```
/*************************************************************************
 *  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 -
 *               (département MIA, UMR AGIR, http://www.modelia.org).
 *
 * 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.
 * 24/10/07, Nathalie Rousse : correction (appel de fclose suite a fopen).
 * 26/10/07, Nathalie Rousse : modification m_type (definition de types : typedef typeValeur et typeTemps, qui "passent
" de float a au moins double).
 *
 * 02/01/08, Nathalie Rousse : modification m_linearInterpolation (ajout prise en compte des entités ModelMaker qui son
t calculées par interpolation linéaire à partir de données lues dans un fichier de données).
 *
 ********************************************************************************/


/* 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>
#include <fstream>
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( typeTemps it, typeTemps 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 équivale
nt).
*/
```

```
/* m_linearInterpolation : dans PL et AL, ajout de la gestion de eil */
/*
 * PL et AL (plus exactement ALenPremier et ALenDernier) sont
 * 2 macros définies pour rendre "le plus transparent possible" :
 * - la gestion du calcul des valeurs en fonction des dérivées,
 * - la gestion du calcul des valeurs par interpolation de données lues.
 */
#define PL(entity) { int crEad=ead.PL(&entity); if (crEad!=CR_OK) crInit.putValue(crEad); \
                     int crEil=eil.PL(&entity); if (crEil!=CR_OK) crInit.putValue(crEil); }
#define ALenPremier() { cr=eil.AL( time.getInstantCourant() ); }
#define ALenDernier() { ead.AL( time.getPasDeTemps() ); }

/* m_linearInterpolation : dans initialiserLesEntites, ajout de la gestion des entités controlled */
void systemParticulier::initialiserLesEntites( void ){

// 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 );

        /* controlled (in et interpolated) */
        I1_pluie.initialiser( nomFichierIn_DONNEE_I1_pluie,
           nbColonnes_DONNEE_I1_pluie, colInstant_DONNEE_I1_pluie, colValeur_DONNEE_I1_pluie );
        I20.initialiser( nomFichierIn_DONNEE_I20,
           nbColonnes_DONNEE_I20, colInstant_DONNEE_I20, colValeur_DONNEE_I20 );
        I25.initialiser( nomFichierIn_DONNEE_I25,
           nbColonnes_DONNEE_I25, colInstant_DONNEE_I25, colValeur_DONNEE_I25 );


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

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

        /* compartiments */
        C_constante.initialiser(7.0);
        C_monte.initialiser(0.0);
        C_descend.initialiser(0.0);
        C_sinusoidale.initialiser(0.0);
        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. */


// 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 et le calcul des valeurs interpolées
         * se fassent ensuite de manière "transparente" (cf "AL" dans
         * actualiserIncrement).
         */
        PL( C3max );
        PL( P1 ); PL( P2 ); PL( P3 ); PL( P4 ); PL( P5 );
        PL( I1_pluie ); PL( I20 ); PL( I25 );
        PL( V1 );
        PL( Fpluie1 ); PL( Fpluie20 ); PL( Fpluie25 );
        PL( F1 ); PL( F2 ); PL( F3 ); PL( F4 );
        PL( C_constante ); PL( C_monte ); PL( C_descend ); PL( C_sinusoidale );
        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".) */

}


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

```
        int cr = CR_OK; // par défaut

/* modeDemploi
         *
         * ATTENTION : NE PAS ENLEVER L'APPEL DE "AL" (plus exactement de "ALenPremier" ni "ALenDernier") : voir commen
taire plus bas à l'appel de ALenDernier.
         */
        ALenPremier(); // notamment maj cr
        if ( cr == CR_OK ){

// 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 */

        /* controlled : le calcul est complètement automatisé (fait par AL) */

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

        /* flux */


        // Flow from I1_pluie to ...
        Fpluie1.value = (typeValeur)( - I1_pluie.value / 10.0 );

        // Flow from I20 to ...
        Fpluie20.value = (typeValeur)( - I20.value / 10.0 );

        // Flow from I25 to ...
        Fpluie25.value = (typeValeur)( - I25.value / 10.0 );


