

Manuel d'utilisation du compilateur Decac

Version 1.0 (release)

- [Introduction](#)
- [Description du compilateur](#)
- [Pour démarrer](#)
- [Options de Decac](#)
- [Limitations du compilateur](#)
- [Messages d'erreur](#)
 - [Erreurs de lexicographie](#)
 - [Erreurs de syntaxe :](#)
 - [Erreurs de compilation](#)
 - [Erreurs à l'exécution](#)
- [Extension de la bibliothèque standard](#)
 - [Manipulation basique de tableaux](#)
 - [Manipulation avancée de tableaux](#)
 - [Tableaux et matrices](#)
 - [Matrices](#)
 - [Classes](#)
 - [Fonctions](#)

Introduction

Ce manuel illustre comment utiliser le compilateur Decac, ainsi que ses fonctionnalités.

Description du compilateur

Le compilateur Decac est un programme qui traduit un fichier source écrit en langage de programmation Deca en un fichier assembleur qui est interprétable par un interprète de Machine Abstraite Ima. Le programme Decac comporte plusieurs options afin de pouvoir réaliser différentes fonctionnalités (voir [Options de Decac](#)). Il est également possible d'utiliser des tableaux comme en Java avec plusieurs opérations de calcul possible dessus (voir [Extension de la bibliothèque standard](#)). Si la compilation s'est bien déroulée le résultat se trouve au même endroit que le fichier source Deca et s'appelle `nomdufichier.r.ass`. Afin de l'exécuter vous pouvez lancer `ima nomdufichier.ass`.

Pour démarrer

Ce dépôt contient le code source du compilateur que nous avons développé. Pour utiliser notre compilateur, vous avez besoin de cloner le dépôt en utilisant git :

```
git clone https://nicolasflamel.ensimag.fr/gl2023/gr3/gl11.git
cd gl11/
mkdir global/
cd global/
tar xvzf ../docker/ima_sources.tgz
make realclean
make
```

Ensuite

```
mvn compile
mvn test-compile
```

Ouvrez votre `$HOME/.bashrc` et ajoutez à la fin du fichier :

```
export PATH="$HOME/.../gl11/global/bin/ima"
alias decac="$HOME/.../gl11/src/main/bin/decac"
```

Une fois cela fait, tapez dans votre terminal :

```
source "$HOME"/.bashrc
```

Une fois le dépôt cloné, vous pouvez créer un fichier de test simple en utilisant un éditeur de texte de votre choix. Par exemple, vous pouvez créer un fichier nommé `test.deca` contenant un programme "Hello World" simple :

```
class HelloWorld {
    void main() {
        print("Hello World!");
    }
}
```

Pour compiler ce fichier, vous pouvez utiliser la commande suivante :

```
decac test.deca
```

Cela générera un fichier `test.ass` que vous pourrez exécuter en utilisant l'émulateur `ima` :

```
ima test.ass
```

Cela devrait afficher "Hello World!" dans la console. Vous pouvez maintenant explorer les fonctionnalités de notre compilateur en modifiant ce fichier de test ou en en créant un nouveau. N'hésitez pas à consulter la documentation pour plus d'informations sur les options et les fonctionnalités disponibles.

Options de Decac

Notre compilateur propose les options suivantes :

- | | |
|------|--|
| -b | Affiche la bannière qui représente le nom de l'équipe de développement. Cette option ne peut être utilisée qu'indépendamment des autres options et ne nécessite pas de fichier source. Lorsque cette option est utilisée, le compilateur s'arrête après avoir affiché la bannière. |
| -p | Arrête la compilation après l'étape de construction de l'arbre, et affiche la décompilation de ce dernier. S'il n'y a qu'un seul fichier source à compiler alors la sortie est un programme Deca syntaxiquement correct. Cette option est incompatible avec l'option -v. |
| -v | Arrête la compilation après l'étape de vérification contextuelle. En l'absence d'erreur, cette option ne produit aucune sortie. Sinon, elle permet l'affichage d'un message d'erreur. Cette option est incompatible avec l'option -p. |
| -n | Ne teste pas les points 11.1 et 11.3 pendant la compilation. Voir [11.1] et [11.3] du cahier des charges. |
| -r x | Limite les registres disponibles à $R_0 \dots R_{\{x-1\}}$ avec x entre 4 et 16. |

