Sommaire

0. Les options du compilateur

La syntaxe d'utilisation de l'exécutable decac est : decac [[-p | -v] [-n] [-r X] [-d]* [-P] ...] | [-b]

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

Option	nom	Description
-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)

1. Limitations et points propres de notre compilateur

L'implémentation du Deca sans objet est complète.

Les limitations ou les points propres à l'implémentation de votre compilateur (par exemple, les portions du langage non ou mal implémentées). On précisera les effets de ces limitations pour l'utilisateur en diagnostiquant éventuellement la cause de ces limitations.

Le compilateur ne fonctionne pas comme attendu avec de grandes conditions. Pour une condition d'un while avec 13785 caractères, 529 ou logiques, 530 et logiques et 1058 comparaisons, le code assembleur ne s'exécute pas correctement. Dans le cas d'un if, avec une condition de 63795 caractères, 2443 ou logiques, 2444 et logiques et 4886 comparaisons, une erreur à la compilation est générée (Stack overflow).

2. Messages d'erreur

2.1 Erreurs syntaxique

Erreur	**Description**	**Exemple**
[**File name**]: [**location**]: token recognition error at: [**token**]	Token inconnu	Program : `[]` `test.deca:1:0: token recognition error at: '['`
[**File name**]: [**location**]: mismatch input [**token**] expecting [**list expected token**]	Le compilateur a reconnu un token qui n'est pas dans la liste de tokens attendus	Program: `{if (true) else}` `test.deca:2:14: mismatched input 'else' expecting {')', '=', '.', '&&', '||', '==', '!=', '<', '<=', '>', '>=', '+', '-', '*', '/', '%', 'instanceof'}`

[**File name**]: [**location**]: no viable alternative at input [**token**]	Le token ne match aucune règle ou des déclaration après des instructions	Program : `class A }`
		`test.deca:1:8: no viable alternative at input '}'`
		Program: `{int a; a = 1; int b}`
		`test.deca:1:19: no viable alternative at input 'b'`

[**File name**]: [**location**]: Missing [**token**] at [**token**] [**File name**]: [**location**]: extraneous input [**token**] expecting [**token**]	Le compilateur a reconnu un token inattendu au lieu d'un token spécifique	Program :`{println(2)}` `test.deca:3:0: missing ';' at '}'`

2.2. Erreurs contextuelles

Erreur	Description	Exemple	Code
Wrong right value type	Erreur de typage lors d'une assignation / instanciation.	Assignation d'une chaîne de caractères à une variable entière.	`int x = 1.1;;`
Wrong condition type	Expression non booléenne dans une condition.	Chaîne de caractères dans la condition d'un *if*.	`if ("a"){}`
Wrong type: expected int or float ≠ current	Erreur de typage dans l'évaluation d'une opération arithmétique.	Multiplication d'une chaîne de caractère par un entier.	`"a" * 2;`
Wrong type - expected: boolean boolean ≠ current	Erreur de typage dans l'évaluation d'une opération booléenne.	Evaluation d'un *ET* entre un booléen et un entier.	`true && 2;`
Wrong left value type	Erreur de typage lors d'une comparaison.	Comparaison entre un entier et un objet de classe *C*.	`Class A {}{A a = new A(); a < 3;}`
Wrong type: boolean (/class /string) is forbidden	Erreur de typage lors d'une comparaison hors "*=*" et "*!=*".	Evaluation d'une comparaison "*>*" entre deux booléens.	`true > true`
Wrong parameter type of print - expected: int, float or string ≠ current	*print* avec un paramètre de type non autorisé.	*print* d'une variable booléenne.	`println(true);`
Wrong type : ... is not defined	Cast en un type non défini.	Cast d'une variable en un type *C* non préalablement défini.	`(C)(x) // C n'est pas definie`
Wrong right value class type - expected: subtype of...	Dans le cas d'un cast d'un objet en une autre classe, les classes source et destination n'ont aucun lien de parenté.	Cast d'un objet de type *A* en type *B* (les deux classes n'ont aucun lien de parenté).	`A a = (A)(new B()) // B et A n'ont pas de lien de parenté`
Wrong cast: Cannot cast ... in ...	Erreur de typage lors d'un Cast (hors classes).	Cast d'une chaîne de caractères en entier.	`int x = (int) ("a");`

Erreur	Description	Exemple	Code
Wrong class name: ... is already defined	Création d'une classe dont le nom est déjà utilisé.	Création d'une classe *A* déjà définie.	`class A {} class A{}`
Wrong superclass name: ... is not defined	Classe mère spécifiée non définie.	Création d'une classe *A* héritant de *B* avec classe *B* non définie.	`class A extends B {} // B n'est pas défini`
Unknown type: ...	Définition dans une classe d'un champ (d'une méthode / d'un paramètre d'une méthode) de type (de retour) inconnu.	Définition dans une classe d'un champ de type *C* non défini.	`class A {C c;} // le type C n'est pas défini`
Forbidden type : ...	Type interdit pour certaines utilisations (délcaration de variables/paramètres/champs, cast, comparaisons).	Déclaration d'une variable de type void.	`void x;`
Wrong field name: ... is already defined in the super class as a ...	Définition dans une classe d'un champ dont le nom est celui d'une méthode dans sa super-classe.	Définition dans une classe d'un champ *a* qui est défini comme une méthode dans sa super-classe.	`class A { void m(){} } class B extends A { int m; }`
Wrong method definition: Method name ... is already taken with a different signature	Redéfinition d'une méthode avec une signature différente.	Double définition dans une classe d'une méthode *m* avec une signature différente.	`class A { void m(int x){} void m(float x){}`
Wrong method definition: Return type of ... doesn't match	Type de retour de la méthode dans la classe fille différent du type de retour de la même méthode dans la classe mère.	Définition d'une méthode dans la classe mère avec un type de retour entier. Redéfinition de la méthode dans une classe fille avec un type de retour void.	`class A { int m(){return 1}} class B extends A {void m(){}}`
Wrong parameter name: ... is already defined	Paramètre déjà utilisé dans la définition de la méthode.	Méthode avec deux paramètres de nom *p*.	`class A { void m(int x, int x){}}}`
Wrong variable name: ... is not defined	Utilisation d'une variable non définie.	Assignation d'une valeur à une variable *x* non préalablement définie.	`x = x + 1 // x n'est pas défini`

