Documentation Utilisateur

Valentin Laclautre, Anthony Dard, Damien Trouche, Martin Gangand,
Basel Darwish Jzaerly

Contents

1 Description du compilateur

Ce compilateur a pour but de compiler des fichiers terminant en *.deca* respectant la syntaxe décrite dans les spécifications du langage deca. Ce compilateur vient avec de nombreuses options qui seront décrites dans la section suivante. Il peut compiler plusieurs programmes en une fois, en spécifiant tout les programmes en argument. Le programme crée par défaut un fichier *.ass* au même endroit que le fichier *.deca*. Ce fichier correspond au programme assembleur pouvant être lu par la machine virtuelle ima. Il est aussi possible de compiler le programme deca en bytecode java (voir section suivante pour plus de détails), afin de pouvoir l'executer avec la machine virtuelle java.

2 commandes et options

2.1 Commandes decac

decac [-p |-v] [-n] [-r X] [-d]* [-P] fichiers.deca... |[-b]

2.2 option -b

Cette option n'est executé que de la manière suivante: **decac -b**. Elle permet d'afficher le numéro du groupe ainsi que les noms des différents membres de l'équipe ayant développé ce compilateur.

2.3 option -p

L'option **-p** permet de parser le (ou les) fichier(s) deca donnés en paramètres. Après cette étape, si aucune erreur n'est survenu, le programme est décompilé et est affiché. Il doit être syntaxiquement correct après la décompilation. Cette option ne peut pas être utilisé avec l'option **-v**

2.4 option -v

Cette option permet d'effectuer l'étape de vérification de l'arbre. Rien n'est affiché s'il n'y a pas d'erreur. Cette option ne peut pas être utilisé avec l'option **-p**

2.5 option -r X

Cette option permet d'effectuer une compilation générant un code source qui n'utilise que X registres banalisés pour s'exécuter. Attention : X doit être

compris entre 4 et 16. Dans le cas contraire, une "UnsupportedOperationException" est levée lors de la compilation. Elle indique le message suivant : "You have to use a number between 4 and 16 after -r". Cette option ne peut pas être utilisée avec l'option -java.

2.6 option -d

Cette option permet d'obtenir des traces de debug: il s'agit de messages s'affichant sur la sortie standard permettant d'en savoir plus ce qu'il s'est passé au cours de la compilation. On peut répéter plusieurs fois l'option (par exemple: decac -d -d -d), et ce jusqu'à trois fois, pour avoir de plus amples informations.

2.7 option -n

Cette option permet de compiler un programme deca sans faire les tests à l'exécution suivants : division entière (et reste de la division entière) par 0, débordement arithmétique sur les flottants (inclut la division flottante par 0.0), absence de return lors de l'exécution d'une méthode, conversion de type impossible, déréférencement de null, débordement mémoire (pile ou tas).

2.8 option -java

Cette option correspond à l'extension byte du compilateur decac. Elle permet de compiler le fichier deca passé en paramètre en bytecode java. Les fichiers créés en sortie sont au format ".class". Pour chaque classe implémentée dans le fichier deca passé en paramètre, un fichier .class est créé. Il peut donc y avoir plusieurs fichiers .class pour un seul fichier deca.

3 Messages d'erreurs

Lors de la vérification de l'arbre, un grand nombre d'erreurs peut survenir lors de la vérification contextuelle du programme deca. Cette section est consacrée à la description de toutes ces erreurs, afin que vous puissiez les résoudre plus facilement.

3.1 IncludeFileNotFound

3.1.1 (...) include file not found

Cette erreur survient lorsque le fichier à importer n'est pas trouvé ou impossible à lire.

3.2 InvalidLValue

3.2.1 left-hand side of assignment is not an lvalue

Cette erreur survient lorsque l'utilisateur tente d'assigner une valeur à une opérande qui n'est pas une lvalue. C'est à dire à laquelle il est impossible d'affecter une valeur.

