

Manuel utilisateur

GL28

Janvier 2020

Table des matières

1	Mode opératoire du compilateur	3
1.1	Commande decac	3
1.1.1	Options	3
1.1.2	Sortie	3
2	Messages d’erreur	4
2.1	Commande decac	4
2.2	Lexicographie	4
2.3	Syntaxe abstraite	4
2.4	Syntaxe contextuelle	5
2.5	Codegen	9
3	Limitations du compilateur	10
4	Extension de la bibliothèque standard	11
4.1	Utilisation	11
4.2	Précision	11
4.2.1	Cosinus et Sinus	11
4.2.2	Arctan et Arcsin	12
4.3	Unit in the last place (ULP)	12
4.4	Limitations de l’extension	13

1 Mode opératoire du compilateur

1.1 Commande decac

`decac [OPTIONS] [DECAFILES]`

1.1.1 Options

- `-b` affiche la bannière du compilateur, à savoir "GL28".
- `-p` le compilateur s'arrête après l'exécution du parseur.
- `-v` le compilateur s'arrête après la vérification de la syntaxe contextuelle.
- `-n` supprime les tests de débordement à l'exécution.
- `-r` limite les registres disponibles.
- `-d` active les traces de debug, répéter l'option plusieurs fois pour avoir plus de traces.

1.1.2 Sortie

Un `<DECAFILE>` est de la forme `<FILENAME>.deca`.

Si la compilation est un succès un fichier `<FILENAME>.ass` est généré, et on peut exécuter ce fichier avec la commande :

```
$ decac <filename>.deca && ima <filename>.ass
```

On peut avoir autant de `<DECAFILE>` que l'on souhaite pour une même commande `decac`. Si le compilateur rencontre un programme deca erroné, il affiche l'erreur et passe au fichier deca suivant.

2 Messages d'erreur

2.1 Commande decac

1. `IllegalArgumentException` : "Missing source file(s) as arguments of decac command"

```
$ decac
```

2. `CLIException` : "invalid argument **<argument>**"

```
$ decac -a <filename>.deca # any letters but b, p, v, n, r, d, P
```

3. `CLIException` : "-v & -p incompatible, please remove one"

```
$ decac -v -p <filename>.deca
```

4. `UnsupportedOperationException` : "Parallel build not yet implemented"

```
$ decac -P <filename>.deca
```

5. `DecacFatalError` : "Failed to open input file *<filename>*"

```
$ decac <filename>.deca // <filename>.deca does not exist in current c
```

2.2 Lexicographie

6. "token recognition error at : '**<name>**'"

```
{  
    @  
}
```

7. "include file not found"

```
#include "unknowFile.deca"
```

8. "circular include for file **<name>**"

```
#include "a_include_b.deca"
```

2.3 Syntaxe abstraite

9. "left-hand side of assignment is not an lvalue"

```

{
    int x = 2;
    1 = x;
}

```

10. "value cannot be stored as an integer value"

```

{
    int x = 4500000000;
}

{
    int x = -4500000000;
}

```

11. "value cannot be stored as a float value"

```

{
    float x = 1.0E50;
}

{
    // arrondi à 0
    float x = 1.0E-50;
}

{
    float x = -1.0E50;
}

```

12. "no viable alternative at input '<name>'"

```

{
    float x = 1.0
}

```

13. "missing '<name>' at '<EOF>'"

```

{
    float x = 1.0;
}

```

2.4 Syntaxe contextuelle

14. "Undefined identifier <name> (0.1)"

15. "Type **<name>** is not defined (0.2)"

```
{ Integer x = y;}
```

16. "**<name>** is a method, please use a method call"

```
class A {  
    int x() {  
        return 1;  
    }  
    int y = x;  
}
```

17. "Variable, parameter or field **<name>** cannot be of type void (2.5 - 2.9 - 3.17)"

```
class A {  
    int x(void y) {} // parameter void  
}
```

```
class B {  
    void y; // field void  
}
```

```
{ void x;} // variable void
```

18. "Class **<name1>** is not a subclass of **<name2>** (not assign_compatible)"

```
class A {}  
class B extends A {}  
{  
    B b = new A(); // A a = new B(); would be correct  
}
```

19. "Expected class **<name1>**, got type **<name2>**"

```
class A {}  
{A x = 3;}
```

20. "Expected type **<name1>**, got type **<name2>**"

```
{boolean x = 3;}
```

21. "Condition must be of type boolean , got type **<name>** (3.29)"

```
{  
    int x = 1;  
    if (x) {}  
}
```

