Manuel utilisateur – Compilateur Deca

Mamadou Thiongane

Omar Guessous

Samy Ahjaou

Hamza Bouihi

Grégoire Sename

Groupe 9 - Équipe 42

22 janvier 2024

Table des matières

1 Commandes de compilation

Le compilateur Deca se lance avec la commande

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

Les options sont les suivantes :

- `-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 $4 \leq$ `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) ; ne pas utiliser avec l'option -v

2 Limitations du compilateur

Le compilateur est 100% fonctionnel pour du code sans-objet. Pour l'étape de génération de code de la partie avec objet, seule la passe 1 a pu être réalisée. En effet, la table des méthodes répond parfaitement aux attentes décrites dans la spécification, néanmoins l'initialisation des champs, des méthodes (et donc la gestion de l'instruction `return`) ainsi que la gestion de la partie objet dans le programme principal n'ont pas pu être implémentées.

3 Messages d'erreur

3.1 Erreurs lexicographiques

Code erreur	Description de l'erreur
token recognition error at : [description]	Chaîne de caractère non finie par exemple : *"chaine non finie*
include file not found	Tentative de *include* un fichier non existant

Table 1 – Liste des erreurs lexicographiques

3.2 Erreurs de syntaxe hors-contexte

Code erreur	Description de l'erreur
mismatched input [caractère écrit] expecting [caractères attendus]	Le caractère tapé dans le code n'est pas attendu, pour que le code soit correct écrivez parmis la liste de [caractères attendus]
no viable alternative at [char]	Le caractère tapé ne s'inscrit pas dans le contexte de cette ligne.
missing [char1] at [char2]	Il manque le caractère 1 avant le caractère 2.
extraneous input [char1] expecting [char2]	Il y a un [char1] écrit en trop, à la place il était attendu un [char2].
left-hand side of assignment is not an lvalue	La partie gauche doit être du type lvalue, par exemple : *i + 1 = 5 ;*

Table 2 – Liste des erreurs de syntaxe hors-contexte

3.3 Erreurs de syntaxe contextuelle

Code erreur	Description de l'erreur
les deux types ne sont pas compatibles	L'affectation entre ces deux types est incompatible, par exemple *boolean* et *int*.
Votre condition doit être de type Boolean	La condition dans une opération *if then else* ou *while* doit être de type *boolean*.
identifiant introuvable.	L'identifiant appelé n'a pas été déifini.
Type invalide	Le type utilisé ne fait pas partie de la liste des types valides pour instancier une variable : *boolean*, *int*, *float*.
Variable déjà définie	Ce nom de variable est déjà utilisé.
Vous ne pouvez pas declarer une variable de type Void !	La déclaration d'une variable de type void est interdite.
Vous ne pouvez pas declarer une variable de type String !	La déclaration d'une variable de type String est interdite.
Le type des deux operandes doit etre int pour l'operation mod	L'opération de modulation ne peut être effectuée qu'entre deux entiers.
Le type attendu après not est boolean	L'opération *not* ne peut être effectuée qu'après un *boolean*.
operation arithmetique impossible avec ces deux types	L'opération arithmétique demandée n'est pas possible entre ces deux type (possible entre float et int).
Un ou deux des operateurs n'est pas de type boolean	Les opération logique *AND*, *OR*, ne peuvent être effectuée qu'entre deux *boolean*.
operation de compraison impossible avec ces types	Les opération de comparaison *GT*, *GEQ*,*LT*, *LEQ* ne peuvent être effectuée qu'entre deux *int*, deux *float* ou un *int* et un *float*. Les opérations *EQ*, *NEQ* entre ceux-ci ou deux *boolean* ou deux *string*.
Type non accepté par Print.	La fonction *Print* attend une *string*, un *int*, ou un *float*.
Le type attendu apres unaryMinus est int ou float	Le type attendu après un signe - dans une opération unaire est un *int* ou un *float*.
Vous ne pouvez pas hériter de ce que vous avez écrit	La classe mère écrite n'est pas une classe.
Env_exp_object non défini	La classe mère n'est pas définie.
La classe mère n'existe pas	La classe mère écrite n'existe pas.
Classe déjà définie	Une classe du même nom est déjà définie.
attribut déjà défini dans la classe mère en tant que methode	Une méthode du même nom a déjà été défini dans la classe mère.
Vous ne pouvez pas déclarer un field de type void	Un attribut ne peut pas être de type *void*.
field déjà défini	Un attribut du même nom a déjà été défini dans la classe.
methode déjà definie	Une méthode du même nom a déjà été définie dans la classe.
methode déjà définie dans la classe mère en tant que field	Un attribut du même nom a déjà été défini dans la classe mère.
le paramètre ne peut pas être de type Void	Une paramètre d'une méthode ne peut pas être de type *void*. Il peut être de type *int*, *float* ou *boolean*.
Cast impossible	Le cast n'est pas possible.