<code>-d</code>	Active les traces de debug. Cette option peut être répétée jusqu'à trois fois pour avoir plus de traces.
<code>-P</code>	Lance la compilation des fichiers en parallèle s'il y a plusieurs fichiers sources.
<code>-w</code>	Active l'affichage de messages d'avertissement ("warnings") pendant la compilation.
<code>-tab</code>	Pour utiliser des tableaux dans votre code, vous pouvez utiliser l'option <code>-tab</code> . Si vous souhaitez utiliser des outils de calcul sur les tableaux, il est nécessaire d'inclure la librairie <code>Ar rayLib.decah</code> ou <code>Matrix.decah</code> en plus de cette option. Pour en savoir plus sur l'utilisation des tableaux, veuillez consulter la section dédiée Bibliothèque tableaux de la documentation.

Les options `-p` et `-v` sont incompatibles.

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

Limitations du compilateur

Notre compilateur a été développé avec soin pour couvrir tous les cas d'utilisation et fonctionner de manière optimale. Nous avons mis en place des tests rigoureux pour garantir la qualité de notre produit. Cependant, étant donné que la partie bibliothèque de tableaux n'a pas encore été pleinement testée, il se peut qu'il y ait des bugs ou des erreurs dans cette partie spécifique. Nous vous encourageons à signaler tout problème rencontré lors de l'utilisation de cette fonctionnalité, afin que nous puissions le corriger rapidement. Cela étant dit, nous sommes confiants dans la qualité générale de notre compilateur et nous sommes convaincus qu'il répondra à vos besoins de compilation de code. Nous continuerons à mettre à jour et à améliorer notre produit pour vous offrir la meilleure expérience possible.

Si vous souhaitez contribuer au développement ou à la maintenance de notre compilateur, vous pouvez nous contacter par courrier électronique à l'adresse contribution@decacompiler.com. Vous pouvez également récupérer le projet à partir du dépôt et commencer à développer dès maintenant. Pour vous aider dans vos démarches, nous vous recommandons de consulter la documentation de conception en plus de ce

document, pour vous familiariser avec les fonctionnalités et les spécificités du compilateur.

Messages d'erreur

Notre compilateur peut renvoyer différents types de messages d'erreur pour vous aider à identifier les problèmes dans votre code. Ces erreurs peuvent être liées à des problèmes de lexicographie, de syntaxe hors-contexte, de syntaxe contextuelle ou d'exécution du code assembleur.

Les erreurs de lexicographie sont des erreurs liées aux mots-clés ou aux identificateurs du langage, par exemple : `"monfichier.java:3:10: mot-clé 'class' attendu mais 'clas' trouvé"`.

Les erreurs de syntaxe hors-contexte sont des erreurs liées à la grammaire du langage, telles que des parenthèses mal fermées, par exemple : `"monfichier.java:5:8: Erreur de syntaxe, ')' attendu mais '}' trouvé"`.

Les erreurs de syntaxe contextuelle sont des erreurs liées à l'utilisation incorrecte des éléments du langage dans un contexte particulier, telles que l'utilisation d'une variable non déclarée, par exemple : `"monfichier.java:7:15: variable 'x' non déclarée"`.

Les erreurs d'exécution du code assembleur sont des erreurs qui surviennent lors de l'exécution du code généré par le compilateur, telles que des erreurs de segmentation ou de dépassement de mémoire, par exemple : `"monfichier.java:4:23: Erreur d'exécution : dépassement de mémoire"`.

