Manuel Utilisateur

Groupe 41

Corentin HEUZÉ

Dorian VERNEY

Adrien AUBERT

Yuxuan LU

Yung Pheng THOR

Table des Matières

1. Introduction

Ce document sert de manuel d'utilisation pour le compilateur Deca. Il met en avant les diverses options de compilation, les contraintes techniques du compilateur, ainsi que la liste des Messages d'erreur et leurs explications. De plus, il présente le mode opératoire de l'extension, décrivant le processus de création des fichiers .class.

2. Mode d'emploi

Afin d'utiliser le compilateur Deca, il est nécessaire d'ajouter d'abord le chemin vers "ProjetGL/src/main/bin" dans le fichier de configuration du shell. Ensuite, il suffit de redémarrer le shell, et la commande "decac" sera prête à être utilisée.

La syntaxe d'utilisation de l'exécutable decac est:

decac [[-p | -v] [-n] [-r X] [-d]* [-P] [-w] [-arm]<*fichier deca*>...] | [-b]

avec <*fichier deca*> des chemins de la forme <*répertoires/nom.deca*> (le suffix .deca est obligatoire). Le résultat <*répertoires/nom.ass*> est dans même répertoire que le fichier source.

La commande decac, sans argument, affichera les options disponibles. On peut appeler la commande decac avec un ou plusieurs fichiers sources Deca.

Si un fichier apparaît plusieurs fois sur la ligne de commande, il n'est compilé qu'une seule fois

Une fois le fichier assembleur généré, il vous suffit d'utiliser la commande "ima" pour l'exécuter.

3. Commandes et Options

-b	(banner)	affiche une bannière indiquant le nom de l'équipe
-p	(parse)	arrête decac après l'étape de construction de l'arbre, et affiche la décompilation de ce dernier (i.e. s'il n'y a qu'un fichier source à compiler, la sortie doit être un programme deca syntaxiquement correct)
-v	(verification)	arrête decac après l'étape de vérifications (ne produit aucune sortie en l'absence d'erreur)
-n	(no check)	supprime les tests à l'exécution spécifiés dans les points 11.1 et 11.3 de la sémantique de Deca.
-r X	(registers)	limite les registres banalisés disponibles à R0 ... R{X-1}, avec 4 <= X <= 16
-d	(debug)	active les traces de debug. Répéter l'option plusieurs fois pour avoir plus de traces.
-P	(parallel)	s'il y a plusieurs fichiers sources, lance la compilation des fichiers en parallèle (pour accélérer la compilation)
-arm	(arm)	Compilation pour ARM.

Les options '-p' et '-v' sont incompatibles. L'option '-b' ne peut être utilisée que sans autre option, et sans fichier source. Dans ce cas, decac termine après avoir affiché la bannière.

4. Message d'erreur

4.1. Les erreurs de syntaxe hors-contexte

4.1.1.token recognition error

Une erreur lorsque la syntaxe du code est incorrecte, cela permet entre autres de vérifier l'orthographe des mots clés, de corriger les parenthèses et les guillemets.

Exemple :

```
../../deca/lexer/invalid/incorrectInclude.deca:10:0: token recognition error at: '#include s'
```

4.1.2.mismatched input error

Cette erreur signifie que le code saisi n'est pas conforme aux règles grammaticales du langage de programmation. Cela peut inclure des fautes d'orthographe, des points-virgules manquants, des crochets qui ne correspondent pas.

Exemple :

```
../../deca/syntax/invalid/listDeclVar/incorrectDeclFloat.deca:12:14: mismatched input ';' expecting {'true', 'new', 'nu
ll', 'readInt', 'false', 'readFloat', 'this', IDENT, '-', '(', '!', INT, FLOAT, STRING}
```

4.1.3.include file not found error

Le fichier est introuvable. Le nom ou le chemin du fichier est incorrect.

Exemple :

```
../../deca/syntax/invalid/include/incorrectInclude.deca:10:20: test.deca: include file not found
```

4.1.4.no viable alternative error

Le code saisi peut contenir des structures ou des symboles qui ne respectent pas les règles grammaticales.

Exemple :

```
../../deca/syntax/invalid/class/incorrectExtends.deca:11:12: no viable alternative at input 'extend'
```

4.1.5.missing IDENT error

Indique qu'il manque un identifiant dans le code.

Exemple :

```
../../deca/syntax/invalid/listDeclVar/incorrectBooleanLiteral.deca:11:15: missing IDENT at '='
```

4.1.6.lvalue error

Génère une erreur lorsque le côté gauche d'une affectation n'est pas une expression attribuable (paramètre, variable ou propriété)

Exemple :

```
../../deca/syntax/invalid/print/incorrectPrint2.deca:11:10: left-hand side of assignment is not an lvalue
```

4.2. Les erreurs de syntaxe contextuelle

4.2.1. Identifier

Cause: opération partielle
Message: No definition for this identifier: (this.name)

4.2.2. Type

Cause: opération partielle
Message: No definition for this type identifier: (name.getName())

4.2.3. decl_class

Cause: opération partielle env_types(super)
Message: Superclass is not declared in Environment Type
Cause: condition class
Message: Superclass is not of type Class
Cause: opération partielle union disjoint
Message: Class identifier: ___ is already declared previously

4.2.4. decl_field

Cause: condition type != void
Message: Field: ___ should not be of type void

Cause: operation partielle union disjoint avec son parent
Message: Field name is declared in its superclass and not declared as field type
Cause: operation partielle union disjoint
Message: Field is redeclared

4.2.5. decl_method

Cause: method(sig2), type2 = env_exp_super(name)
Message: method identifier declared in superclass and not declared as method type
Cause: condition sig=sig2
Message: method is redeclared in superclass and has different signature from its superclass
Cause: condition subtype(env_types, type, types2)
Message: the return type of redeclared method is not a subtype of the return type of its superclass method
Cause: operation partielle union disjoint
Message: Method is redeclared

4.2.6. decl_param

Cause: condition type != void
Message: Parameter type should not be void

4.2.7. list_decl_param

Cause: opération partielle union disjoint
Message: Parameter is redeclared

4.2.8. decl_var

Cause: condition type != void
Message: Variable type should not be null.
Cause: Partial operation (Disjoint Union)
Message: Variable is redeclared.

