© *Jean-Baptiste APOUNG KAMGA <jean-baptiste.apoung@math.u-psud.fr>*

Fiche de TDM : Normes et styles de programmation et auto-documentation

Il est question dans cette fiche de

1. s'approprier un code et de le transformer selon une norme de programmation fournie,
2. générer une documentation automatique du code à l'aide de l'outil *doxygen*

Thème - 1 *Normes de programmation*

Exercice-1 **:** Récupération du Projet

Q-1-1 **:** Récupérer le projet **miraMeshSplitter.tar.gz** et décompresser le.

```
tar -zxvf miraMeshSplitter.tar.gz
```

Q-1-2 **:** Décompresser le fichier dans votre répertoire de travail. Compiler le.

```
cd miraMeshSplitter
./configure
make
```

Q-1-3 **:** Exécuter le code et commenter. (*Vous aurez besoin du fichier* **sample_mesh.msh**)

```
./mesh2dT
```

Assurrez-vous d'obtenir la sortie suivante :

```
./mira2dT [OPTIONS]...[FILES]....
Options are as follows type them to see the entries requirements :
--partition
--split
--relabel
 --bcbyvertex
 --pressuresplit
 --prolongate
--reduce
 --tosendbc
 --toreceivbc
 --localbcprolong
 --merge
--setconnect
Options in white are not yet available

 Please sent comments to:  jean-baptiste.apoung@math.u-psud.fr
```

Tester alors certaines des options proposées.

Exercice-2 : Transformation de code

Q-2-1 : Identification des incohérences

En vous basant des notions vues en cours,
- Vérifiez si dans son développement, les auteurs on respecté une certaine norme de programmation
- Les exigences de nomenclatures sont-elles respectées ?
- La structuration des fichiers entêtes et sources est-elle favorable à la maintenance, à l'utilisation ?

Q-2-2 : Transformation

Il est question de transformer le code de sorte à respecter une norme de programmation *Cette transformation pouvant nécessiter le changement de dénomination des fichiers, il est vivement conseillé de commencer par porter le code dans un environnement de développement intégré. Dans cette direction, vous avez le choix entre les outils suivants : anjuta* `http://www.anjuta.org/`*, geany* `http://www.geany.org/`*, kdevelop* `http://kdevelop.org/`*, qtcreator* `http://qt.nokia.com/products/developer-tools/`

En vous aidant des normes de programmations en **C++** vues en cours, transformez le code de sorte à respecter une norme de programmation.

Vous pourrez aussi consulter les liens et références suivants :
- `http://www.abxsoft.com/ccs/ccs-std.html`
- `http://www.possibility.com/Cpp/CppCodingStandard.html`
- `http://www.doc.ic.ac.uk/lab/cplus/c++.rules/`

Thème - 2 *Documentation*

Exercice-1 : Prise en main de Doxygen

1. Placez vous dans le répertoire du code

```
cd miraMeshSplitter
```

2. Générer un fichier de configuration doxygen

```
doxygen  -g
```

3. Ouvrez le fichier générer **Doxifile** et modifier le comme indiquer en cours de sorte à générer la documentation du code uniquement aux format .htm et .tex dans les répertoires respectifs doc/html et doc/latex. *La modification devra autoriser la génération des entités non documentées, car par défaut, doxygen ne génère pas la documentation des entités non documentées.*
4. Générer la documentation

```
doxygen  Doxifile
```

5. Générer la documentation .pdf

```
cd doc/latex
pdflatex refmanual
```

Exercice-2 : Applications

1. Modifier à nouveau le fichier **Doxifile** de sorte à ne générer de documentation que pour des objets documentés.
2. Documenter à présent le code **miraMeshSplitter** et générer sa documentation aux format .html et .pdf (*La documentation consistera en la documentation des fichiers, fonctions, classes etc. Il est vivement conseillé de recourir à la programmation parpseudocode (PPP)*)

   Vous pourrez aussi consulter les liens et références suivants :
   – `http://www.stack.nl/~dimitri/doxygen/`