ELEFTHERIOU Theophanis - ELIEZER Gwennan - GINDRIER Clément
PETITEAU Matthieu - POLICET Rémi

MANUEL UTILISATEUR

Vous trouverez dans ce document les spécificités de notre compilateur

Equipe GL60

GRENOBLE INP UGA Ensimag

SOMMAIRE

Spécifications de notre compilateur

Langage supporté

Le compilateur rendu supporte l'essentiel des scripts deca. Vous pouvez l'utiliser pour n'importe lequel de vos programmes deca tant qu'il ne comporte pas de conversion (cast) ou de test d'appartenance (instanceof).
Il supporte ainsi les déclarations de variables de types primitifs (Object, void, boolean, string, int, float ou null) ou de classes, mais aussi les principales fonctions pré implantées (print, println, printx, printlnx, readInt, readFloat).
Il accepte les opérations classiques (addition, soustraction, multiplication, division, reste de la division euclidienne, assignation, négation), les opérations booléennes (and, or et négation) et les relations d'ordre. Il gère aussi les boucles (uniquement avec while) et les conditionnelles (if, elsif et else). Néanmoins, au sein d'un bloc, il est impossible d'alterner déclarations de variables et instructions : toutes les déclarations de variables doivent être faites au début du bloc.
À l'aide de notre compilateur, vous pouvez aussi réaliser des déclarations de classes (étendues ou non). De plus, il permet de déclarer les classes dans n'importe quel ordre: on peut en particulier déclarer une classe fille avant d'avoir déclaré sa classe mère. Au sein des déclarations de classes, le compilateur gère les déclarations d'attributs et de méthodes (qui peuvent même être alternées !). Pour les attributs, deux visibilités sont acceptées : Si l'on ne précise pas, les attributs sont visibles partout, sinon ils peuvent être PROTECTED, visibles seulement par la classe la déclarant ou par les classes descendantes. Les méthodes peuvent être écrites en deca ou en assembleur avec le mot clé 'asm' comme suit :

```
typeDeMaMethode maMethode
      asm("contenu_de_ma_methode");
```

Finalement notre compilateur supporte la compilation séparée (pour plus de détails, voir la partie *Extension Link*).

Commandes et options

Les commandes demandées sont implémentées :

-b : Affiche une bannière indiquant le nom de l'équipe.

-p : Décompile l'arbre abstrait obtenu à la suite du lexing et du parsing.

-v : Vérifie que le programme peut compiler.

-P : S'il y a plusieurs fichiers sources, lance la compilation des fichiers en parallèle (pour accélérer la compilation).

-n : Supprime les tests de division par 0, de débordement arithmétique sur les flottants, d'absence de `return` à une méthode, de conversion de type impossible, de déréférencement de null, de débordement mémoire, d'accès à des variables non initialisées, d'utilisation d'une méthode écrite en assembleur non compatible avec l'assembleur généré par le compilateur.

-d : Active les traces de debug. Répéter l'option jusqu'à 4 fois pour avoir plus de traces.

-r : Limite les registres banalisés disponibles. Cette limite doit être comprise entre 4 et 16 inclus.

Nous avons également ajouté les options suivantes :

-h : Affiche de l'aide sur les options du compilateur.

-c : Compile en fichier objet `.deco`, et autorise l'import de fichiers.

Messages d'erreurs

Erreurs de lexicographie :

Erreur	Cause
no viable alternative at input 'clas'	Token non reconnu par le lexer Le mot (ici "clas") n'est pas dans le lexique accepté.

Erreurs de syntaxe :

Erreur	Cause
mismatched input 'class' expecting {'instanceof', ';', '.', ',', '=', '+', '-', '*', '/', '%', '>', '<', '==', '!=', '>=', '<=', '&&', '||'}	Erreur de grammaire La suite de mots n'est pas reconnue par la grammaire. Ici le mot 'class' est suivi d'un '=' dans le code alors que les mots autorisés derrière 'class' sont {'instanceof', ';', '.', ',', '=', '+', '-', '*', '/', '%', '>', '<', '==', '!=', '>=', '<=', '&&', '||'}.
missing '}' at '<EOF>' missing ')' at '{' missing IDENT at '='	Oubli d'un mot pour respecter la grammaire. Le message d'erreur indique le mot suivant l'erreur.