22. "Arithmetic operation **<opName>** is not supported for types **<name1>** and **<name2>** (3.33)"
- ```
{ int x = true + 1;}
```
23. "Invalid type for boolean operation **<opName>** : left operand is of type **<name1>** instead of boolean (3.33)"
- ```
{ boolean x = 1 && false;}
```
24. "Invalid type for boolean operation **<opName>** : right operand is of type **<name1>** instead of boolean (3.33)"
- ```
{ boolean x = true && 3*4;}
```
25. "Comparison operation **<opName>** : is not supported for types **<name1>** and **<name2>** (3.33)"
- ```
{ boolean y = 3.14 < false;}
```
26. "Exact comparison operation **<opName>** : is not supported for types **<name1>** and **<name2>** (3.33)"
- ```
{ boolean y = 3.14 == false;}
```
27. "Types of print arguments type must be String, int or float - not **<name>** (3.31)"
- ```
{ print(true);}
```
28. "Cannot extends class **<name>**, it is a predefined type"
- ```
class A extends boolean {}
```
29. "Superclass **<name>** is not defined (1.3)"
- ```
class A extends B {}
```
30. "Cannot declare class **<name>** it is a predefined type"
- ```
class int {}
```
31. "Class **<name>** is already defined (1.3)"
- ```
class A {}  
class A {}
```
32. "Field or method **<name>** is already defined in this class"

```
class A { int x; int x;}
```

33. "Field **<name>** should redefine another field"

```
class A { void x() {} }
class B extends A { int x;}
```

34. "Variable **<name>** is already declared at **<location>** (3.17)"

```
{ int x; int x;}
```

35. "Expression at **<location>** is not a class object, cannot do a method call"

```
class A {
    int one() {
        return 1;
    }
}
{
    int x;
    int y = x.one();
}
```

36. "Method **<name>** is not defined in class **<className>**"

```
class A {
    int x = this.getX();
}
```

37. "**<name>** is not a method, cannot do a method call"

```
class A {
    int y;
    int x = this.y();
}
```

38. "Wrong signature for call of method **<name>** : does not match its definition (3.71) : expected **<signature1>**, got **<signature2>**"

```
class A {
    int get2D(int x, float y) {}
    int x = get2D(3.14, 3.14);
}
\end{enumerate}
```

39. "Modulo operation is not supported for types **<name1>** and **<name2>** (3.51)"

```
{ int x = 3.14%4; }
```

40. "<**name**> is not a class type, cannot use New"
- ```
{ int x = new int();}
```
41. "Invalid type for unary operation "not" : <**name**> instead of boolean (3.37)"
- ```
{ boolean x = !1;}
```
42. "Expression at <**location**> is not a class object, cannot use ".""
- ```
{
 int x;
 int y = x.x;
}
```
43. "\"This\" cannot be used in main (3.43)"
- ```
{ int x = this.y;}
```
44. "Invalid type for operand "-" : <**name**> instead of int or float"
- ```
{ int x = -true;}
```
45. "Two parameters share the same name <**name**>"
- ```
class A {
    void get(int x, int x) {}
}
```

2.5 Codegen

46. "Assembly code does not support \$ character in method name <**name**>"
- ```
class A {
 void me$thod() {}
}
```
47. "Assembly code does not support \$ character in class name <**name**>"
- ```
class A$BC {}
```
48. "Error : Stack overflow" : si trop de variables sont déclarées dans le programme (dépend de la taille de la pile de la machine).
49. "Error : Impossible allocation : heap overflow" : si trop d'allocations mémoires à l'aide de l'instruction *new*


```
class A {}
{ A a = new A(); }
```

50. "Error : Division by zero"

```
{
    int x = 1/0;
    int y = 1%0;
    float z = 1.0/0.0;
}
```

51. Error : "Overflow during arithmetic operation" : lors d'une opération arithmétique flottante dont le résultat ne peut pas être codé sur un flottant

```
{ float x = 5000000000.0 * 5000000000; }
```

52. "Error : Invalid input" : si une instruction *readInt()* (resp. *readFloat()*) reçoit autre chose qu'un *int* (resp. *float*).

53. "Error : Cannot acces null object" : lors de l'accès à un champ d'un objet non initialisé, ou lors de l'appel d'une méthode d'un objet non initialisé

```
class A {
    int x = 1;
}
{
    A a = new A();
    a.x;
}
```

54. "Error : Impossible converion to float" : lors de la conversion d'un *int* en *float*, si le resultat de la conversion ne peut pas être codé sur un flottant

```
{ float x = 5000000000 * 5000000000; }
```

3 Limitations du compilateur

La compilation en parallèle de plusieurs fichiers deca n'est pas supportée par ce compilateur.

Le cast explicite et la méthode *instanceof* ne sont pas supportés par ce compilateur.