Erreurs de lexicographie

- `<nom de fichier.deca>:<ligne>:<colonne>: token recognition error at <'token pas valide'>`

Erreurs de syntaxe :

- `<nom de fichier.deca>:<ligne>:<colonne>: no viable input at input <token pas dans la grammaire>`

Erreurs de compilation

- `<nom de fichier.deca>:<ligne>:<colonne>:"La variable <name> n'a pas été déclaré : rule 0.1"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"The type <type> doesn't exist : rule 0.2"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"The field <field> is already declared : rule 2.4"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"The method <method> is already declared : rule 2.6"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"<name> already declared as field in super class : rule 2.7"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"The parameter <name> is already used : rule 3.12"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"Can't do instanceof between <name> and <type> : rule 3.40"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"Can't call method on <name> : rule 3.71"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"The signature doesn't match expected : rule 3.73"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"Can't do new on <name> + <type> + : rule 3.42"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"Return can't be void : rule 3.24"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"Can't select field from <type> : rule 3.65"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"Can't acces a protected field in main : rule 3.66"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"Can't acces a protected field from a foreign class : rule 3.66"`
- `<nom de fichier.deca>:<ligne>:<colonne>:"Can't call this in the main program : rule 3.43"`

Erreurs à l'exécution

`<nom de fichier.deca>:<ligne>:<colonne>: <description informelle du problème>`

Extension de la bibliothèque standard

Manipulation basique de tableaux

Pour utiliser les fonctionnalités de manipulation de tableaux de notre compilateur, vous devrez utiliser la commande `decac` avec l'option `-tab` lors de la compilation de vos fichiers. Cela permettra au compilateur de détecter l'utilisation de tableaux. Il n'est pas obligatoire d'inclure la bibliothèque `ArrayLib.decah` pour déclarer et utiliser des tableaux, cependant si vous souhaitez utiliser des fonctionnalités avancées de manipulation de tableaux il vous faudra l'inclure. Les informations sur la déclaration et l'utilisation de tableaux sont disponibles ci-après.

Syntaxe de déclaration de tableaux

```
int[] tab;  
int[][] tab;  
int[]...[] tab;
```

Syntaxe de déclaration et d'initialisation de tableaux

```
int[] tab = new int[N]();  
int[][] tab = new int[N][]();  
int[][] tab = new int[N][M]();  
int[]...[] tab = new int[N][N1]...[Nk]();  
int[]...[] tab = new int[N][N1]...[Nk][]...[]();
```

Syntaxe de d'initialisation explicite

```
int[] tab = {1, 2, 3};  
int[][] tab = {{0, 1}, {0, 1}, {0, 1}};
```

Syntaxe d'initialisation avec sous-tableaux de taille variée

```
int[][] tab = {{1}, {2, 2}, {3, 3, 3}};  
int[][][] tab = {{{1}, {2, 2}}, {{3, 3, 3}, {1}}, {}};
```

Manipulation de tableaux

```
// Déclaration et initialisation de deux tableaux
int[] t1 = {1, 2, 3};
int[] t2 = {{1, 2, 3}, {1, 1, 1}};

// Affichage du premier élément du premier tableau
println("t1[0] = ", t1[0]);

// Affichage du deuxième tableau
int i = 0, j = 0;
while (i < t2.length) {
    while (j < t2[i].length) {
        print(t2[i][j], ' ');
        j = j + 1;
    }
    i = i + 1;
    println();
}
```

Cela fonctionne de manière similaire pour des tableaux de flottants, de booléens et d'objets.

Manipulation avancée de tableaux

Cette section fournit des informations sur les fonctionnalités avancées de manipulation de tableaux et de matrices de notre compilateur. Pour bénéficier de ces fonctionnalités, il est nécessaire d'inclure la bibliothèque `ArrayLib.decah` au début de votre code source avec l'instruction `#include "ArrayLib.decah"`. De même, pour bénéficier des fonctionnalités sur les matrices, il est nécessaire d'inclure `MatrixLib.decah`. Cela vous permettra d'accéder à un ensemble de fonctions supplémentaires pour travailler efficacement avec des tableaux et des matrices.