Erreurs mémoire :

Erreur	Cause
Utilisation d'un flottant non supporté sur 32 bits.	Assignation d'un flottant dépassant les 32 bits autorisés.
Utilisation d'un entier non supporté sur 32 bits.	Assignation d'un entier dépassant les 32 bits autorisés.

Erreurs de Contexte :

Class	
Erreur	**Cause**
This class is already defined in this context	Définition de deux classes avec le même nom.
It seems that there is a fishy loop in the class hierarchy	Une classe a été déclarée comme sa propre mère, ou est une sous-classe d'elle même
the parent of this class should be a class	Une classe ne peut hériter que d'un objet de type classe.

Expressions	
Erreur	**Cause**
Error: Unsupported operands. Expected : int	L'expression devrait être un entier uniquement et ce n'est pas le cas ici.
Error: Unsupported operands. Expected : int or float	L'expression devrait être un entier ou un flottant et ce n'est pas le cas ici.
Error: Unsupported operands. Expected : boolean	L'expression devrait être un booléen et ce n'est pas le cas ici.
the expression f is not recognised	'f' n'est pas défini dans l'environnement et ne peut donc pas être appelé correctement. Soit c'est une variable inexistante, soit un attribut inexistant, soit une méthode inexistante.
Type int is not a class type, and has thus no field.	On cherche à appeler une variable définie dans un type qui n'est pas une classe.
Cannot use the symbol 'this' outside a class	'this' ne peut pas être appelé à l'extérieur d'une classe, dans Main par exemple.
the type ClasseIndefinie is not recognized	On fait appel à un type non existant.
t is undefined in this context	On fait appel à une expression non définie.
This variable is already defined in this context	On définit une variable qui a déjà été définie.
This expression cannot be defined as a field because its parent already define it another way	On définit une expression d'une certaine manière alors que sa classe mère la définit d'une autre manière. Par exemple une méthode qui serait Override par un Field.
the type pangolin is not recognized	Un identifiant dans le code est censé être un type, mais il ne correspond ici ni à un type primitif, ni à une classe déclarée.
A field cannot be a 'void' type	Un field ne peut pas être de type void.

Field	
Erreur	**Cause**
This expression cannot be defined as a field because its parent already define it another way	On définit une expression d'une certaine manière alors que sa classe mère la définit d'une autre manière. Par exemple une méthode qui serait Override par un Field.
the type pangolin is not recognized	Le type du field n'est pas reconnu.
A field cannot be a 'void' type	Un field ne peut pas être de type void.
The returned type of this method should be a sub-type of the one of its parent	Le type de retour d'une méthode doit être un sous type du type de retour de la méthode parente.
The method signature doesn't match its parent	La signature d'une méthode doit correspondre à la signature de son parent.
Cannot pass a voidType object in argument	Une méthode ne peut pas prendre de paramètre de type void.
Type TYPE is not a class type, and has thus no method.	Un appel de méthode a été fait sur une variable de type primitif TYPE, qui n'est pas une classe et n'a donc pas d'attribut.
The identifier IDENT in class CLASS is not a field	Dans le code, IDENT est utilisé comme un attribut sur une variable de classe CLASS, mais dans cette classe, IDENT est une méthode.
Cannot access CLASS's field "FIELD" from Main	L'attribut FIELD de la classe CLASS est protégé, son accès est donc interdit dans le programme principal.
Cannot access CLASS1's attributes from class CLASS2	Un accès à un attribut protégé d'un objet de la classe CLASS1 est fait dans une méthode de la classe CLASS2, mais elle n'est pas fille de la classe CLASS1, et n'a donc pas accès à ses attributs protégés.
Cannot access field of class CLASS1 from class CLASS2	Un accès à un attribut protégé défini dans une classe CLASS1 est fait dans une méthode de la classe CLASS2, mais elle n'est pas fille de la classe CLASS1, et n'a donc pas accès à cet attribut. (Différence avec l'erreur

	au dessus: on regarde la classe dans laquelle l'attribut est défini, et pas la classe de l'objet auquel on sélectionne un attribut)