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

        // Flow from C2_reservoir to C3_reservoir
        F2.value = (typeValeur)( V1.value * C2_reservoir.value );
```

```
        // Flow from C3_reservoir to C4_puits
        F3.value = (typeValeur)( P3.value * C3_reservoir.value );

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

        /* compartiments */

#define PI 3.14159265 // constante pour les calculs qui suivent
long double PIsur20 = PI / 20.0; // constante pour les calculs qui suivent
long double deuxPI = PI * 2.0; // constante pour les calculs qui suivent

        C_constante.ddt = (typeValeur)( 50.0 );
        C_monte.ddt = (typeValeur)( 200.0 + 30.0 * time.t ); // droite passant par les points (t=40,1400) et (t=160,500
0)
        C_descend.ddt = (typeValeur)( 6200.0 - 30.0 * time.t ); // droite passant par les points (t=40,5000) et (t=160,
1400)
        C_sinusoidale.ddt = (typeValeur)( sin( ( time.t * PIsur20 ) - deuxPI ) ); // sin qui vaut 0 pour t=40s, t=80s,
t=120s et t=160s

        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" (plus exactement de "ALenPremier" ni "ALenDernier"), qui effectue
 de manière "transparente" le calcul des valeurs en fonction des dérivées et le calcul des valeurs interpolées.
         * (Ce n'est pas recommandé, mais quelqu'un qui choisirait d'écrire lui-même le calcul des valeurs en fonction
des dérivées ET le calcul des valeurs interpolées n'aurait pas besoin d'appeler "AL", ni "PL" avant. En remplacement, i
l devrait ici écrire lui-même le calcul :
```

```
         * - pour chaque donnée de type entityAvecDerivee (ou ses filles), selon l'exemple suivant avec la donnée C3_re
servoir : C3_reservoir.value = C3_reservoir.value + C3_reservoir.ddt * time.getPasDeTemps();
         * - pour chaque donnée de type entityInterpolee (ou ses filles) ; ...attention dans ce cas à bien gérer ce qui
 est actuellement fait dans PL !
         */
        ALenDernier();

        } // fin de if ( cr == CR_OK )

        return cr;
}


/*
 *
 * 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)
 */

// Créer 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 donnée value au format float
// 24/10/07 correction
#define ajouterDonnee( nomFichier, value ){ FILE *file=fopen((const char *)nomFichier,(const char *)"a"); fprintf( file
, "%f\n", (float)value ); fclose(file); }

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

// 26/10/07 m_type : cast float
        printf( " ***** les entités du système : \n" );
        printf( "C3max= %f, " , (float)C3max.value );
        printf( "P1= %f, ", (float)P1.value );
        printf( "P2= %f, ", (float)P2.value );
        printf( "P3= %f, ", (float)P3.value );
        printf( "P4= %f, ", (float)P4.value );
        printf( "P5= %f. ", (float)P5.value );
        printf( "\n" );
        printf( "I1_pluie entité interpolée selon fichier %s, \n", I1_pluie.getNomFichierDonneesAinterpoler() );
        printf( "I20       entité interpolée selon fichier %s, \n", I20.getNomFichierDonneesAinterpoler() );
        printf( "I25       entité interpolée selon fichier %s.", I25.getNomFichierDonneesAinterpoler() );
        printf( "\n\n" );

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

        // affichage à l'écran de l'information jour
        printf( "(t/86400)\n" );
```

```
        printf( "jour   " );

        printf( "t      " );

        printf( "I1_pluie       " );
        printf( "I20    " );
        printf( "I25    " );

        printf( "V1     " );

        printf( "Fpluie1        " );
        printf( "Fpluie20       " );
        printf( "Fpluie25       " );
        printf( "F1     " );
        printf( "F2     " );
        printf( "F3     " );
        printf( "F4     " );

        printf( "C_constante    " );
        printf( "C_monte        " );
        printf( "C_descend      " );
        printf( "C_sinusoidale  " );
        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 résultats sortis sont enregistrés dans des fichiers
        preparerFichier( nomFichierOut_t );
        preparerFichier( nomFichierOut_I1_pluie );
        preparerFichier( nomFichierOut_I20 );
        preparerFichier( nomFichierOut_I25 );
        preparerFichier( nomFichierOut_V1 );
        preparerFichier( nomFichierOut_Fpluie1 );
        preparerFichier( nomFichierOut_Fpluie20 );
        preparerFichier( nomFichierOut_Fpluie25 );
        preparerFichier( nomFichierOut_F1 );
        preparerFichier( nomFichierOut_F2 );
        preparerFichier( nomFichierOut_F3 );
        preparerFichier( nomFichierOut_F4 );
        preparerFichier( nomFichierOut_C_constante );
        preparerFichier( nomFichierOut_C_monte );
```