4 Extension de la bibliothèque standard

4.1 Utilisation

Les fonctions implémentées sont les suivantes :

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

Pour utiliser l'une de ces fonctions dans un programme *.deca*. Il faut mettre en tête du programme (i.e. avant la déclaration des classes) :

```
#include "Math.decah"
```

Puis en début de main :

```
{
    Math m = new Math();
    ...
}
```

ainsi on peut appeler les fonctions listées si-dessus sans oublier le préfixe **m.** .

4.2 Précision

Pour toutes les fonctions nous allons tracer

$$\left| \frac{fonctionDeca - fonctionJava}{ulp} \right|$$

afin de mesurer la précision.

4.2.1 Cosinus et Sinus

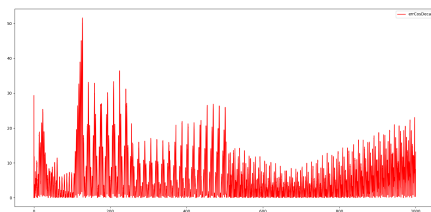


FIGURE 1 – Erreur cosinus

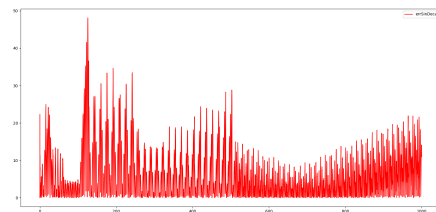


FIGURE 2 – Erreur sinus

Ainsi on remarque que pour des valeurs inférieures à 1000. La précision pour *sin* et *cos* est inférieure à 50 ULP.

4.2.2 Arctan et Arcsin

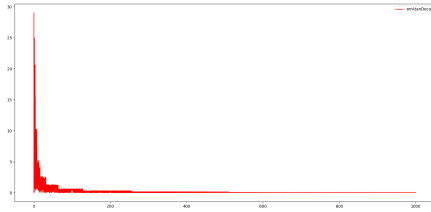


FIGURE 3 – Erreur Arctan

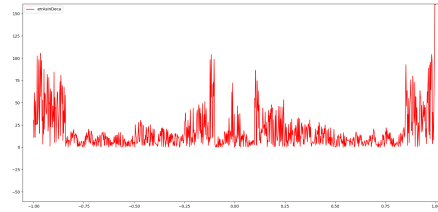


FIGURE 4 – Erreur Arcsin

De même pour Arctan on peut assurer une précision inférieure à 30 ULP. Quant à Arcsin la précision est inférieure à 100 ULP.

4.3 Unit in the last place (ULP)

Sur ce graphe on a pas tracé l'erreur en ULP mais en valeur réelle i.e.

$$|fonctionDeca - fonctionJava|$$

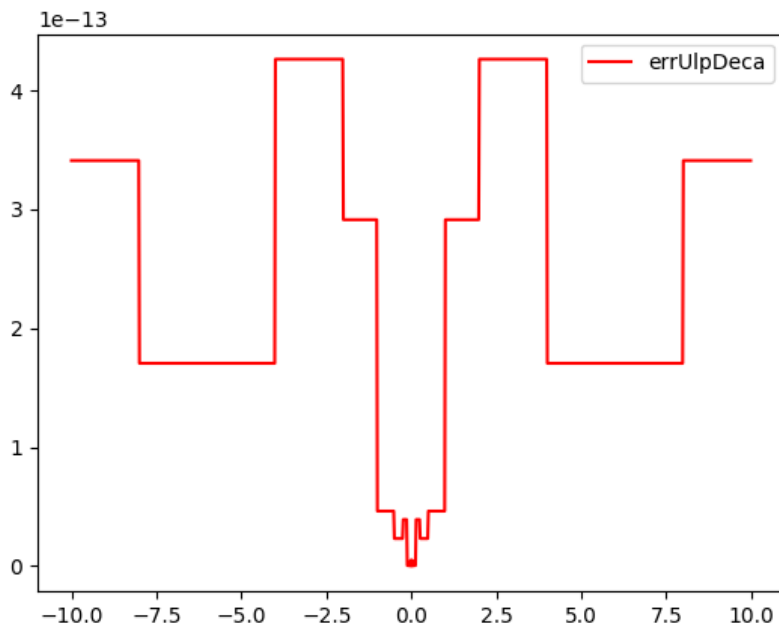


FIGURE 5 – Erreur ULP (absolue)

On constate donc une précision de l'ordre de 10^{-13} aux alentours de 0.

4.4 Limitations de l'extension

Attention, $ulp(0)$ ne fonctionne pas, on peut remplacer de manière arbitraire par $ulp(0.00001)$ par exemple.

Pour plus d'informations concernant les précisions et calculs de coût consultez la documentation de conception.