Pour une documentation plus détaillée, notamment sur les spécificités de l'implémentation, veuillez consulter le guide intitulé "Documentation de l'extension".

Tableaux et matrices

La syntaxe d'utilisation des méthodes de manipulation avancées des tableaux et des matrices est illustrée ci-après.

```
int[] tab = new int[100](); // création d'un tableau à 100 éléments
ArrayLib.fill(tab, 1);     // remplit le tableau avec des 1
```



```

int[] tab2 = {1, 3, 2, 4};
Index index = Array.search(tab2, 3);
// renvoie l'indice qui est la position de l'entier 3 dans `tab2`

ArrayLib.compare(tab, tab2);
// renvoie `false` car les tableaux `tab` et `tab2` ne sont pas
identiques

int[] tab2Trie = ArrayLib.quicksort(tab2, 0, tab2.length);
// renvoie le tableau `tab2` trié : {1, 2, 3, 4}

int[] tabCopy = ArrayLib.deepCopy(tab2);
// créé un nouveau tableau qui est une copie de tab2

int[] tabCopyRange = ArrayLib.deepCopyrange(tab2, 0, 2);
// créé un nouveau tableau qui est une copie de tab2 des éléments de
l'incide 0 à 2
// tabCopyRange vaut : {1, 3, 2}

```

`void printTabFloat` **Affiche un tableau.**

(float[] tab)

`void printTabInt`(int

[] tab)

`void printTabBoolean`

(boolean[] tab)

`void fillFloat`(float **Remplie le tableau passé en argument avec l'élément donné.**

[] tab, float
element)

`void fillInt`(int[]

tab, int element)

`void fillBoolean`

(boolean[] tab,
boolean element)

`int searchFloat`(float **Renvoie un entier qui indique la position dans le tableau tab**
[] tab, float **du premier élément recherché , s'il a été trouvé. Sinon, l'**
element) **entier renvoyé vaut -1.**

`int searchInt`(int[]

tab, int element)

```
int      searchBool  
(boolean[] tab,  
boolean element)
```

```
int      searchObject  
(Object[] tab,  
Object element)
```

```
void quicksortFloat  
(float[] tab)
```

Trie le tableau `tab` passé en argument, de manière croissante.

```
void quicksortInt(int  
[] tab)
```

```
int partitionFloat  
(float[] tab, int  
low, int high)
```

Bouge les éléments du tableau par rapport au pivot.

```
int partitionInt(int  
[] tab, int low, int  
high)
```

```
v o i d  
quicksortRecFloat  
(float[] tab, int  
low, int high)
```

Trie le tableau passé en argument, de l'index `low` à l'index `high`, de manière croissante.

```
void quicksortRecInt  
(int[] tab, int low,  
int high)
```

Matrices

Les fonctionnalités spécifiques à la manipulation de matrices proposées par notre compilateur sont destinées uniquement pour les matrices contenant des entiers ou des nombres flottants. Il est à noter que la méthode Gauss-Jordan ne prend en compte que des matrices contenant des nombres flottants. Si vous souhaitez utiliser cette méthode sur une matrice contenant des entiers, il vous faudra convertir explicitement la matrice entière en matrice flottante (`castMatrixIntToFloat`).

Les structures de données et méthodes spécifiques aux matrices sont résumées ci-dessous.


Classes

La classe suivante permet de stocker deux valeurs importantes pour le calcul matriciel, le rang et le déterminant.

