

```

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

```

```

/*****
 * Licence :

```

Le logiciel portageMM2CPP est une réécriture en langage C++ du modèle écrit dans le fichier ModelMaker "modele2.mod" (situé dans le répertoire "leSourceModelMaker"). Voir informations dans ../laDocumentation/help.

Ce logiciel est régi par la licence CeCILL soumise au droit français et respectant les principes de diffusion des logiciels libres. Vous pouvez utiliser, modifier et/ou redistribuer ce programme sous les conditions de la licence CeCILL telle que diffusée par le CEA, le CNRS et l'INRIA sur le site "<http://www.cecill.info>".

En contrepartie de l'accessibilité au code source et des droits de copie, de modification et de redistribution accordés par cette licence, il n'est offert aux utilisateurs qu'une garantie limitée. Pour les mêmes raisons, seule une responsabilité restreinte pèse sur l'auteur du programme, le titulaire des droits patrimoniaux et les concédants successifs.

A cet égard l'attention de l'utilisateur est attirée sur les risques associés au chargement, à l'utilisation, à la modification et/ou au développement et à la reproduction du logiciel par l'utilisateur étant donné sa spécificité de logiciel libre, qui peut le rendre complexe à manipuler et qui le réserve donc à des développeurs et des professionnels avertis possédant des connaissances informatiques approfondies. Les utilisateurs sont donc invités à charger et tester l'adéquation du logiciel à leurs besoins dans des conditions permettant d'assurer la sécurité de leurs systèmes et ou de leurs données et, plus généralement, à l'utiliser et l'exploiter dans les mêmes conditions de sécurité.

Le fait que vous puissiez accéder à cet en-tête signifie que vous avez pris connaissance de la licence CeCILL, et que vous en avez accepté les termes.

Les fichiers Licence\_CeCILL\_V2-fr.txt et Licence\_CeCILL\_V2-en.txt du répertoire ../LICENCE fournissent le texte de la licence, dans sa version 2, en langue française et en langue anglaise. (ces 2 fichiers sont aussi directement dans le répertoire courant)

```

*****/
/*****
 *

