```
/*****************************************************************************
 *  Logiciel portageMM2CPP de portage de code ModelMaker en langage C++      *
 *  Copyright INRA, février 2006                                             *
 *****************************************************************************/

/*****************************************************************************
 *
 * Fichier     : help_modeDemploi
 *
 * Auteur(s)   : Nathalie Rousse, Nathalie.Rousse@toulouse.inra.fr
 *               de l'INRA - Institut National de la Recherche Agronomique -
 *               ( Département MIA, UMR AGIR, RMT modelia ).
 *
 * Description : Fichier faisant partie de la documentation du logiciel
 * portageMM2CPP. L'ensemble des fichiers de documentation est récapitulé
 * dans le fichier help.
 *
 *****************************************************************************
 * Historique :
 *
 * 23/11/07, Nathalie Rousse : création du fichier.
 *
 * Le fichier est créé par déplacement de texte qui se trouvait jusque là
 * dans le fichier help.
 *
 *****************************************************************************/
/*


Rappel :
Pour le guide d'utilisation global, voir le fichier help_guideDutilisation.


****** MODE D'EMPLOI ************************************************************

     ******************************************************************
     *                                                                *
     *               Comment, à partir de ce logiciel,                *
     *        construire son propre logiciel en C++ (.cpp, .h)        *
     *     relatif à son propre modèle ModelMaker (fichier ".mod").   *
     *                                                                *
     ******************************************************************

Les modifications qu'il faut apporter au code C++ sont notées directement dans le code (*.cpp et *.h) : ce sont les com
mentaires marqués du label "modeDemploi" (b). De plus on trouve dans les représentations UML une aide à la mise en oeuv
re des modifications : ce sont les commentaires sur fond jaune dans le "Diagramme des classes spécifiques" et le complé
ment de ce diagramme dans « UMLmaj_portageMM2CPP.ppt » (a).
```

Le déroulement de la procédure à suivre pour construire son propre logiciel C++ est décrit ci-dessous :

1) Récupérer en l'état le logiciel exemple "portageMM2CPP" :

Copier/coller dans son propre espace de travail tout l'espace de travail (répertoires sources, tests, exécution ...) :

- leSourceModelMaker
- leSourceCPP
- laConfigCPP
- executionCPP
- lesEntreesCPP
- lesSortiesCPP
- lesTestsCPP
- laDocumentation
- LICENCE

2) Modifier le contenu du répertoire leSourceModelMaker

Remplacer modele2.mod de l'exemple par son propre fichier ModelMaker, de même pour modele2.txt (voir explications dans "readme" de "leSourceModelMaker").

Remarque : Le fichier "modele2.txt" pourra par ailleurs servir pour la vérification de cohérence entre versions logicielles. En effet, une fois que le modèle aura été réécrit en C++, il existera deux formes logicielles du modèle (formes ModelMaker et C++) susceptibles d'évoluer en parallèle. Pour repérer les différences/changements entre une version "vCPP" du logiciel C++ (dont le .txt associé est modele2.txt.vCPP) et une version "vMM" du modèle ModelMaker modele2.mod (dont le .txt est modele2.txt.vMM), on pourra comparer leur .txt : commande "diff modele2.txt.vCPP modele2.txt.vMM".

3) Modifier le contenu des répertoires leSourceCPP et laConfigCPP

3.1) la partie générique (genericitePortageMM2CPP.cpp, genericitePortageMM2CPP.h et genericiteConfigPortageMM2CPP.h) :

A priori il n'est pas besoin de retoucher à cette partie.

Balayer toutefois les instructions marquées du label "modeDemploi" jalonnant le code (voir "(b)").

En effet il peut être nécessaire de faire évoluer ce code pour différentes raisons :
- besoin d'enrichir/modifier cette partie si le nouveau modèle, plus élaboré que celui de l'exemple, utilise des possibilités de ModelMaker qui ne sont pas prises en compte actuellement (voir les commentaires marqués du label "modeDemploiEvolutions" dans le code).
- besoin/choix d'implémenter une(de) nouvelle(s) fonction(s) de simulation "simuler" par rapport à l'utilisation prévue du logiciel C++ (voir description dans les représentations UML : "Diagramme des classes génériques" et "Diagramme d'activité de méthodes" et le complément de ces diagrammes dans « UMLmaj_portageMM2CPP.ppt », voir les commentaires marqués du label "modeDemploiEvolutions" dans le code).
- dimensionnements : voir "3.4)"

3.2) la partie spécifique (specificitePortageMM2CPP.cpp, specificitePortageMM2CPP.h et specificiteConfigPortageMM2CPP.h) :

Reécrire le code de ces fichiers : la déclaration des données, le contenu des fonctions. Pour cela suivre les instructions marquées du label "modeDemploi" jalonnant le code (voir "(b)") et s'aider des explications et illustrations de "(a)".

Par ailleurs, cette partie est susceptible de bouger au titre d'évolutions. Par exemple, quelqu'un peut être conduit à reprendre les fonctions qui restituent les résultats sortis des simulations ("afficherEntetesEntites" et "afficherEntites") s'il a besoin de sortir des résultats à d'autres formats... (voir les commentaires marqués du label "modeDemploiEvolutions" dans le code).

3.3) le main (mainPortageMM2CPP.cpp) :

Reprendre le code pour appeler la simulation comme voulu (paramètres d'appel de "simuler" ...). Pour cela suivre les instructions marquées du label "modeDemploi" jalonnant le code (voir "(b)") et s'aider des explications et illustrations de "(a)".

3.4) Configuration/dimensionnement (les fichiers particulièrement dédiés à la configuration sont genericiteConfigPortageMM2CPP.h et specificiteConfigPortageMM2CPP.h) :

Dans tous les fichiers, vérifier les configurations/dimensionnements actuellement codés/définis. Ces éléments sont repérables grâce au label "modeDemploi_ConfDim" dont ils sont marqués. Si besoin, les modifier : changer des constantes/valeurs par défaut, augmenter des tailles de tableaux...

3.5) label "modeDemploi"

Cette étape "3)" n'est pas achevée tant qu'on ne s'est pas assuré, dans TOUT LE CODE de TOUS LES FICHIERS, d'avoir consulté et suivi les consignes/indications marquées du label "modeDemploi" (voir "(b)").

4) Essai et validation.

Mettre en place et effectuer les tests permettant de vérifier son propre logiciel : dans son propre fichier ModelMaker (".mod") et dans le répertoire "lesTestsCPP" ; voir "Tests du logiciel" dans le fichier help.

5) Utilisation du logiciel

Une fois le logiciel validé, l'intégrer dans son propre contexte. Il peut s'agir de remplacer le main par une IHM plus élaborée, ou encore d'invoquer la fonction de simulation "simuler" dans un autre programme/logiciel C++ (d'un autre modèle, d'une application logicielle ...) etc.

****

```
*/
```