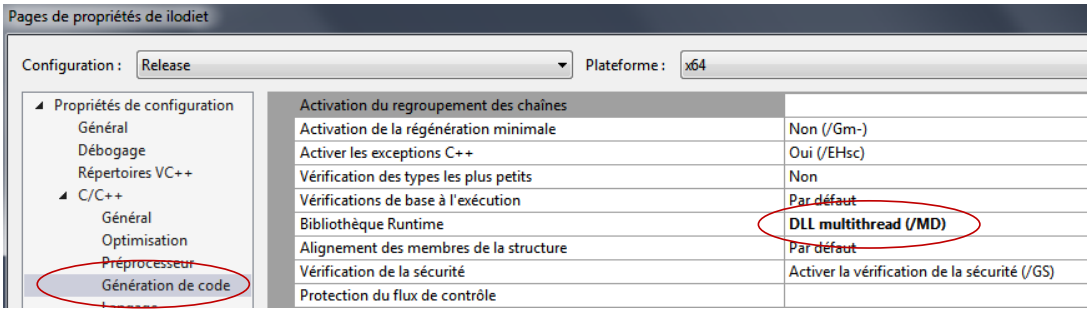

Figure 8. Choisir la "bibliothèque runtime"

2.7 Supprimer la directive préprocesseur **_DEBUG** (conf **debug**)

Dans **C/C++ / Préprocesseur** supprimer **_DEBUG** des définitions de préprocesseur (Figure 9).

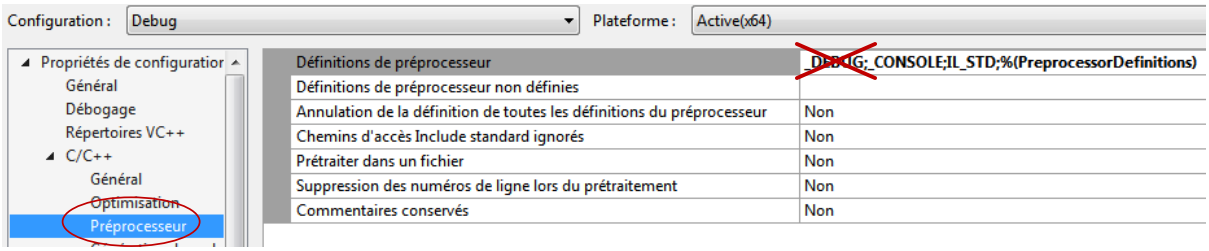

Figure 9. Supprimer _DEBUG des définitions de préprocesseur

En effet laisser **_DEBUG** provoque des erreurs : **"LNK2038: discordance détectée pour '_ITERATOR_DEBUG_LEVEL'"**.

NB. Heureusement supprimer ce flag ne vous empêchera pas de debugger, à condition bien sûr de laisser OUI dans la case "**editeur de liens / debogage / générer des informations de debogage**".

3 Références

[1] Microsoft, https://msdn.microsoft.com/fr-fr/library/ms235629.aspx