Method	
Erreur	**Cause**
The returned type of this method should be a sub-type of the one of its parent	Le type de retour d'une méthode doit être un sous type du type de retour de la méthode qui est 'Override'
The method signature doesn't match its parent	La signature d'une méthode doit correspondre à la signature de son parent
Cannot pass a voidType object in argument	Une méthode ne peut pas prendre de paramètre de type void
Type TYPE is not a class type, and has thus no method.	Un appel de méthode a été fait une sur variable de type primitif TYPE, qui n'est pas une classe et n'a donc pas de méthode.
The identifier IDENT in class CLASS is not a method	Dans le code, IDENT est utilisé comme une méthode sur une variable de classe CLASS, mais dans cette classe, IDENT est un attribut.

Erreurs assembleur :

Erreur	Cause
Error: Division by zero	Opération de division entière / flottante avec le membre de droite égal à 0 / 0f.
Error: Modulo by zero	Opération de modulo entier avec le membre de droite égal à 0.
Error: Input/Output error	L'utilisateur rentre un nombre du mauvais type, ou avec un overflow.
Error: Overflow during arithmetic operation	Une opération flottante a dépassé les 32 bits.
Error: Stack Overflow	La pile n'est pas assez grande et débordera si on continue l'exécution du programme.
Error: Expected a return at the end of a no void method	Si la méthode retourne un type autre que void. Elle doit forcément passer par un return.

Erreur : dereferencement de null	Soit "a" un objet avec comme valeur : null. Les trois codes suivants provoquent cette erreur :
	a.x = 5;
	int b = a.x;
	a.méthode();

Limitations

Notre compilateur ne supporte pas la totalité du langage deca: les tests d'appartenance et les conversions de types ne sont pas acceptées.
En outre, avec notre compilateur, il est impossible de déclarer une variable de type chaîne de caractère. En effet, les chaînes de caractères sont stockées à l'aide de tableaux. Le langage deca sans extension ne comprenant pas la gestion des tableaux, nous n'avons pas ajouté cette option à notre compilateur.

Extension LINK

Présentation de l'extension

L'extension LINK a pour but de permettre la compilation séparée, suivie d'une édition de liens. Grâce à cette extension, l'utilisateur peut donc importer d'autres fichiers deca dans son programme deca, compiler chaque programme indépendamment, et ensuite les recombiner. Cela lui permet de mieux organiser et structurer son code.

Lors d'un import, toutes les classes définies dans le fichier importé sont importées, et peuvent être utilisées dans le programme entier. Lors de l'édition de liens, seul le programme principal (main) du premier fichier compilé est gardé, les autres sont supprimés.

Utilisation de l'extension

L'utilisation de l'extension se veut simple et intuitive. Elle se fait en 3 étapes :

1. Importer les fichiers qu'il faut dans chaque programme
2. Compiler les programmes séparément avec l'option `-c`, créant des fichiers objets, d'extension `.deco`
3. Combiner les fichiers objets avec l'éditeur de liens `linker`

Exemple rapide

Pour un fichier "progs/a.deca" utilisant une classe définie dans "modules/b.deca", ajouter dans "a.deca":

```
import "../modules/b.deca"
```

Compiler les fichiers :

```
decac -c progs/a.deca
decac -c modules/b.deca
```

Édition des liens :

```
linker progs/a.deco modules/b.deco
```

Exécution :

```
ima progs/a.ass
```

Imports des fichiers

Pour importer d'autres fichiers deca, il faut ajouter une ligne par fichier importé commençant par le mot clé 'import', puis d'écrire le chemin vers le fichier importé entre guillemets ("chemin/du/fichier/nom_fichier.deca"). Les chemins relatifs comme absolus sont acceptés, et sont calculés à partir du fichier compilé.
Les imports doivent absolument se trouver avant les déclarations de classes.
Enfin les noms des fichiers deca ne peuvent pas contenir les 3 caractères spéciaux n'apparaissant pas dans les String, à savoir : '\n', '"' et '\\' (Si c'était le cas, le Lexer ne reconnaîtrait pas le nom du fichier et soulèverait une erreur).
Voici un exemple fonctionnel et un exemple non fonctionnel de fichier déca :

Exemple fonctionnel :

```
import "/user/6/elefthet/Projet_GL/src/test/deca/syntax/valid/tests/import_de_bonne_qualite.deca"

class Test {
    int attribut
    void fct(int a){
        this.attribut = a;
    }
}



{
    println("Je");
    println("m'en");
    println("vais");
    println("comme");
    println("un");
    println("prince");
}
```

Ici l'import est réalisé au bon endroit (au tout début du programme), l'adresse du fichier est bien encadrée de doubles guillemets et le mot réservé import est orthographié correctement. Ainsi, en supposant que l'adresse du fichier soit correcte, l'import sera réalisé sans problème.