<code>class RangDet</code>	Structure contenant deux valeurs : le rang d'une matrice (<code>int</code>) et son déterminant (<code>float</code>).
----------------------------	--

Fonctions

 Les méthodes suivantes sont implémentées par la méthode de Gauss-Jordan :

<code>IntFloat gaussJordan</code> <code>(float[][] matrix)</code>	 Attention, cette méthode modifie totalement la matrice passée en argument.
--	--

<code>float[][] inverse</code> <code>(float[][] matrix)</code>	Renvoie une nouvelle matrice, inverse de la matrice <code>matrix</code> .
---	---

<code>float det(float[][]</code> <code>matrix)</code>	Calcul du déterminant d'une matrice.
--	--------------------------------------

<code>float rang(float[][]</code> <code>matrix)</code>	Calcul du rang d'une matrice.
---	-------------------------------

 Fonctions arithmétiques classiques sur les matrices :

<code>float traceFloat</code> <code>(float[][] matrice)</code>	Calcul de la trace d'une matrice carrée.
---	--

<code>int traceInt(int[][]</code> <code>matrice)</code>	
--	--

<code>float[][] addFloat</code> <code>(float[][] A, float[]</code> <code>[] B)</code>	Renvoie une nouvelle matrice, résultat de l'addition de la matrice <code>A</code> par la matrice <code>B</code> .
---	---

<code>int[][] addInt(int[]</code> <code>[] A, int[][] B)</code>	
--	--

<code>float[][] subFloat</code> <code>(float[][] A, float[]</code> <code>[] B)</code>	Renvoie une nouvelle matrice, résultat de la soustraction de la matrice <code>A</code> par la matrice <code>B</code> .
---	--

```
int[][] subInt(int[]  
[] A, int[][] B)
```

```
float[][]  
multScalarFloat(float  
[][] A, float lambda)
```

```
int[][] multScalarInt  
(int[][] A, int  
lambda)
```

```
float[][]  
multMatrixFloat(float  
[][] A, float[][] B)
```

```
int[][] multMatrixInt  
(int[][] A, int[][]  
B)
```

```
float[]  
multArrayMatrixFloat  
(float[] A, float[]  
[] B)
```

```
int[]  
multArrayMatrixInt  
(int[] A, int[][] B)
```

```
float[]  
multMatrixArrayFloat  
(float[][] A, float  
[] B)
```

```
int[]  
multMatrixArrayInt  
(int[][] A, int[] B)
```

Renvoie une nouvelle matrice, résultat de la multiplication de `A` par le scalaire `lambda`.

Renvoie une nouvelle matrice, résultat de la multiplication des matrices `A` et `B`.

Renvoie un tableau (vecteur), résultat du produit matriciel de `A` (tableau) par `B` (matrice).

Renvoie un tableau (vecteur), résultat du produit matriciel de `A` (matrice) par `B` (tableau).

🔗 Fonctions utiles sur les matrices :

```
float[][] deepCopy  
(float[][] matrix)
```

Crée une nouvelle matrice à partir de la matrice `matrix` passée en paramètre. La méthode `deepCopy` n'est pas implémentée pour les matrices d'Objets.

```
float[][][]  
deepCopyOfRange(float  
[][][] matrix, int  
linStart, int  
colStart, int  
linEnd, int colEnd)
```

```
void fill(float[][]  
matrice, float  
element)
```

```
int[] search(float[]  
[] matrice, float  
element)
```

```
boolean compare(float  
[][] A, float[][] B)
```

```
float[][][]  
castMatrixIntToFloat  
(int[][][] matrix)
```

```
int[][][]  
castMatrixFloatToInt  
(float[][][] matrix)
```

Crée une nouvelle matrice à partir de la matrice `matrix` passée en paramètre, en ne prenant que les éléments dans le rectangle `[linStart, colStart]x[linEnd, colEnd]`. La méthode `deepCopyOfRange` n'est pas implémentée pour les matrices d'Objets.

Remplie la matrice passée en argument avec l'élément donné.

Renvoie un tableau de longueur deux, qui indique la position dans la matrice `matrice` du premier élément recherché, s'il a été trouvé. Sinon, le tableau renvoyé est `{-1, -1}`.

Sur une matrice d'entiers ou de flottants, renvoie `true` si les matrices sont identiques, `false` sinon.

Renvoie une nouvelle matrice flottante correspondant à la matrice entière passée en argument (et inversement).