Override impossible, verifiez la signature de votre fonction	L'override de la méthode n'est pas possible, veuillez vérifier la signature de votre méthode.
Override impossible, verifiez le type que renvoie votre fonction	L'override de la méthode n'est pas possible, veuillez vérifier le type de votre méthode.
Paramètre dèjà utilisé	Le paramètre est déjà utilisé.
Verifiez le nombre de parametre de la fonction	Le nombre de parametre utilisé ne correspond pas au nombre de paramètres attendus.
Methode [nom de la méthode] non définie dans la classe	Vous tentez d'appeler une méthode qui n'existe pas dans cette classe.
Vous ne pouvez pas appeler New sur ce type	Ce type n'est pas une classe
Votre fonction ne renvoie rien	La fonction devrait renvoyer quelque chose mais ne renvoie rien.
Champ protégé	Cet attribut est protégé, vous ne pouvez pas y avoir accès en dehors de sa classe ou des sous-classes de celle-ci.
Vous ne pouvez pas utiliser This dans le Main	Vous ne pouvez pas utiliser *this* dans le main, seulement dans une classe.

Table 3 – Liste des erreurs de syntaxe contextuelle

3.4 Erreurs d'exécution du code assembleur

Code erreur	**Description de l'erreur**
stack_overflow_error	La taille maximum allouée pour ce programme a été depassée.
overflow_error	Débordement lors d'une opération arithmétique sur les flottants ou d'une division par zéro des flottants
zero_division_error	La division par 0 sur les entiers est interdite.
zero_modulo_error	L'opération modulo sur les entiers ne peut pas être faite par 0.
io_error	L'entrée ne correspond pas à celle attendue pour la méthode `readInt()` ou `readFloat()`.

Table 4 – Liste des erreurs d'exécution du code assembleur

4 Extension TRIGO

4.1 Mode opératoire de l'extension TRIGO

L'extension TRIGO est implémentée dans le fichier `src/main/resources/include/Math.decah`. Il fait partie de la bibliothèque standard de Deca, et donc est utilisable en incluant simplement `Math.decah` quelque soit l'emplacement du fichier de test dans le projet.

Chaque fonction prend en paramètre un `float` et renvoie un `float`. Les différentes fonctions sont :
- `cos` : calcul de cosinus
- `sin` : calcul de sinus
- `tan` : calcul de tangente
- `atan` : calcul de arctangente
- `asin` : calcul de arcsinus
- `ulp` : calcul de l'ulp (Unit of Least Precision)

On peut également récuperer la valeur de π avec la variable `pi` dans la bibliothèque. Voici un exemple d'utilisation de cette bibliothèque.

```
// Fichier exempleMath.deca

#include "Math.decah"

{
    Math Math = new Math();

    float cos3pi = Math.cos(3 * Math.pi);

    println(cos3pi);
}
```

4.2 Précision de l'extension TRIGO

Cette bibliothèque utilisant la norme *IEEE 754 simple précision*, les calculs sont limités par l'*ulp* (Unit of Least Precision).

Ainsi, pour un calcul de *sinus*, *cosinus* ou *tangente*, l'erreur moyenne est de l'ordre de 10^{-7}. Ceci est dû au fait que nous utilisons la periodicité des fonctions pour recentrer les valeurs dans l'intervalle $[-\pi, \pi]$, ou l'*ulp* est de l'ordre de 10^{-7} (err $\in [10^{-9}, 10^{-6}]$).

La fonction asin est définie dans l'intervalle $[-1, 1]$, et donc a une erreur de l'ordre de 10^{-8} mais qui augmente en s'approchant des bornes de son ensemble de définition.

La fonction atan est définie sur $\mathbb{R}$, et a une précision qui reste inférieure à 10^{-9} sur tout son ensemble de définition.