Erreur	Description	Exemple	Code
Wrong type - expected: Class type ≠ current	*instanceof* sur une variable qui n'est pas de type Classe.	*a instanceof A* avec *a* de type entier.	`int x = 1; x instanceof Object`
Wrong method call: Cannot call method on null	Appel d'une méthode sur un objet null.		`A a = null; null.m();`
Wrong method call: mismatched number of arguments in the call of method	Appel d'une méthode avec un nombre d'arguments différent que dans la signature.	Appel d'une méthode avec des arguments manquants	`class A {void m(int x, int y){}} { (new A).m(1);}`
Wrong initialization object: Type ... is not defined	*New* avec un type non défini.		`new A(); // A n'est pas défini`
Wrong instruction: return statement not allowed in void method	*return* dans une méthode de type de retour void.		`Class A { void exemple() {return 1;}`
Wrong return type: it does not match method return type	Renvoie d'une valeur de mauvais type dans une méthode.	Méthode de type de retour entier renvoyant un flottant.	`Class A { int exemple() {return 2.0;}`
Wrong expression: cannot select a field of a non-class type	Tentative d'accès à un champ sur une variable de type non-classe.	Tentative d'accès à un champ sur une variable entière.	`Class A {int x;} {int y; y.x;}`
Wrong field access: Cannot access protected field: ...	Tentative d'accès à un champ *protected* depuis une variable dont le type n'est pas un sous-type de la classe concernée.		`Class A {protected int x;} Class B{} {B b=new B(); b.x;`

2.3. Erreurs d'exécution

2.3.1 Erreurs IMA

Toutes ces erreurs sont désactivables via l'option `-n` du compilateur sauf mention contraire.

Erreur	Description	Exemple
Error: Input/Output error	Non désactivable par l'option `-n`. Erreur d'entrée/sortie.	Saisie d'une lettre lors d'un `readInt()`.
Error: Overflow	Débordement arithmétique sur les flottants et division par 0.	Une opération arithmétique comportant un opérande flottant qui provoque un résultat flottant non codable sur 32 bits (trop grand en valeur absolue).
Error: Underflow	Débordement arithmétique sur les flottants par valeur inférieure.	Une opération arithmétique comportant un opérande flottant qui provoque un résultat flottant non codable sur 32 bits (trop proche de zéro).
Error: Null error	Déréférencement de null	Accès à un champs ou appel de méthode sur une référence null.
Error: Missing return statement	Absence de `return` lors de l'exécution d'une méthode	Une méthode non void qui ne comporte par d'instruction `return`
Error: Stack Overflow	Débordement de pile	Instanciation d'un objet dont un attribut est initialisé avec un objet de la même classe.
Error: Heap Overflow	Débordement de tas	Instanciation d'objet dans une boucle infinie

2.3.2 Erreurs IMA

Erreur	Description	Exemple
Comment [commentaire] contains [newline character|carriage return]	Impossibilité de faire un saut de ligne (\n) ou un retour charriot (\r) dans un commentaire ima	`;\n`

2.4. Erreurs Compilateur

Erreur	Description	Exemple
[options] are incompatible	Certaines options ne peuvent être lancées en même temps via decac, car leur exécution est incompatible.	`decac -p -v file`
Failed to open [out|in]put file: [file]	Le fichier d'entée ou de sortie n'a pas pu être ouvert, sûrement un problème de chemin de fichier ou de permissions.	`decac file.dec`

3. Extensions de la bibliothèque standard

Trigonométrie : math.decah

Nous avons implémenté une extension mathématique trigonométrique pour résoudre des calculs avec des fonctions sinusoïdales usuelles:

- sin
- cos
- arcsin
- arctan

La méthode ulp permet de contrôler la précision du résultat lors de la validation. Un ULP d'une valeur flottante est la distance positive entre la valeur donnée et la valeur suivante qui est plus grande en amplitude.

4. Utilisation des extensions

Signatures des méthodes math.deca

```
float sin(float f)
float cos(float f)
float asin(float f)
float atan(float f)
float ulp(float f)
```

Utilisation de math.decah

Les étapes pour utiliser les méthodes trigonométriques :

- Ajouter la bibliothèque math.decah avec : `#include "Math.decah"`
- Instancier la classe Math avec : `Math m = new Math();`
- Appeler les méthodes souhaitées avec les paramètres nécessaires : `m.sin(3.18)`

5. Limitation des extensions

Ensemble des valeurs entrées supportées

Les méthodes sin, cos, atan, ULP supportent tous les flottants (32 bits) sur R.

La méthode asin supporte tous les flottants (32 bits) sur l'intervalle [-1,1].

Limitations des précisions des résultats

Se référer au document BibliothèqueStandard.