3.3 Circular include for file

3.3.1 Circular include for file (...)

Cette erreur survient lorsque l'utilisateur fait une inclusion de fichier circulaire. C'est à dire lorsqu'un fichier en inclut un autre qui a déjà été inclut.

3.4 ContextualError

3.4.1 (0.1) The identifier is not declared

Cette erreur survient lorsque l'utilisateur a tenté d'utiliser une variable qui n'a pas été déclarée avant. Celle-ci n'est donc pas initialisée et est inconnue. Pour la résoudre, il faut déclarer cette variable.

3.4.2 (0.2) The identifier has an invalid type

Cette erreur survient lorsque l'utilisateur tente d'utiliser un type inconnu. En effet, les types autorisés sont : int, float, string, et booléen.

3.4.3 (3.17) Variable declaration with type void is forbidden

Cette erreur survient lorsque l'utilisateur déclare une variable avec le type void, il est imossible d'effectuer cela.

3.4.4 (3.17) The identifier is already declared

Cette erreur survient lorsque l'utilisateur déclare une variable qui a déjà été déclarée préalablement. En effet, il est impossible de déclarer deux variables qui ont le même nom, même si leur type est différent.

3.4.5 (3.28) Expression type is not compatible

Cette erreur survient lorsque l'utilisateur tente d'affecter une opérande à une variable dont le type est incompatible. Par exemple, il n'est pas possible d'affecter un booléen à un float. Cependant, il est possible d'affecter un int à une variable de type float, dans ce cas la conversion est implicite.

3.4.6 (3.29) Condition must return a boolean

Cette erreur survient lorsque l'utilisateur utilise une condition dont le type de retour n'est pas un booléen.

3.4.7 (3.31) Wrong type for the print function. It should be int, float or string

Cette erreur survient lorque l'utilisateur a tenté d'utiliser une fonction d'affichage (print, println, printx, printlnx), mais qu'il a rentré un mauvais type en paramètres. En l'occurence, les seuls type autorisés dans une fonction d'affichage sont: int, float ou string.

3.4.8 (3.33) Arithmetic operation only: (...) accept (int, int) as operands type

Cette erreur survient lorsque l'utilisateur tente d'effecter l'opération arithmétique modulo entre 2 variables qui ne sont pas de type int. En effet l'opération arithmétique modulo n'est autorisée qu'entre 2 entiers.

3.4.9 (3.33) Arithmetic operation: (...) only accept ([int—float], [int—float]) as operands type

Cette erreur survient lorsque l'utilisateur tente d'effectuer une opération arithmétique sur des variables qui ne sont pas de de type int ou float. En effet, les opérateurs arithmétiques +, -, *, et / n'acceptent que des entiers et des flottants.

3.4.10 (3.33) Boolean operation: (...) only accept ([int—float], [int—float]) or objects for == and !=, as operands type"

Cette erreur survient lorsque l'utilisateur utilise l'opérateur "==" ou "!=" avec des types différents de int et float, ou qui ne sont pas des objets. En effet, il n'est possible de tester une égalité ou inégalité que sur des entiers, des flottants, ou des objets.

3.4.11 (3.33) Comparison operation: (...) only accept ([int—float], [int—float]) or objects for a comparison, as operands type

Cette erreur survient lorsque l'utilisateur utilise un opérateur de comparaison avec des types différents de int et float, ou qui ne sont pas des objets. En effet, il n'est possible de faire une comparaison que sur des entiers, des flottants, ou des objets.

3.4.12 (3.37) Not operator only accept boolean operand

Cette erreur survient lorsque l'utilisateur utilise l'opérateur unaire "not" sur un type différent de booléen.

3.4.13 (3.37) UnaryMinus operator only accept int or float operand

Cette erreur survient lorsque l'utilisateur utilise l'opérateur unaire "-" sur un type différent de int ou float.