```

```
* Fichier      : specificitePortageMM2CPP.cpp
*
* Auteur(s)    : Nathalie Rousse, Nathalie.Rousse@toulouse.inra.fr
*               de l'INRA - Institut National de la Recherche Agronomique -
*               ( UMR ARCHE / Plate-forme INRA-ACTA-ICTA ).
*
* Description  :
*
* Ce fichier fait partie du code source C++ du logiciel portageMM2CPP.
*
* Il en contient la partie complètement spécifique du modèle traité
* (corps des classes). Voir ../laDocumentation/help, en particulier
* "Présentation générale" et "Spécification".
*
*****
* Historique  :
*
* 03/02/06, Nathalie Rousse : création du fichier.
*
*****/

/* modeDemploi
 * Instruction s'adressant à quelqu'un qui est en train de construire son propre logiciel C++ relativement à son propre
 * modèle ModelMaker : dans tout le code de ce fichier, consulter et suivre les consignes/indications marquées du label "
 * modeDemploi". Voir aussi "Mode d'emploi" dans le fichier "help".
 */

#include <stdio.h>
#include <iostream>
using namespace std;

#include <math.h>

#include "genericitePortageMM2CPP.h"
#include "specificitePortageMM2CPP.h"

/*****
 *
 * corps de la classe spécifiant le système particulier
 *
 *****/

/*
 *
```

```
* Constructeurs, initialisation de l'ensemble du système
*
*/

/*
* Initialisation de l'ensemble du système avec horloge locale par défaut
*/
systemParticulier::systemParticulier( void ){

    // initialisation avec horloge locale par défaut
    genericSystem::initialiser();

    initialiserLesEntites();
}

/*
* Initialisation de l'ensemble du système
* avec configuration de l'horloge locale
*/
systemParticulier::systemParticulier( float it, float deltatValue ){

    /* initialisation avec configuration de l'horloge locale :
    * instant initial et pas de temps (plus petit increment) */
    genericSystem::initialiser( it, deltatValue );

    initialiserLesEntites();
}

/*
* Initialisation des entités du système
*/

/* modeDemploiEvolutions
* Actuellement les valeurs d'initialisation sont codées/définies "en dur" dans initialiserLesEntites. Une évolution se
rait de rendre variables les valeurs d'initialisation afin d'en permettre le choix au niveau du "main.cpp" (ou équiva
nt).
*/

/* PL, AL : 2 macros définies pour rendre "le plus transparent possible" la
* gestion du calcul des valeurs en fonction des dérivées */
#define PL(entity) { int cr=ead.PL(&entity); if (cr!=CR_OK) crInit.putValue(cr); }
#define AL() { ead.AL( time.getPasDeTemps() ); }

void systemParticulier::initialiserLesEntites( void ){

// modeDemploiDebut
```

```

/*
 * Appel de "PL" pour chaque entité définie (ie pour chaque entité
 * ModelMaker déclarée comme donnée de la classe systemParticulier).
 * Ces appels de "PL" sont indispensables pour que le calcul des
 * valeurs en fonction des dérivées se fasse ensuite de manière
 * "transparente" (cf "AL" dans actualiserIncrement).
 */
PL( C3max );
PL( P1 ); PL( P2 ); PL( P3 ); PL( P4 ); PL( P5 );
PL( V1 );
PL( F1 ); PL( F2 ); PL( F3 ); PL( F4 );
PL( C1_source ); PL( C2_reservoir ); PL( C3_reservoir ); PL( C4_puits ); PL( C5_debordement );

/* modeDemploi
 *
 * remplacer le code situé entre ici et "modeDemploiDebut" ci-dessus par le code adapté à son système particuli
er (à ses propres entités ModelMaker déclarées comme données de la classe systemParticulier). (Ce n'est pas recommandé,
mais quelqu'un qui choisirait d'écrire lui-même le calcul des valeurs en fonction des dérivées n'aurait pas besoin d'a
ppeler "PL" ni "AL".) */

// modeDemploiDebut

/* constantes */
C3max.initialiser( 0.2 );

/* paramètres */
P1.initialiser( 0.08 );
P2.initialiser( 0.06 );
P3.initialiser( 0.04 );
P4.initialiser( 50.0 );
P5.initialiser( 20.0 );

/* variables : pas d'initialisation (nuls par défaut) */

/* flux : pas d'initialisation (nuls par défaut) */

/* compartiments */
C1_source.initialiser(1.0);
C2_reservoir.initialiser(0.0);
C3_reservoir.initialiser(0.0);
C4_puits.initialiser(0.0);
C5_debordement.initialiser(0.0);

/* modeDemploi
 *
 * remplacer le code situé entre ici et "modeDemploiDebut" ci-dessus par le code adapté à son système particuli
er : se conformer au MAIN extrait du fichier ModelMaker (les valeurs initiales), s'aider des commentaires sur fond jaun