Exemple non fonctionnel :

```
class Test {
    int attribut
    void fct(int a){
        this.attribut = a;
    }
}

import /user/6/elefthet/Projet_GL/src/test/deca/syntax/valid/tests/import_de_mauvaise_qualite.deca

{
    println("Je");
    println("m'en");
    println("vais");
    println("comme");
    println("un");
    println("crapaud");
}
```

Ici l'import est réalisé au mauvais endroit (après les déclarations de classes) et l'adresse n'est pas encadrée de doubles guillemets.

Compilation séparée

Pour compiler les fichiers en fichier objet, il faut rajouter l'option -c lors de la compilation. Ainsi, si le fichier montré ci-dessus s'appelle "test_import.deca", on peut le compiler à l'aide de la commande suivante :

```
decac -c test_import.deca
```

Le compilateur va de lui-même récupérer les déclarations des classes des fichiers importés grâce aux instructions import.
Le résultat d'une compilation avec l'option -c se retrouve non pas dans un fichier .ass, mais dans un fichier .deco (pour **Dec**a **O**bject file).

L'option -c est incompatible avec l'option -v ou -p: aucun fichier objet ne sera généré si l'une de ces deux options est présente en plus de l'option -c.

Édition de liens

Pour faire l'édition de liens, il faut avoir au préalable compilé le programme que l'on souhaite exécuter ainsi que toutes ses dépendances (c'est-à-dire les fichiers qu'il importe, ainsi que les dépendances des fichiers qu'il importe).
Ensuite, il faut lancer la commande linker, avec comme premier argument le fichier objet du programme que l'on souhaite exécuter, puis comme arguments suivants les fichiers objets des dépendances du programme.

```
linker <prog_principal>.deco <module1>.deco <module2>.deco ...
```

L'éditeur de liens va ainsi récupérer les fichiers .deco qui lui sont donnés, et en créer un fichier .ass, contenant le programme principal du premier fichier, et les classes de tous les fichiers donnés.

Ce fichier pourra ensuite être exécuté avec ima.

Messages d'erreurs

Erreur	Cause
Erreur dans le nommage du/des fichier(s) importé(s).	Le nom du fichier a mal été orthographié dans l'import du fichier source. Le chemin absolu du fichier dans le fichier principal est incorrect.
token recognition error at: '"/Chemin/absolu/de/mon/fichier/nom_de_mon_fichi\n'	Le nom du fichier comporte un des caractères interdits : \n', '"' et/ou '\\'. (\n dans l'exemple d'erreur).

Limites de notre extension

La principale limite de l'extension de notre compilateur est que l'outil `linker` ne vérifie pas que les fichiers liés sont les bons. Ainsi, si un fichier objet est manquant ou en trop, c'est `ima` qui va le détecter au début de l'exécution (label défini en double ou indéfini).
De plus, si un fichier voit sa définition des classes être modifiée (ajout/suppression d'une classe, ajout/suppression d'une méthode, ajout/suppression d'un champ), il faut recompiler tous les fichiers qui en dépendent. Si la recompilation n'a pas lieu, le comportement de l'exécution du fichier assembleur résultant de l'édition de liens n'est pas défini.
Une seconde limite est la non-gestion des boucles d'import. Ainsi si un fichier A importe un fichier B et que le fichier B importe le fichier A, alors la commande 'decac -c nom_du_fichier_A' ne termine pas.
De plus, dans l'état actuel des choses, les fichiers doivent être importés un à un. La notion de paquetage n'existe pas, on ne peut donc pas directement importer un dossier comprenant plusieurs fichiers.