```
        preparerFichier( nomFichierOut_C_descend );
        preparerFichier( nomFichierOut_C_sinusoidale );
        preparerFichier( nomFichierOut_C1_source );
        preparerFichier( nomFichierOut_C2_reservoir );
        preparerFichier( nomFichierOut_C3_reservoir );
        preparerFichier( nomFichierOut_C4_puits );
        preparerFichier( nomFichierOut_C5_debordement );

/* 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
// 26/10/07 m_type : cast float

        // affichage à l'écran de l'information jour
        printf( "%f      ", (float)(time.t/86400.0) );

        printf( "%f      ", (float)time.t );

        printf( "%f      ", (float)I1_pluie.value );
        printf( "%f      ", (float)I20.value );
        printf( "%f      ", (float)I25.value );

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

        printf( "%f      ", (float)Fpluie1.value );
        printf( "%f      ", (float)Fpluie20.value );
        printf( "%f      ", (float)Fpluie25.value );
```

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

        printf( "%f      ", (float)C_constante.value );
        printf( "%f      ", (float)C_monte.value );
        printf( "%f      ", (float)C_descend.value );
        printf( "%f      ", (float)C_sinusoidale.value );
        printf( "%f      ", (float)C1_source.value );
        printf( "%f      ", (float)C2_reservoir.value );
        printf( "%f      ", (float)C3_reservoir.value );
        printf( "%f      ", (float)C4_puits.value );
        printf( "%f      ", (float)C5_debordement.value );

        printf( "\n" );

#endif /* CLE_AVEC_TRACE_ECRAN_DONNEES */

        // les résultats sortis sont enregistrés dans des fichiers
        ajouterDonnee( nomFichierOut_t, time.t );
        ajouterDonnee( nomFichierOut_I1_pluie, I1_pluie.value );
        ajouterDonnee( nomFichierOut_I20, I20.value );
        ajouterDonnee( nomFichierOut_I25, I25.value );
        ajouterDonnee( nomFichierOut_V1, V1.value );
        ajouterDonnee( nomFichierOut_Fpluie1, Fpluie1.value );
        ajouterDonnee( nomFichierOut_Fpluie20, Fpluie20.value );
        ajouterDonnee( nomFichierOut_Fpluie25, Fpluie25.value );
        ajouterDonnee( nomFichierOut_F1, F1.value );
        ajouterDonnee( nomFichierOut_F2, F2.value );
        ajouterDonnee( nomFichierOut_F3, F3.value );
        ajouterDonnee( nomFichierOut_F4, F4.value );
        ajouterDonnee( nomFichierOut_C_constante, C_constante.value );
        ajouterDonnee( nomFichierOut_C_monte, C_monte.value );
        ajouterDonnee( nomFichierOut_C_descend, C_descend.value );
        ajouterDonnee( nomFichierOut_C_sinusoidale, C_sinusoidale.value );
        ajouterDonnee( nomFichierOut_C1_source, C1_source.value );
        ajouterDonnee( nomFichierOut_C2_reservoir, C2_reservoir.value );
        ajouterDonnee( nomFichierOut_C3_reservoir, C3_reservoir.value );
        ajouterDonnee( nomFichierOut_C4_puits, C4_puits.value );
        ajouterDonnee( nomFichierOut_C5_debordement, C5_debordement.value );

/* modeDemploi
         *
         */

        return CR_OK;
```

```
}

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


/*****************************************************************************
 *                                                                           *
 * Fin du fichier : specificitePortageMM2CPP.cpp                             *
 *                                                                           *
 *****************************************************************************/
```