```

```
e des représentations UML. */
}

/*
 *
 * Fait évoluer (incrémente) le système (les entités ModelMaker) en
 * fonction du pas de temps horloge "deltat"
 *
 */
int systemParticulier::actualiserIncrement( void ){

// modeDemploiDebut

/*
 * Ecriture de toutes les relations entre toutes les entités (ie
 * les entités ModelMaker déclarées comme données de la classe
 * systemParticulier). L'ordre d'écriture n'est pas quelconque.
 */

/* constantes : pas de calcul */

/* paramètres : pas de calcul */

/* variables */
V1.value = P2.value * ( 1.0 + (float)sin((double)(time.t/5.0)) );

/* flux */

// Flow from C1_source to C2_reservoir
F1.value = ( P1.value * time.t / (time.t + P4.value) ) * C1_source.value;

// Flow from C2_reservoir to C3_reservoir
F2.value = V1.value * C2_reservoir.value;

// Flow from C3_reservoir to C4_puits
F3.value = P3.value * C3_reservoir.value;

// Flow from C3_reservoir to C5_debordement
if ( C3_reservoir.value >= C3max.value ){
    F4.value = P5.value * ( C3_reservoir.value - C3max.value );
} else {
    F4.value = 0.0;
}
}
```

```
/* compartiments */

C1_source.ddt = -F1.value;
C2_reservoir.ddt = +F1.value-F2.value;
C3_reservoir.ddt = +F2.value-F3.value-F4.value;
C4_puits.ddt = +F3.value;
C5_debordement.ddt = +F4.value;

/* modeDemploi
*
* remplacer le code situé entre ici et "modeDemploiDebut" ci-dessus par le code adapté à son système particuli
er : se conformer au MAIN extrait du fichier ModelMaker (les équations, relations), s'aider des commentaires sur fond j
aune des représentations UML.
* Avertissement : veiller à l'ordre dans lequel sont écrites les relations, vérifier que cet ordre est fidèle
au modèle d'origine (ie dans sa forme ModelMaker).
*/

/* modeDemploi
*
* ATTENTION : NE PAS ENLEVER L'APPEL DE "AL", qui effectue de manière "transparente" le calcul des valeurs en
fonction des dérivées.
* (Ce n'est pas recommandé, mais quelqu'un qui choisirait d'écrire lui-même le calcul des valeurs en fonction
des dérivées n'aurait pas besoin d'appeler "AL", ni "PL" avant. En remplacement, il devrait ici écrire lui-même le calc
ul, pour chaque donnée de type entityAvecDerivee (ou ses filles), selon l'exemple suivant avec la donnée C3_reservoir :
* C3_reservoir.value = C3_reservoir.value + C3_reservoir.ddt * time.getPasDeTemps();
*/
AL();

return CR_OK;
}

/*
*
* Affichages, restitution des résultats sortis des simulations
*
*/

/*
* Description de la forme actuelle de restitution des résultats :
*
* 1) Il est fait des affichages à l'écran.
*
* Les affichages écran sont plus ou moins détaillés (voir commentaires
* de CLE_AVEC_TRACE_ECRAN_DONNEES plus bas dans le code).
*
* 2) Il est aussi sorti des résultats dans des fichiers résultats :
```

```

*
*   Il est produit des fichiers de résultats : un fichier pour l'instant
*   courant "t" de l'horloge du système, et un fichier pour chaque entité
*   du système (ie pour chaque entité ModelMaker déclarée comme donnée de
*   la classe systemParticulier). Actuellement, les noms de ces fichiers sont
*   codés/définis "en dur" dans afficherEntetesEntites et afficherEntites.
*   Chaque fichier contient la liste des valeurs successives au cours du
*   temps de la donnée concernée ("value") : affichage au format float, une
*   donnée par ligne.
*/

/* modeDemploiEvolutions
* La forme de restitution actuellement implémentée est assez "basique" : le
* format est volontairement simpliste/peu contraignant/peu particulier,
* dans l'idée de servir de base pour d'éventuelles évolutions. L'intention
* est de faciliter le travail de quelqu'un qui viendrait reprendre
* afficherEntetesEntites et afficherEntites, par exemple s'il a besoin
* de sortir des résultats à d'autres formats ...
*/

/*
* "preparerFichier", "ajouterDonnee" : utilisées avec la forme actuelle de
* restitution des résultats (écran et fichiers)
*/
// creer le fichier nomFichier si non existant et sinon le vider
#define preparerFichier( nomFichier ){ fopen((const char *)nomFichier,(const char *)"w"); }
// Ajouter dans le fichier nomFichier la donnee value au format float
#define ajouterDonnee( nomFichier, value ){ FILE *file=fopen((const char *)nomFichier,(const char *)"a"); fprintf( file
, "%f\n", value ); }

int systemParticulier::afficherEntetesEntites(void){ // doit être "conforme" à afficherEntites

// modeDemploiDebut

/*
* Les résultats de simulation affichés à l'écran sont plus ou moins détaillés
* en fonction de la valeur de la clé de compilation
* CLE_AVEC_TRACE_ECRAN_DONNEES (configurée dans la commande de compilation,
* voir "compiler").
* Si la clé de compilation CLE_AVEC_TRACE_ECRAN_DONNEES est activée, alors
* les valeurs au cours du temps des données de simulation sont affichées à
* l'écran (en plus de l'affichage des conditions de simulation).
*/
#ifdef CLE_AVEC_TRACE_ECRAN_DONNEES

    printf( " ***** les entités du système : \n" );
    printf( "C3max= %f, " , C3max.value );
    printf( "P1= %f, " , P1.value );