4.2.9. return

Cause: condition type != void
Message: type ne doit pas être void

4.2.10. print

Cause: Les types ne sont int float ou string
Message: Invalid type to print : (printType)

4.2.11. println

Cause: Les types ne sont int float ou string
Message: "Invalid type to print : (printType)

4.2.12. rvalue

Cause: condition assign_compatible(env_types, type1, type2)
Message: Expected type: ( ) but the returned type: ( )

4.2.13. condition

Cause: condition boolean type
Message: Condition does not return a boolean type

4.2.14. exp_print

Cause: condition type non compatible
Message: type doit être int, float ou string

4.2.15. Opérations binaires

Cause: OpArith
Message: The type is not integer or float
Cause: OpBool
Message: The type is not boolean
Cause: OpCmp
Message: The type is not integer or float
Cause: Modulo
Message: The type is not integer

4.2.16. Opération Unitaire

Cause: UnaryMinus
Message: The type is not integer or float
Cause: Not

Message: The type is not boolean

4.2.17. Cast

Cause: cast_compatible(env_types, type2, type)
Message: Error: Compatibility for conversion

4.2.18. New

Cause: type != class type
Message: Type error : should be a class type

4.2.19. This

Cause: type != 0
Message: ‘This’ should be inside a class

4.2.20. decl_class

Cause: opération partielle env_types(class)
Message: Superclass: C is not declared in Environment Type
Cause: condition class
Message: identificateur de classe attendu

4.2.21. Selection

Cause: env_types(class2) = class(, …)
Message: Type error : should be a class type
Cause: subtype(env_types, type_class(class2), type_class(class)) for field_ident=protected
Message: Error: is not a subtype
Cause: subtype(env_types, type_class(class), type_class(class_field)) for field_ident=protected
Message: Error: is not a subtype

4.2.22. Field_ident

Cause: identifier return type not field
Message: Type error: should be a field

4.2.23. MethodCall

Cause: env_types(class2) = (class, …)
Message: Type error: should be a class type

4.2.24. Method_ident

Cause: identifier return type not method
Message: Type error: not a method

4.3. Les erreurs de Codegen

4.3.1 Heap Overflow Error

Cause: Erreur de débordement de tas
Message: Heap Overflow

4.3.2 Input/Output Error

Cause: Erreur de non-compatibilité avec Int ou Float
Message: Input/Output Error

4.3.3 Null Dereferencing

Cause:Erreur d'utilisation d'un objet qui référence à null
Message: Object is null

4.3.4 Overflow Error

Cause: Erreur de débordement lors d'une opération arithmétique
Message: Overflow during arithmetic operation

4.3.5 Stack Overflow Error

Cause: Erreur de débordement de pile
Message: Stack Overflow

5. Limitations

5.1. Partiellement implémenté avec des erreurs persistantes

5.1.1 Appel de méthode

Les appels de méthode fonctionnent dans la globalité seulement. On ne peut pas “Print” directement l’appel d’une méthode. Nous devons passer par une variable intermédiaire. De même pour les appels récursifs, nous devons conserver le résultat dans une variable car “Return” ne fonctionne pas avec les appels.

5.1.2 Gestion de la mémoire

Notre compilateur ne va pas utiliser de registres qui lui sont interdits Nous arrivons à allouer et désallouer des registres pour effectuer nos calculs (Nous avons joué avec les priorités pour utiliser moins de registres). En parallèle nous avions commencé un gestionnaire de mémoire plus complet en utilisant la pile, mais ce dernier ne fonctionnait seulement que pour certaines opérations.

Donc des opérations utilisant trop de registres par rapport au nombre autorisé mène à une erreur.

5.1.3 Comparaisons

Nous traitons les cas “simples” de comparaisons, i.e. “a == b” mais des cas plus complexes autorisés par la grammaire comme “a == b == c”.

5.1.4 Optimisations des opérations arithmétiques

Nous avions seulement pensé, mais pas implémenté, à des optimisations de calculs si on divise ou bien on multiplie par une puissance de 2.

5.2. Non implémenté

5.2.1. Cast entre Objets

Seul le transtypage entre un float et un int et entre un int et un float fonctionne. De plus si B hérite de A alors “A a = new B()” fonctionne. Mais (B)(a) ne marche pas.

5.2.2 Visibilité des champs

Nous ne tenons pas compte de la visibilité des champs dans genCode. Cependant la fonctionnalité marche jusqu'à la partie contextuelle.

5.2.3 TSTO

Non implémenté, donc les erreurs de dépassement de pile ne sont pas détectées (car aussi pas d'appel récursif).

5.2.4 InstanceOf

Nous avons seulement commencé à implémenter la fonction. Nous étions sur le parcours des types jusqu'à "Objet".