Recommandations pour le développement de Cast3M

1 Introduction

Nous allons préciser dans cette note quelques informations utiles aux développeurs du logiciel Cast3M. La lecture préalable des rapports portant sur le langage ESOPE, sur les objets de Cast3M et sur les opérateurs de Cast3M est vivement recommandée.

Certaines des recommandations auront en fait un caractère obligatoire, d'autres sont juste des règles de bonnes conduites qui malheureusement n'ont pas toujours été suivies. Ces recommandations sont là pour assurer la qualité et le portage de Cast3M sur tous types de machine.

Le langage utilisé pour développer Cast3M est ESOPE qui est lui-même une extension de fortran.

La dernière étape du développement est la fourniture d'un cas test.

Nous nous excusons pour le coté rébarbatif de cette liste et nous vous souhaitons bonne lecture.

2 Liste des recommandations

Les opérateurs : La pertinence d'un opérateur et sa syntaxe sont des éléments primordiaux du logiciel Cast3M. Il est très vivement recommandé d'en discuter avec l'équipe de développement du logiciel avant de se lancer dans la programmation.

Les variables : Elles ont six lettres au maximum et évite les mots clés du fortran (do, if, ...)

Elles respectent la carte : IMPLICIT REAL*8(A-H,O-Z) qui doit être systématiquement placée en tête de chaque subroutine

Les fonctions génériques : Il faut utiliser, quand cela est possible, les fonctions génériques du fortran 77, C'est à dire les fonctions qui ne présupposent pas le type de la varieble passée en argument. Par exemple, pour calculer le cosinus de xva (double précision) utiliser COS(XVA) et non pas DCOS(XVA). Ceci facilite le portage sur différentes machines. Les principales sont :

max, min, mod

acos, asin, atan, atan2

cos, sin, tan

log, exp, erf

int, real, nint, sign

sqrt

Les structures de données : Les COMMON fortran et les déclarations de SEGMENT apparaissant dans plusieurs opérateurs ont des déclarations identiques. Ces déclarations doivent être isolées dans des fichiers séparés et incluses à l'aide d'une instruction –INC ...

L'usage du COMMON interne à un opérateur devient interdit pour des raisons de parallélisation du logiciel.

Les structures de données de type SEGMENT doivent être systématiquement supprimées en fin de travail ou au moins elles doivent être désactivées. L'opérateur « TEMPS » suivi de l'option « place » renseigne sur l'état des segments, nombre de segments actifs, nombre total de segments....

Les sous-programmes : Leur nom ne doit pas dépasser six caractères.

Juste après la carte subroutine ... ; il faut placer l'instruction implicite de type suivante :

IMPLICIT REAL*8(A-H,O-Z)

Un des critères de qualité formelle de Cast3M est de vérifier l'analyse d'intercompilation. Il faut veiller à respecter le nombre et les types des arguments lors des appels de sous-programmes.

Un sous programme ne peut recevoir en argument une variable ou un tableau appartenant à une structure SEGMENT s'il désire lui-même utiliser des instructions ESOPE

Commentaires, indentation : La programmation doit être lisible et les algorithmes utilisés doivent pouvoir s'en déduire facilement. Des commentaires aident si nécessaire la compréhension.

Les structures logiques utilisées (IF,DO,..) doivent être soulignées par l'indentation.

Chaque sous programme devra posséder un commentaire précisant sa fonction et au besoin une liste des entrées/sorties devra être commentée.

Entrées Sorties : En standard, un sous-programme ne doit effectuer ni entrée ni sortie. Il doit appeler les fonctions correspondantes de GIBIANE (LIROBJ, ECROBJ, ...) pour communiquer avec l'analyseur de syntaxe. Normalement, un opérateur travaille silencieusement, c'est à dire qu'il n'émet aucun message. Malgré tout, des impressions supplémentaires peuvent être prévues. Elles devront se faire ou non suivant la valeur de la variable IIMPI du common COPTIO, qui en standard se trouve être égale à zéro et qui peut être modifiée par l'utilisateur avec la directive : « OPTION IMPI NN ;». Elles se font alors sur l'unité logique IOIMP. IOIMP se trouve aussi dans le common COPTIO et peut être modifié par :« OPTION IMPR 'mon.fic'; ».

Les erreurs : La validité des structures de données utilisées sera systématiquement contrôlée. La justesse des données également. En cas d'erreur, un message d'erreur sera émis par appel au sous programme ERREUR et le contrôle est rendu au sous programme PILOT. La variable IERR située dans le common COPTIO sera systématiquement testée après chaque appel de fonction pour contrôler la validité de l'exécution.

Pour imprimer un message d'erreur, il faut appeler le sous programmes ERREUR avec en argument un numéro d'erreur. Les données supplémentaires sont transmises par les variables INTERR, REAERR et MOTERR du common COPTIO. Si le message n'existe pas il faut l'ajouter dans le fichier 'GIBI.ERREUR' en respectant les règles suivantes :

Un message contient deux ou quatre lignes. La première (et troisième) contient le numéro de l'erreur (sur 4 caractères), un blanc et le niveau de l'erreur sur un caractère. Ce niveau est égal à 2 pour arrêter l'exécution de l'opérateur, de la boucle répéter et des procédures actives

La deuxième ligne (et la quatrième) contient le texte du message en format A80.

Des informations passées par les variables INTERR, REAERR, MOTERR peuvent être écrites dans le corps du message

%i1,%i2	représentent INTERR(1), INTERR(2) ...
%r1,%r2	représentent REAERR(1), REAERR(2) ...
%m3:8	représente MOTERR(3:8)

Pour ajouter un message d'erreur, il faut prendre un numéro positif consécutif à celui du dernier message et écrire le message dans sa version française et dans sa version anglaise.

Il est aussi possible d'introduire des messages d'information. Pour les différencier des messages d'erreur, leur numéro est négatif et le niveau d'erreur vaut 0.

Les cas tests : Nous rappelons que la fonction d'un cas test est de vérifier la non-régression du logiciel, c'est à dire vérifier que les résultats fournis demain seront les mêmes que ceux fournis hier. Il ne s'agit en aucun cas de faire une démonstration de la validité des résultats obtenus.

Les cas tests de validation des résultats, s'ils sont trés rapides d'exécution, peuvent être utilisés. Sinon, il faut en inventer d'autres ou tronquer leur exécution. De nombreux cas tests utilisent une variable COMPLET positionnée en gibiane à VRAI ou FAUX pour faire la différence entre une exécution complète ou tronquée.

Ne pas faire de dessin dans les cas test ou en contrôler l'exécution par une variable gibiane GRAPH qui est positionnée à FAUX par l'instruction gibiane GRAPH=FAUX ;

3 Conclusion

Le travail d'incorporation des développements externes se trouvera facilité si vous avez suivi ces quelques recommandations.