```

```
printf( "P2= %f, ", P2.value );
printf( "P3= %f, ", P3.value );
printf( "P4= %f, ", P4.value );
printf( "P5= %f. ", P5.value );
printf( "\n\n" );

printf( " ***** les resultats sortis : \n" );

printf( "t      " );

printf( "V1      " );

printf( "F1      " );
printf( "F2      " );
printf( "F3      " );
printf( "F4      " );

printf( "C1_source      " );
printf( "C2_reservoir    " );
printf( "C3_reservoir    " );
printf( "C4_puits        " );
printf( "C5_debordement      " );

printf( "\n\n" );

#endif /* CLE_AVEC_TRACE_ECRAN_DONNEES */

// les "resultats sortis sont enregistres dans des fichiers
preparerFichier( "../lesSortiesCPP/t.res" );
preparerFichier( "../lesSortiesCPP/V1.res" );
preparerFichier( "../lesSortiesCPP/F1.res" );
preparerFichier( "../lesSortiesCPP/F2.res" );
preparerFichier( "../lesSortiesCPP/F3.res" );
preparerFichier( "../lesSortiesCPP/F4.res" );
preparerFichier( "../lesSortiesCPP/C1_source.res" );
preparerFichier( "../lesSortiesCPP/C2_reservoir.res" );
preparerFichier( "../lesSortiesCPP/C3_reservoir.res" );
preparerFichier( "../lesSortiesCPP/C4_puits.res" );
preparerFichier( "../lesSortiesCPP/C5_debordement.res" );

/* modeDemploi
 *
 * remplacer le code situé entre ici et "modeDemploiDebut" ci-dessus par le code adapté à son système particuli
er, aux sorties souhaitées. Veiller à respecter la cohérence avec la fonction afficherEntites (ordre des affichages ...
).
 */
```



```
        return CR_OK;
    }

int systemParticulier::afficherEntites(void){ // doit être "conforme" à afficherEntetesEntites

// modeDemploiDebut

/*
 * Les résultats de simulation affichés à l'écran sont plus ou moins détaillés
 * en fonction de la valeur de la clé de compilation
 * CLE_AVEC_TRACE_ECRAN_DONNEES (voir détails dans afficherEntetesEntites)
 */
#ifdef CLE_AVEC_TRACE_ECRAN_DONNEES

    // trace
    printf( "%f      ", time.t );

    printf( "%f      ", V1.value );

    printf( "%f      ", F1.value );
    printf( "%f      ", F2.value );
    printf( "%f      ", F3.value );
    printf( "%f      ", F4.value );

    printf( "%f      ", C1_source.value );
    printf( "%f      ", C2_reservoir.value );
    printf( "%f      ", C3_reservoir.value );
    printf( "%f      ", C4_puits.value );
    printf( "%f      ", C5_debordement.value );

    printf( "\n" );

#endif /* CLE_AVEC_TRACE_ECRAN_DONNEES */

    // les "resultats sortis sont enregistres dans des fichiers
    ajouterDonnee( "../lesSortiesCPP/t.res", time.t );
    ajouterDonnee( "../lesSortiesCPP/V1.res", V1.value );
    ajouterDonnee( "../lesSortiesCPP/F1.res", F1.value );
    ajouterDonnee( "../lesSortiesCPP/F2.res", F2.value );
    ajouterDonnee( "../lesSortiesCPP/F3.res", F3.value );
    ajouterDonnee( "../lesSortiesCPP/F4.res", F4.value );
    ajouterDonnee( "../lesSortiesCPP/C1_source.res", C1_source.value );
    ajouterDonnee( "../lesSortiesCPP/C2_reservoir.res", C2_reservoir.value );
    ajouterDonnee( "../lesSortiesCPP/C3_reservoir.res", C3_reservoir.value );
    ajouterDonnee( "../lesSortiesCPP/C4_puits.res", C4_puits.value );
    ajouterDonnee( "../lesSortiesCPP/C5_debordement.res", C5_debordement.value );
```

```
/* modeDemploi
   *
   * remplacer le code situé entre ici et "modeDemploiDebut" ci-dessus par le code adapté à son système particuli
er, aux sorties souhaitées. Veiller à respecter la cohérence avec la fonction afficherEntetesEntites (ordre des afficha
ges ...).
   */

    return CR_OK;
}

/*
 * PHOTO DU MAIN de ModelMaker : voir "specificitePortageMM2CPP.h"
 */
```